FISHER NETWORKS: A PRINCIPLED APPROACH TO RETRIEVAL-BASED CLASSIFICATION

Héctor Ruiz

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

Liverpool – April 2013

Abstract

Due to the technological advances in the acquisition and processing of information, current data mining applications involve databases of sizes that would be unthinkable just two decades ago. However, real-word datasets are often riddled with irrelevant variables that not only do not generate any meaningful information about the process of interest, but may also obstruct the contribution of the truly informative data features. Taking into consideration the relevance of the different measures available can make the difference between reaching an accurate reflection of the underlying truth and obtaining misleading results that cause the drawing of erroneous conclusions.

Another important consideration in data analysis is the interpretability of the models used to fit the data. It is clear that performance must be a key aspect in deciding which methodology to use, but it should not be the only one. Models with an obscure internal operation see their practical usefulness effectively diminished by the difficulty to understand the reasoning behind their inferences, which makes them less appealing to users that are not familiar with their theoretical basis.

This thesis proposes a novel framework for the visualisation and categorisation of data in classification contexts that tackles the two issues discussed above and provides an informative output of intuitive interpretation. The system is based on a Fisher information metric that automatically filters the contribution of variables depending on their relevance with respect to the classification problem at hand, measured by their influence on the posterior class probabilities.

Fisher distances can then be used to calculate rigorous problem-specific similarity measures, which can be grouped into a pairwise adjacency matrix, thus defining a network. Following this novel construction process results in a principled visualisation of the data organised in communities that highlights the structure of the underlying class membership probabilities. Furthermore, the relational nature of the network can be used to reproduce the probabilistic predictions of the original estimates in a case-based approach, making them explainable by means of known cases in the dataset.

The potential applications and usefulness of the framework are illustrated using several real-world datasets, giving examples of the typical output that the end user receives and how they can use it to learn more about the cases of interest as well as about the dataset as a whole.

Acknowledgments

First of all, I would like to thank the School of Computing and Mathematical Sciences at LJMU for these three years of financial support. The opportunity they gave me has not only substantially enriched my academic life, but has also allowed me to experience a different country and culture, which I consider just as valuable as a university degree.

Of course, a huge thank you goes to my supervisory team. It is the biggest cliché of acknowledgments, but I truly believe I would not have been able to get to this point without your support and encouragement. I am lucky to have enjoyed so many interesting discussions with Professor Paulo Lisboa; I was impressed by his brightness and natural talent for science when I first met him, and I am even more so after these years. It was also great having Dr. Ian Jarman on board; he combines a strong statistical background with a useful pragmatic character that research often lacks. And, of course, Dr. José Martín, whose sound advise brought me into this project. Despite living in a different country, he was always keen to help and contribute, and his regular visits were very appreciated. It has not been an easy journey, but it is safe to say that without the three of you it would have been much tougher.

And last, but certainly not least, a mention to my friends and family in Spain, in particular to my parents. It is not always easy when you are far from home, but I have always felt your love and support.

Contents

1 Introduction **1**

1.1 Statistical models . . . 2
1.2 Rule extraction . . . 6
1.3 Graphical models . . . 12
1.4 Aims and objectives . . . 15
1.5 Structure of the thesis . . . 16

2 Literature review **18**

2.1 Distance metric learning . . . 18
2.1.1 Unsupervised metric learning . . . 19
Principal component analysis (PCA) . . . 19
Multidimensional scaling (MDS) . . . 22
Isomap . . . 22
Locally linear embedding (LLE) . . . 24
2.1.2 Supervised metric learning . . . 26
Nearest neighbour classification and distance metric learning . . . 26
Metric learning using similarity-based optimisation . . . 28
Metric learning based on relevance measures . . . 31
Metric learning with kernels . . . 37
Fisher information metric in the input space . . . 41
2.2 Analysis of community structure in networks . . . 42
2.2.1 Traditional community extraction . . . 44
The graph partitioning problem . . . 44
Hierarchical clustering . . . 48
2.2.2 Modern community extraction . . . 50
Edge removal algorithms . . . 50
Modularity optimisation algorithms . . . 53
2.3 Methodology selection . . . 55
2.3.1 Selection of the distance metric . . . 56

2.3.2 Selection of the probability estimation method . . . 56
Generative models . . . 56
Discriminative models . . . 57
2.3.3 Selection of the community detection algorithm . . . 58
2.4 Novel contributions . . . 59
Fisher network framework for data visualisation and classification . . . 59
FI metric derivation using discriminative models . . . 59
Closed-form distance measure for linear models and iterative geodesic finder 59
FI metric and blind signal separation . . . 60
Reference case identification and informative backgrounds . . . 60
Related work . . . 60
2.5 Chapter summary . . . 61

3 Methodology description 63
3.1 Probability estimation . . . 63
3.1.1 Binary classification . . . 64
Linear estimators: Linear logistic regression . . . 65
Non-linear estimators: Multilayer perceptron . . . 66
3.1.2 Multiclass classification . . . 69
3.2 Fisher information matrix . . . 69
3.2.1 FI matrix for two classes . . . 70
3.2.2 FI matrix for multiple classes . . . 71
3.3 Fisher information metric . . . 71
3.3.1 The Fisher metric and the KL divergence . . . 71
3.3.2 Global distances for linear estimators . . . 72
3.3.3 Global distances for non-linear estimators: finding geodesic distances . . . 73
Straight path approximation . . . 73
Graph-based approximation . . . 74
The free point approach . . . 74
3.4 Fisher networks . . . 76
3.4.1 Selection of the network locality parameter . . . 77
Faithfulness of the network predictions . . . 77
Cramer's V statistic . . . 78
McNemar's test . . . 78
3.5 Community detection . . . 79
3.6 Practical uses of Fisher networks . . . 81
3.6.1 Dataset visualisation . . . 81
Informative backgrounds . . . 82

3.6.2 Identification of reference cases . . . 83
3.6.3 Case-based classification . . . 84
3.7 Chapter summary . . . 85

4 Experimental results **86**
4.1 The effect of the Fisher metric . . . 86
4.2 Fisher metric and KNN classification . . . 88
4.3 Comparison of geodesic calculation methods . . . 92
4.3.1 Performance comparison on real-world data . . . 92
4.3.2 The effect of the sample size . . . 94
4.4 Fisher networks for real-world data . . . 96
4.4.1 MLP classification benchmark . . . 97
4.4.2 Construction of Fisher networks . . . 100
Empirical selection of the network locality parameter . . . 100
Fisher network examples . . . 106
Fisher networks with informative backgrounds . . . 115
Reference cases in Fisher networks . . . 122
4.5 Supervised blind signal separation using FI metrics . . . 127
Convex non-negative matrix factorisation . . . 127
Description of the MRS data . . . 128
Results . . . 129
4.6 Chapter summary . . . 132

5 Review, conclusions and future work **133**
5.1 Review . . . 133
5.2 Conclusions . . . 135
5.3 Future work . . . 135

A Derivation of the FI matrix **138**
A.1 FI matrix derivation for two classes . . . 138
A.2 FI matrix derivation for multiple classes . . . 139

B Interpretation of the FI metric in the context of information and divergence **141**
B.1 Information . . . 141
B.2 Divergence . . . 142
B.3 Regularity conditions . . . 143
B.4 Connection with the Fisher information . . . 144

C Derivation of the distance expression for linear estimators **145**

References **147**

Publications **155**

Chapter 1

Introduction

Machine learning methods are designed to discover the underlying processes and relationships that may exist in data. With this purpose in mind, users often find themselves trying to fit models to the data as closely as possible, in an effort for them not to miss any of the patterns present in the available information. However, as desirable as a good model-to-data fit is, it should not overshadow the importance of interpretability. There is always a gap between data modelling and knowledge extraction [1], and only by reducing it can we improve the understanding of the hidden mechanisms that generated the data.

In this context, as well as throughout this thesis, a model is deemed *interpretable* if the results that it provides can be explained in the language of the domain of the problem at hand in a way that is intuitive to the human cognition of the users. In practical applications, the success of inference models depends as much on the ability to interpret their internal operation as it does on the accuracy of their predictions. From a pragmatic point of view, a model is only useful if the user accepts the answers that it produces, and that is unlikely to happen unless it is clear to them which are the factors that explain the conclusions obtained.

Interpretability is particularly important in fields where the potential impact of the decisions taken is large, as is the case, for example, of medical decision support systems. In that area, the factors that determine the recommendations produced and the relationship between them must be well understood by the clinicians, otherwise they will ignore the system and use the corresponding default protocol. A similar need for interpretability can also be found in other scenarios, e.g., retail and industry.

This means we need to open up the black boxes and provide intuitive ways to understand how our methods do what they do. This chapter summarises some of the existing approaches to bridge this gap in different areas of data mining, either by suggesting new ways to look at known models or by creating systems that are naturally interpretable.

1.1 Statistical models

The issue of interpretability arises even in the simplest statistical models, such as linear regression, where a variable y is approximated by a linear combination of a set of explanatory variables x_i:

$$y \approx \tilde{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_N x_N. \tag{1.1}$$

The approximation $\tilde{y}$, given by Eq. (1.1), represents a hyperplane in the N-dimensional space of the covariates. This plane is fitted to the true surface that y defines, and therefore interpreting the model comes down to interpreting the regression coefficients β_i. The immediate meaning of a coefficient β_i is the change in $\tilde{y}$ when x_i is increased by 1, while the rest of the variables are kept constant. However, because variables are normally measured in different units, the significance of a unit increase may vary greatly from one variable to another, making the comparison of coefficients meaningless. To prevent this, regression coefficients can be standardised by using normalised variables [2], resulting in directly comparable coefficients that measure change in the output when the corresponding input increases by one standard deviation and the rest of variables remain constant.

Another important detail is that if variables are added or removed from the set of inputs, like in

$$\begin{aligned} \tilde{y} &= \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3, \\ \tilde{y} &= \gamma_1 x_1 + \gamma_2 x_2, \\ \tilde{y} &= \delta_1 x_1 + \delta_4 x_4, \end{aligned} \tag{1.2}$$

the coefficient corresponding to a particular variable usually changes from one model to another [3] (i.e., in Eq. (1.2), β_1, γ_1 and δ_1 are different in general). In other words, the meaning of a coefficient also depends on which other variables are available in the model.

Of course, this interpretation of the coefficients assumes that it is possible to change each x_i without affecting the rest of variables, which is seldom true in practice due to correlations usually existing between variables. This shows that interpreting models, even simple ones, is not always straightforward.

There also exist graphical approaches to interpret linear models, such as the use of nomograms. Originally, nomograms [5] were devised as a graphical aid for engineers to be able to quickly calculate complex formulas. More recently, they have been used as visual representations of regression models, allowing the user to see the impact that the different variables have on the outcome. Fig. 1.1 shows the use of a nomogram to visualise the result of a Cox proportional hazards regression analysis, a type of regression

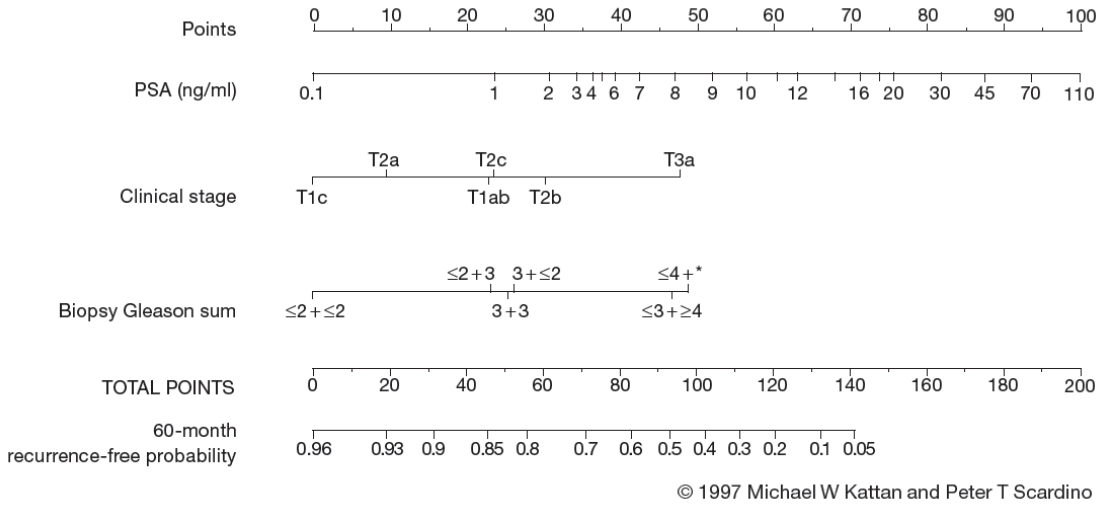


Figure 1.1: *Preoperative nomogram based on 983 patients treated at Baylor College of Medicine, Houston, TX, for predicting freedom from recurrence after radical prostatectomy* [4]. *The authors provided the following instructions of use for physicians:*
"Locate the patient's PSA on the PSA axis. Draw a line straight upwards to the Points axis to determine how many points towards recurrence the patient receives for his PSA. Repeat this process for the Clinical Stage and Biopsy Gleason Sum axes, each time drawing straight upward to the Points axis. Sum the points achieved for each predictor and locate this sum on the Total Points axis. Draw a line straight down to find the patient's probability of remaining recurrence-free for 60 months assuming he does not die of another cause first".
Image from [4]. *Reprinted with kind permission from Oxford University Press.*

used to predict the probability of some event taking place past a certain time. In this example [4], the model predicts the probability of treatment failure before the actual treatment (a surgical procedure) is carried out. The nomogram combines the three available preoperative factors to assist the physician and patient in deciding whether or not to proceed with the treatment.

Interpretability in non-linear models is just as important, if not more so, than it is in linear cases. The internal operation of these models is usually complex, resulting in an intricate input/output relationship. For this reason, it is usually difficult for the user to identify the factors that have a bigger impact on the results obtained. One way of approaching this is through the use of variable selection [6], normally as a preprocessing step preceding the main algorithm. The idea is to reduce the initial set of available variables to a subset that contains a selection of those that are most relevant according to some criteria previously specified. Similarly, feature selection derives a compact set of features by transforming and combining the original variables. By discarding irrelevant variables, data visualisation and interpretation becomes simpler, and the use of more informative data during the training phases of the models improves their prediction capabilities.

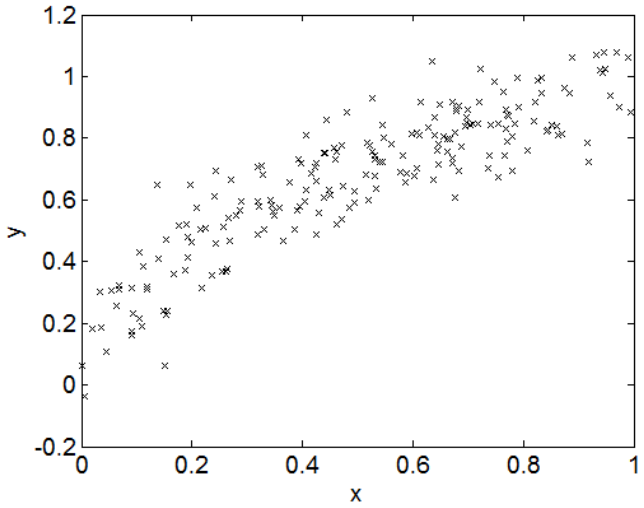


Figure 1.2: *Scatterplots are simple sensitivity analysis tools that allow to visualise the dependence between two variables. In this case,* $y = \sqrt{x}$*, with some added noise.*

The most common criterion to guide variable selection is the sensitivity of the covariates. Sensitivity analysis observes the variation in the output of a system and studies to what extent it is explained by changes in each of its inputs. There is a plethora of sensitivity analysis methods [7], but all of them share the aim of finding out the contribution of the inputs to the output variance, thus determining the relative importance of each of them. This helps in understanding the relationship between the variables involved in the process under study as well as their interactions, while providing a measure of the influence of the different factors. A simple example of the analysis of the sensitivity of one variable with respect to another is given by scatterplots (see Fig. 1.2).

A variation of the score-based approach that the nomogram follows also exists for non-linear models. The interval coded scoring (ICS) system [8] also assigns points to each of the variables included in the model and then uses the aggregated score to determine the outcome. The method is based on a non-linear support vector machine (SVM) [9], with some differences: Here, the non-linear transformation of the variables is performed using a step function, dividing their range into a number of consecutive intervals, within which the response of the estimated variable is assumed to be constant. Each of these intervals then receives a score depending on the effect they have on the output, hence the name *interval coded score*. At the same time, the sparsity of the model is enhanced by favouring the use of a small number of variables and intervals in order to produce a simpler interpretation of the system.

An example of the use of the ICS system is displayed in Fig. 1.3, where the risk of a non-viable pregnancy is predicted based on a set of six variables. The non-linear transformation of the variables is applied separately to each of them, meaning any interactions

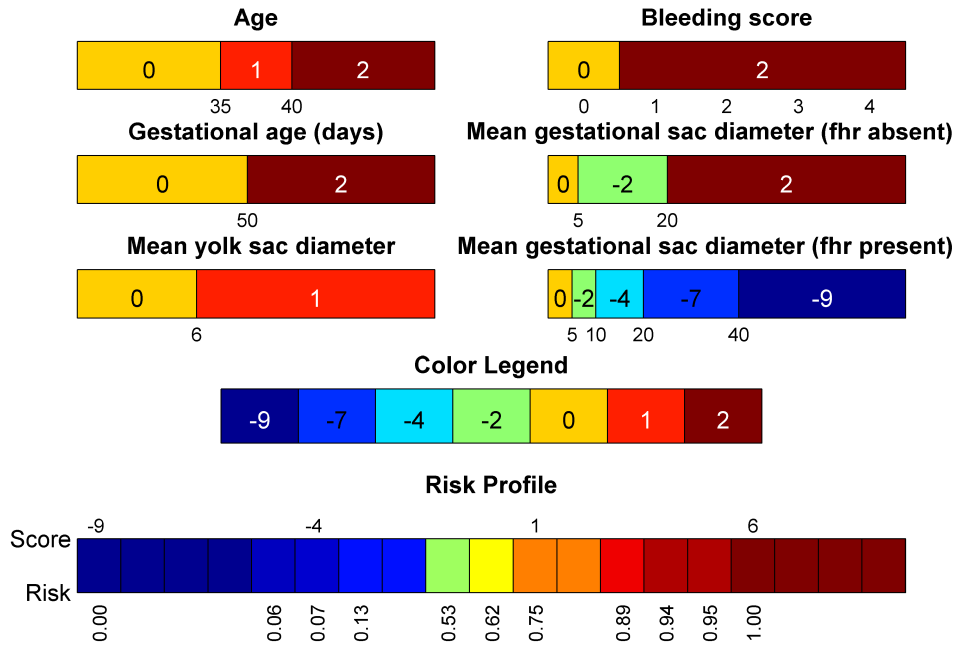


Figure 1.3: *Application of the ICS approach to the prediction of non-viable pregnancies* [8]. *"Picture-based representation by means of colour bars, representing the intervals in which the variable effect is estimated to be constant. For each of the represented bars, the points corresponding to the value of the patient's covariates are obtained. The total score is obtained by adding all points. The colour bar at the bottom represents the predicted risk associated with the final score".*
Image from [8]. *Reprinted with kind permission from the authors.*

between them must be manually built into the model by including additional covariates. In this case, the author has prior knowledge that the diameter of the gestation sac has a different effect for fetuses with or without a visualised heart rate, and takes that into account in the construction of the model. Although the fact that variables are transformed individually simplifies the interpretation of the system, it also represents a limitation, and it would be desirable for the method to automatically detect relevant interactions between the different factors.

This framework inherently offers the interesting possibility of converting the resulting model into a questionnaire, which in this particular example could be used by the clinicians to easily obtain predictions. This idea resembles the approach followed by the techniques in Section 1.2, where the objective is to obtain a set of simple rules that can accurately replace a given statistical model.

1.2 Rule extraction

Rule extraction methods address the problem of interpretability in a different fashion: The starting point is also a black box model, but instead of trying to directly give a meaning to its internal structure, the size of its coefficients, etc., the aim is to produce an equivalent rule-based system that mimics its behaviour but that is significantly simpler and easier to understand. The reason to derive these rules from an intermediate model instead of starting directly from the data is that a model with an appropriate training and a good generalisation ability is usually a better representation of the data than the data itself, because it filters out the noise while retaining the knowledge present in the samples [10].

The interest in rule extraction spans nearly three decades [11]. During that time, many algorithms have been developed in this field (see [12, 10] for comprehensive overviews). The main differences between them have to do with the task they are designed to perform (regression, classification or both), the models they can be applied to (some rule extractors require models with a specific structure) and the type of rules they produce. Regarding the latter aspect, two main formats are used: Boolean and fuzzy rules. Boolean rules are expressions of the type *if-then-else* where the conditions under evaluation are Boolean expressions that involve the inputs to the model. Rules are designed so that they cover the whole of the input space and, in the Boolean case, also so that only one rule can be triggered by a given set of inputs. Fuzzy rules, on the other hand, allow the simultaneous activation of more than one rule by replacing the two-valued Boolean logic by conditions that can be fulfilled partially by means of membership functions.

Fig. 1.4 shows an example of Boolean rules extracted from a simple classifier. The three different sets of rules illustrate the characteristic tradeoff faced by these type of algorithms, that is, the balance between interpretability of the rules and fidelity to the model they approximate. The simplest of the three sets in Fig. 1.4, given by the solid blue line, can be expressed as:

if $(x_1 < 2.2) \wedge (x_2 < 3.9)$ **then**
  $class = *$
else
  $class = \square$
end if

This approximates the original boundary reasonably well, and it is very simple to apply and understand in terms of the values of the variables. The dotted blue line can be expressed as:

if $(x_1 < 2.4) \wedge (x_2 < 2)$ **then**
  $class = *$

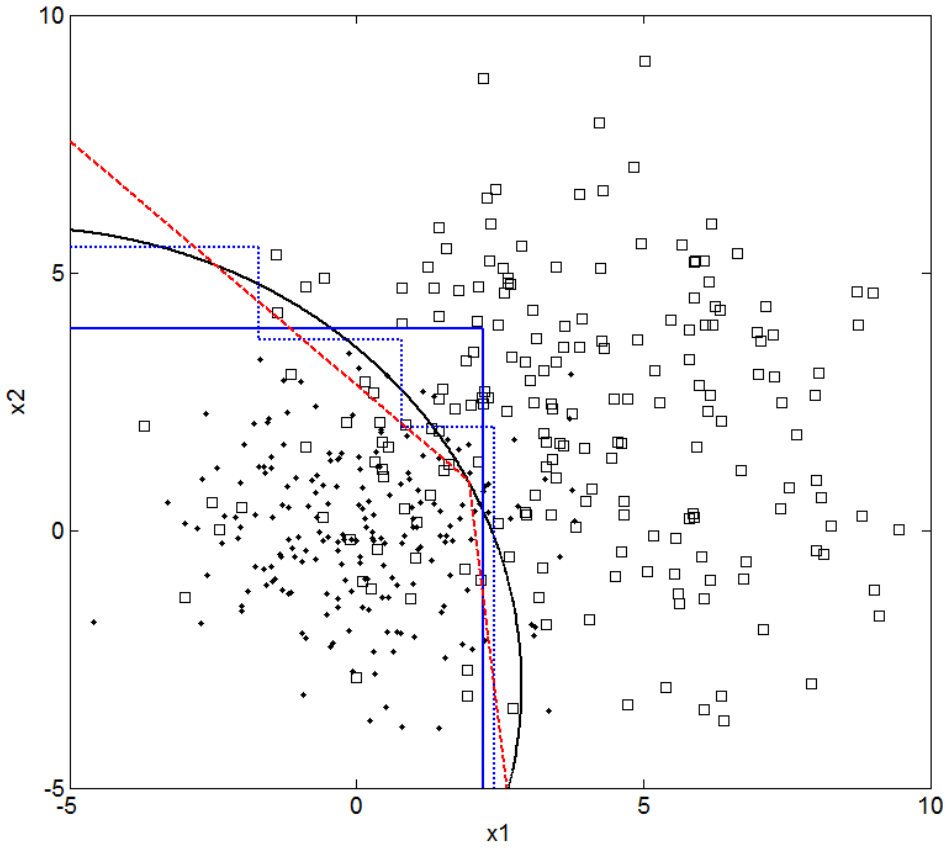


Figure 1.4: *Example of different sets of rules extracted from a classifier. Each marker type represents one class, both of them Gaussian distributions with* $\boldsymbol{\mu} = [0, 0]$, $\boldsymbol{\Sigma} = 1.5\mathbf{I}$ $(*)$ *and* $\boldsymbol{\mu} = [4, 2]$, $\boldsymbol{\Sigma} = 2.5\mathbf{I}$ $(\square)$. *The black solid line defines the border between the classes, calculated using Bayes' theorem to find out the posterior probabilities,* $p(c_i|\mathbf{x}) = p(\mathbf{x}|c_i)p(c_i)/\sum_j p(\mathbf{x}|c_j)p(c_j)$, *and then finding the discriminant surface* $p(c_1|\mathbf{x}) = p(c_2|\mathbf{x})$, *which is quadratic in* $\mathbf{x}$. *The solid blue line represents a simple approximation to the optimal boundary that can be represented with a single Boolean condition. The dotted blue line approximates the boundary more accurately due to a more complex propositional structure. Finally, the dashed red line also provides a close approximation, but at the cost of less intuitive Boolean expressions.*

else if $(x_1 < 0.8) \wedge (x_2 < 3.7)$ **then**
  $class = *$
else if $(x_1 < -1.7) \wedge (x_2 < 5.5)$ **then**
  $class = *$
else
  $class = \square$
end if

This set represents a better approximation, but at the expense of using a more complicated structure, going from one to three Boolean conditions. Note that this is only a 2-dimensional problem; the additional complexity of the rule set required to obtain

an improvement in fidelity will grow exponentially with the dimensionality of the space. This stresses the importance of simplicity in the rule structure, especially when the number of variables in the model is large. The last set, represented by the dashed red line, can also be translated into basic Boolean expressions as:

if $(x_1 < 1.9) \wedge (x_1 + x_2 < 2.8)$ **then**
  $class = *$
else if $(8.7x_1 + x_2 < 17.8)$ **then**
  $class = *$
else
  $class = \square$
end if

In this case, the price to pay for an improvement in fidelity with respect to the original model is the evaluation of more complex conditions, which now involve a linear combination of the two variables. The consequence is the deterioration of the interpretability of the rules. The error that the approximation makes can be reduced as much as desired, regardless of the complexity of the model, by increasing the number of conditions and by including non-linear effects and variable interactions. However, this would defeat the purpose of rule extraction, as the resulting rules would be as obscure as the original model itself, hence why the aforementioned tradeoff between interpretability and fidelity is crucial for the usefulness of these methods.

Classification and regression trees (CART) algorithms [11], also known as decision trees, are a very popular example of binary rule extractors. Trees can be derived directly from labelled data or, from a rule extraction perspective, using the predictions of a model trained on the data to benefit from its generalisation ability. Both classification and regression trees work under the same principle: iteratively split the data into subsets of smaller size. Each data subset corresponds to a node in the tree, starting from the root node on top, which contains the whole data space, and going down towards the leaves. The splitting process continues until a node is determined to be terminal (i.e., a leaf node), meaning that branch of the tree is not divided any further, and data points that fall into this leaf will be assigned a specific class (classification) or $\tilde{y}$ value (regression). Fig. 1.5 shows an example of a classification tree that replicates the Boolean rule set represented by the dotted blue line in Fig. 1.4.

The CART construction process is determined by three factors [11]:

1. Selection of the splitting criterion: The splits that take place in the nodes of the tree are designed so that the data subsets in the level below are *purer* than the set they come from. It therefore becomes a requirement to have a way of measuring the *purity* of the subset of data contained in a node, e.g., a measure of the type

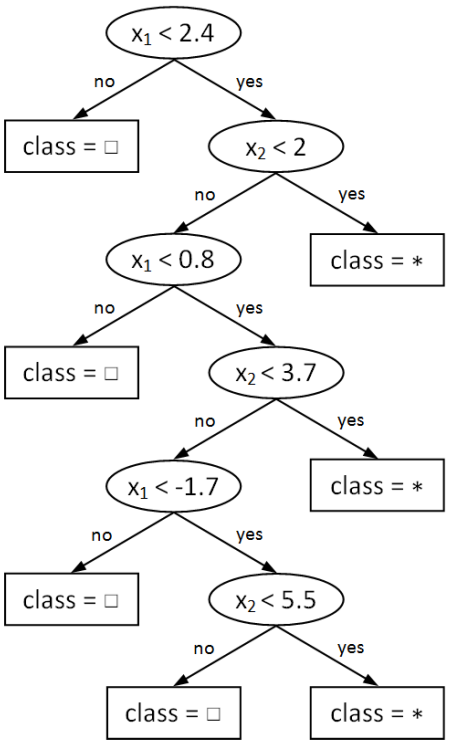


Figure 1.5: *The classification tree shown here represents a set of rules equivalent to the dotted blue line in Fig. 1.4. Each of the round nodes splits the data in two groups according to the result of evaluating the corresponding Boolean condition. To classify a data point, start from the root node (the topmost node in the figure) and follow the path given by the binary splits until a terminal node is reached, then assign the class indicated inside.*

$I = -\sum_i p(c_i) log(p(c_i))$ for classification, where $p(c_i)$ is the proportion of points from class i; or $I = (1/L)\sum_l^L (y_l - \bar{y})^2$ for regression, where y_l are the output values of the points within the node and $\bar{y}$ is their average [10]. The goodness of the split is then defined as the difference in the impurity measure, $G = I(t) - (I(t_1) + I(t_2))$, where t_1 and t_2 are the nodes resulting from splitting node t.

2. Deciding when a node is declared terminal: In principle, the splitting process can continue until each node of the tree contains only one training sample. However, this would lead to a large and overfitted model, so the growth of the tree must be controlled in some way. A simple option is to define a threshold on the goodness of each split, so that if the division of a subset does not provide a large enough improvement, it is rejected. More generally, pruning methods take a fully grown tree and recursively merge its leaves until some optimal point is reached. This is what *minimal cost complexity pruning* [11] does in CART by defining an objective function that involves both the prediction error and the number of nodes of the tree.

3. Assignment of predictions to terminal nodes. Finally, when the set of terminal

nodes is defined, each of them must be assigned a class or a $\tilde{y}$ value prediction. In the case of classification, a majority vote can be taken to decide the class, while a regression tree can simply use the average of the y value of the training points that fall into the node.

Although it is the most common approach, not all rule extraction methods produce rules that are mutually exclusive. In [13], for instance, a decision surface is approximated using as many rules as there are training data points available, where each rule is the largest multivariate hypercube that, while centered in a data point, stays on one side of the class boundary without crossing it. This preliminary set is then refined by eliminating redundant and low-performing rules, with performance measured using the sensitivity and specificity of the binary classification associated with each rule. The fact that rules are defined directly in the data space, rather than sequentially looking at each of the covariates, and not forcing them to be mutually exclusive, means that the set of rules can generally be of low order while retaining a good classification performance, making the interpretability of the rules easier [14].

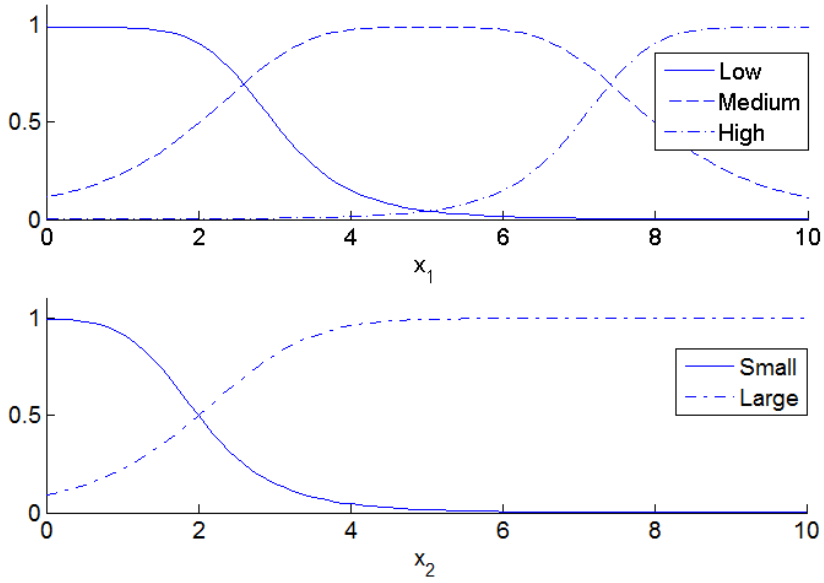


Figure 1.6: *Membership functions for two variables, x_1 and x_2. In this example, there are three fuzzy groups for x_1 and two for x_2.*

Fuzzy inference methods [15] also use rules to approximate the behaviour of models but, in contrast to Boolean rule extractors, they use fuzzy conditional statements to determine the trigger of the rules. Examples of fuzzy rules are:

if (x_1 is HIGH)$\wedge$(x_2 is SMALL) **then** y is LOW

if (x_1 is MEDIUM)$\wedge$(x_2 is LARGE) **then** y is HIGH,

where *LOW*, *MEDIUM* and *HIGH* are fuzzy sets associated to the variable x_1, and

SMALL and *LARGE* to x_2. In fuzzy logic, the result of evaluating a condition like x_1 *is HIGH* is not restricted to a yes/no answer, instead it returns values within a continuous interval (generally from 0 to 1). Given a particular observation of a variable, these continuous values are interpreted as the membership of that sample to each of the sets defined for that variable. Each fuzzy set is therefore characterised by its corresponding membership function. Fig. 1.6 shows examples of fuzzy sets for two variables, x_1 and x_2.

Fuzzy sets are interesting from the point of view of interpretability because they resemble the linguistic labels commonly used to capture human knowledge in imprecise situations (e.g., when someone says *it is cold*). Using this type of logic, fuzzy rules imitate the quantitative decision making process typical of human reasoning (e.g., *it is cold, so turn up the heating*), which is inherently very intuitive.

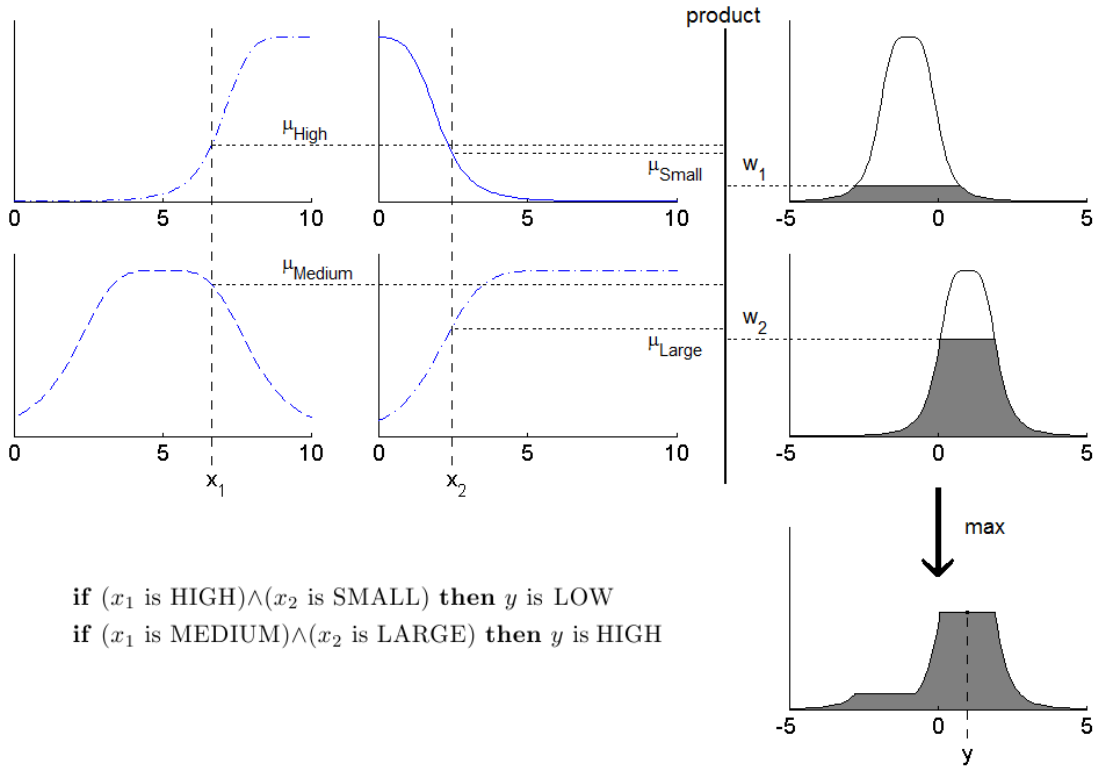


Figure 1.7: *Mamdani fuzzy logic inference system with two rules, two inputs and one output. The memberships μ of the fuzzy sets involved in the premise part of the rules are combined to produce the weights w_1 and w_2, which then determine the contribution of the output functions of the rules. These weighted contributions are aggregated to produce a fuzzy output, from which a crisp value of the output variable y is obtained.*

The reasoning process of a method based on fuzzy rules is detailed in [15] and illustrated in Fig. 1.7. The first step is the *fuzzification* of the inputs, where the membership functions are used to determine the degree of membership of the inputs to the fuzzy sets involved in the antecedent of the rules. The membership values are then combined to calculate the firing strength of each rule (this is usually done using the product or

minimum operators, but others may be more suitable depending on the rule structure). The appropriate consequence of each rule is then generated depending on the firing strengths. Depending on whether the outcome of the rules is fuzzy or crisp (a scalar), two types of fuzzy models can be distinguished: Mamdani [16] (pictured in Fig. 1.7) and TSK (Takagi-Sugeno-Kang) [17]. In Mamdani models, the consequent part of rules is a fuzzy membership function, needing an additional step to calculate a crisp output from it (normally referred to as *defuzzification*). In the example, the output y is selected to be the mean of the values for which the function is at its maximum. In the case of TSK models, the consequence of each rule is an analytical function of the inputs, so that each rule directly generates a crisp value which is then combined with the rest, usually through a weighted sum of the form $y = (w_1 y_1 + w_2 y_2)/(w_1 + w_2)$, where $y_i = f(x_1, x_2)$. Although Mamdani models produce more interpretable rule consequences by keeping the concept of *fuzziness* throughout the process, the lack of a time-consuming *defuzzification* step and the higher-precision output make the TSK model the most popular in practical applications.

It arises from the description of fuzzy models that a key step in the design of the system is the definition of the membership functions and the set of fuzzy *if-then* rules. To do so, the designer uses any available information about the target system, which normally consists on their own knowledge about the model or information provided by human experts. In the absence of prior information, membership functions can be obtained by evenly dividing the input space or by performing clustering in the inputs to obtain meaningful partitions of the data (see *input space partitioning* in [15]). Once these elements are defined, the parameters that determine the membership functions (both in the premise and consequent parts of the rules) are tuned to minimise the output error using regression and optimisation techniques. On that note, ANFIS (Adaptive-network-based fuzzy inference system) [18] proposes an automated way to refine those parameters by creating a class of neural networks that are functionally equivalent to fuzzy systems. Under this framework, membership functions are represented by adaptive nodes in a neural network. The parameters of the premise functions are tuned using a back-propagation gradient descent learning rule, while the optimal consequent parameters are identified with the least squares estimate (provided the output function is linear). Applications of this method, as well as of fuzzy systems in general, can be found in areas like classification [19], control [20] and prediction [21].

1.3 Graphical models

The third and last group of models discussed in this introductory section on interpretability are graphical models [22, 23], also known as independence networks. The way these

methods achieve interpretability is completely different from the approaches described in the previous sections. Graphical models are probabilistic models that produce a representation of the dependencies present in a set of random variables as edges in a graph. In particular, Bayesian networks, the most common type of graphical model, factorise the joint distribution of the whole variable set into a product of conditional dependencies, identifying interactions between the variables involved in each of these conditional distributions:

$$p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i|A_i), \tag{1.3}$$

where A_i is the subset of variables upon which the variable x_i depends. The variables in A_i are normally referred to as *parents* of x_i in this conditional dependence relationship, and this is indicated visually in the network with a directed edge from the elements in A_i to x_i. Fig. 1.8 represents a simple Bayesian network example.

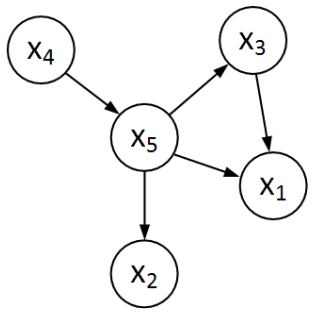


Figure 1.8: *Example of a Bayesian network with five nodes and five edges. In this case, the factorisation of the joint probability* $p(x_1, x_2, x_3, x_4, x_5)$ *is given by* $p(x_1|x_3, x_5)p(x_2|x_5)p(x_3|x_5)p(x_4)p(x_5|x_4)$.

The process of learning the structure of a Bayesian network comprises two phases: First, an undirected network known as the *skeleton* is constructed that contains the edges that should appear in the final graph. Then, the edges are oriented to produce the final model. The presence of an edge between nodes x_i and x_j in the skeleton of the network means that at least one combination A of the remaining variables exist for which the test of conditional independence $I(x_i, x_j|A)$ is false. Computationally, this means the independence test (e.g., likelihood ratio test, conditional mutual information, etc.) must be carried out for all those combinations, so it is important to use an efficient calculation algorithm when the size of the dataset increases. This and other practical considerations are addressed in [24].

Graphical models are useful because they provide a visual insight into the process of interest by identifying the relationships between the different factors present in the dataset. For instance, Fig. 1.9 shows an example of graphical models in a real-world scenario. In this application, the data consisted on the answers to a mental wellbeing survey taken

by 18,500 people in the North West of England in 2009 [25].

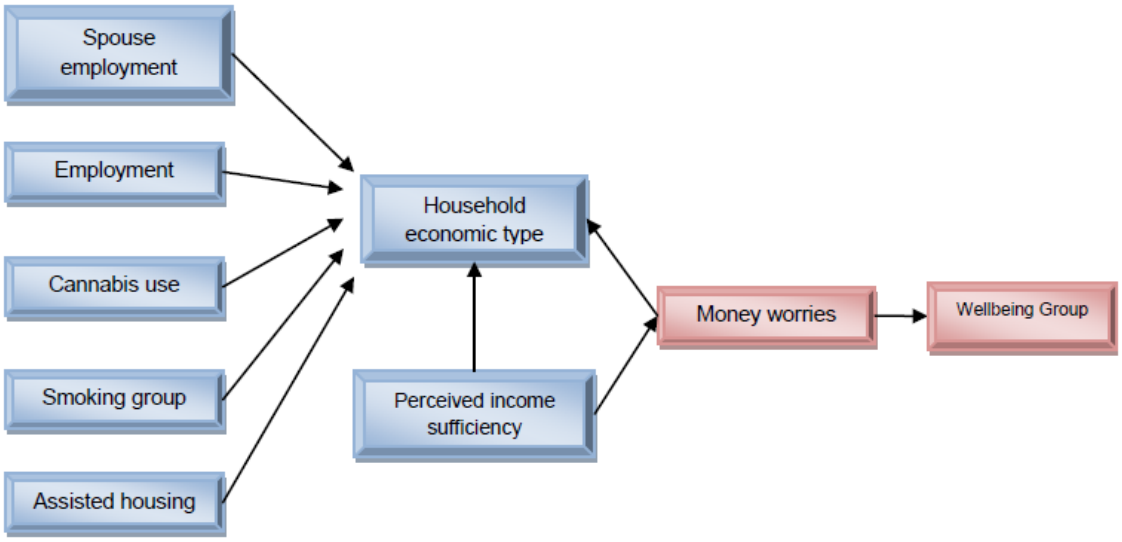


Figure 1.9: *The influence of financial worries on wellbeing, as part of the 2009 North West Mental Wellbeing Survey.*
Image from [25]. *Reprinted with kind permission from the authors.*

The purpose of this questionnaire was to provide a measure of the wellbeing of the population in that region of the country, for which 44 questions were included that covered a series of relevant factors (e.g., employment, area of residence, health, income, etc.). The answers to these questions contained valuable information about the factors that influence wellbeing, so an exploratory analysis was carried out to study the relationships between them.

Four variables were identified as the primary influencing factors: life satisfaction, sedentary time, money worries and the possibility of borrowing a small amount of money in a financial emergency. Fig. 1.9 illustrates the factors that affect one of this primary variables, money worries. This variable measures how often the participants were worried about money in the weeks prior to taking the survey. The graph indicates that financial worries are influenced primarily by the person's perceived income sufficiency and their household economic type (employed, unemployed, retired or in education), with the latter being related to five other factors. A detailed analysis of the different variables and their connections can be found in the original report, this small example only intends to give an idea of how graphical models can be used to extract knowledge from data in an intuitive way. This type of output is useful not only from an explanatory point of view, but also from an interactive perspective. If the interest is to modify a target variable (e.g., improve mental wellbeing), independence networks are a good way to find out which variables should be actuated on to obtain the desired effect.

1.4 Aims and objectives

The aim of this thesis is to develop a novel mathematical framework for semi-supervised visualisation and classification of data that provides a balance between prediction accuracy and interpretability of the output. In order to achieve that, several specific objectives must be met:

1. Accurate estimation of class membership probabilities from a dataset with auxiliary information in the form of true class labels.

2. Derivation of a metric that is informed about the posterior class probabilities using the corresponding estimates. Such metric should consider only those features of the data space that have an impact on the classification problem, combining ideas from feature selection techniques and sensitivity analysis.

3. Definition of similarity measures from pairwise distances calculated using the metric. This determines the structure of a similarity network, where the strength of the connections between pairs of points is quantified by these measures.

4. Analysis of the structure of similarity networks using a community detection algorithm. This will allow the identification of network clusters that reflect the structure of the data in the input space with regard to the underlying posterior probability surfaces.

5. Replication of the classification capability of the original probabilistic model using case-based inference, relating class predictions to known instances in the dataset.

6. Visualisation of similarity networks, making data structure, community membership and class prediction immediately apparent to the user.

7. Illustration of the practical use, potential applications and usefulness of the framework using real-world datasets.

Several interesting benefits arise from the use of this framework:

- By using a distance metric that knows about the relevance of the different areas of the data space, the resulting affinity measure will produce network structures that accurately reflect the similarity between nodes. As a consequence, similar nodes will be grouped together and connected by strong links, resulting in networks with local neighbourhoods that are homogeneous in terms of the classification probabilities.

- A common obstacle found in data mining applications is the difficulty in visualising high-dimensional data directly, obscuring the data structure as well as the

interpretation of the derived models. The use of networks for data representation is a natural way of overcoming this hurdle, making the visualisation of a dataset as simple as displaying the nodes that form the network and the connections between them. Thanks to the use of the FI metric, the underlying structure of the network will reflect the statistical geometry of the original data space as determined by the density function estimates. The representation of the network will therefore provide an informative picture of the data, even in high-dimensional problems.

- Because connection weights contain accurate information about the similarity between data points, the usefulness of the framework goes beyond the visualisation of data, and can also be used as an interpretable retrieval-based classifier. By definition, each node in the network is connected to other similar nodes. This set of adjacent nodes make an ideal cohort from which to extract a prediction based on their posterior probabilities. Furthermore, the result of this classification is directly interpretable as a contribution from the most similar nodes within the network, that is, cases in the dataset. This makes it easy to understand where each class prediction comes from, which is especially useful when the end user is not familiar with how the model estimates the class membership probabilities.

- The derivation of the FI metric is a supervised process that requires labelled data. However, once this process is complete, the metric can be applied to out of sample data. This approach can be very useful in practical applications as a semi-supervised classifier/visualisation tool, using the labelled portion of the data to train the probability estimator and derive the FI metric in order to later build a relational network containing both labelled and unlabelled samples. In a practical application, this would provide the end-user with not only the basic information about class membership prediction of unlabelled cases, but also their location within the network, the network subgroup that they belong to, the set of known training cases that explain their class prediction, etc.

In summary, this thesis is about relevance and interpretability. Relevance because a framework is sought that focuses only on the information that is important for the topic of interest, and interpretability because it ought to provide an intuitive understanding of the structure and patterns present in the data.

1.5 Structure of the thesis

This thesis is divided into five chapters. Chapter 1 discusses the importance of interpretability in data mining models, with examples of how this is achieved in existing

methods. A qualitative description of the expected characteristics of the framework is included.

Chapter 2 reviews the existing work on the two main areas that this framework relates to: metric learning and similarity networks, in particular the identification of network structure. This is followed by a description of the novel contributions of this work, and how they fit into the existing literature.

The methods included in the framework are described individually in Chapter 3, detailing their role within the system. This includes the estimation of the posterior probability surfaces, the derivation of the distance metric, calculation of geodesic distances and the construction and analysis of the networks.

Chapter 4 studies the empirical results obtained from the different experiments in which the framework has been tested, discussing their implications and comparing them with other methods in the literature when appropriate.

Finally, Chapter 5 concludes the thesis by revisiting the initial project objectives and discussing the methodology developed to meet them. To finish off, some interesting possible lines of future work are suggested.

Chapter 2

Literature review

This chapter reviews the most significant published work in the two areas of knowledge that the framework presented in this thesis aims to combine: the derivation of a distance metric that informs about the relevance of the input space in classification scenarios and the analysis of a network based on the pairwise similarities of its elements.

2.1 Distance metric learning

Many widely-used data analysis algorithms involve in their operation measures of similarity between examples given by the distance between them in the input space. Using a distance function to measure similarity between pairs of elements of a space is an intuitive way to understand their relationship in the context of a particular problem domain, but it is crucial to appreciate that the specifics of the chosen metric play a central role in the performance of such methods. That is the case, for instance, in K-nearest neighbours classification, where the definition of the neighbourhood for a given test instance depends heavily on the distance metric, which in turn determines the resulting class prediction [26]. Or in k-means clustering, where points are assigned to clusters depending on the distance that separates them from their centroids. When explicit metrics are used in practice, the Euclidean distance is a common choice because of its simplicity and low computational cost, often overlooking the equal weighting of dimensions that it implies. However, this may be very misleading, especially in high-dimensional applications, where the issue of variable relevance becomes an essential consideration.

It is only recently that this topic has been addressed and work has been directed towards measures of distance that stress those features of the space that have discriminative properties, whether it is according to classification, regression or any other criterion.

In terms of notation, a distance function is a mapping $d : X \times X \to \mathbb{R}$ from pairs of elements in the vector space $\mathbb{R}^N$ to the set of positive real numbers, $\mathbb{R}^+$. Such mapping

is called a metric if, given $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in X$, it satisfies the following four conditions:

1. **non-negativity:** $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$
2. **symmetry:** $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$
3. **triangle inequality:** $d(\mathbf{x}_i, \mathbf{x}_k) \leq d(\mathbf{x}_i, \mathbf{x}_j) + d(\mathbf{x}_j, \mathbf{x}_k)$
4. **discernibility:** $d(\mathbf{x}_i, \mathbf{x}_j) = 0 \Longleftrightarrow \mathbf{x}_i = \mathbf{x}_j$

A metric that does not satisfy the fourth condition is known as a *pseudometric*. Because that is often the case in distance metric learning, this chapter uses the term *metric* for both cases, specifying when appropriate.

2.1.1 Unsupervised metric learning

Unsupervised approaches to distance metric learning use only the geometric properties of the data in the input space (distances, variable correlations, etc.), without any additional source of information. There is a strong connection between unsupervised metric learning algorithms and dimensionality reduction in that they learn a low-dimensional manifold that preserves the geometric relationships between the samples in the original input space under some standard (e.g., Euclidean) metric. Although the goal of these methods is to map data into a lower-dimensional space rather than learning a metric, an immediate consequence of the mapping is that distances between examples in the new space become more meaningful than they were in the original one, which implicitly defines a metric in the original space.

Principal component analysis (PCA)

Principal component analysis (PCA) [27] is probably the best-known data projection algorithm for its simplicity and efficiency. PCA finds the orthogonal linear transformation that best preserves the variance of the input data. The transformed data is expressed as a linear combination of independent variables, known as *principal components*, which are used as coordinates of the new space.

For a given dataset $\mathbf{X}$ of size $N \times L$ (features $\times$ samples, observations as columns), PCA starts by calculating the covariance matrix,

$$\mathbf{S} = \bar{\mathbf{X}}\bar{\mathbf{X}}^T/(L-1), \tag{2.1}$$

where $\bar{\mathbf{X}}$ is the original dataset with rows centered in the origin by subtracting their mean. The principal components of the data are given by the eigendecomposition of the covariance matrix,

$$\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}, \tag{2.2}$$

where $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N)$ is the $N \times N$ matrix that contains the eigenvectors $\mathbf{u}_i$ as columns and $\mathbf{\Lambda}$ is a diagonal matrix with elements $\Lambda_{ii} = \lambda_i$ corresponding to the eigenvalues of $\mathbf{u}_i$. Eigenvalues obtained from a symmetric positive semidefinite matrix, as is the case of $\mathbf{S}$, are always non-negative. The first principal component is the eigenvector with the largest eigenvalue, and corresponds to the direction of the space along which the data spreads most, or equivalently, the line that minimises the sum of squared distances between itself and the data points. The second principal component is the direction of maximum variance if deviations of the data along the first component are suppressed, and so on.

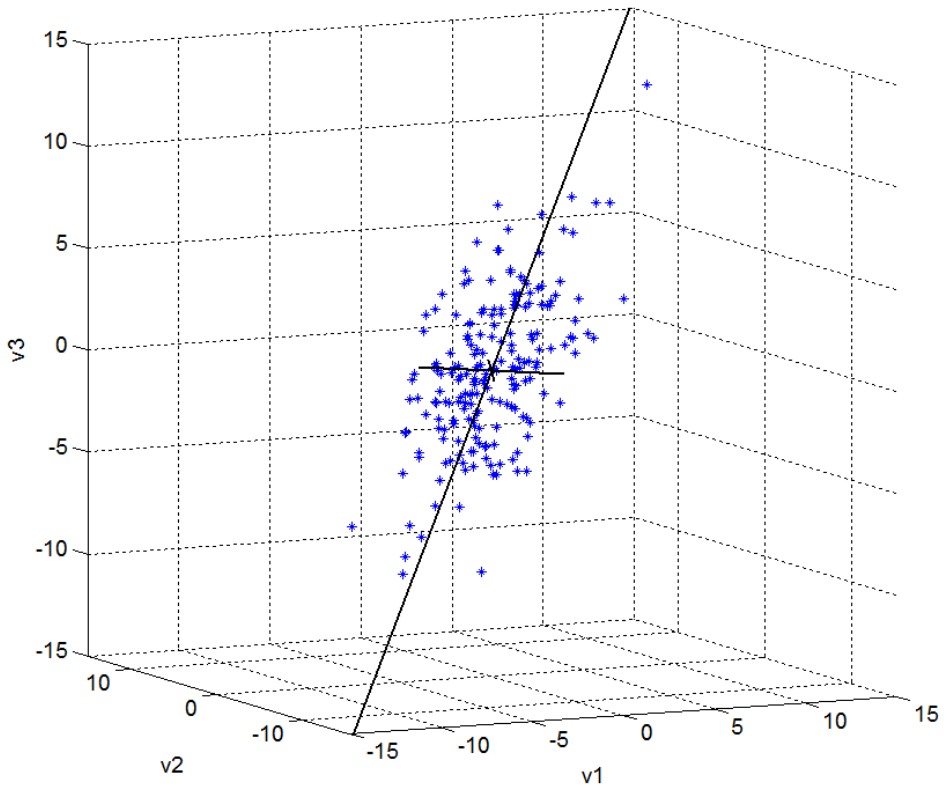


Figure 2.1: *Two-dimensional Gaussian (200 points) with mean* $\boldsymbol{\mu} = [0, 0]$ *and covariance matrix* $\mathbf{\Sigma} = \left(\begin{smallmatrix} 1 & 3 \\ 3 & 1 \end{smallmatrix}\right)$ *embedded in the plane* $-v_1 - v_2 + 2v_3 = 0$ *with noise added to the* v_3 *component. The solid black lines represent the direction of the three eigenvectors, with length proportional to their corresponding eigenvalue.*

Fig. 2.1 shows an example application of PCA to a three-dimensional dataset in which data approximately lie on a two-dimensional manifold (a subspace of the original input

space). The plot shows how the eigenvalues are proportional to the variance of the data along the direction of their associated eigenvectors.

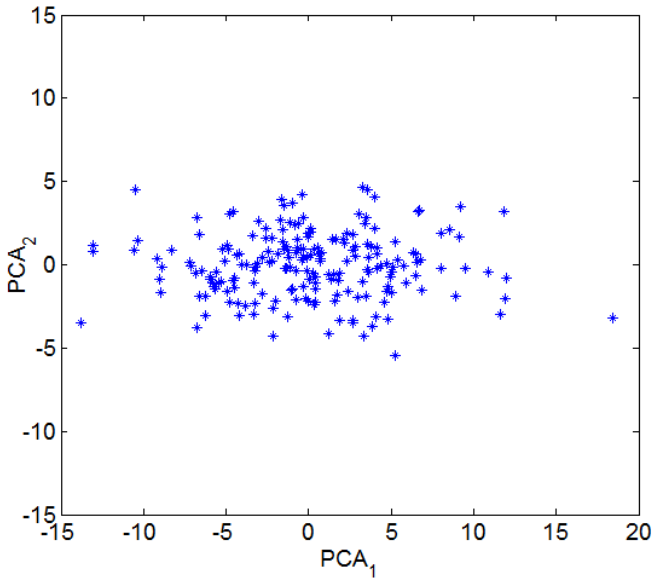


Figure 2.2: *Projection of the original data using the first (PCA_1) and second (PCA_2) principal components.*

After calculating the eigendecomposition in Eq. (2.2), the transformation of the data can be obtained as

$$\mathbf{Y} = \mathbf{U}_M^T \mathbf{X}, \tag{2.3}$$

where $\mathbf{U}_M$ is a matrix that contains the M eigenvectors included in the transformation as columns. In order to preserve the variance of the data, these should be the eigenvectors with the largest eigenvalues. In this particular example, the transformation involved the two principal components, and because the deviations of the data along the third component were small, a mapping into a two-dimensional space was possible without losing much information, as shown in Fig. 2.2.

The metric defined by PCA is obtained by measuring Euclidean distances between transformed points:

$$\begin{aligned} d_{PCA}(\mathbf{x}_i, \mathbf{x}_j) = d_E(\mathbf{U}_M^T \mathbf{x}_i, \mathbf{U}_M^T \mathbf{x}_j) = \|\mathbf{U}_M^T(\mathbf{x}_i - \mathbf{x}_j)\|_2 = \\ = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{U}_M \mathbf{U}_M^T (\mathbf{x}_i - \mathbf{x}_j)}, \end{aligned} \tag{2.4}$$

where $\|\cdot\|_2$ denotes the Euclidean norm from here on. Eq. (2.4) is the expression of the metric associated with any linear transformation of the form described in Eq. (2.3), of which PCA is an example. This type of metric is known in the distance metric literature as a *Mahalanobis* metric, and constitutes a metric when $\mathbf{U}_M$ is full rank and

a pseudometric when it is not, as in the example above.

As noted in [28], the traditional use of the term *Mahalanobis* refers to metrics of the form

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j)}, \tag{2.5}$$

where $\mathbf{S}$ represents the covariance matrix of the data sample. In this chapter, however, as well as in the bulk of the metric learning literature, the term is applied to metrics where $\mathbf{S}$ is any positive semidefinite matrix.

Multidimensional scaling (MDS)

The term *multidimensional scaling* (MDS) [29] applies to a variety of algorithms that, starting from a matrix of pairwise dissimilarities between instances, produce a representation of those entities as points in a space in such a way that the distances between them approximate as closely as possible the dissimilarities between the corresponding instances in the original matrix.

Specifically, classical MDS [30, 31] techniques take an $L \times L$ pairwise distance matrix with elements d_{ij} and find a configuration of L points $\mathbf{x}_i$ in a Euclidean M-dimensional space that minimises some cost function, for instance

$$\varepsilon = \sum_{i}^{L} \sum_{j}^{L} (d_{ij} - d_E(\mathbf{x}_i, \mathbf{x}_j))^2. \tag{2.6}$$

It can be shown that, when the distances d_{ij} are Euclidean, Eq. (2.6) can be rearranged into a matrix expression and solved using eigenvectors, producing identical results as PCA [29]. However, variations of the classical MDS algorithms exist that, using different cost functions and optimisation methods, can overcome the linearity-related limitations of PCA and perform nonlinear data projections (e.g., Sammon mapping [32]). Alternative optimisation techniques used in MDS include gradient descent and iterative majorisation [33].

Isomap

Isomap [34] is a nonlinear dimensionality reduction algorithm that can deal with very nonlinear data structures that MDS methods or PCA would struggle with. To do so, Isomap assumes that the data in the original space lies on a manifold of lower dimensionality, like the two-dimensional Swiss roll in the dataset in Fig. 2.3.

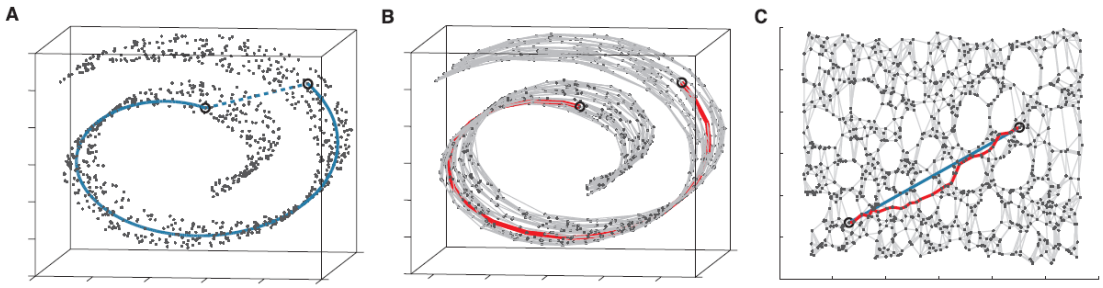


Figure 2.3: *The Swiss roll dataset (1000 points). Similarly to the data in Fig. 2.1, points in this dataset lie on a manifold of lower dimensionality than that of the input space. However, in this case the structure of the manifold is highly nonlinear, and a linear projection method like PCA would fail to learn it. Euclidean distances between points that are distant within the manifold are a poor approximation of their geodesic distance* **(A)**. *Isomap builds a network weighted using Euclidean distances between nearby points* **(B)**, *then approximates geodesics by finding the shortest path between nodes (red line), and learns a low-dimensional embedding that preserves them* **(C)**. *Image from* [34]. *Reprinted with kind permission from AAAS.*

The difficulty in cases like this is that points that are close in the original high-dimensional space may be far from each other within the manifold (see Fig. 2.3(A)), which means that Euclidean distances do not always adequately reflect the intrinsic similarities between examples. Isomap addresses this problem by generating a matrix of pairwise geodesic distances with values that correspond to the shortest path between each pair of points within the manifold (Fig. 2.3(B)), rather than using the straight-line Euclidean distances. A standard MDS algorithm is then applied to this matrix to obtain a low-dimensional representation with coordinates that capture the inherent degrees of freedom of the data (Fig. 2.3(C)).

The key part of the algorithm is the way geodesic distances are approximated. For each point $\mathbf{x}_i$, a set of neighbours is identified as the closest K points according to Euclidean distance $d_E(\mathbf{x}_i, \mathbf{x}_j)$ in the original space. The assumption made is that these K points are also close to $\mathbf{x}_i$ on the manifold. The next step is to define a network where each point is connected to its K neighbours. The connection between points $\mathbf{x}_i$ and $\mathbf{x}_j$ receives a weight $w_{ij} = d_E(\mathbf{x}_i, \mathbf{x}_j)$ if they are neighbours and $w_{ij} = \infty$ otherwise. Fig. 2.3(B) shows the network connections for the Swiss roll example for $K = 7$ as grey edges. Finally, the geodesic distance between points on the manifold is estimated as the length of the shortest path from $\mathbf{x}_i$ to $\mathbf{x}_j$ in the network, $d_G(\mathbf{x}_i, \mathbf{x}_j)$, which can be calculated using graph search algorithms like Dijkstra's or the Floyd-Warshall algorithm. Based on the fact that the local geometry of the data is approximately the same in the input space as it is on the manifold, Isomap uses information about local structure to build a global estimate of the manifold via geodesic distances.

The main limitation of the Isomap algorithm has to do with the neighbour selection. If the size of the neighbourhood is too small, it may result in groups of points becoming

isolated from the rest, dividing the manifold into disconnected pieces. On the other hand, a large size may cause short-circuits between distant points that are close in the input space due to the curvature of the manifold. This would affect not only those two points, but also the region surrounding them, producing a deformation in the learnt embedding. In practice, especially when dealing with noisy data, there may be cases where it is not possible to select a neighbourhood size that prevents both of these effects. The number of data samples available is also an important factor, as it conditions how well $d_G(\mathbf{x}_i, \mathbf{x}_j)$ approximates the true geodesic.

Locally linear embedding (LLE)

The locally linear embedding (LLE) [35] also aims to find a low-dimensional embedding that preserves the data relationships present in the original space. The difference in LLE is that the mapping is determined entirely using local geometry, which means global structure is characterised through the collective analysis of the local neighbourhoods.

The starting point in LLE is a set $\mathbf{X}$ of L points in an N-dimensional space that are assumed to lie in a nonlinear manifold of dimensionality M, $M < N$. The first step in LLE is to express each sample $\mathbf{x}_i$ as a linear combination of its neighbours (again defined as the K closest points under the Euclidean metric). The reconstruction of a point from its neighbours will not be perfect in general, and therefore a cost function can be defined as

$$\varepsilon_{rec}(\mathbf{W}) = \sum_{i}^{L} \|\mathbf{x}_i - \sum_{j}^{L} w_{ij}\mathbf{x}_j\|_2^2, \tag{2.7}$$

where $\hat{\mathbf{x}}_i = \sum_j^L w_{ij}\mathbf{x}_j$ is the reconstruction of $\mathbf{x}_i$ and the weight w_{ij} represents the contribution of $\mathbf{x}_j$ to it. The weight matrix $\mathbf{W}$ characterises the local structure of the L neighbourhoods, and its components are found by minimising Eq. (2.7) subject to two constraints, i) a point $\mathbf{x}_j$ will only be used in the reconstruction (i.e. $w_{ij} \neq 0$) if it is a neighbour of $\mathbf{x}_i$ and ii) the contributions for each sample must add up to one ($\sum_j^L w_{ij} = 1$). An additional constraint may be imposed such that iii) the weights are non-negative ($w_{ij} \geq 0$), which implies that the reconstruction lies in the convex envelope of the neighbour set.

The local relationships contained in $\mathbf{W}$ are invariant to rotations, rescalings and tranlations of the data $\mathbf{X}$. This is important because, going back to the assumption that the data is approximately embedded in a smooth M-dimensional manifold, there is a local linear transformation associated to each neighbourhood that takes the points in it from the original space to the coordinates of the manifold. And precisely because $\mathbf{W}$ is invariant to such transformation, the linear reconstruction of the samples defined by the weights is also valid in the manifold coordinates. Consequently, the last step in the LLE

algorithm is to find a set of points $\mathbf{Y}$ in the manifold for which the relationships in $\mathbf{W}$ are preserved as closely as possible. This is achieved by minimising the embedding cost function

$$\varepsilon_{emb}(\mathbf{Y}) = \sum_{i}^{L} \|\mathbf{y}_i - \sum_{j}^{L} w_{ij}\mathbf{y}_j\|_2^2, \tag{2.8}$$

which, in contrast to Eq. (2.7), is minimised with respect to the location of the points instead of the value of the weights. The complete LLE embedding process is captured in Fig. 2.4.

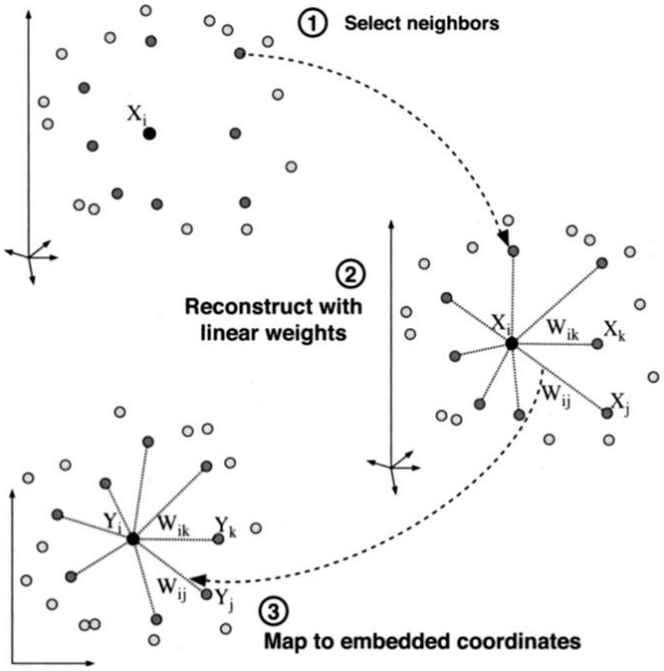


Figure 2.4: *LLE involves three different stages. First, a set of K neighbours is identified for sample $\mathbf{x}_i$. Then, the weights $\mathbf{W}$ are calculated that minimise the linear reconstruction error $\|\mathbf{x}_i - \sum_j^L w_{ij}\mathbf{x}_j\|_2^2$. Finally, the same linear combination is carried over to the manifold, producing the corresponding image $\mathbf{y}_i$.*
Image from [35]. *Reprinted with kind permission from AAAS.*

LLE shares with Isomap the requirement of a uniform and dense enough sampling of the underlying manifold in order to produce an embedding that approximates it accurately. Other limitations of the algorithm include the amount of parameters involved in the optimisations ($L \times (L-1)$ weights in Eq. (2.7) and $L \times M$ coordinates in Eq. (2.8)), which can become a problem in applications with large datasets; and the fact that it does not produce an explicit expression of the mapping that can be applied to new data without having to repeat the process.

2.1.2 Supervised metric learning

As mentioned in the beginning of Section 2.1, distance functions are often used to measure similarity between elements in a dataset. In such context, an ideal distance metric should focus only on those features of the covariate space that are taken into account when it comes to assessing similarity. A pair of examples that differ only by a small amount in an attribute that is known to be critical in some situation should not be considered similar, the same way that a large divergence in an irrelevant property should not be given much importance.

It is therefore obvious that the definition of similarity must be a central factor in the derivation of a distance metric. For instance, if a dataset contains information about the characteristics of different car models, similarity between individuals could be defined with respect to driving style, comfort, maintenance costs, etc., giving rise to different metrics that would measure distances in the space of the car characteristics differently. For that reason, the distance metric used in a particular situation should be tailored to the problem at hand.

In practice, the specification of the meaning of *similar* is materialised in the form of side-information, generally as an auxiliary variable that accompanies the data and helps determine which points are similar and which are not depending on their values. This gives supervised methods the ability to focus on the features that determine similarity and provides them with an informative criterion that the learning process of the metric can optimise for.

Nearest neighbour classification and distance metric learning

A large proportion of the existing research on supervised metric learning is focused on improving the performance of nearest neighbour (NN) classifiers, but the principles involved are also valid for other classification methods. For this reason, this section starts by briefly discussing the influence of the distance metric in K-nearest neighbours (KNN) classification [26]. NN methods [36, 37] provide a nonparametric prediction of a target function $f(\mathbf{x})$ at a given point $\mathbf{x}_0$ in the input space based on known values of the function in the surrounding area:

$$f(\mathbf{x}_0) = \frac{1}{|C(\mathbf{x}_0)|} \sum_{\mathbf{x}_i \in C(\mathbf{x}_0)} f(\mathbf{x}_i), \tag{2.9}$$

where $C(\mathbf{x}_0)$ denotes the set of neighbours of $\mathbf{x}_0$ used for the prediction. The assumption is that $f(\mathbf{x})$ is a smooth function that is approximately constant within the neighbourhood, i.e. $f(\mathbf{x}_0 + d\mathbf{x}) \approx f(\mathbf{x}_0)$ for a small $d\mathbf{x}$.

Let us now consider a classification setting with a set of data points $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_L\}$, $\mathbf{x}_i \in \mathbb{R}^N$, with known class labels $\mathbf{Y} = \{y_1, \cdots, y_L\}$, $y_i \in \{1, \cdots, J\}$. An intuitive choice for the target function in this context is the conditional probability of the classes, $f_j(\mathbf{x}) = p(c_j(\mathbf{x}) = 1|\mathbf{x})$, where $c_j(\mathbf{x}) = 1$ implies that $\mathbf{x}$ belongs to class j. If the probabilities $p(c_j(\mathbf{x}) = 1|\mathbf{x})$ are known, the optimal classification is given by the Bayes' rule, which assigns $\mathbf{x}$ to the class with the highest posterior probability (provided that the cost of misclassification is equal for all classes). These probabilites are, however, rarely known in practice, so the estimation has to come from its samples, the pairs $\{\mathbf{x}_i, y_i\}$. Using the same principle of Eq. (2.9), $p(c_j(\mathbf{x}) = 1|\mathbf{x}_0)$ can be estimated as

$$\hat{p}(c_j|\mathbf{x}_0) = \frac{1}{K} \sum_{\mathbf{x}_i \in C(\mathbf{x}_0)} c_j(\mathbf{x}_i), \tag{2.10}$$

where the compact notation $p(c_j|\mathbf{x})$ replaces $p(c_j(\mathbf{x}) = 1|\mathbf{x})$ and $K = |C(\mathbf{x})|$, the number of neighbours, is used from here on throughout this section.

The accuracy of the predictions given by Eq. (2.10) increases asymptotically,

$$\lim_{L \to \infty} \hat{p}(c_j|\mathbf{x}_0) = p(c_j|\mathbf{x}_0), \tag{2.11}$$

provided that the number of neighbours K increases with the number of samples but represents a vanishing fraction of it [37]. In other words, for a large enough training set, the KNN classification error approximates the minimum value possible, given by the Bayes' rule.

Although Eq. (2.11) represents a desirable convergence property for NN methods, it also carries a crucial implication. The accuracy of KNN classification is based on the K neighbours being close to the query point, and therefore having similar values of the class membership probabilities. This, as seen above, will be true as long as L is sufficiently large, which is to say, as long as the true underlying probabilities are densely sampled. That immediately brings the problem of high-dimensionality into the discussion. An illustrative example of this is given in [26], where the expected diameter of the $K = 1$ neighbourhood (effectively the distance to the closest point) is measured for a uniform distribution of increasing dimensionality. The result is a neighbourhood that increases exponentially with the number of dimensions, breaking the condition that the target function be constant within it. The sample size required to compensate this effect is also exponential and quickly becomes unfeasible in practice.

KNN classification can still perform well in some cases even if the probability estimates are not accurate. This is because the classification of a sample depends only on which class has a maximum $\hat{p}(c_j|\mathbf{x}_0)$, so if this matches the true label, the deviation in the

estimation will not produce an error. This gives the method some level of robustness against the dimensionality issue, but additional action must be taken to further reduce the effect of the problem. The following subsections review some interesting possibilities.

Metric learning using similarity-based optimisation

It has already been discussed how different directions of the space have different degrees of importance. In general, the probabilities $p(c_j|\mathbf{x})$ will not change equally along all directions, and the analysis of these differences can help attenuate the effects of high dimensionality. The objective is to obtain a metric that weights important directions (those where the probabilities change fast) strongly, so that they have more influence over the calculation of distances. Using a metric like that in NN methods elongates neighbourhoods along the less relevant directions and compresses them along the most important ones, resulting in areas with a lesser variation of the class probabilities. This means enforcing the assumption on which KNN relies, compensating the adverse effect of dimensionality by appropriately filtering the influence of each covariate.

The **large margin nearest neighbour (LMNN)** classification algorithm [38] learns a Mahalanobis metric

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{A}^T(\mathbf{x}_i - \mathbf{x}_j)\|_2 = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{A}\mathbf{A}^T(\mathbf{x}_i - \mathbf{x}_j)} \tag{2.12}$$

from pairs $\{\mathbf{x}_i, y_i\}$ that tries to rearrange the data space so that the K nearest neighbours of each point always belong to the same class, while keeping points from other classes further apart. To find a transformation matrix $\mathbf{A}$ that achieves that, the following cost function must be minimised:

$$\begin{aligned} \varepsilon_{LMNN}(\mathbf{A}) &= \sum_{i,j}^{L} \eta_{ij} d(\mathbf{x}_i, \mathbf{x}_j)^2 + \\ +\alpha \sum_{i,j,l}^{L} \eta_{ij}(1 - m_{il})[1 + d(\mathbf{x}_i, \mathbf{x}_j)^2 - d(\mathbf{x}_i, \mathbf{x}_l)^2]_+, \end{aligned} \tag{2.13}$$

where $\eta_{ij} = 1$ if $\mathbf{x}_j \in C(\mathbf{x}_i)$ and 0 otherwise, $m_{il} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_l$ belong to the same class, $[z]_+ = \max(z, 0)$ and α is a positive constant. The first term in the cost function is responsible for pulling neighbours together, while the second pushes points away from neighbourhoods of points with a different label. A contribution to the second term appears when $1 + d(\mathbf{x}_i, \mathbf{x}_j)^2 > d(\mathbf{x}_i, \mathbf{x}_l)^2$, that is, when the distance between a pair of points $\mathbf{x}_i$ and $\mathbf{x}_l$ from different classes is smaller than the distance between $\mathbf{x}_i$ and any of its neighbours. The unit constant defines the minimum width of the margin or minimum

distance from the neighbourhood of $\mathbf{x}_i$ that $\mathbf{x}_l$ must clear in order to avoid a penalty in the cost function. The effect of the metric is illustrated in Fig. 2.5.

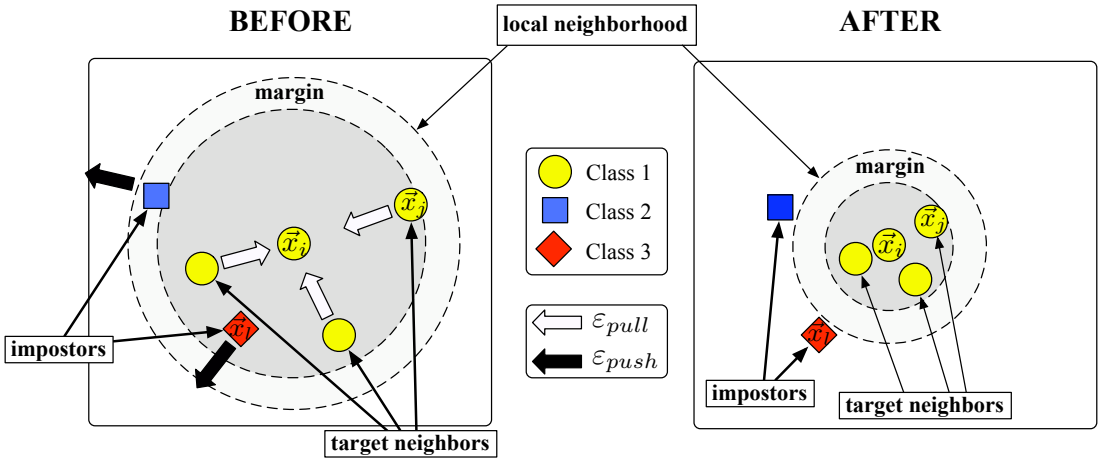


Figure 2.5: *Transformation of the neighbourhood of point* $\mathbf{x}_i$ *when the LMNN algorithm is applied. The minimisation of Eq. (2.13) pulls the neighbours together and pushes the* impostors *away from them, so they are further from* $\mathbf{x}_i$ *than (at least) the distance to its most distant neighbour plus a security margin.*
Image from [28]. *Reprinted with kind permission from the authors.*

The optimisation of the cost function is posed as a semidefinite programming problem, ensuring that the obtained matrix $\mathbf{A}\mathbf{A}^T$ is positive semidefinite. Because this is a convex optimisation problem, a global minimum is guaranteed.

The LMNN algorithm is partly inspired by a previous method, **neighbourhood component analysis (NCA)** [39]. NCA designs a Mahalanobis metric to improve the performance of KNN classifiers by minimising a stochastic variant of the leave-one-out NN misclassification rate. The difference here is that, instead of selecting a fixed number of neighbours, the algorithm assigns fuzzy neighbourhood relationships between the examples: given a pair of points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the training set, the probability that $\mathbf{x}_i$ selects $\mathbf{x}_j$ as its neighbour is

$$p_{ij} = \frac{\exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2)}{\sum_k^L \exp(-d(\mathbf{x}_i, \mathbf{x}_k)^2)}, \tag{2.14}$$

where $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{A}^T(\mathbf{x}_i - \mathbf{x}_j)\|_2$ is the distance function learnt. If $\mathbf{x}_i$ picks $\mathbf{x}_j$ as its neighbour, it automatically receives its class, y_j. In this setting, the probability that $\mathbf{x}_i$ is classified correctly is

$$h_i = \sum_{j|y_j=y_i} p_{ij}, \tag{2.15}$$

determined by the probability of selecting a point from the same class as a neighbour,

which depends on the metric. From Eq. (2.15), the objective function is defined as the expected accuracy rate over all samples,

$$\varepsilon_{NCA}(\mathbf{A}) = \sum_{i}^{L} h_i. \tag{2.16}$$

An important contribution of this method is the removal of the neighbour selection stage. In most NN methods (e.g., LMNN, discussed above) a fixed number of neighbours are selected as the K closest examples to the query point (or, alternatively, those within a certain distance), and then their membership probabilities are averaged. The probabilistic approach in Eq. (2.14) integrates the selection process in the optimisation, giving the algorithm the ability to automatically adjust the size of the neighbourhood considered by modifying the matrix $\mathbf{A}$ accordingly. Effectively, all training points are used as neighbours and contribute to the class assignment, but the contribution of each of them is weighted by p_{ij} so that the closest points are the most influential.

Another consequence of the stochastic selection of neighbours is that the objective function in Eq. (2.16) is continuous and differentiable with respect to $\mathbf{A}$, as opposed to the leave-one-out error rate using standard KNN classification, where a differential change in $\mathbf{A}$ may cause a finite variation of the accuracy. This allows a simple maximisation of the cost function using gradient descent, which would not be possible in the standard formulation. The downsides of this optimisation method are its susceptibility to local minima and the computational cost associated.

The distance metrics generated by LMNN and NCA are optimised so that nearby points that are similar according to the auxiliary information get closer together while surrounding points that are marked as dissimilar are pushed away. An alternative to this is to enforce such displacements on a global basis rather than at a neighbourhood level. That is the goal in [40], to learn a metric (we will refer to it as **global similarity distance metric (GSDM)**) that respects the given similarity information globally. This is reflected in the cost function,

$$\begin{aligned} \varepsilon_{GSDM}(\mathbf{A}) = & \sum_{i,j|y_i=y_j} d(\mathbf{x}_i, \mathbf{x}_j)^2 \\ s.t. & \sum_{i,j|y_i\neq y_j} d(\mathbf{x}_i, \mathbf{x}_j) \geq 1 \end{aligned} \tag{2.17}$$

where $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{A}^T(\mathbf{x}_i - \mathbf{x}_j)\|_2$ is the desired Mahalanobis distance and the constraint on dissimilar pairs is introduced to avoid the trivial solution $\mathbf{A} = 0$. Note that, unlike in Eq. (2.13), here there is no restriction on $\mathbf{x}_i$ and $\mathbf{x}_j$ being neighbours or close to each other, and all pairs are considered. As in LMNN, this optimisation problem is convex,

so a global minimum can be found. The authors propose two ways of minimising it, i) applying the Newton-Raphson method for a diagonal $\mathbf{A}$ and ii) using gradient descent + iterative projections for a full $\mathbf{A}$ (in this case the Newton method is computationally too expensive).

The main limitation of the algorithms discussed (and of Mahalanobis metrics in general) comes from the fact that the matrix $\mathbf{A}$ that defines the metric is constant in the whole input space. That means the weight assigned to each direction is the same everywhere, which is equivalent to assuming that the relative importance of each variable is the same regardless of the location in the space. While that is partially true in problems where the boundaries between classes are linear, the assumption is not valid for complex nonlinear decision boundaries. And even in the simpler linear case, the relevance of the variables in areas close to the boundary is different than in areas far away, where the target functions $p(c_j|\mathbf{x})$ are constant and all variables are equally relevant (or irrelevant) for the calculation of distances. For a metric to be able to capture these relevance changes, it must be able to adapt to the different regions of the space by modifying the weight of each direction appropriately.

Metric learning based on relevance measures

The previous subsection showed how a distance metric can be designed to respect similarity relationships by optimising a cost function that includes explicit instructions (pull similar points together, move different points away). The algorithms in this section take a different path to the derivation of the metric: based on statistical reason, they first define a measure of relevance of the data variability which is then used to determine the parameters of the metric.

For categorised data, a measure with such properties is naturally given by **Fisher's linear discriminant analysis (LDA)** [41]. For the set of examples $\mathbf{X}$ and the corresponding set of class labels $\mathbf{Y}$ defined above, LDA finds the linear transformation that produces the maximum separation between the classes (provided they are normally distributed), which is equivalent to maximising the between- to within-class variance ratio,

$$s = \sum_{i}^{M} \frac{\mathbf{v}_i^T \mathbf{S}_b \mathbf{v}_i}{\mathbf{v}_i^T \mathbf{S}_w \mathbf{v}_i}, \tag{2.18}$$

where the vectors $\mathbf{v}_i$ form the transformation matrix, $\mathbf{V}_M = (\mathbf{v}_1, \cdots, \mathbf{v}_M)$; $\mathbf{S}_b$ and $\mathbf{S}w$

are the between-class and within-class covariance matrices,

$$\begin{aligned} \mathbf{S}_b &= \frac{1}{J}\sum_{j}^{J}(\boldsymbol{\mu}_j - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_j - \bar{\boldsymbol{\mu}})^T \\ \mathbf{S}_w &= \frac{1}{L}\sum_{j}^{J}\sum_{i|y_i=j}(\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T, \end{aligned} \tag{2.19}$$

whose sum gives the overall variance $\mathbf{S} = \mathbf{S}_b + \mathbf{S}_w$; $\boldsymbol{\mu}_j$ is the mean of class j and $\bar{\boldsymbol{\mu}}$ is the mean of the class means. Similarly to the PCA case, the transformation $\mathbf{V}_M$ that minimises Eq. (2.18) is given by the leading eigenvectors of, in this case, $\mathbf{S}_w^{-1}\mathbf{S}_b$. In fact, this method can be seen as a supervised version of PCA where the additional information available is used to differentiate between-class from within-class variance.

Inspired by LDA, the **discriminant adaptive nearest neighbour (DANN)** classification algorithm [42] defines an adaptive metric

$$\begin{aligned} d(\mathbf{x}, \mathbf{x}_0) &= \sqrt{(\mathbf{x} - \mathbf{x}_0)^T\mathbf{A}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)}, \\ \text{with } \mathbf{A}(\mathbf{x}_0) &= \mathbf{S}_w^{-1}(\mathbf{x}_0)\mathbf{S}_b(\mathbf{x}_0)\mathbf{S}_w^{-1}(\mathbf{x}_0), \end{aligned} \tag{2.20}$$

similar to the LDA metric but where the between and within covariance matrices are calculated using only points located within the neighbourhood of $\mathbf{x}_0$. The effect of the metric is that distances along directions of the space where the local class centroids coincide (those parallel to the class boundary) are ignored due to the absence of between-class variance. On the other hand, distances along directions where the class centroids are separated are indicators of relevant variance and are weighted as important by $\mathbf{A}(\mathbf{x}_0)$. The result is a neighbourhood that stretches along the less important directions and shrinks along the relevant ones to avoid including points from other classes.

Because $\mathbf{S}_b(\mathbf{x}_0)$ and $\mathbf{S}_w(\mathbf{x}_0)$, and consequently $\mathbf{A}(\mathbf{x}_0)$, depend on the location of interest and on the selected neighbours, in practice DANN is applied as an iterative algorithm:

1. Initialise $\mathbf{A}(\mathbf{x}_0) = \mathbf{I}$ (Euclidean metric).
2. Select the set of neighbours of $\mathbf{x}_0$ as the closest K points according to Eq. (2.20).
3. Calculate $\mathbf{S}_b(\mathbf{x}_0)$ and $\mathbf{S}_w(\mathbf{x}_0)$ from the set of neighbours (see [42] for details).
4. Calculate $\mathbf{A}(\mathbf{x}_0) = \mathbf{S}_w^{-1/2}\left(\mathbf{S}_w^{-1/2}\mathbf{S}_b\mathbf{S}_w^{-1/2} + \epsilon\mathbf{I}\right)\mathbf{S}_w^{-1/2}$, where the regularisation term $\epsilon\mathbf{I}$ is added so that the neighbourhood does not expand indefinitely in any direction.
5. Repeat steps 2 to 4 until convergence.

6. Use $\mathbf{A}(\mathbf{x}_0)$ it to perform KNN classification on the query point $\mathbf{x}_0$.

A related method, **relevant component analysis (RCA)** [43, 44], calculates a metric based on the same principle for data that, although unlabelled, is divided into groups of points, called *chunklets*, that are known to belong to the same class. RCA treats each of these *chunklets* as a separate class, calculates the associated covariance matrices and obtains a Mahalanobis metric from them. This method is designed for specific unsupervised applications, such as video segmentation or speech recognition, where data naturally arrives as chunks of similar points, and these *partial* labels can be assigned to the data without much effort.

Back to DANN, the authors also provide an interesting connection between the metric in Eq. (2.20) and the *chi-squared* distance,

$$d_\chi(\mathbf{x}, \mathbf{x}_0) = \sum_j^J \frac{[p(c_j|\mathbf{x}) - p(c_j|\mathbf{x}_0)]^2}{p(c_j|\mathbf{x}_0)}, \tag{2.21}$$

provided that the $p(\mathbf{x}|c_j)$ are normally distributed in the neighbourhood of $\mathbf{x}_0$. This distance function is useful from a KNN classification perspective because it separates examples $\mathbf{x}$ from the query point $\mathbf{x}_0$ when they belong to a class j for which $p(c_j|\mathbf{x}_0)$ is small. At the same time, points with similar class probabilities receive small distances, making it easier for them to stay in the neighbourhood.

The **adaptive metric for nearest neighbour (ADAMENN)** classification approach [45] makes use of the local discriminative properties of the chi-square distance in Eq. (2.21) to derive a measure of relevance:

$$r_i(\mathbf{z}) = \sum_j^J \frac{[p(c_j|\mathbf{z}) - \bar{p}(c_j|x_i = z_i)]^2}{\bar{p}(c_j|x_i = z_i)}. \tag{2.22}$$

where x_i is the value of the i-th component of $\mathbf{x}$ and $\bar{p}(c_j|x_i = z_i)$ is the expected value of $p(c_j|\mathbf{x})$ conditional to $x_i = z_i$. Eq. (2.22) measures, at a particular location $\mathbf{z}$ in the input space, how much the probabilities $p(c_j|\mathbf{z})$ are expected to change if we move from that point while keeping the i-th coordinate fixed. If the expected variation (and therefore $r_i(\mathbf{z})$) is small, it means that as long as $x_i = z_i$, the class probabilities will not change much with respect to their value at $\mathbf{x} = \mathbf{z}$, so knowing the value of the i-th feature is very informative to predict $p(c_j|\mathbf{x})$ in the vicinities of $\mathbf{z}$. On the other hand, if the variation is large, then knowing the value of that variable is not decisive in the prediction.

The measure $r_i(\mathbf{z})$ can be generalised to a test point $\mathbf{x}_0$ by averaging its value over the

neighbourhood,

$$\bar{r}_i(\mathbf{x}_0) = \frac{1}{K} \sum_{\mathbf{z} \in C(\mathbf{x}_0)} r_i(\mathbf{z}), \tag{2.23}$$

indicating the local relevance of the i-th coordinate of the data space around $\mathbf{x}_0$. This is then used to define the weights of the metric, described by a diagonal matrix:

$$w_i(\mathbf{x}_0) = \frac{\exp(\beta R_i(\mathbf{x}_0))}{\sum_j^N \exp(\beta R_j(\mathbf{x}_0))} \tag{2.24}$$

$$d(\mathbf{x}, \mathbf{x}_0) = \sqrt{\sum_i^N w_i (x_i - x_{0i})}, \tag{2.25}$$

where $R_i(\mathbf{x}_0) = \max_k(\bar{r}_k(\mathbf{x}_0)) - \bar{r}_i(\mathbf{x}_0)$ inverts the range of $\bar{r}_i(\mathbf{x}_0)$ so that large values correspond to relevant features. The metric weights w_i are then calculated as an exponential function of those values. The parameter β controls the effect of the metric, and can go from 0 (equal weights, Euclidean distance) towards larger values, exponentially increasing the effect of the dominant feature (the one with the largest $R_i(\mathbf{x}_0)$ score) over the rest. This also introduces a regularisation effect like the term $\epsilon\mathbf{I}$ in DANN, preventing zero-weighted directions.

The effect of the metric can be explained in terms of the relevance scores: a large $R_i(\mathbf{x}_0)$ implies that if the value of the i-th covariate is kept close to x_{0i}, the class probability will be similar to that of $\mathbf{x}_0$. This is encouraged with a large w_i that makes displacements from x_{0i} expensive. However, if $R_i(\mathbf{x}_0)$ is small the points with x_i close to x_{0i} are less likely to be similar to x_0 in terms of class-membership, and therefore moving along that direction is favoured by a small w_i.

As in DANN, the metric obtained by ADAMENN also depends on the neighbours of $\mathbf{x}_0$ selected, so it can also be applied iteratively following the six steps of DANN (calculating the relevance measures $\bar{r}_i(\mathbf{x}_0)$ in step 3 and the weights $w_i(\mathbf{x}_0)$ in step 4). In practice, only the labels of the samples are known, so $p(c_j|\mathbf{z})$ and $\bar{p}(c_j|x_i = z_i)$ must be estimated. This is done by considering the class prevalences in the neighbourhood of $\mathbf{z}$ (see [45] for details). Also note that, because the scores are calculated for each covariate independently, the method is limited to individual effects and it has no way of measuring the relevance of variable interactions.

Other ways of defining the local relevance of the inputs have been studied. For instance, the **local flexible metric based on support vector machnes (LFM-SVM)** algorithm [46] uses the class boundary learnt by an SVM to determine the importance of the variables. An SVM [47] is a supervised classifier that learns a nonlinear discriminant function $g(\mathbf{x})$ from binary labelled data and then uses it to classify unknown examples

according to $\text{sign}(g(\mathbf{x}_0))$. The idea in LFM-SVM is to use the class boundary $g(\mathbf{x}) = 0$ to define the direction of maximum relevance. For a point $\mathbf{b}$ on the decision boundary, the direction that best separates the two classes is given by $\mathbf{n}(\mathbf{b}) = \nabla_{\mathbf{x}} g(\mathbf{x})|_{\mathbf{x}=\mathbf{b}}$, the gradient of the function at $\mathbf{b}$. In areas close to the boundary, $\mathbf{n}(\mathbf{b})$ gives the direction perpendicular to the decision surface, and therefore the one where the class probability changes most. The local importance of any given direction, e.g., the i-th coordinate of the space $\mathbf{u}_i$, at $\mathbf{b}$ can be measured by comparing it to the gradient, and the immediate way to do it is using the dot product:

$$R_i(\mathbf{b}) = \mathbf{n}(\mathbf{b}) \cdot \mathbf{u}_i = \mathbf{n}(\mathbf{b})^T \mathbf{u}_i. \tag{2.26}$$

These relevance scores can then be used to weight the displacement along the different coordinates and construct a distance metric using the expressions in Eq. (2.24) and Eq. (2.25).

In general, the query point $\mathbf{x}_0$ may not lie on the boundary, in which case $g(\mathbf{x}_0) \neq 0$. Because it is at the boundary that the gradient is considered most informative, the algorithm estimates the closest point to $\mathbf{x}_0$ on the boundary, $\mathbf{b}_0 = \arg\min_{\mathbf{b}} d(\mathbf{b}, \mathbf{x}_0)$, and calculates the gradient there.

The effect of the metric should be emphasised in the boundary regions, where the class probabilities are less homogeneous and change differently depending on the direction of movement, therefore needing supervision more than in flatter areas of the surface. With that idea in mind, LFM-SVM determines the value of the β parameter in Eq. (2.24) based on the distance from the query point to the boundary. Specifically, $\beta = \max(\bar{d} - d(\mathbf{b}_0, \mathbf{x}_0), 0)$, where $\bar{d}$ is a constant, for example the average distance between the training points and the boundary. That way, points near the boundary receive a large β and vice versa, automatically adjusting the metric to the local characteristics of the region of the query point. This is an improvement over ADAMENN, where this parameter is set manually. Another advantage is the fact that the calculation of the relevance scores is not based on a set of neighbours, which removes yet another parameter and makes the algorithm non-iterative. On the downside, LFM-SVM obviously requires training an SVM and, in multiclass problems, adapting the model from binary classification, which is not as simple as in NN methods.

Before the ADAMENN algorithm was proposed, the idea of a relevance measure based on the predictive power of an input had been used in the **Machete** [26], a KNN classification algorithm in which the neighbourhood of the query point $\mathbf{x}_0$ is determined by recursively partitioning the input space. The Machete implements a CART algorithm (see Section 1.2) that iteratively divides the input space into subregions increasingly homogeneous in terms of class membership. The process follows these steps:

1. Initialise the neighbourhood of the query point as the whole dataset, $C_n(\mathbf{x}_0) = \mathbf{X}$.

2. Calculate the relevance of the coordinates of the input space at $\mathbf{x}_0$, $R_i(\mathbf{x}_0)$, using the points in $C_n(\mathbf{x}_0)$.

3. Partition $C_n(\mathbf{x}_0)$ into two regions along the most relevant covariate, $i' = \arg\max_i R_i(\mathbf{x}_0)$. The resulting subregion $C_{n+1}(\mathbf{x}_0)$ is centered in $\mathbf{x}_0$ and includes the $K_{n+1} = \alpha K_n$ closest points to $\mathbf{x}_0$ along the i'-th direction, where K_n in the number of points contained in the previous partition $C_n(\mathbf{x}_0)$, and $0 < \alpha < 1$.

4. Assign $C_n(\mathbf{x}_0) \leftarrow C_{n+1}(\mathbf{x}_0)$, $K_n \leftarrow K_{n+1}$ and repeat steps 2 and 3. Do this until K_{n+1} has the desired value.

5. Take $C(\mathbf{x}_0) = C_{n+1}(\mathbf{x}_0)$ as the neighbourhood of $\mathbf{x}_0$ and perform KNN classification (Eq. (2.10)).

In each iteration, the algorithm partitions the current neighbourhood along the direction that presents more variation of the class probabilities, keeping the most similar points to $\mathbf{x}_0$ in that respect. When the process ends, the resulting $C(\mathbf{x}_0)$ is an axis-oriented hyperrectangle centered in $\mathbf{x}_0$ with edges of length $l_i(\mathbf{x}_0)$. This is equivalent to applying standard KNN classification using a diagonal metric with components $w_i = 1/l_i(\mathbf{x}_0)$ and a ∞-norm. The result is, again, a smaller neighbourhood along directions of rapid variation of $p(c_j|\mathbf{x})$ and longer along those where the probability changes more slowly, producing an area with a smooth class probability surface.

The parameter α controls the number of iterations, and represents a tradeoff between the number of features considered and the computational cost. A small value will require fewer iterations to obtain the final neighbourhood, but because of the finite sample size L, there may be relevant variables that do not get involved. Similarly, a large number will consider all relevant directions, but at a higher computational cost.

To make sure all relevant variables affect the final neighbourhood, the **Scythe** [26] changes the way the regions are partitioned in step 3 of the Machete algorithm so that every variable affects the shape of the new subregion proportionally to their relevance. Each time step 3 is run, $C_{n+1}(\mathbf{x}_0)$ is obtained as a hyperrectangle that expands, centered in $\mathbf{x}_0$, along every direction i with a speed of growth different for each of them, and inversely proportional to $R_i(\mathbf{x}_0)$. The expansion stops when the number of test points contained within $C_{n+1}(\mathbf{x}_0)$ is equal to K_{n+1}, then the algorithm goes back to step 2 and updates the relevance scores. This ensures that all the local relevances are taken into account, regardless of the value of α.

Like ADAMENN and LFM-SVM, the Machete and the Scythe consider variables individually, so they do not measure the relevance of directions other that the input axis.

A possibility is to manually include in the process features derived from custom combinations of the input variables to capture otherwise ignored interactions, which could be identified for example using linear or quadratic discriminant analysis [26].

Metric learning with kernels

A kernel method is an algorithm that maps data points from the input space to a higher dimensional space, $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, and then applies a linear procedure there. The motivation is that, because the mapping $\boldsymbol{\phi}(\mathbf{x})$ can be arbitrarily complex, a simple linear method (e.g., a linear discriminant) in the feature space can have a very nonlinear effect in the original space. The main advantage of kernel methods is that they operate in the feature space without having to calculate the coordinates of the mapped points explicitly. In fact, not even defining the mapping is required. Instead, only inner products between mapped points are needed, and they are obtained from the kernel function that defines them, $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j)$. This is known as the *kernel trick*, and increases the power of algorithms that depend only on dot products, also making them applicable to inputs of any format, as long as a dot product can be defined between the examples.

Kernel functions represent measures of similarity between pairs of points in the feature space, and can be designed in many different ways: polynomial, $K(\mathbf{x}_i, \mathbf{x}_j) = (a\mathbf{x}_i^T\mathbf{x}_j + b)^c$; Gaussian, $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/2\sigma^2)$; sigmoid, $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(a\mathbf{x}_i^T\mathbf{x}_j + b)$; etc.

As measures of similarity, kernels are closely related to distances. The consequence of the map $\mathbf{x} \in \mathbb{R}^N \rightarrow \boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^F$, $F >> N$, is that the whole input space is embedded into a region $\mathcal{M}$ in the high-dimensional feature space. The geometric structure of $\mathcal{M}$ is completely determined by the mapping function but, as long as $\boldsymbol{\phi}(\mathbf{x})$ satisfies some smoothness assumptions (namely having fixed rank $N_{\boldsymbol{\phi}}$, usually $N_{\boldsymbol{\phi}} = N$, and having all derivatives defined and continuous everywhere in $\mathbb{R}^N$), $\mathcal{M}$ will be an N-dimensional differentiable manifold in the feature space containing the image of $\mathbb{R}^N$ [48].

Another consequence of the mapping is that, by defining this embedding, $\boldsymbol{\phi}(\mathbf{x})$ is also inducing a metric in the input space. This metric corresponds to measuring distances between pairs of images $\boldsymbol{\phi}(\mathbf{x}_i)$ and $\boldsymbol{\phi}(\mathbf{x}_j)$ in the feature space along the surface $\mathcal{M}$:

$$d_{\boldsymbol{\phi}}(\mathbf{x}_i, \mathbf{x}_j) = d_{\mathcal{M}}(\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)). \tag{2.27}$$

Although for particular cases the manifold generated by the mapping function may be flat (a hyperplane), in general $\mathcal{M}$ will be a curved surface. In such case, $\mathcal{M}$ is called a *Riemannian* manifold. Distances in these manifolds are calculated locally using a

Riemannian metric tensor $\mathbf{G}(\mathbf{x})$ [49]:

$$d_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{x}+d\mathbf{x})^2 = d\mathbf{x}^T \mathbf{G}(\mathbf{x}) d\mathbf{x}. \tag{2.28}$$

$\mathbf{G}(\mathbf{x})$ is a symmetric positive definite matrix that determines the metric, and it is possible to express it in terms of the kernel function by considering the length of an infinitely short segment $d\mathbf{z}$ in $\mathcal{M}$. Let us take $\mathbf{z} = \boldsymbol{\phi}(\mathbf{x})$ and $\mathbf{z} + d\mathbf{z} = \boldsymbol{\phi}(\mathbf{x} + d\mathbf{x})$, the mapped segment and its length are

$$\begin{aligned} d\mathbf{z} &= \boldsymbol{\phi}(\mathbf{x}+d\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \boldsymbol{\phi}(\mathbf{x}) d\mathbf{x} \\ d_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{x}+d\mathbf{x})^2 &= \|d\mathbf{z}\|_2^2 = d\mathbf{x}^T \nabla_{\mathbf{x}} \boldsymbol{\phi}(\mathbf{x})^T \nabla_{\mathbf{x}} \boldsymbol{\phi}(\mathbf{x}) d\mathbf{x}, \end{aligned} \tag{2.29}$$

using the first order Taylor approximation of $\boldsymbol{\phi}(\mathbf{x} + d\mathbf{x})$. Finally, using the definition of the kernel function, $K(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y})$,

$$d_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{x}+d\mathbf{x})^2 = d\mathbf{x}^T \, \nabla_{\mathbf{x}} \nabla_{\mathbf{y}} K(\mathbf{x}, \mathbf{y})|_{\mathbf{y}=\mathbf{x}} \, d\mathbf{x}, \tag{2.30}$$

which gives the expression of the Riemannian metric in terms of the kernel function used. To calculate distances between non-adjacent points in $\mathcal{M}$, Eq. (2.28) must be transformed into a path integral:

$$d_{\boldsymbol{\phi}}(\mathbf{x}_i, \mathbf{x}_j) = \left| \int_{t_i}^{t_j} \sqrt{\dot{\mathbf{x}}(t)^T \mathbf{G}(\mathbf{x}(\mathbf{t})) \dot{\mathbf{x}}(t)} dt \right|, \tag{2.31}$$

where $t \in [t_i, t_j]$, $\mathbf{x}(t)$ is the path that goes from $\mathbf{x}_i = \mathbf{x}(t_i)$ to $\mathbf{x}_j = \mathbf{x}(t_j)$ and $\dot{\mathbf{x}}(t) = d/dt \; \mathbf{x}(t)$. The reason why Eq. (2.28) cannot be used for points distant in $\mathcal{M}$ is that the straight line connecting them will, in general, leave the manifold and therefore the result will not represent a distance in the original space. Only when the embedding defined by the kernel function is flat, i.e., when $\mathbf{G}(\mathbf{x})$ is constant, can $d_{\boldsymbol{\phi}}(\mathbf{x}_i, \mathbf{x}_j)$ be calculated directly using the Euclidean metric in the feature space. An example of this situation is the kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{A}\mathbf{A}^T \mathbf{x}_j$. For this kernel, the metric induced in the input space is

$$\begin{aligned} d_{\boldsymbol{\phi}}(\mathbf{x}_i, \mathbf{x}_j)^2 &= d_E(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))^2 = \\ &= \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_i) + \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_j) - 2\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = \\ &= K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j) = \\ &= (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{A}\mathbf{A}^T (\mathbf{x}_i - \mathbf{x}_j), \end{aligned} \tag{2.32}$$

the Mahalanobis distance metric in Eq. (2.12).

The strong relationship between kernel functions and metrics suggests that if a kernel is

selected that is informed about the problem of interest, so will the associated distance metric. In classification tasks, the ideal kernel function is [50]:

$$\mathring{K}(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \text{if } y_i = y_j \\ 0 & \text{if } y_i \neq y_j \end{cases} \tag{2.33}$$

Based on that concept, [51] proposes a method for **kernel idealisation using distance metric learning**. Given a kernel $K(\mathbf{x}_i, \mathbf{x}_j)$, idealising it means making it more similar (according to the *alignment* measure [50]) to the ideal kernel $\mathring{K}(\mathbf{x}_i, \mathbf{x}_j)$, and this can be done easily as follows:

$$\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\gamma}{2}\mathring{K}(\mathbf{x}_i, \mathbf{x}_j), \tag{2.34}$$

where $\gamma > 0$ is a control parameter (see [51] for proof and details). The problem is that, in practice, the analytic expression of $\mathring{K}(\mathbf{x}_i, \mathbf{x}_j)$ will not be available, and only values for training data $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ will be known. To be able to apply the idealisation to test examples, the authors take advantage of the kernel/distance connection and combine Eqs. (2.32), (2.33) and (2.34), obtaining the idealised distance

$$\tilde{K}(\mathbf{x}_i, \mathbf{x}_i) + \tilde{K}(\mathbf{x}_j, \mathbf{x}_j) - 2\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = \tilde{d}_{ij}^2 = \begin{cases} d_{ij}^2 & \text{if } y_i = y_j \\ d_{ij}^2 + \gamma & \text{if } y_i \neq y_j \end{cases}, \tag{2.35}$$

where d_{ij} is the distance measure that corresponds to the original kernel. The algorithm then assumes $\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{A}\mathbf{A}^T \mathbf{x}_j$ and optimises the value of $\mathbf{A}\mathbf{A}^T$ so that the idealised distances are as close as possible to the desired values in Eq. (2.35). Once the optimal value of $\mathbf{A}\mathbf{A}^T$ is determined, the kernel function (and therefore the corresponding metric) can be applied to test and training examples alike.

In [49], the authors take a more general approach to the problem of transforming a kernel to improve classification, in this case for SVM models. The modification of the function pursues the traditional goal discussed above: to increase the weight of directions of significant variation and reduce the rest. The transformation they propose is

$$\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{x}_i)h(\mathbf{x}_j)K(\mathbf{x}_i, \mathbf{x}_j), \tag{2.36}$$

known as a *conformal transformation* of the kernel and where $h(\mathbf{x})$ is a scalar function. This also transforms the metric, $\tilde{\mathbf{G}}(\mathbf{x}) = \nabla_{\mathbf{x}}\nabla_{\mathbf{y}}\tilde{K}(\mathbf{x}, \mathbf{y})\Big|_{\mathbf{y}=\mathbf{x}}$, and for the Gaussian kernel

selected in their experiments the result is:

$$\tilde{\mathbf{G}}(\mathbf{x}) = \nabla_{\mathbf{x}} h(\mathbf{x}) \nabla_{\mathbf{x}} h(\mathbf{x})^T + h(\mathbf{x})^2 \mathbf{G}(\mathbf{x}). \tag{2.37}$$

Now let $h(\mathbf{x}) = \exp(-d(\mathbf{x}, \mathbf{b}(\mathbf{x}))^2 / 2\tau^2)$, where $\mathbf{b}(\mathbf{x})$ is, as in LFM-SVM, the closest point to $\mathbf{x}$ in the decision boundary of the SVM. The result of the transformation is that the weights $\tilde{g}_{ij}(\mathbf{x})$ increase in relative importance with respect to $g_{ij}(\mathbf{x})$ when i) $h(\mathbf{x})$ changes along the corresponding direction and/or ii) $\mathbf{x}$ is close to the boundary, providing the effect intended.

The **Fisher kernel** [52] is another noteworthy example of a kernel function. It was developed with the motivation to combine the power of generative and discriminative models in classification. The idea is for a probabilistic generative model to estimate the distribution of the input data, and use this information to define a mapping function, and by extension a kernel, that will then be introduced into a discriminative classifier. By doing so, the resulting framework takes advantage of the ability of generative models to deal with difficult data types of variable length (speech and text recognition, biosequences, etc.) and the superior classification performance of discriminative approaches, which generally struggle with data formats where it is difficult to define measures of similarity between samples.

The Fisher kernel considers the generative model formed by a set of probability density functions $p(\mathbf{x}|\boldsymbol{\theta})$, parameterised by $\boldsymbol{\theta}$. This model constitutes a manifold in the space of all the density functions $p(\mathbf{x})$, which it is characterised by the following Riemannian metric [53],

$$\begin{aligned} d(\boldsymbol{\theta}, \boldsymbol{\theta} + d\boldsymbol{\theta})^2 &= d\boldsymbol{\theta}^T \mathbf{G}(\boldsymbol{\theta}) d\boldsymbol{\theta}, \\ \text{with } \mathbf{G}(\boldsymbol{\theta}) &= E_{\mathbf{x}}(\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}|\boldsymbol{\theta})^T), \end{aligned} \tag{2.38}$$

where $E_{\mathbf{x}}$ denotes the expectation with respect to $p(\mathbf{x}|\boldsymbol{\theta})$, and $\mathbf{G}(\boldsymbol{\theta})$ is known as the Fisher information [54]. The distance between two points $\boldsymbol{\theta}$ and $\boldsymbol{\theta} + d\boldsymbol{\theta}$ in this metric corresponds to the distance between the two corresponding densities $p(\mathbf{x}|\boldsymbol{\theta})$ and $p(\mathbf{x}|\boldsymbol{\theta} + d\boldsymbol{\theta})$ along the manifold, and measures how different they are in terms of the expected variation in the log-likelihood of $\mathbf{x}$. The consequence of the metric being Riemannian and not Euclidean is that the expected variation of $\log p(\mathbf{x})$ caused by a distortion in $\boldsymbol{\theta}$ is different depending on the location of the space in which it is measured.

In the problem of learning the underlying distribution of $\mathbf{x}$, it is interesting to find the direction in the parameter space that moves $\boldsymbol{\theta}$ closer to the optimal parameter vector $\mathring{\boldsymbol{\theta}}$ for a small displacement $d\boldsymbol{\theta}$. If the optimal vector $\mathring{\boldsymbol{\theta}}$ is taken as the one that gives a

maximum value of the log-likelihood, this direction is given by the natural gradient [55],

$$\mathbf{n}(\boldsymbol{\theta}, \mathbf{x}) = \mathbf{G}(\boldsymbol{\theta})^{-1} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}|\boldsymbol{\theta}), \tag{2.39}$$

which gives the direction of steepest ascent of the log-likelihood at a point $p(\mathbf{x}|\boldsymbol{\theta})$ within the manifold.

The Fisher kernel is based on the idea that, given a realisation $p(\mathbf{x}|\boldsymbol{\theta})$ of the generative model, two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ that are similar will have similar values of the gradient, and a kernel function is defined accordingly:

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{n}(\boldsymbol{\theta}, \mathbf{x}_i)^T \mathbf{G}(\boldsymbol{\theta}) \mathbf{n}(\boldsymbol{\theta}, \mathbf{x}_j) = \\ = \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_i|\boldsymbol{\theta})^T \mathbf{G}(\boldsymbol{\theta})^{-1} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_j|\boldsymbol{\theta}). \end{aligned} \tag{2.40}$$

A large value of the inner product of the gradients implies similarity from the point of view of the underlying generative model in the sense that both examples require a similar adaptation of the parameters to improve the fit. To apply the kernel in practice, the training data is used to obtain an estimated density function $p(\mathbf{x}|\hat{\boldsymbol{\theta}})$, which enables the calculation of pairwise similarities $K(\mathbf{x}_i, \mathbf{x}_j)$. The kernel can then be applied to test points for classification. For example, with an SVM, the prediction for a new point $\mathbf{x}_0$ would be

$$y_0 = \text{sign}\left(\sum_i^L \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_0) \right), \tag{2.41}$$

where α_i are parameters that determine the influence of each training point and, in this case, $y_i = \{-1, +1\}$ represents a binary classification.

Fisher information metric in the input space

It has been discussed above how a family of probability densities $p(\mathbf{x}|\boldsymbol{\theta})$ defines a manifold where each point represents the function for a particular $\boldsymbol{\theta}$. Distances between functions within the manifold are calculated using the Fisher information (FI) metric in Eq. (2.38) and provide a measure of the local relevance of small variations of $\boldsymbol{\theta}$. The metric is informed about the generative properties of the data, and it is thus able to assess the importance of directions in the space of the parameters. Furthermore, there exists a connection between the FI metric and the Kullback-Leibler (KL) divergence [56]:

$$\begin{aligned} I_{KL}(p(\mathbf{x}|\boldsymbol{\theta}), p(\mathbf{x}|\boldsymbol{\theta} + d\boldsymbol{\theta})) = d\boldsymbol{\theta}^T \mathbf{G}(\boldsymbol{\theta}) d\boldsymbol{\theta}, \\ \text{where } I_{KL}(p(\mathbf{x}|\boldsymbol{\theta}), p(\mathbf{x}|\boldsymbol{\theta} + d\boldsymbol{\theta})) = -\int \log \left(\frac{p(\mathbf{x}|\boldsymbol{\theta} + d\boldsymbol{\theta})}{p(\mathbf{x}|\boldsymbol{\theta})} \right) p(\mathbf{x}|\boldsymbol{\theta}) d\mathbf{x}, \end{aligned} \tag{2.42}$$

which reinforces the role of the FI metric as a measure of the difference between adjacent probability distributions on the manifold.

In the literature, the majority of the work done involving the FI metric has used the classical formulation in Eq. (2.38), that is, defining the metric in the space of the parameters and using it to measure relevance of displacements there [55, 49, 57, 52, 56, 54]. However, it is possible to exchange the roles of $\mathbf{x}$ and $\boldsymbol{\theta}$ so that the metric is defined in the input space, now measuring the effect that moving $\mathbf{x}$ has on the probability of some parameter. This is the idea proposed in [58, 59], where the authors change $p(\mathbf{x}|\boldsymbol{\theta})$ by $p(y|\mathbf{x})$ in the definition of the metric:

$$\begin{aligned} d(\mathbf{x}, \mathbf{x}+d\mathbf{x})^2 &= d\mathbf{x}^T \mathbf{G}(\mathbf{x}) d\mathbf{x}, \\ \text{with } \mathbf{G}(\mathbf{x}) &= E_y \left(\nabla_{\mathbf{x}} \log p(y|\mathbf{x}) \nabla_{\mathbf{x}} \log p(y|\mathbf{x})^T \right). \end{aligned} \tag{2.43}$$

In this new setting, the probability function $p(y|\mathbf{x})$ represents the discrete probability distribution, or probability mass function, of the class $y \in \{1, \cdots, J\}$ conditional on $\mathbf{x}$. The expectation E_y is now over $p(y|\mathbf{x})$, thus giving the FI matrix as a summation over the classes:

$$\mathbf{G}(\mathbf{x}) = \sum_{j=1}^{J} \nabla_{\mathbf{x}} \log p(c_j|\mathbf{x}) \nabla_{\mathbf{x}} \log p(c_j|\mathbf{x})^T p(c_j|\mathbf{x}). \tag{2.44}$$

The interpretation of the metric in the input space is analogous to the traditional case. The distance between two points $\mathbf{x}$ and $\mathbf{x} + d\mathbf{x}$ now reflects the variation that the displacement $d\mathbf{x}$ from one to the other produces on the class probabilities. When that translation does not change the posteriors, the distance given by the metric will be zero, indicating that, locally, movement along $d\mathbf{x}$ is irrelevant in terms of the auxiliary information. Similarly, large variations produce longer distances and imply relevance. This extends to global distances through the solution of Eq. (2.31), where the shortest distance is the one calculated along the geodesic path between $\mathbf{x}_i$ and $\mathbf{x}_j$. To approximate the geodesic, the authors [60] adopt a graphical approach similar to that suggested in the Isomap algorithm, discussed earlier in this chapter.

2.2 Analysis of community structure in networks

A network is a representation of a set of elements and the pairwise relations between them using nodes and edges (in this document, as well as in most of the related literature, the terms network/graph, community/group/cluster, edge/link/connection and node/vertex are considered equivalent). This type of representation arises naturally in fields where connections between pairs of elements are clearly defined, as is the case with information

networks (e.g., the Internet) and social networks [61]. In those two examples, links are determined by data transfers and social interaction, and it is simple to identify them and build them into a network by placing edges between the corresponding nodes.

In data mining scenarios, the subject of interest of this thesis, the elements that form the dataset are normally independent from each other and it is not straightforward to establish connections between them. For example, in a medical database of patients there is not an immediate way of linking related patients together, as they do not interact between each other like members of a social network do. Nevertheless, network representations can still be used in these situations by focusing on similarity between the examples, understanding the presence of an edge as an indicator of similarity between the nodes that it connects. Moreover, edges may have associated weights that determine the strength of the links so, in this context, networks can accurately reflect different degrees of similarity between pairs of nodes. We will refer to graphs built based on this idea as *similarity networks*.

Networks provide a simple an intuitive visual representation of the arrangement of node connections, which in turn reveals information about the data structure. In practice, edges will not spread homogeneously across the network. Instead, it is common to find areas in the graph with a high concentration of connections and others where the amount of edges is low, suggesting the presence of network substructures. In particular, it is frequent to look for communities within the graph in order to identify groups of nodes that have similar characteristics or that play similar roles in some underlying process. Fig. 2.6 illustrates this with a simple example.

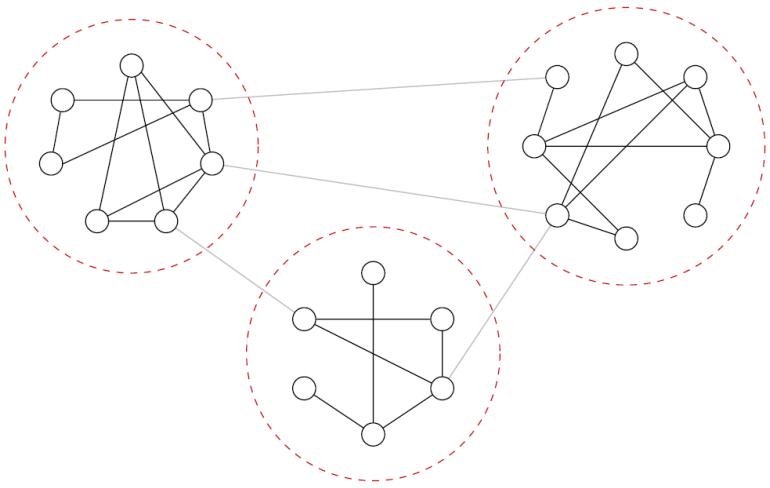


Figure 2.6: *Example of a graph with a strong community structure, as reflected by the large number of connections within each of the three groups compared to the amount of edges between them.*
Image from [62]. *Reprinted with kind permission from Springer Science and Business Media.*

Although the concept of a network community is not clearly defined and each detection algorithm uses its own criteria, most methods agree on the intuition that a community is formed by a group of nodes densely connected between each other which, at the same time, have few links with vertices from other groups [63, 64, 62].

Due to the wide variety of applicable areas for network analysis, the problem of detecting communities has been studied from various perspectives in different disciplines. Next subsection briefly reviews the most important traditional methods and then introduces modern algorithms and the advances they bring. Here, the focus is on the simplest (and most common) type of network: unipartite, i.e., all vertices are equal and can be connected to any other vertex; and undirected, so there is no orientation associated to the edges.

2.2.1 Traditional community extraction

The graph partitioning problem

The problem of optimal graph partitioning arises in important practical situations such as parallel computation and electronic circuit layout. In the former, processes must be grouped and allocated to the CPU's in a way that minimises the communication required between the different units to avoid slowing down the global execution. In the latter, the position of the electronic components in a circuit board is normally arranged so that the total length of the conducting tracks is minimised, reducing fabrication costs and the intensity of parasitic effects. These two scenarios can be expressed in network terms by allowing nodes to represent processes and components, and edges to symbolise process interactions and electrical connections. Based on this situation, graph partition algorithms try to find the division of a network into two groups that produces the minimum number of edges between them. The process can be repeated to obtain further subdivisions; this is known as *iterative bisection*. There are two main bisection methods: the Kernighan-Lin algorithm and spectral partitioning.

The objective of the **Kernighan-Lin algorithm** [65] is to divide a given graph $\mathcal{G}$ of size $2L$ vertices into two disjoint partitions $\mathcal{A}$ and $\mathcal{B}$ of size L. The connections existing in $\mathcal{G}$ are contained in the adjacency matrix $\mathbf{A}$. Elements of $\mathbf{A}$ satisfy $A_{ij} \geq 0$, $A_{ij} = A_{ji}$ and $A_{ii} = 0$, and represent the weight of the connection between a pair of nodes $x_i, x_j \in \mathcal{G}$, where a value of 0 implies no connection and a large value corresponds to a strong link. Edge weights can be thought of as the number of edges connecting a pair of nodes given by a real number instead of an integer; unweighted networks are a particular case where $\mathbf{A}$ has only binary values. The algorithm defines the ideal partitions as those that minimise

the number of edges that go from $\mathcal{A}$ to $\mathcal{B}$:

$$\mathcal{A}, \mathcal{B} = \arg \min_{\mathcal{A},\mathcal{B}} \sum_{x_i \in \mathcal{A}} \sum_{x_j \in \mathcal{B}} A_{ij}. \tag{2.45}$$

A guaranteed way to obtain the optimal solution is to try all possible partitions and select those that produce the minimum cost as given by Eq. (2.45), but that is computationally unfeasible in most applications. The Kernighan-Lin algorithm proposes an heuristic approximation to the true minimum that can be realistically used in practice. The method takes two initial partitions $\mathcal{A}$ and $\mathcal{B}$ and tries to move them closer to the ideal partitions $\mathring{\mathcal{A}}$ and $\mathring{\mathcal{B}}$ by swapping nodes between them. The fundamental idea is that, for any given partitions, two subsets $\mathcal{X} \subset \mathcal{A}$ and $\mathcal{Y} \subset \mathcal{B}$ exist such that swapping them results in the ideal bisection, i.e. $\mathring{\mathcal{A}} = \mathcal{A} - \mathcal{X} + \mathcal{Y}$ and $\mathring{\mathcal{B}} = \mathcal{B} - \mathcal{Y} + \mathcal{X}$. The question addressed by the algorithm is how to approximate these subsets as closely as possible in a reasonable time.

To aid the selection of the nodes that will be exchanged, the external and internal costs of a vertex $x_i \in \mathcal{A}$ are defined as

$$\begin{aligned} E_i &= \sum_{x_j \in \mathcal{B}} A_{ij} \\ I_i &= \sum_{x_j \in \mathcal{A}} A_{ij}, \end{aligned} \tag{2.46}$$

and conversely for a point in $\mathcal{B}$. Further to that, $D_i = E_i - I_i$ is the difference between the external and internal connections of x_i. It is easy to show that swapping a pair of nodes $x_i \in \mathcal{A}$ and $x_j \in \mathcal{B}$ reduces the cost of the partition in Eq. (2.45) by an amount $g = D_i + D_j - 2A_{ij}$, so g represents the cost improvement provided by the swap. The cost decrease associated to a node exchange is used to sequentially determine the swap subsets $\mathcal{X}$ and $\mathcal{Y}$, resulting in the following algorithm:

1. Given some initial partitions $\mathcal{A}$ and $\mathcal{B}$, find all D_i values.

2. Select the pair of nodes $x_{A0} \in \mathcal{A}$ and $x_{B0} \in \mathcal{B}$ whose swap produces the maximal cost decrease, g_0.

3. Assume x_{A0} and x_{B0} have been swapped and calculate the new D_i values, D'_i, so they *know* about the swap. Then select the pair $x_{A1} \in \mathcal{A} - \{x_{A0}\}$ and $x_{B1} \in \mathcal{B} - \{x_{B0}\}$ that produces the maximal cost decrease, g_1. Note that points swapped before are not eligible to swap again. Repeat this process until every point has been exchanged once, and therefore the partitions are $\mathcal{A}$ and $\mathcal{B}$ again.
   This step yields a series of L cost gains g_k that, added together, sum up to 0. The

algorithm then chooses k so that $\sum_1^k g_k$ is maximised. The value of k determines the subsets $\mathcal{X} = \{x_{A0}, \cdots, x_{Ak}\}$ and $\mathcal{Y} = \{x_{B0}, \cdots, x_{Bk}\}$ and the decrease in the partition cost that swapping them produces, $g_0 + \cdots + g_k$.

4. Exchange $\mathcal{X}$ and $\mathcal{Y}$ and return to step 2 with the new partitions. The process continues until a partition is reached with a minimum cost, that is, one for which no swap is possible that improves the split. At this point, the algorithm could be executed again using different initial partitions to see whether a better solution is reached.

The algorithm is an efficient bisection method and it is reported to perform well in practice, but it has some limitations. First, it requires knowing the size of the partitions beforehand; the algorithm can be adapted to produce partitions of different sizes, but they still need to be specified. Note that, for the cost function in Eq. (2.45), a free choice of the sizes of $\mathcal{A}$ and $\mathcal{B}$ would result in the trivial solution of putting all nodes in one partition and none in the other. Also, the method is designed to find optimal bisections, and although the authors suggest ways to divide graphs into more than two partitions, they are based on finding pairwise optimal solutions between the subsets, and not a global one. The dependence of the result on the initialisation of the partitions is also an undesirable feature. The first two points are not an issue in the problem for which the algorithm was devised, but they do represent an obstacle for general community detection, where in general neither the number of partitions nor their size will be known in advance.

Spectral partitioning [66] avoids some of those drawbacks by relying on the spectral properties of the Laplacian matrix to perform the bisection. The Laplacian of a graph is a matrix calculated as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix that contains the degrees of the vertices (the number of edges adjacent to each node).

The Laplacian contains information about many properties of the network. In particular, the eigenvectors and eigenvalues of the matrix can be used to identify communities in the network. Fig. 2.7 illustrates this with an example of a network with a clear structure. Because there are three disconnected communities, the graph Laplacian is block diagonal, each block the same size as the corresponding community. As a consequence, the Laplacian has as many null eigenvalues as there are disconnected groups in the graph, three in this case. By observing which components of those eigenvectors are different than zero, it is simple to identify the node-community memberships.

Fig. 2.7 is obviously a special case where the communities are perfectly defined. Fig. 2.8 is a variation of that example where edges have been added between the communities, producing a connected graph. In this case, there is only a null eigenvalue, which corresponds to the eigenvector of *all ones*; this is now of little use to identify communities

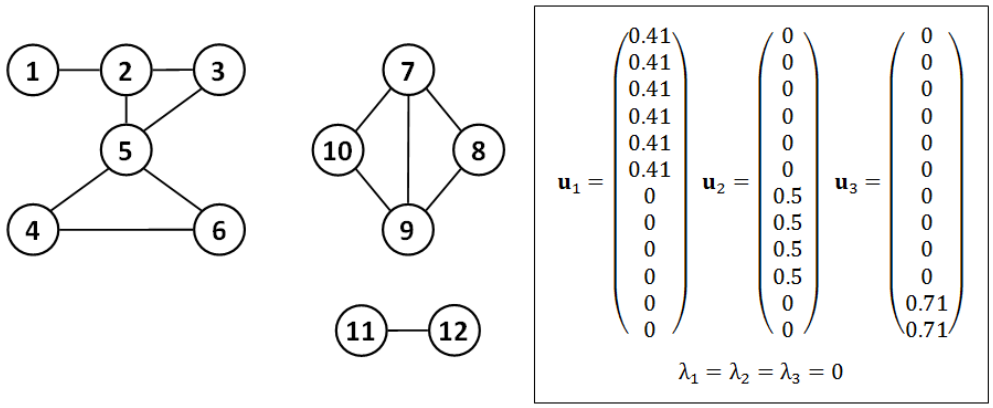


Figure 2.7: *Example network with three separate communities. The decomposition of the Laplacian matrix returns three null eigenvalues, one for each module.*

because it would lead to putting all vertices into the same group. There is, however, useful information in the eigenvector with the second smallest eigenvalue. This vector is known as Fiedler vector in honour of the mathematician who observed that its components can be used to find good bisections of connected graphs [67]. The eigenvalue of the Fiedler vector, λ_2, receives the name of *algebraic connectivity* and gives an idea of the modularity of the network, with small values indicating that a good bisection of the graph (with few edges between the communities) is possible.

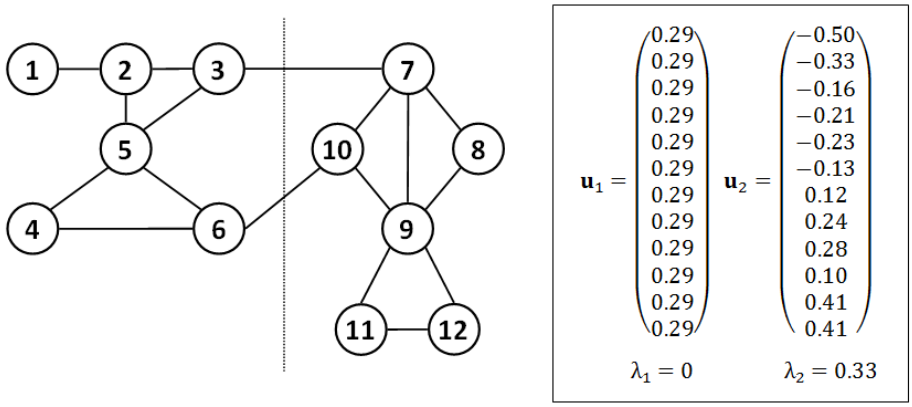


Figure 2.8: *Modified version of the network in Fig. 2.7. In connected graphs there can only be a single null eigenvalue, λ_1, and the algebraic connectivity λ_2 is always greater than zero. Spectral bisection uses the signs of $\mathbf{u}_2$ to split the network. The vertical dotted line indicates the resulting partition.*

The reason (and the way) to use the Fiedler vector $\mathbf{u}_2$ to divide the graph arises from the aforementioned definition of the cost of a bisection as the number of edges running from one partition to the other. If a vector $\mathbf{s}$ is defined such that $s_i = \{+1, -1\}$ determines the partition into which node i falls, the cost of the bisection is given by $\varepsilon = \mathbf{s}^T\mathbf{L}\mathbf{s}$ times an irrelevant constant. The vector $\mathbf{s}$ can be expressed in terms of the eigenvectors of $\mathbf{L}$

as $\mathbf{s} = \sum_i a_i \mathbf{u}_i$, with $a_i = \mathbf{u}_i^T \mathbf{s}$. Using that, the cost of the partition is

$$\varepsilon = \mathbf{s}^T \mathbf{L} \mathbf{s} = \sum_i a_i^2 \mathbf{u}_i^T \mathbf{L} \mathbf{u}_i = \sum_i a_i^2 \lambda_i = \sum_i (\mathbf{u}_i^T \mathbf{s})^2 \lambda_i. \tag{2.47}$$

Minimising Eq. (2.47) is a difficult task, but a good approximation to the minimum can be obtained if λ_2 is small by taking $\mathbf{s}$ parallel to $\mathbf{u}_2$ [64]. This would reduce the cost of the partition to λ_2. Notice that the minimum (zero cost) corresponds to using $\mathbf{u}_1$, but that is not useful, as discussed above. Because the values of the components of $\mathbf{s}$ are restricted, the vector cannot be made exactly parallel to $\mathbf{u}_2$; the best approximation is to set $\mathbf{s}$ according to the signs of the Fiedler vector, $s_i = +1$ if $u_{2i} > 0$ and $s_i = -1$ if $u_{2i} < 0$. Fig. 2.8 shows an example of the application of this method.

Also note that, since the Fiedler vector $\mathbf{u}_2$ is perpendicular to $\mathbf{u}_1$, it must have positive and negative values and will normally produce partitions of balanced sizes. This is a desirable property, as it prevents the method from returning close to trivial partitions, e.g., separating the node with the lowest degree from the rest.

Spectral bisection is a fast algorithm and finds good divisions (provided there is one) [64, 62]. Additionally, it is possible to refine the communities obtained by using them as the initialisation of the Kernighan-Lin algorithm. This not only improves the cost of the split, but also removes the need to specify the size of the clusters, which is automatically determined by the signs of the Fiedler vector. Despite the various alluring features of the method, it still has flaws from the point of view of community detection. Firstly, spectral partitioning is, like the Kernighan-Lin algorithm, a bisection method, and even though it can be applied recursively to obtain more than two partitions, it is not clear which order to follow nor how many communities the algorithm should look for. Secondly, and most importantly, the algorithm will always divide the network, even in cases where no good partitions exist. A community detection algorithm should only divide a graph into subgroups if there is a community structure that justifies doing it, otherwise leaving the network as it is. In that sense, spectral bisection, as well as the Kernighan-Lin algorith, lacks a way of assessing whether a network should be partitioned or not depending on its structure.

Hierarchical clustering

Hierarchical clustering [68] is, along with graph bisection, one of the most popular approaches to community detection, especially when dealing with data that, by nature, is structurally hierarchical. In the context of networks, this is characterised by a structure consisting of a few large communities that divide into smaller nested groups and so on, as is the case, for example, in social networks. To model this type of structure, hierar-

chical clustering takes the set of graph vertices and iteratively merges them, one by one, until the initial L communities (one per vertex) eventually reduces to a single one that contains all nodes. This is known as the *agglomerative* approach. Alternatively, *divisive* algorithms start with all vertices together in one group and repeatedly split it until the L nodes are separated. In both cases, the result is a set of L different solutions that range from one extreme to the other, and which are normally represented using a *dendrogram* like the one in Fig. 2.9.

The key aspect of hierarchical clustering methods is the criterion used to choose which communities should be merged (or split) and in which order. These decisions are determined by a similarity measure that quantifies the affinity between two vertices, which then extends to communities by taking the minimum, maximum, or average value between the nodes involved. In each iteration, the algorithm merges the two most similar communities according to the measure (or, in the divisive case, the community division is carried out that produces the most dissimilar subgroups). Several vertex similarity measures have been suggested in the literature [64, 62], such as *structural similarity*, which looks at how many neighbours the two vertices share; *independent path* counts (two paths are vertex/edge-independent if they have no vertices/edges in common); *random walks*, which measure dissimilarity as the number of steps needed by a random walker to go from one node to the other; etc.

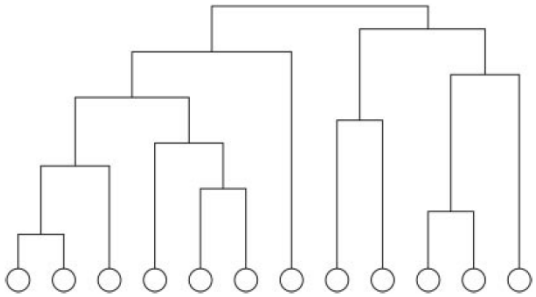


Figure 2.9: *Dendrogram representing an example of hierarchical clustering of a network with 12 nodes. Agglomerative methods start from the bottom of the tree and group nodes together until reaching the top, whereas divisive algorithms start from the top and split the communities until getting to the bottom. A horizontal cut of the tree returns the set of communities into which the graph is divided at the point where the cut is taken.*
Image from [69]. *Reprinted with kind permission from the National Academy of Sciences.*

An advantage of hierarchical clustering methods is that they do not require knowing the number or size of communities in advance, only the pairwise similarities between nodes are needed. Also, the tree representation of the different partitions provided by dendrograms is an intuitive visualisation of the nested community structure of the network. On the downside, these algorithms do not specify which of the different found partitions better represents the actual graph structure, and it is the user who must decide where to cut the

tree. Another characteristic inconvenience of hierarchical clustering is that, although it is good at identifying the central parts of communities, it usually leaves peripheral vertices (i.e., nodes that are connected to a cluster by very few and/or weak edges) unassigned, forming communities on their own [69, 70].

2.2.2 Modern community extraction

Traditional techniques for graph partitioning and clustering have proven useful for detecting communities in specific situations. Inspired by the strengths and limitations of those methods, researchers have more recently worked towards methods that focus on the general case of identifying communities in networks whose structure is unknown. That implies that algorithms must be capable of determining the number and configuration of communities that best describe the internal structure of the network. This subsection discusses the most relevant advances that have appeared in this field in the last decade.

Edge removal algorithms

One of the most popular recent contributions to community detection is the **Girvan-Newman algorithm** [69]. It is based on the principle of divisive hierarchical methods, but it differs from them in a fundamental aspect: instead of eliminating edges between nodes with low similarity, the Girvan-Newman algorithm removes the edges that connect the different modules together. In the classic hierarchical approach the first edges to go are those that connect the most distinct nodes, but that does not necessarily mean they belong to different communities. The Girvan-Newman method focuses on finding the edges that lie between the communities and removes them even if they connect two very similar nodes (according to traditional similarity measures).

It is clear that if such edges are identified and removed, the communities will appear as disconnected groups of vertices. So the question arises of how to define a measure that highlights the edges between the different communities. The authors found the answer in *vertex betweenness*, a measure introduced in [71] to give an idea of the influence of a node within a network. The betweenness of a node is calculated as the number of shortest paths connecting pairs of other vertices that run through it. They then generalised this measure for edges, defining *edge betweenness* as the number of geodesic paths in a network that include that particular edge. The intuition is that, if indeed there are communities in the network under study, they will be, recalling the properties mentioned above, connected with each other by only a few edges. Consequently, shortest paths running between communities will likely include these edges, resulting in a high value of betweenness that increases their chances of being removed from the graph. Other betweenness measures

exist, such as *random-walk betweenness* and *current-flow betweenness* [70], based on the same idea as the *shortest-path betweenness* just explained.

Once the betweenness measure is defined, the algorithm is ready to run. It involves four steps:

1. Calculate the betweenness of every edge in the network.
2. Identify the edge with the highest betweenness and remove it from the graph.
3. Recalculate the betweennesses for the remaining edges.
4. If any edges remain, return to step 2 and repeat.

The recalculation of the betweenness measures in step 3 is the other main improvement over hierarchical clustering that the Girvan-Newman algorithm brings. Instead of calculating all the pairwise measures at the beginning of the process and then using them to define the order of edge removal, now they are updated in each iteration. This is important in situations where there are communities connected to each other by a small number of edges; it usually happens that most of the inter-community shortest paths run through one of those edges only, giving it a high betweenness while the rest receive a low value. When this edge is removed, the shortest paths will have to run through the remaining inter-community edges, increasing their betweenness. The recalculation step updates these values and prevents the potentially wrong removal of other edges that should have taken place at a later stage. Experiments show that this step is crucial for the performance of the algorithm [70].

The introduction of betweenness measures overcame the unwelcome typical behaviour of traditional hierarchical methods discussed in Section 2.2.1, but the Girvan-Newman algorithm was still unable to decide which of the partitions contained in the resulting dendrogram was the best. The solution to that problem came two years later, when the same authors introduced the concept of **modularity** [70], a measure of the quality of the division of a network. To derive this score, the first step is to consider the division of a network into K communities. A matrix $\mathbf{e}$ is then defined whose elements e_{ij} contain the proportion of edges that connect communities i and j in the original network (without removing any edges). With this formulation, an intuitive quality measure of the partition could be defined as the total proportion of intra-community connections, $\mathrm{Tr}(\mathbf{e}) = \sum_i e_{ii}$, since a good division should be characterised by a high value of this sum. However, a partition with a maximum value of this measure could be easily obtained by assigning all nodes to any of the K communities.

To prevent this, the modularity measure takes into account the structure of an analogous network which has the same number of edges as the original, but placed randomly between

the vertices, ignoring the community they belong to. The probability that an edge in the network involves at least one node from community i is $a_i = \sum_j e_{ij}$, the proportion of edges that connect to that community. Therefore, the probability of a randomly located edge to connect communities i and j is $e'_{ij} = a_i a_j$. The modularity of a network partition is defined as the difference between the proportion of intra-community edges in the original network minus the proportion of such edges that one could expect to find in an analogous network with the same node memberships but where edges are placed at random:

$$Q = \sum_{i}^{K} e_{ii} - e'_{ii} = e_{ii} - a_i a_i = \mathrm{Tr}(\mathbf{e}) - \|\mathbf{e}^2\| \tag{2.48}$$

The reason for evaluating the community structure of a network by comparing its edge allocation with that of a network constructed randomly is that in the latter one does not expect to find a group structure, but rather a homogeneous mesh of edges connecting the different communities. Thus, a particular assignment of vertices to communities is considered to reflect an underlying community structure if it results in more intra-community edges than it would be expected by chance. In practice, the Girvan-Newman algorithm calculates the modularity for each of the solutions found, i.e., for each of the possible cuts of the dendrogram, and selects the partition with a highest value. Note that, whenever Q is computed, the original topology of the graph is used. Naturally, one should look for partitions with a positive value of Q. Experiments suggest that a modularity score larger than around 0.3 indicates the presence of significant structure [72]. Although its maximum is 1, values of Q over 0.7 are rare in practice [70].

The main limitation of the Girvan-Newman algorithm is that it is computationally expensive due to the constant recalculation of the betweenness measures. Several modifications of the algorithm have been proposed to make it faster: In [73], the calculation of the edge betweenness was approximated using only the shortest paths between some pairs of vertices chosen randomly, rather than considering all of them. This implies a loss in the accuracy of the measures, but results show that a substantial speed-up can be obtained without much performance deterioration (note that the algorithm only needs to know which edge has the highest betweenness). A different approximation of the betweenness measure is suggested in [74]. Other algorithms simply replace shortest-path betweenness with a different measure. For example, [75] uses the number of short loops, e.g., triangles, that include a particular edge, based on the fact that edges between communities will normally not belong to many such loops, as opposed to edges located deep within a cluster.

Modularity optimisation algorithms

Ever since its introduction, Newman and Girvan's modularity has been the most used quality measure in community detection algorithms [64]. As discussed above, modularity helps evaluate how good the partition of a network is, enabling an objective comparison between different solutions. But that is not the only way it can be beneficial to the process of finding communities. Because modularity is independent of the method used, it is also possible to design algorithms that try to find the best graph partition by explicitly optimising the measure. As in the case of the Kernighan-Lin algorithm, the simplest solution is to calculate the modularity for all possible partitions and select the one that yields the maximal value, but again, that would be too costly. Viable algorithms must therefore approximate the optimal division of the network; some of the most representative approaches are briefly described below.

The first algorithm to directly maximise Q was a **greedy optimisation** method based on agglomerative hierarchical clustering [72]. As such, it starts with each vertex forming a separate community and iteratively merges them until they are all in the same group. In each iteration, the algorithm calculates which merger results in the largest increase in modularity ΔQ and carries it out. The modularity score of the different partitions is stored, and when the fusion process is finished, the division corresponding to the maximal Q value is selected. There are many variations of the algorithm [64], aimed at improving its computational cost and accuracy, such as [76], which speeds up the update of the matrix $\mathbf{e}$; [77], which favors the formation of balanced communities; [78], which studies the benefits of using predefined communities as an initial stage for the agglomerative process; and [79], a version of the algorithm that starts by forming small communities restricted to neighbouring nodes that are then used as *supervertices* of a new network on which the algorithm is applied again, resulting in a very fast recursive method.

If the accuracy of the approximation to the modularity maximum takes priority over execution time, the best results are given by using **simulated annealing** [80]. Simulated annealing [81] is a probabilistic algorithm for global optimisation that performs a stochastic exploration of the state space. The system transitions from one state to another are affected by a noise component that helps avoid local minima. The method allows the exploration of states even when they result in a decrease in modularity, with the hope that they may lead to posterior configurations that are closer to the global minimum. As the algorithm progresses, the probability of accepting transitions that decrease modularity is reduced, and the system eventually converges to a partition. Simulated annealing outperforms all popular community detection algorithms [63], but it does so at the cost of complexity; it is a computationally very demanding method, and can be applied to networks with at most a few thousand vertices.

A less expensive yet competitive alternative consists on the maximisation of modularity

using **extremal optimisation** [82]. This method, reminiscent of the Kernighan-Lin algorithm, starts with an initial division of the network in two random sets of nodes and repeatedly moves the vertex whose *fitness* (a measure of its contribution to the modularity score) is minimal from one community to the other. When no vertex movement is possible that increases Q, the bisection is considered optimal. The process then continues recursively on each of the two resulting communities independently.

Spectral optimisation of the modularity is yet another possibility of addressing the problem, as suggested in [83]. This approach is inspired by the spectral bisection algorithm discussed in Section 2.2.1, but in this case the objective function used to determine the best partition is the modularity. For a particular bisection of a network, the original expression of Q in Eq. (2.48) can be reformulated and expressed in terms of individual node contributions:

$$Q = \frac{1}{4m}\sum_{i}^{L}\sum_{j}^{L}\left(A_{ij} - \frac{k_i k_j}{2m}\right)s_i s_j = \frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s}, \tag{2.49}$$

where k_i is the degree of vertex i, m is the total number of edges in the network (or the sum of all edge weights in weighted networks), and the vector $\mathbf{s}$ contains the node memberships, $s_i \in \{+1, -1\}$. Eq. (2.49) considers, for each pair of vertices, the difference between the weight of the edge connecting them and the expected weight if edges were placed randomly, given by $k_i k_j/2m$. The matrix $\mathbf{B}$ is called *modularity matrix* and, following the reasoning that led to Eq. (2.47), its eigenvectors and eigenvalues can be used to rewrite Q by taking $\mathbf{s} = \sum_i a_i \mathbf{u}_i$:

$$Q = \mathbf{s}^T\mathbf{B}\mathbf{s} = \sum_i a_i^2 \mathbf{u}_i^T\mathbf{B}\mathbf{u}_i = \sum_i a_i^2\lambda_i = \sum_i (\mathbf{u}_i^T\mathbf{s})^2\lambda_i, \tag{2.50}$$

where $\mathbf{u}_i$ and λ_i are the eigenvectors and eigenvalues of $\mathbf{B}$. In this case, the interest lies in maximising the function, so analogously to the approximation taken to minimise Eq. (2.47), $\mathbf{s}$ is chosen as close as possible to the eigenvector with the largest eigenvalue. Again, due to the binary restriction of the components of the vector, the values are assigned according to the signs of the eigenvector.

To put the algorithm into practice, the user simply calculates the matrix $\mathbf{B}$ and obtains its eigendecomposition. Then, the signs of the eigenvector with the most positive eigenvalue are used to bisect the graph. It may happen that the largest eigenvalue is zero, the rest of them being negative (note that, as with the Laplacian, the vector $(1, 1, \cdots, 1)$ is always an eigenvector with null eigenvalue, since the rows and columns of $\mathbf{B}$ add up to 0), which means that no partition exists that results in a positive modularity. In that situation, the elements of the leading eigenvector are all equal, and therefore no partition is performed.

This is a significant advantage over spectral partitioning, where the bisection was always carried out, regardless of whether it produced meaningful communities or not.

The magnitude of the eigenvector components is also informative. Large absolute values in the elements of the leading eigenvector imply large contributions of the corresponding nodes towards the modularity score, and vice versa. Moving those vertices from one community to the other would result in an important deterioration of the modularity, which suggests they are central to the community they belong. Similarly, elements of the eigenvector close to zero correspond to nodes that are not influential on the modularity, meaning they are not so closely related to their community.

Like other bisection methods, this algorithm approaches the problem of identifying multiple communities by iteratively splitting clusters. However, in contrast to other algorithms, it appreciates that divisions following the initial one must not treat subcommunities as independent groups of nodes because they are still connected to vertices outside of them in the original network, and therefore splitting them while ignoring those outbound connections would not be maximising the global modularity. Instead, if a subgroup of size L' is to be divided, an $L' \times L'$ modularity matrix $\mathbf{B}'$ is calculated that is informed about the structure of the rest of the initial network. $\mathbf{B}'$ is then decomposed and its leading eigenvector is used to divide the subgroup. The bisection of this subgroup will produce a change in the global modularity, ΔQ. If this value is positive, it means that partitioning the subcommunity improves the global score, and therefore the split is accepted. If, on the other hand, it is negative, the division worsens the measure and it is rejected, leaving the subcluster as it was. The iterative bisection process stops when no subdivision is possible that improves the global modularity.

2.3 Methodology selection

As discussed in previous sections, the main objective of this work is to develop a framework for the construction and analysis of informative networks that generate interpretable insights of classification datasets, both in terms of visualisation of data structure and prediction of class labels. Accordingly, the skeleton of this methodology comprises two well differentiated parts: the distance metric, responsible for producing pairwise measures of similarity between points in the dataset; and the community detection algorithm, which performs the analysis of the network that those similarities define. After a review of the most representative methodologies in both of those fields, this section specifies which methods are used to achieve the objectives listed in Section 1.4, making way to Section 2.4 for a discussion on how this work contributes to the existing knowledge on which it finds support.

2.3.1 Selection of the distance metric

Probably the most influential factor within the framework, the distance metric is solely responsible for capturing the (dis)similarity between the elements of the database, and it is imperative that it does so in a way that adjusts to the classification task at hand, weighting displacements according to their influence on the classification probability surfaces. It is also desirable that the behaviour of the metric adapts to the characteristics of the surroundings of the location of the input space where it is applied, taking the focus off flat areas and putting it on the fast-changing boundary regions. Several of the reviewed supervised metrics satisfy these properties (DANN, ADAMENN, LFM-SVM, etc.), but only the **Fisher information metric** provides an elegant, clearly defined and statistically rigorous solution. Furthermore, because it depends explicitly on the variation of the class-membership probabilities, it leaves the user the freedom to estimate those surfaces with their method of choice.

Based on those features, we follow a similar approach as that suggested in [58, 59] and opt to derive a variation of their FI metric based on discriminative probability estimators.

2.3.2 Selection of the probability estimation method

The Fisher information metric is determined by the FI matrix in Eq. (2.44), which, in turn, depends on the posterior probabilities $p(c_j|\mathbf{x})$. In most applications, these functions are not known a priori, so they need to be estimated before the metric can be applied. Probability density estimation can be carried out using any of the various methods found in the machine learning literature, which can be divided in two categories: generative and discriminative.

Generative models

Generative probability estimators approximate the joint distributions $p(\mathbf{x}|c_j)$ of the observed variable set from the available data, resulting in a full model of the data that allows sampling any of the input variables. Once the generative densities have been estimated, Bayes' theorem can be used to obtain conditional probabilities as

$$\hat{p}(c_j|\mathbf{x}) = \frac{\hat{p}(\mathbf{x}|c_j)\hat{p}(c_j)}{\sum_j \hat{p}(\mathbf{x}|c_j)\hat{p}(c_j)}. \tag{2.51}$$

Most generative models assume the data has been generated by a distribution whose parameters are estimated using maximum likelihood. A very simple and common example is to assume that a given variable is normally distributed, estimating the Gaussian

parameters as the mean and variance of the samples available,

Mixture models add an additional level of complexity by assuming that each sample is generated by a weighted contribution of individual probability distributions [84],

$$\hat{p}(\mathbf{x}|c_j) = \sum_i w_{ij} f_i(\mathbf{x}). \tag{2.52}$$

The densities $f_i(\mathbf{x})$ represent different populations within the whole data space. In principle, $f_i(\mathbf{x})$ can be any probability function; common choices are Gaussian and Bernoulli distributions for continuous and discrete data respectively. In this case, an optimisation algorithm like *Expectation Maximisation* is required to calculate the parameters of each function as well as the weights mixing weights w_{ij}.

Discriminative models

Discriminative estimators model the probability distribution of a target variable conditional on the values of the observed covariates. In contrast to generative methods, discriminative models estimate the posterior distributions $p(c_j|\mathbf{x})$ directly, without first modelling the density function in the input space. This is usually all that is needed, since in many applications (e.g., classification and regression) the full probabilistic model of the data is not required.

Nearest-neighbours methods constitute one of the most widely used discriminative estimators. As seen earlier in the chapter, NN techniques follow a non-parametric approach that estimates the value of the target function $p(c_j|\mathbf{x})$ based on its value at nearby locations where it is known (see Eq. (2.10)).

Another representative example of discriminative model is Fisher's linear discriminant, also discussed earlier in the chapter. For the binary case, this model classifies samples according to the sign of $y = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w}$ defines a discriminative hyperplane between the two classes, and whose values are obtained by maximising the between- to within-class variance ratio. However, the projection y does not represent a probability estimate; the correct calibration is given by the family of generalised linear models,

$$y = \Phi(\mathbf{w}^T\mathbf{x} + w_0), \tag{2.53}$$

where $\Phi(\cdot)$ is known as the activation function. A widely used $\Phi(\cdot)$ is the sigmoid function, which gives rise to linear logistic regression, a well calibrated model even when the classes are imbalanced. The main alternative to discriminative linear estimators is the multilayer perceptron (MLP), where a layered structure of artificial neurons with sigmoid outputs

turns the linear approach into a flexible non-linear model capable of fitting any probability surface [85].

Other examples of flexible discriminants include radial basis function (RBF) networks [86], another type of neural network that uses non-linear activation functions based on the distance between the inputs and the functions' centres; and relevance vector machines (RVMs) [87], a variation of the non-probabilistic SVM classifier that generates probabilistic predictions.

For the particular application that concerns this work, only the posterior class-membership probabilities are required, and therefore discriminative estimation methods seem more appropriate given their purpose. Moreover, they do not make any assumptions on the generative distributions of the data, avoiding any practical limitations in that respect. Specifically, sigmoid-output estimators are the methods selected for this framework because of their flexibility, simple implementation and differentiability of the surfaces they produce. Furthermore, by choosing these models, the differential expression of the Fisher information metric can be calculated generally in terms of the sigmoidal formula of the posterior probabilities, $\hat{p}(c_j|\mathbf{x}) = \Phi(a(\mathbf{x}))$, and then adapted to the corresponding linear/non-linear case by specifying the argument $a(\mathbf{x})$.

2.3.3 Selection of the community detection algorithm

Although there is not as much variety in this field as there is in distance metric learning, there are still some interesting differences between the existing methods that can help decide on which to rely. Firstly, it is clear that only algorithms based on modularity should be considered unless there is a good reason to use others, given the acceptance and proven usefulness of the measure. From the different alternatives available, we decided to maximise modularity using **spectral optimisation**, as suggested in [83]. The choice of this method over the rest is justified for the simplicity of its implementation and, especially, for the repeatability of the partitions it finds, which do not depend on a stochastic procedure, as in simulated annealing, or in any kind of initialisation, as in the extremal optimisation algorithm.

There are several other methodologies involved at different stages of the framework; they will be introduced as they appear throughout the next chapter.

2.4 Novel contributions

Fisher network framework for data visualisation and classification

The main contribution of this thesis is the combination of a Fisher information metric and a network community extraction algorithm into a semi-supervised framework for interpretable data visualisation and classification [88]. Successful applications of FI metrics in classification scenarios are not new [59, 60], but to our knowledge such a metric has never been used in the construction of affinity networks before.

FI metric derivation using discriminative models

The derivation of the FI metric is inspired by that of Kaski et al. [58], but it differs from it in one crucial aspect: the estimation of the posterior probabilities. Their approach uses generative models such as Gaussian mixtures or Parzen estimators in conjunction with Bayes' theorem to obtain the posterior class probabilities $p(c_j|\mathbf{x})$. In this work, the probability estimation is carried out by a linear logistic regressor (LLR) or a multilayer perceptron (MLP), both discriminative methods that, as such, avoid density estimation as an intermediate step, resulting in a more direct metric learning process. This work does not intend to contribute to the debate of whether generative of discriminative models are to be preferred in machine learning applications, as there is plenty of literature available on that topic [89, 90]. In this context, our contribution is an original derivation of the metric for LLR and MLP. A connection between the interpretations of the FI metric in the data and parameter spaces is provided in Section 3.3.1 that relates to the traditional relationship between the metric and the KL divergence [56].

Closed-form distance measure for linear models and iterative geodesic finder

For the linear case of the LLR, we show that the path integral in Eq. (2.31) can be solved analytically, resulting in a closed-form expression that enables the calculation of global distances without having to approximate the integral. For the MLP, and non-linear methods in general, the issue of geodesic paths comes into play; we propose an iterative greedy algorithm based on gradient descent optimisation to find shortest paths in the input space, and compare it with the alternatives used in the literature [60].

FI metric and blind signal separation

Before building the first similarity networks [88, 91], studies of the benefits of the FI metric in KNN classification [92] and in a blind signal separation problem are carried out. In the latter, the categorisation of brain tumour spectroscopy data is performed using an original combination of the Fisher metric and non-negative matrix factorisation [93]. This includes prior knowledge extracted from known class labels in an otherwise unsupervised method through the use of the metric, improving the performance of the original algorithm.

Reference case identification and informative backgrounds

When it comes to the representation of networks, the proposed framework automatically arranges the nodes in a 2 or 3-dimensional Euclidean space using MDS techniques so that the distances between them approximate those in the Riemannian space, making the data structure immediately apparent. After the community detection stage, one or more reference cases may be identified for each cluster using centrality measures; this allows a simplified representation where only intra-community edges that connect to a reference node are visible, highlighting the cluster structure. Node classification can also be performed accepting only contributions from reference nodes or using the full network.

Additionally, *informative backgrounds* provide an even richer visualisation of Fisher information networks by colouring the background of network plots according to the predicted values of the class membership probabilities, graphically signaling the different class regions and borders, and so defining a visual *classification map* for enhanced interpretability.

Related work

Going back to the framework as a whole, there is a recent methodology called CaseReasoner [94] that is similar to our work in several respects. CaseReasoner is part of a project to develop a healthcare platform for a grid of clinical centres, and it is aimed at providing clinicians with a flexible way to access and analyse the clinical records of all centres simultaneously. To visualise the data, CaseReasoner uses three different representations: treemaps, heatmaps and similarity networks; it is the latter that is related to our framework, as it also produces affinity networks based on the pairwise similarities of the samples in the database. There are, however, two important differences in the process:

- CaseReasoner often works with categorised data, so it uses similarity measures

that exploit the available auxiliary labels, as advised in the literature. It offers two measures, one calculates similarity as the inverse of the divergence between the posterior class probabilities of the two points of interest, and the other trains an ensemble of classification trees and looks at how many of them put the two points in the same class. Both are non-metric *global* measures, since they only consider the predictions of a pair of examples to determine how similar they are, ignoring their location in the input space and what lies between them there. In effect, this overrides the topology of the data space, and therefore the resulting similarities miss the structural information of the data. In contrast, the FI metric looks at *local* divergences between points and extends them to distant examples through a path integral. This is a geometrically rigorous distance measure that preserves the topology of the input space, producing affinity measures that reflect the data structure, which is later infused into the similarity network. This is important because it helps to distinguish class substructure in multimodal distributions, something impossible to detect using global divergences. A simple example of this effect is discussed in [60].

- To construct the network from the affinity measures, CaseReasoner uses one of three neighbourhood methods: relative neighbourhood, ϵ-neighbourhood or K-nearest neighbours. The first method connects two points only if there is no other point that is more similar to both of them than they are between each other, the second connects points whose similarity is above a certain threshold, and the third connects each point to its K most similar neighbours. These methods generally produce sparse networks, which is good for a tidy representation. However, the placement of vertex connections also affects the clusters found by the community detection algorithm. Our framework builds fully connected networks, so none of the similarity measures are ignored. By doing so, we make sure that the communities found in the subsequent stage are the result of a perfectly informed process. This is likely to be irrelevant for points with a high class membership, as they will probably fall in the same community even if only a few of their strongest connections are considered. For borderline points, though, snapping any potentially informative connections may affect their assignment and make them fall in the wrong cluster.

2.5 Chapter summary

This chapter takes a detailed look at the most relevant methods in the different areas of machine learning that concern this framework. After exploring the various techniques that the distance metric learning literature has to offer, the chosen metric is based on the Fisher information matrix calculated in the input space, as it brings together all the

properties desired for the underlying similarity measure. As for the posterior probability estimates required to derive the metric, these will be calculated using sigmoid functions with linear and non-linear arguments. Regarding network community extraction, the existing research clearly suggests the choice of a modularity-based algorithm, from which a spectral optimisation method is selected.

The combination of these techniques results in a novel similarity network construction methodology. The structure of the generated graphs reflect that of the data in the original space, using only the information that is objectively relevant to the class membership probabilities.

Chapter 3

Methodology description

From a practical point of view, the framework proposed in this thesis takes a dataset with samples divided into classes and generates a network representation that can then be used to gain insight into the data structure and extract class predictions for new unknown points. The production of these *Fisher information networks* entails several internal subtasks; the purpose of this chapter is to describe each of the different steps in the process and connect them together to present the complete methodology of the framework.

The diagram in Fig. 3.1 summarises the steps required to go from a dataset to the corresponding FI network. The first stage consists on estimating the class-membership probabilities $p(c_j|\mathbf{x})$ from the data with the supervision of auxiliary variables in the form of class labels (Section 3.1). The probabilities are then introduced in the definition of the FI matrix (Section 3.2). With this, it is possible to calculate infinitesimal distances directly using the differential metric expression, which can be extended to global distances by approximating geodesic paths and calculating a path integral along them (Section 3.3). Using a Gaussian kernel, pairwise distances are transformed into connectivity weights (Section 3.4), giving rise to the adjacency matrix that determines the structure of the Fisher network. A community detection algorithm then analyses this matrix and extracts the clusters present in the network (Section 3.5). Once the community structure of the network is known, it can be used to visualise the original dataset, find the most relevant nodes in each cluster and perform interpretable classification based on known cases (Section 3.6).

3.1 Probability estimation

Because the Fisher information focuses on the variation of the conditional class probabilities, having a good estimation of the true functions is essential in order for the FI metric

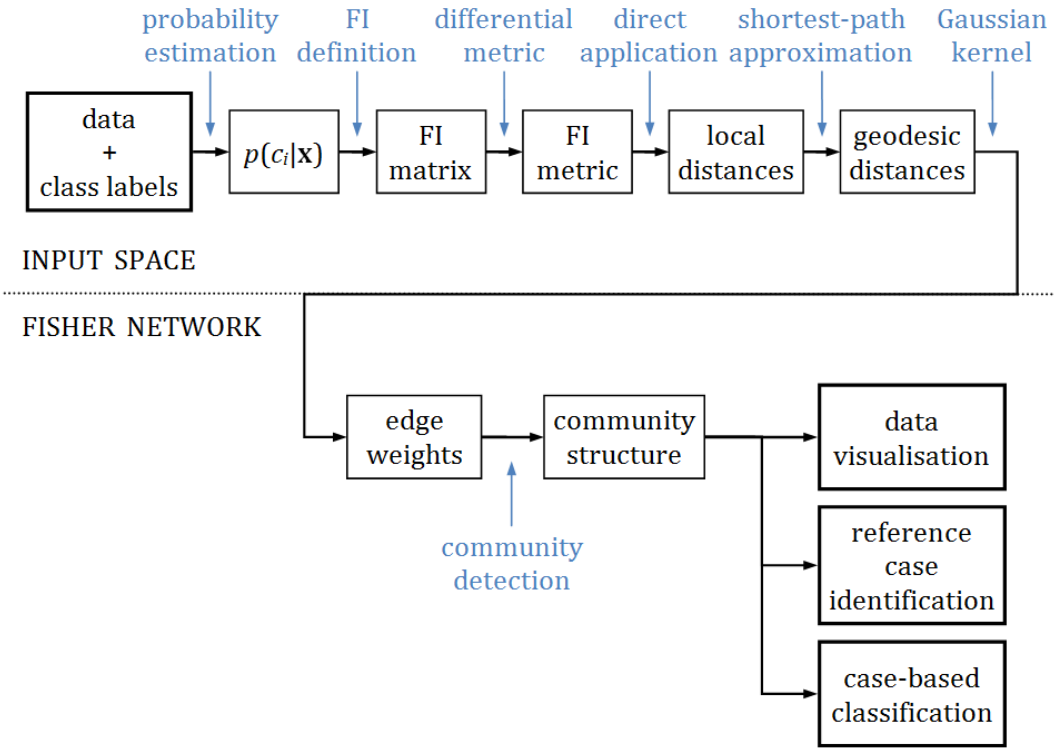


Figure 3.1: *The diagram illustrates the Fisher network constructing procedure. Each section in this chapter describes one of the different steps involved in the process.*

to accurately reflect those changes, which in turn results in a sound similarity measure.

This section describes the methods implemented in the framework to obtain such estimates. For the sake of convenience, the binary and multiclass problems are considered separately, since the former leads to a simpler expression of the FI matrix. Furthermore, binary classification problems are very frequent in practice, so having a specific formulation of the metric is useful.

3.1.1 Binary classification

For the binary case, the probability estimators considered have an output characterised by a sigmoid function,

$$\hat{p}(c_1|\mathbf{x}) = \frac{1}{1+\exp(-a(\mathbf{x}))}, \tag{3.1}$$

where, following the notation used in Section 2.1.2, $\hat{p}(c_1|\mathbf{x}) = \hat{p}(y=1|\mathbf{x})$ is the estimated probability that $\mathbf{x}$ belongs to class 1. Since the probabilities must add up to one, the conditional probability of class 0 can be calculated from Eq. (3.1) as $\hat{p}(c_0|\mathbf{x}) = 1-\hat{p}(c_1|\mathbf{x})$. The scalar function $a(\mathbf{x})$ represents a map of the input space into real values, and it

is the gateway through which the input variable information enters the model. The characteristics of $a(\mathbf{x})$ determine how the different variables are combined to produce the posterior probabilities, thus defining the estimation. The process of training the model consists on adjusting the parameters in the function so the predictions are as close as possible to the true labels. Depending on the complexity of this function, the resulting estimators can be divided in two categories: linear and non-linear.

Linear estimators: Linear logistic regression

Linear estimators use, as their name indicates, a function $a(\mathbf{x})$ that is a linear combination of the input variables,

$$a(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_N x_N = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}, \tag{3.2}$$

which, in combination with the sigmoid function in Eq. (3.1), gives a binary linear logistic regressor (LLR). In this case, $a(\mathbf{x})$ is usually known as *generalised linear estimator*, and its parameters $\boldsymbol{\beta}$ and β_0 can be determined by optimising the sum of squared errors,

$$\varepsilon_{SSE} = \sum_{i=1}^{L} (y_i - \hat{p}(c_1|\mathbf{x}_i))^2. \tag{3.3}$$

The sum of squares error is, however, an objective function better suited for linear regression. A more appropriate function for classification is the log-likelihood,

$$\varepsilon_{LL} = -\sum_{i=1}^{L} (y_i \log(\hat{p}(c_1|\mathbf{x}_i)) + (1 - y_i) \log(1 - \hat{p}(c_1|\mathbf{x}_i)), \tag{3.4}$$

which, when optimised, maximises the likelihood of the estimator making the correct predictions.

Linear probability estimation is simple and efficient, but it is limited by the complexity of the problems it can tackle. Because $a(\mathbf{x})$ is linear in $\mathbf{x}$, the boundary between classes, $\hat{p}(c_1|\mathbf{x}) = \hat{p}(c_0|\mathbf{x}) = 0.5 \Longleftrightarrow a(\mathbf{x}) = 0$, is a hyperplane in the input space, with equation $\boldsymbol{\beta}^T \mathbf{x} = 0$. Such a boundary can only solve linearly separable classification problems; for more complicated cases a non-linear method is required.

Non-linear estimators: Multilayer perceptron

The model chosen to provide the non-linear version of $a(\mathbf{x})$ is a multilayer perceptron (MLP). The MLP is a type of neural network and, as such, its computational power is based on the combined operation of a number of smaller units called *neurons* (see Fig. 3.2). These neurons were introduced in 1943 by McCulloch and Pitts [95] as a simplified representation of brain neurons, with the idea that circuits made of these basic units properly interconnected could implement logical functions of arbitrary complexity. Furthermore, these units can *learn* by adapting the weights of their synaptic connections, so they can be used for regression and classification through the appropriate modification of those parameters.

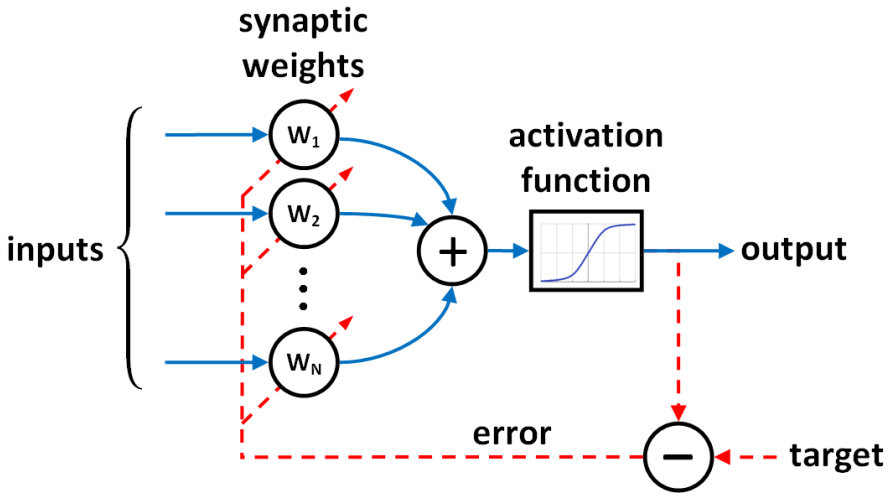


Figure 3.2: *Representation of the McCulloch-Pitts neuron. When the weighted sum of the inputs is positive, the neuron is fired and the output is active. Negative weights correspond to inhibitory inputs, whereas positive ones are associated with excitatory signals. One of the inputs is normally a constant that determines the value of the activation threshold, also known as bias.*

Fig. 3.3 shows the structure of the MLP implemented in the framework for binary classification problems, which contains two layers. First, a *hidden layer* with S neurons takes the components of $\mathbf{x}$ as inputs and generates as many non-linear intermediate signals. These are then combined in the *output layer* by another neuron, resulting in a non-linear expression of $a(\mathbf{x})$,

$$a(\mathbf{x}) = \mathbf{W}^O \Phi(\mathbf{W}^H \mathbf{x} + \mathbf{B}^H) + B^O, \tag{3.5}$$

where $\mathbf{x}$ is the $N \times 1$ input vector, $\mathbf{W}^H$ is the $S \times N$ hidden layer weight matrix, $\mathbf{B}^H$ is the $S \times 1$ hidden layer bias weight vector, $\mathbf{W}^O$ is the $1 \times S$ output layer weight vector, B^O is the output layer bias weight and $\Phi(z) = 1/(1 + \exp(-z))$ is the sigmoid function (applied element by element for vectors).

As with the LLR, training the MLP consists on minimising the log-likelihood error

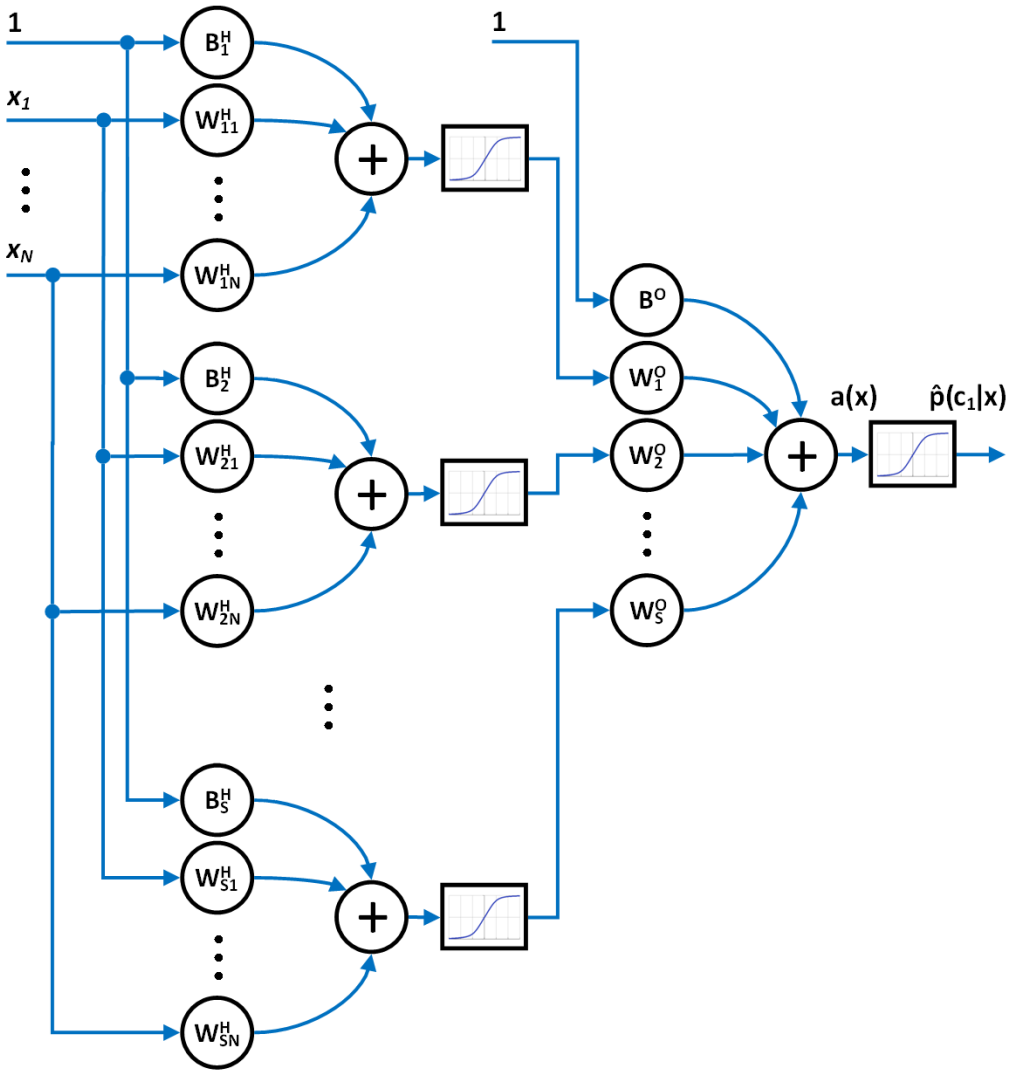


Figure 3.3: *An MLP with S nodes in the hidden layer and a single-neuron output layer. If the hidden neurons are equipped with exponential non-linear activation functions (normally sigmoid or hyperbolic tangent functions), and provided there is a large enough number of these units, an MLP can approximate arbitrarily complex functions to the desired accuracy* [84].

function, this time with respect to the synaptic weights. The simplest method to do this is gradient descent, which iteratively updates a parameter w with an increment $\Delta w = -\alpha \nabla_w \varepsilon_{LL}$, where α is the step size or learning rate. Interestingly, when gradient descent is applied to these models, their layer structure causes the propagation of the estimation error from the output back through the network, controlling the update of the weights inside. This is widely known as **error backpropagation** training [96, 97, 98]. For instance, in the MLP in Fig. 3.3, the variation of the objective function with respect

to the output of the network is

$$\delta = \frac{d\varepsilon_{LL}}{da(\mathbf{x})} = -\sum_{i=1}^{L}(y_i(1-\hat{p}(c_1|\mathbf{x}_i)) - (1-y_i)\hat{p}(c_1|\mathbf{x}_i)), \tag{3.6}$$

which, at the same time, is an aggregate measure of the individual estimation errors made. Using the chain rule, the weight updates can be expressed in terms of δ, which represents an error signal that propagates backwards to guide the update process:

$$\begin{aligned} \Delta\mathbf{W}^O &= -\alpha\frac{d\varepsilon_{LL}}{d\mathbf{W}^O} = -\alpha\delta\frac{da(\mathbf{x})}{d\mathbf{W}^O} \\ \Delta\mathbf{W}^H &= -\alpha\frac{d\varepsilon_{LL}}{d\mathbf{W}^H} = -\alpha\delta\frac{da(\mathbf{x})}{d\mathbf{h}(\mathbf{x})}\frac{d\mathbf{h}(\mathbf{x})}{d\mathbf{W}^H}, \end{aligned} \tag{3.7}$$

where $\mathbf{h}(\mathbf{x}) = \Phi(\mathbf{W}^H\mathbf{x} + \mathbf{B}^H)$ is the $S \times 1$ vector of hidden layer outputs.

Regarding the structure design of the implementation of the MLP, the number of free coefficients must be large enough to generate accurate posterior probability surfaces that provide the desired performance. However, an excess of the number of neurons results in a too complex output response that overfits the training data as well as the noise that it may include. Consequently, the network makes very small prediction errors on the training examples but its estimation power does not generalise well to new data. To mitigate overfitting, **regularisation** techniques modify the objective function by including a term ε_{reg} that penalises overly complex networks,

$$\varepsilon_{total} = \varepsilon_{LL} + \gamma\varepsilon_{reg}, \tag{3.8}$$

where the parameter γ controls the influence of the regularisation term. This framework uses a *weight decay* regulariser [85], which defines ε_{reg} as the sum of all the weights in the network squared,

$$\varepsilon_{reg} = \frac{1}{2}\sum_{w\in \substack{\mathbf{W}^H,\ \mathbf{B}^H \\ \mathbf{W}^O,\ B^O}} w^2. \tag{3.9}$$

The update of a generic weight using this regulariser is given by

$$\Delta w = -\alpha\nabla_w\varepsilon_{total} = -\alpha(\nabla_w\varepsilon_{LL} + \gamma\nabla_w\varepsilon_{reg}) = -\alpha(\nabla_w\varepsilon_{LL} + \gamma w). \tag{3.10}$$

In the absence of the first term, the second term of the update equation induces a decay of all weights to zero, hence the name of the regulariser. This effect means that for a weight to have a value different than zero, it must provide a contribution to the reduction of

the prediction error ε_{LL}. In other words, complex neural networks with very non-linear output surfaces, which are given by large weights, must be justified by a significant improvement of the model performance over simpler structures. Otherwise, the output function is smoothed to provide a better balance between simplicity and accuracy.

3.1.2 Multiclass classification

In multiclass problems, data points belong to one of J classes, $y_i \in \{1, \cdots, J\}$, so $a(\mathbf{x})$ is replaced by J functions $a_j(\mathbf{x})$. To ensure that the J estimated probabilities add up to one, $\sum_j \hat{p}(c_j|\mathbf{x})) = 1$, the sigmoid output of the binary estimator in Eq. (3.1) is replaced by a softmax activation function,

$$\hat{p}(c_j|\mathbf{x})) = \frac{\exp(a_j(\mathbf{x}))}{\sum_{k=1}^{J} \exp(a_k(\mathbf{x}))}. \tag{3.11}$$

To obtain the functions $a_j(\mathbf{x})$, the framework implements an MLP like the one in Fig. 3.3 but with J neurons in the output layer. Therefore, the functions $\mathbf{a}(\mathbf{x}) = (a_1(\mathbf{x}), \cdots, a_J(\mathbf{x}))^T$ are

$$\mathbf{a}(\mathbf{x}) = \mathbf{W}^O \Phi(\mathbf{W}^H \mathbf{x} + \mathbf{B}^H) + \mathbf{B}^O, \tag{3.12}$$

where $\mathbf{W}^O$ is now a $J \times S$ matrix and $\mathbf{B}^O$ is a $J \times 1$ vector. The training process is the same as in the binary classification case, adapting the log-likelihood objective function to multiclass prediction,

$$\varepsilon_{LLm} = -\sum_{i=1}^{L} \sum_{j=1}^{J} (c_j(\mathbf{x}_i) \log(\hat{p}(c_j|\mathbf{x}_i)) + (1 - c_j(\mathbf{x}_i)) \log(1 - \hat{p}(c_j|\mathbf{x}_i)), \tag{3.13}$$

and updating the weights using regularised backpropagation.

3.2 Fisher information matrix

Obtaining the closed-form probability estimates in Section 3.1 enables the next step in the process: calculating the Fisher information matrix. In this context of conditional class

probabilities, the FI matrix is given by any of the following two equivalent definitions:

$$\begin{aligned} &1) \quad \mathbf{FI}(\mathbf{x}) = \sum_{j}^{J} (\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))(\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))^T \hat{p}(c_j|\mathbf{x}) \\ &2) \quad \mathbf{FI}(\mathbf{x}) = -\sum_{j}^{J} (\nabla^2_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))\hat{p}(c_j|\mathbf{x}) \end{aligned} \tag{3.14}$$

Combining these definitions with Eqs. (3.1) and (3.11), this section is devoted to the expression of the FI matrix in terms of the parameters of the estimators considered above.

In the binary classification problem, the summation involved in the definitions reduces to a simple closed-form expression, so it is convenient to calculate the FI matrix expressions separately for the binary and multiclass cases.

3.2.1 FI matrix for two classes

The binary class problem only requires the estimation of one of the two posterior probability functions, since the other can be calculated from it. Consequently, the FI matrix can be written in terms of that probability. For estimators with a sigmoid output function, the matrix is

$$\mathbf{FI}(\mathbf{x}) = (\nabla_{\mathbf{x}} a(\mathbf{x}))(\nabla_{\mathbf{x}} a(\mathbf{x}))^T \hat{p}(c_1|\mathbf{x})(1 - \hat{p}(c_1|\mathbf{x})). \tag{3.15}$$

The calculations leading to this expression have been omitted here for brevity; they can be found in Appendix A.1. As discussed in Section 3.1, $a(\mathbf{x})$ is the mapping of the input that determines the estimation, and is given by

$$\begin{aligned} a(\mathbf{x}) &= \beta_0 + \boldsymbol{\beta}^T \mathbf{x} && \text{(LLR)} \\ a(\mathbf{x}) &= \mathbf{W}^O \Phi(\mathbf{W}^H \mathbf{x} + \mathbf{B}^H) + B^O && \text{(MLP)} \end{aligned} \tag{3.16}$$

for the linear and non-linear methods considered, respectively. Eq. (3.15) implies $a(\mathbf{x})$ must be a continuous function defined and differentiable everywhere in the input space. That is the case; its derivatives are

$$\begin{aligned} \nabla_{\mathbf{x}} a(\mathbf{x}) &= \boldsymbol{\beta} && \text{(LLR)} \\ \nabla_{\mathbf{x}} a(\mathbf{x}) &= (\mathbf{W}^O(\mathbf{W}^H \circ ((\mathbf{h}(\mathbf{x}) \circ (\mathbf{1}_{S\times 1} - \mathbf{h}(\mathbf{x})))\mathbf{1}_{1\times N})))^T && \text{(MLP)}, \end{aligned} \tag{3.17}$$

where $\mathbf{1}_{a\times b}$ is an $a \times b$ matrix with unit components and the '$\circ$' operator denotes the element-wise Hadamard product.

3.2.2 FI matrix for multiple classes

For multiple classes, the class-membership probabilities are given by Eq. (3.11) and, when inserted in Eq. (3.14), the expression of the FI matrix is

$$\mathbf{FI}(\mathbf{x}) = \sum_{j}^{J}\sum_{k}^{J}\sum_{l}^{J}(\nabla_{\mathbf{x}}(a_j(\mathbf{x}) - a_k(\mathbf{x})))(\nabla_{\mathbf{x}}(a_j(\mathbf{x}) - a_l(\mathbf{x})))^T \cdot \\ \cdot\hat{p}(c_j|\mathbf{x})\hat{p}(c_k|\mathbf{x})\hat{p}(c_l|\mathbf{x}). \tag{3.18}$$

Again, the steps to obtain this expression are detailed in Appendix A.2. As in the binary case, the functions $a_j(\mathbf{x})$ must be continuous and differentiable. Their derivatives, $\nabla_{\mathbf{x}}\mathbf{a}(\mathbf{x}) = (\nabla_{\mathbf{x}}a_1(\mathbf{x}), \cdots, \nabla_{\mathbf{x}}a_J(\mathbf{x}))$ are given by the binary MLP formula in Eq. (3.17), but with a $J \times N$ output weight matrix $\mathbf{W}^O$.

3.3 Fisher information metric

As Kaski et al. propose in [58, 59], the FI matrix in Eq. (3.14) can be used to define a metric in the input space, analogously to the traditional application of this idea in the parameter space. To do so, a differential distance is defined as

$$d(\mathbf{x}, \mathbf{x} + d\mathbf{x})^2 = d\mathbf{x}^T\mathbf{FI}(\mathbf{x})d\mathbf{x}, \tag{3.19}$$

which gives the distance between adjacent points in the input space.

3.3.1 The Fisher metric and the KL divergence

There is a widely known link between the traditional FI metric in the parameter space and the Kullback-Leibler (KL) divergence of generative distributions (see Eq. (2.42)). That connection is described in detail in [56] where, starting from the definition of the KL divergence and applying a Taylor expansion, it is shown that the measure is related locally to a differential metric determined by the FI matrix.

An analogous derivation of that relationship can be carried out in the input space for the classification context on which this work focuses. Given two neighbouring points $\mathbf{x}$ and $\mathbf{x} + d\mathbf{x}$ and a set of conditional class probabilities $\hat{p}(c_j|\mathbf{x})$, the KL divergence is given by

$$I_{KL}(\mathbf{x} : \mathbf{x} + d\mathbf{x}) = -\sum_{j}\log\left(\frac{\hat{p}(c_j|\mathbf{x} + d\mathbf{x})}{\hat{p}(c_j|\mathbf{x})}\right)\hat{p}(c_j|\mathbf{x}). \tag{3.20}$$

Appendix B shows that this divergence is related to the distance between the points under the FI metric,

$$I_{KL}(\mathbf{x} : \mathbf{x} + d\mathbf{x}) = \frac{1}{2} d(\mathbf{x}, \mathbf{x} + d\mathbf{x})^2 = \frac{1}{2} d\mathbf{x}^T \mathbf{FI}(\mathbf{x}) d\mathbf{x}. \tag{3.21}$$

The interpretation of the divergence measure and its derivation are described in detail in Appendix B. As a summary, Eq. (3.21) shows that the distance between two nearby points under the Fisher metric is connected to the divergence between the corresponding probabilities $\hat{p}(c_j|\mathbf{x})$ and $\hat{p}(c_j|\mathbf{x} + d\mathbf{x})$. Therefore, infinitesimal displacements along directions of the space that produce large divergences in the posterior probabilities will result in long differential Fisher distances, indicating high relevance with respect to the external class labels, and vice versa.

3.3.2 Global distances for linear estimators

Given the FI matrix, Eq. (3.19) calculates the distance between infinitely close points. In order to extend this local metric to any two points in the input space, the differential distance must be integrated along the corresponding path,

$$d(\mathbf{x}_A, \mathbf{x}_B) = \left| \int_{t_A}^{t_B} \sqrt{\dot{\mathbf{x}}(t)^T \mathbf{FI}(\mathbf{x}(t)) \dot{\mathbf{x}}(t)} dt \right|, \tag{3.22}$$

where $\mathbf{x}(t)$ represents a path in the input space that goes from $\mathbf{x}_A = \mathbf{x}(t_A)$ to $\mathbf{x}_B = \mathbf{x}(t_B)$ and $\dot{\mathbf{x}}(t) = d/dt\ \mathbf{x}(t)$; $\mathbf{x}(t)$ is the path along which the distance is measured.

Interestingly, when the FI matrix is calculated using a linear estimator like that in Eq. (3.2), the path integral can be solved analytically (see Appendix C for detail on the solution) resulting in the following closed-form expression,

$$d(\mathbf{x}_A, \mathbf{x}_B) = \left| 2 \left[\arctan \left(\exp \left(-\frac{a(\mathbf{x}(t))}{2} \right) \right) \right]_{\mathbf{x}_A}^{\mathbf{x}_B} \right|, \tag{3.23}$$

which does not depend on the particular path from $\mathbf{x}_A$ to $\mathbf{x}_B$. That is, any path connecting $\mathbf{x}_A$ and $\mathbf{x}_B$ has the same length under the Riemannian metric defined by the Fisher information.

Although this may seem counter-intuitive, it becomes clear when the detail of the method is scrutinised more closely. When the class probability is estimated using a binary model from the exponential family with a linear score function $a(\mathbf{x})$, the whole data space in effect collapses into a straight line. This is because $a(\mathbf{x})$ only takes into account

displacements along a single direction in the space. For the particular case of the LLR, this direction is given by the vector of regression coefficients $\boldsymbol{\beta}$ due to the dot product $\boldsymbol{\beta}^T\mathbf{x}$ present in Eq. (3.2), which results in the estimated probability changing only along the vector $\boldsymbol{\beta}$. The Fisher metric focuses on variations of the probability, so for any given pair of points it is only the distance between their projections onto $\boldsymbol{\beta}$ what determines the distance between them in the Riemannian space. In summary, because only one direction of the space is relevant, every path that connects the two points of interest will have the same length regardless of its shape.

3.3.3 Global distances for non-linear estimators: finding geodesic distances

In the case of the MLP estimators discussed in Section 3.1, the score $a(\mathbf{x})$ is a non-linear function of $\mathbf{x}$, and consequently its first derivative, included in the FI matrix expression, depends on $\mathbf{x}$. This complicates the integral in Eq. (3.22), making an analytical solution impossible to find. The same happens in the multiclass problem, this time due to the FI matrix having a more complex formulation. Distances in these cases depend on the path considered, and a method is required to find the length of the shortest path or *geodesic distance*.

This subsection describes several methods to approximate the minimum distance between a pair of points. First, the straight path and graph-based algorithms suggested in [60] are described, which are then followed by an alternative proposal developed specifically for this framework.

Straight path approximation

The simplest approximation to the true geodesic distance between two points $\mathbf{x}_A$ and $\mathbf{x}_B$ is to apply the differential expression in Eq. (3.19) directly, even if the points are not close to each other,

$$d_1(\mathbf{x}_A, \mathbf{x}_B)^2 = (\mathbf{x}_A - \mathbf{x}_B)^T \mathbf{FI}(\mathbf{x}_A)(\mathbf{x}_A - \mathbf{x}_B). \tag{3.24}$$

This considers the shortest path to be the straight line between the two points, and approximates the integral along that path by assuming that the FI matrix is constant throughout, with value $\mathbf{FI}(\mathbf{x}(t)) = \mathbf{FI}(\mathbf{x}_A)$. The FI matrix may be evaluated at the midpoint $(x_A + x_B)/2$ to make the distance symmetric.

To improve the accuracy of this approach, the path integral can be approximated nu-

merically using interpolation methods (e.g., the rectangle or trapezoid rules). By doing this, the FI matrix is evaluated at T points along the straight path,

$$d_T(\mathbf{x}_A, \mathbf{x}_B) = \sum_{t=1}^{T} d_1 \left(\mathbf{x}_A + \frac{t-1}{T}(\mathbf{x}_B - \mathbf{x}_A), \mathbf{x}_A + \frac{t}{T}(\mathbf{x}_B - \mathbf{x}_A) \right), \tag{3.25}$$

which can approximate the true length of the straight line as closely as desired by choosing a large enough T. The authors suggest $T = 10$ as a rule of thumb based on empirical classification performance.

Graph-based approximation

The immediate way to improve on the T-point approximation is to allow non-straight geodesic paths. The authors do this by using a graph-based geodesic estimation similar to the solution implemented in the Isomap algorithm [34]. The method starts by calculating all pairwise distances between the points in the dataset using the T-point algorithm. These distances are then used to define a fully connected graph where the weight of the edge between points $\mathbf{x}_A$ and $\mathbf{x}_B$ is given by $d_T(\mathbf{x}_A, \mathbf{x}_B)$. Finally, a graph search method such as the Floyd-Warshall algorithm is used to find the shortest path between the selected pair of points, which in the general case will consist of a series of hops between nodes in the graph. The length of the resulting path, $d_G(\mathbf{x}_A, \mathbf{x}_B)$, is calculated as the sum of the weights that it contains, and is taken as the geodesic distance.

The free point approach

By definition, the result of the graph-based geodesic calculation above depends on the particular set of data points available to use as vertices. This results in a restricted set of candidate paths from which the algorithm chooses the shortest. To remove that limitation, we propose an iterative approach to build an unconstrained approximation to the true geodesic.

The method starts by dividing the straight path connecting the points at hand, $\mathbf{x}_A$ and $\mathbf{x}_B$, into T+1 segments. A hyperplane is then defined in the point between each pair of consecutive segments. The resulting T hyperplanes are parallel to each other and orthogonal to the straight line path.

A point is defined within each of the hyperplanes, with the idea of establishing a path from $\mathbf{x}_A$ to $\mathbf{x}_B$ by connecting it to the points in the previous and next hyperplanes, as depicted in Fig. 3.4. The T points can move freely within their respective hyperplanes, and therefore any direct path between the two points can be formed provided T is large

enough.

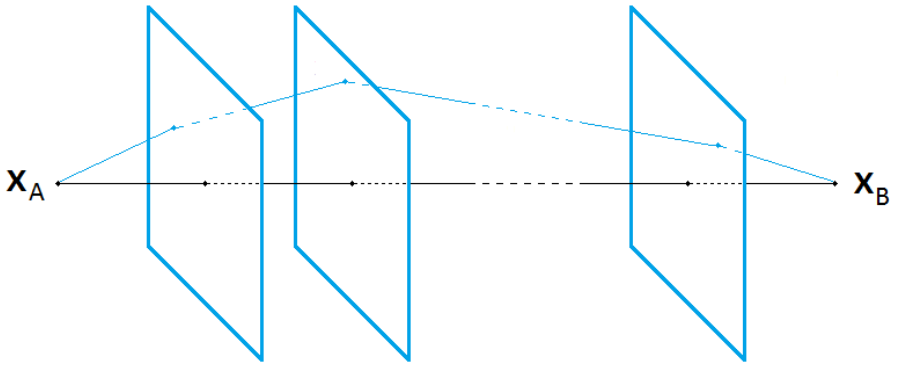


Figure 3.4: *The free point approach defines the path between* $\mathbf{x}_A$ *and* $\mathbf{x}_B$ *as the consecutive connection of a set of points. The location of these points within their hyperplane containers is iteratively adjusted to minimise the total length of the path.*

In order for the composed path to have minimum length, it is expressed in terms of the free points by adding the length of all segments,

$$d_{FP}(\mathbf{x}_A, \mathbf{x}_B) = \sum_{i=0}^{T} d(\mathbf{x}_F^i, \mathbf{x}_F^{i+1}), \tag{3.26}$$

where $\mathbf{x}_F^i$ are the free points, with $\mathbf{x}_F^0 = \mathbf{x}_A$ and $\mathbf{x}_F^{T+1} = \mathbf{x}_B$. The length of the segments can be calculated using the straight line methods discussed above and, for binary classifiers, also with Eq. (3.23) applied locally. The total distance is used as objective function, and it is minimised using gradient descent with respect to the position of the points,

$$\Delta \mathbf{x}_F^i = -\alpha \nabla_{\mathbf{x}_F^i} d_{FP}(\mathbf{x}_A, \mathbf{x}_B). \tag{3.27}$$

This optimisation setting raises the question of how to parameterise the point to hyperplane membership so the free points move within their corresponding hyperplane without leaving it during the updates. A solution to this problem is the use of the *Gram-Schmidt orthonormalisation*. This method takes a finite set of vectors as input and generates an orthogonal set $\mathbf{Z} = (\mathbf{z}_1, \cdots, \mathbf{z}_N)$, which in this case will be used as alternative axes for the representation of the free points. If the normalisation is applied so that one of those axes, for instance $\mathbf{z}_1$, has the same direction as the vector $\mathbf{x}_B - \mathbf{x}_A$, the rest of them will, by definition, be perpendicular to it and therefore be contained in a hyperplane orthogonal to the straight path connecting the two points (Fig. 3.5). In such a scenario, all points within each hyperplane will have the same value of the first coordinate, so the only condition for a point to lie within a particular hyperplane is to have the corresponding coordinate value.

The vectors $\mathbf{z}_i$ can be normalised to have unit length, forming a transformation matrix

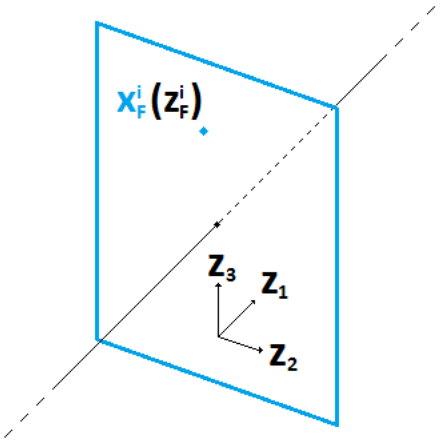


Figure 3.5: *Gram-Schmidt orthonormalisation applied in the context of the method. The axis* $\mathbf{z}_1$ *is perpendicular to the* T *hyperplanes, providing each of them with a unique coordinate value.*

$\mathbf{U} = (\mathbf{u}_1, \cdots, \mathbf{u}_N)$ that allows the transition from the coordinates of the free points in the original axes to the new ones and vice versa,

$$\mathbf{x}_F^i = \mathbf{U}\mathbf{z}_F^i \Longleftrightarrow \mathbf{z}_F^i = \mathbf{U}^T\mathbf{x}_F^i. \tag{3.28}$$

In this new representation, the objective function can be written as

$$d_{FP}(\mathbf{x}_A, \mathbf{x}_B) = d_{FP}(\mathbf{U}\mathbf{z}_A, \mathbf{U}\mathbf{z}_B) = \sum_{i=0}^{T} d(\mathbf{U}\mathbf{z}_F^i, \mathbf{U}\mathbf{z}_F^{i+1}). \tag{3.29}$$

Similarly, the gradient descent formulation is now

$$\Delta\mathbf{z}_F^i = -\alpha\nabla_{\mathbf{z}_F^i} d_{FP}(\mathbf{U}\mathbf{z}_A, \mathbf{U}\mathbf{z}_B). \tag{3.30}$$

The update of the free points is carried out normally except for their first coordinate, which is kept constant so they stay within the hyperplanes.

3.4 Fisher networks

After selecting a distance calculation method, the next step is the construction of similarity networks based on the Fisher metric. Distances are, however, inversely correlated to similarity, i.e., points separated by a small distance are considered very similar, and conversely for large distances. A further step is therefore required to transform a distance value into a measure of the similarity between a pair of points, which may then be used

to weight the edge connecting the corresponding pair of nodes in the network.

Assuming that zero distance maps to maximum similarity and recognising that the anisotropy of the projective space of covariates has been captured by the Fisher metric, it is natural to transform distances into similarity indicators using a Gaussian radial kernel,

$$A_{ij} = \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_G^2}\right), \tag{3.31}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ under the Fisher metric and σ_G is the *network locality parameter*, which determines the width of the Gaussian kernel. The distances $d(\mathbf{x}_i, \mathbf{x}_j)$ define the meaning of the similarities A_{ij} with respect to the estimate of the conditional class probabilities. These measures, which lie in the range $A_{ij} \in (0, 1]$, form the adjacency matrix $\mathbf{A}$ that contains the structure of the Fisher network. Note that, although the diagonal elements A_{ii} should be equal to 1 given that $d(\mathbf{x}_i, \mathbf{x}_i) = 0$, they are explicitly set to zero to avoid self-connections in the network.

The parameter σ_G controls the width of the kernel and, by extension, how local the resulting similarities are: a small value of σ_G will result in only very close points having significant connection weights, while larger values will reduce this effect and produce meaningful values also for points that are further away. Notice that the weights A_{ij} characterise the network and determine the communities extracted from it and the predictions drawn, so it is important to establish some criteria to choose σ_G.

3.4.1 Selection of the network locality parameter

The suggested method to select σ_G is the empirical study of its effect on different measures used as indicators of the goodness of a particular value of the parameter. The idea is to calculate the adjacency matrix using the Gaussial kernel for a range of values of σ_G, evaluating the indicators for each of the resulting networks. Based on the measures, a recommended value of the parameter (or range thereof) is provided. When selecting σ_G in the experiments shown in Chapter 4, the following three measures are used.

Faithfulness of the network predictions

The first measure has to do with the accuracy with which the Fisher network is capable of replicating the class-membership probabilities given by the original estimator, $\hat{p}(c_j|\mathbf{x})$. The way predictions are drawn from the network in this framework is by approximating the score $a_k(\mathbf{x})$ given by the estimator for a point $\mathbf{x}_i$ as a weighted sum of the rest of the

nodes' scores,

$$a'_k(\mathbf{x}_i) = \frac{1}{\sum_j A_{ij}} \sum_j A_{ij} a_k(\mathbf{x}_j), \tag{3.32}$$

which is then inserted into the output function of the estimator to calculate the probability $\hat{p}'(c_k|\mathbf{x}_i)$. These network estimates are computed for all points in the dataset, and their quality is measured using the KL divergence between them and the original estimates, used as reference,

$$\varphi_{KL} = -\frac{1}{L} \sum_i^L \sum_k^J \hat{p}(c_k|\mathbf{x}_i) \log \left(\frac{\hat{p}'(c_k|\mathbf{x}_i)}{\hat{p}(c_k|\mathbf{x}_i)} \right). \tag{3.33}$$

The measure gives an idea of how well the network predictions replicate those of the original estimator, with the minimum value $\varphi_{KL} = 0$ representing identical estimates. The term $1/L$ is used to normalise the measure and make it comparable across datasets.

Cramer's V statistic

The second measure is Cramer's V [99], an association score included here to provide an evaluation of how coherent the communities found in the network are with respect to the class labels of the data. It is calculated from the contingency table of the vector of community memberships and the vector of true labels as

$$\varphi_{CV} = \sqrt{\frac{\chi^2}{L \min(C_1 - 1, C_2 - 1)}}, \tag{3.34}$$

where χ^2 is the chi-squared statistic obtained from the table, C_1 is the number of communities and C_2 is the number of classes in the dataset. This measure ranges from 0 to 1, indicating the concordance between the community allocation of the nodes and their original class membership, with $\varphi_{CV} = 0$ meaning no association and $\varphi_{CV} = 1$ corresponding to complete concordance.

McNemar's test

The third measure has to do with the classification accuracy associated with the network predictions, $\hat{p}'(c_k|\mathbf{x}_i)$, and the primary ones, $\hat{p}(c_k|\mathbf{x}_i)$, used as reference. Because the two classifiers are applied to the same data, the extent to which the differences between the

errors they make can be attributed to chance is quantified with McNemar's test [86],

$$\varphi_{MN} = \frac{|n_A - n_B| - 1}{\sqrt{n_A + n_B}}, \tag{3.35}$$

where n_A denotes the number of errors made by the network predictions and not by the original ones and vice versa for n_B. Under the null hypothesis that the classifiers are not significantly different, φ_{MN} is given by a normal distribution $N(0, 1)$, so it is possible to obtain a p-value from the test and therefore define a threshold for significance. This is of interest for the evaluation, since the selected value of σ_G is expected to produce a network that retains the classification power of the initial estimator.

Section 4.4.2 shows examples of the parameter selection process using these criteria. Note that the three measures described above are suggestions adequate for classification and community detection purposes; other measures can be incorporated to this analysis to suit the user's interests.

3.5 Community detection

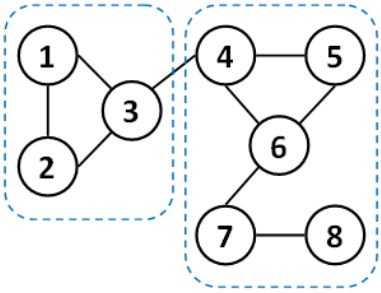
Eq. (3.31) represents the transition from the space of the covariates to the network sphere. The adjacency matrix $\mathbf{A}$ represents a fully connected graph, the Fisher network, whose structure contains the geometrical relationships between the points in the original dataset under the FI metric. The next stage of the process is the study of that structure in order to find meaningful communities with respect to the classification problem. This functionality is implemented by Newman's spectral algorithm for community detection [83], whose theoretical basis is described in Section 2.2.2. This section covers the application of the method in practice with a small example.

Fig. 3.6 shows a simple network whose community structure is desired. The algorithm obtains it by iteratively splitting the network in a way that maximises a measure of modularity. For the initial split, the resulting modularity score is given by

$$Q = \frac{1}{4m} \sum_{i}^{L} \sum_{j}^{L} \left(A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s}, \tag{3.36}$$

where $\mathbf{s}$ contains the node memberships $s_i \in \{+1, -1\}$ after the partition. The modularity matrix $\mathbf{B}$ is calculated from A_{ij}, the weight of the edge connecting nodes $\mathbf{x}_i$ and $\mathbf{x}_j$; k_i and k_j, the degree or total weight of the edges adjacent to these two nodes, and m, the aggregate weight of all edges in the network. The optimum split is given by the signs of $\mathbf{v}_1$, the eigenvector of $\mathbf{B}$ with the highest eigenvalue. Fig. 3.6 contains the values of $\mathbf{A}$,

B and $\mathbf{v}_1$ for the example, as well as the resulting division.



$$\mathbf{A}=\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{B}=\begin{bmatrix} -.22 & .78 & .67 & -.33 & -.22 & -.33 & -.22 & -.11 \\ .78 & -.22 & .67 & -.33 & -.22 & -.33 & -.22 & -.11 \\ .67 & .67 & -.50 & .50 & -.33 & -.50 & -.33 & -.17 \\ -.33 & -.33 & .50 & -.50 & .67 & .50 & -.33 & -.17 \\ -.22 & -.22 & -.33 & .67 & -.22 & .67 & -.22 & -.11 \\ -.33 & -.33 & -.50 & .50 & .67 & -.50 & .67 & -.17 \\ -.22 & -.22 & -.33 & -.33 & -.22 & .67 & -.22 & .89 \\ -.11 & -.11 & -.17 & -.17 & -.11 & -.17 & .89 & -.06 \end{bmatrix}$$

$$\mathbf{v}_1=\begin{bmatrix} .46 & .46 & .41 & -.16 & -.30 & -.41 & -.30 & -.16 \end{bmatrix}^{\mathsf{T}}$$

Figure 3.6: *Example network with 8 nodes and 9 edges with unit weights. The community detection algorithm starts by splitting the network according to the signs of the leading eigenvector of the modularity matrix* **B**. *The resulting node assignment, highlighted with a dashed line, increases the modularity from the initial* $Q = 0$ *(no division) to* $Q = 0.36$.

After the first division, the algorithm continues the splitting process with each of the resulting partitions. To do it, a reduced modularity matrix is defined for each of them,

$$B_{ij}^{(g)} = A_{ij} - \frac{k_i k_j}{2m} - \delta_{ij}\left(k_i^{(g)} - k_i \frac{d_g}{2m}\right), \tag{3.37}$$

where $k_i^{(g)}$ is the degree of node $\mathbf{x}_i$ within subgraph g and d_g is the sum of the degrees of the nodes that form the subgraph, $d_g = \sum_{l \in g} k_l$. That expression reduces the initial **B** to a $L_g \times L_g$ matrix and adjusts the diagonal elements so that its rows and columns add up to zero. Then, like in the initial stage, the leading eigenvector of $\mathbf{B}^{(g)}$ is calculated and the subgraph is split according to its signs. This produces a change in the modularity of the network given by

$$\Delta Q = \frac{1}{4m}\mathbf{s}^{(g)T}\mathbf{B}^{(g)}\mathbf{s}^{(g)}. \tag{3.38}$$

Fig. 3.7 illustrates how the splitting procedure continues after Fig. 3.6. The smaller of the two partitions is indivisible, i.e., there is no way to split so that $\Delta Q > 0$, but the other one can still produce a slight increase in modularity if appropriately divided.

The algorithm continues the recursive division of the network for as long as there are

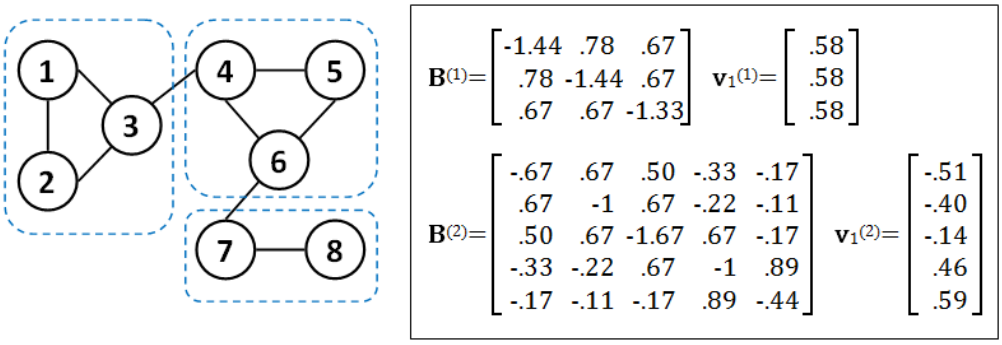


Figure 3.7: *Further partitioning of the network in Fig. 3.6. The largest eigenvalue of the left hand side subgraph is zero, and consequently no subdivision is possible that increases modularity. In contrast, the larger subgraph is divisible, producing an improvement* $\Delta Q = 0.04$*, for a total* $Q = 0.40$*.*

splits that increase the global modularity. At some point, all remaining communities will be indivisible, and the execution of the algorithm will conclude. This is the case after carrying out the partition in Fig. 3.7: none of the three remaining communities have positive eigenvalues, and so the process is over and the final communities are defined.

3.6 Practical uses of Fisher networks

The final section of this chapter discusses the functionality that Fisher networks bring into the framework in practical classification problems, namely data representation, identification of central nodes within the communities and case-based class prediction.

3.6.1 Dataset visualisation

One of the motivations for this work is to produce a dataset representation method that highlights the structure of the data in a way that is meaningful with respect to the classification problem at hand. After building the Fisher network and analysing its structure, this task is as simple as plotting the nodes and the edges connecting them.

Since the network, as defined by the adjacency matrix $\mathbf{A}$, is fully connected, some form of edge filtering must be carried out prior to the representation. Possible approaches include pruning edges with weights below a certain threshold, keeping only the K strongest connections, displaying only the spanning tree of the network, etc. The experiments in this work display two different types of edges: connections between nodes from the same community and connections between the most central node in each cluster and the rest of the members.

The other visualisation factor is the location of the nodes in the representation of the network, which directly affects the clarity with which the structure of the graph is perceived by the user. This is especially important for large networks, where careless positioning of the nodes will likely result in a confusing mesh of edges. To avoid this, nodes should be arranged in a way that places points that are close under the Fisher metric also close in the graph, and vice versa. A good method to obtain such placement is to make use of the MDS techniques described in Section 2.1.1. In particular, this framework uses the Sammon mapping [32], a non-linear projection algorithm that takes the $L \times L$ matrix of pairwise Fisher distances $d_F(\mathbf{x}_i, \mathbf{x}_j)$ of the points in the N-dimensional space and returns the coordinates of a new set of L points in a low-dimensional space (2 or 3-D in this context) for which the Euclidean distances $d_E(\mathbf{x}'_i, \mathbf{x}'_j)$ approximate the original ones as closely as possible, minimising the objective function

$$\varepsilon = \sum_{i,j=1}^{L} \frac{(d_F(\mathbf{x}_i, \mathbf{x}_j) - d_E(\mathbf{x}'_i, \mathbf{x}'_j))^2}{d_F(\mathbf{x}_i, \mathbf{x}_j)}. \tag{3.39}$$

This is a simple way of automatically locating the nodes that helps make the data structure apparent (when there is any).

Informative backgrounds

The Fisher network framework offers an optional visualisation setting that enhances interpretability of the data by colouring the background of the plot of the graph according to the predicted local class probabilities . To do so, a colour is assigned to each of the J classes present in the dataset as an RGB triplet. The colour of a point in the background is then calculated as a sum of the J colours, where the contribution of each class' colour is weighted by the corresponding posterior probability,

$$\mathbf{C}(\mathbf{x}'_b) = \sum_{j}^{J} \hat{p}'(c_j|\mathbf{x}'_b)\mathbf{C}_j, \tag{3.40}$$

where $\mathbf{C}(\mathbf{x}'_b)$ contains the red, green and blue components of the colour at point $\mathbf{x}'_b$ in the 2-D background and $\mathbf{C}_j$ is the colour assigned to class j, also in RGB format. This expression effectively modulates the colour of the background using the local probabilities in the area. The class membership probabilities at $\mathbf{x}'_b$ are calculated by applying the softmax output function (sigmoid function for binary classification) to the estimated

scores

$$a'_j(\mathbf{x}'_b) = \frac{1}{\sum_i A'_{bi}} \sum_i A'_{bi} a_j(\mathbf{x}_i),$$
$$\text{where } A'_{bi} = \exp\left(-\frac{(d_E(\mathbf{x}'_b, \mathbf{x}'_i))^2}{\sigma_G^2}\right). \tag{3.41}$$

Eq. (3.41) estimates the class membership probability of points in the background using a weighted average similar to that in Eq. (3.32), with the difference that here the Fisher distances required to calculate the contribution of each point in that expression are approximated using Euclidean distances obtained from the 2-D projection. This is a sensible solution, given that the objective of the Sammon mapping is to preserve Fisher distances in the Euclidean space, and it provides probability figures for background pixels $\mathbf{x}'_b$ that, due to the nature of the projection, do not have an explicit counterpart in the original input space.

To apply this to the whole background, it is first divided into as many *pixels* as desired, which are then coloured using the RGB vector corresponding to the centre point of the square, as given by Eq. (3.40). The result is a very intuitive representation that gives the user an immediate idea of the most probable class memberships for a node or group of nodes upon observation of the relevant plot area.

For problems with many different classes, or where a simpler representation is preferred, there is also the possibility of colouring the background in greyscale. In this case, the particular tone of the pixels does not depend on individual class probabilities. Instead, the two classes j and k with the highest posterior probabilities at $\mathbf{x}'_b$ are selected and the difference between those values is calculated, $d = \hat{p}'(c_j|\mathbf{x}'_b) - \hat{p}'(c_k|\mathbf{x}'_b)$. That value determines the colour applied, from white when $d = 0$ to black when $d = 1$. Therefore, boundary regions will be easily recognisable by their light grey and white tones, while areas with a strong class membership will be characterised by a dark grey, close to black colour. Section 4.4.2 contains examples of both types of informative backgrounds applied in practice.

3.6.2 Identification of reference cases

The partition of the network into communities provides information about the way nodes are grouped, which at the same time, given the construction process of the graph, describes the data structure in the original space. However, the study of the network does not have to stop there; it may also be interesting in certain applications to identify the most important nodes of each community so they can be used as representatives of the whole cluster. Locating such elements in the graph can be useful, for instance, to perform case-based classification considering only a reduced set of instances instead of the whole

dataset, or to give an idea of the membership strength of a node to its community using the distance that separates it from the corresponding reference cases.

To identify such nodes, a *centrality* score is calculated that gives a measure of the importance of each node or, equivalently, how central it is within its community. In this framework, the centrality of a node $\mathbf{x}_i$ within a community g is calculated as the sum of the weights that connect it with members of the same group,

$$k_i^{(g)} = \sum_{j \in g} A_{ij}, \tag{3.42}$$

which is normally known as the degree of $\mathbf{x}_i$ within g. For that reason, this measure is known as degree centrality. There are more complex centrality measures in the literature [61], such as eigenvector centrality, Katz centrality, *PageRank*, etc. However, it is precisely its simplicity that makes degree centrality a good choice for this framework. It requires very little computational effort and, more importantly, it is based on a very intuitive concept, which makes the choice of the resulting reference cases easy to explain.

3.6.3 Case-based classification

An important functionality of the framework is to produce interpretable semi-supervised class predictions for new data that replace the less intuitive class assignments given by the original estimator. Fisher networks tackle this problem by expressing their predictions for a new point in terms of the estimates of the nodes present in the network. Specifically, an uncategorised test point $\mathbf{x}_t$ is incorporated to the Fisher network of training data following these steps:

1. Calculate the L pairwise Fisher distances between $\mathbf{x}_t$ and the points $\mathbf{x}_i$ already in the network, $d_F(\mathbf{x}_t, \mathbf{x}_i)$, using the geodesic calculation method of choice.

2. Obtain the associated weights A_{ti} using Eq. (3.31).

3. Attach the weights to the initial adjacency matrix and apply Newman's algorithm to extract communities.

4. Identify reference cases in the communities using degree centrality.

5. Denoting by g the community containing node $\mathbf{x}_t$ and by $\mathcal{R}$ the set of reference cases, estimate the scores $a'_j(\mathbf{x}_t)$ from i) all nodes in the network, ii) community neighbours or iii) reference cases using the corresponding expression in Eq. (3.43). Use the softmax function (sigmoid function in the binary case) to calculate the class-membership probabilities $\hat{p}'(c_j|\mathbf{x}_t)$.

$$
\begin{aligned}
a'_j(\mathbf{x}_t) &= \frac{1}{\sum_{i=1}^{L} A_{ti}} \sum_{i=1}^{L} A_{ti} a_j(\mathbf{x}_i) \quad \text{(network)}, \\
a'_j(\mathbf{x}_t) &= \frac{1}{\sum_{\mathbf{x}_i \in g} A_{ti}} \sum_{\mathbf{x}_i \in g} A_{ti} a_j(\mathbf{x}_i) \quad \text{(community)}, \\
a'_j(\mathbf{x}_t) &= \frac{1}{\sum_{\mathbf{x}_i \in \mathcal{R}} A_{ti}} \sum_{\mathbf{x}_i \in \mathcal{R}} A_{ti} a_j(\mathbf{x}_i) \quad \text{(reference cases)}.
\end{aligned} \tag{3.43}
$$

At the end of the process, the user has information about the new point $\mathbf{x}_t$ regarding location within the network representation, community membership, similar nodes and reference cases and estimated class probabilities calculated from known cases in the database, with individual contributions indicated by the weights A_{ti}. All this information is available for the user to analyse and decide whether they accept the class assignment recommended by the system.

3.7 Chapter summary

In this chapter, a descriptive explanation is provided of the construction process of Fisher information networks. Implementation details are given for each of the stages involved, from the estimation of the posterior class probabilities $p(c_j|\mathbf{x})$ to the identification of network communities and reference cases, covering important topics such as the approximation of geodesic distances and the selection of the network locality parameter σ_G.

The different functionalities and potential practical usefulness of the framework are then discussed, explaining how to visualise the networks and obtain interpretable class predictions from them.

Chapter 4

Experimental results

Following the description of the theoretical basis that supports the Fisher network framework, this chapter presents a series of empirical tests designed to put the system into practice. The experiments involve synthetic and real-world classification data coming from binary and multiclass problems, and their objective is to show examples of Fisher networks and discuss practical issues related to the methodology and the graph construction process.

The first two sections study the effect of the Fisher information metric on the data geometry (Section 4.1) and high-dimensional KNN classification (Section 4.2). After that, Section 4.3 provides a detailed comparison between the free point approach and the graph-based approximation; both geodesic calculation methods described in Section 3.3.3. Then follows Section 4.4, which contains examples of Fisher networks built from different real-world datasets. Finally, Section 4.5 introduces a related application where the FI metric is used to improve the performance of an unsupervised blind signal separation method through the use of auxiliary data.

4.1 The effect of the Fisher metric

The aim of the first application of this chapter is to show how the FI metric affects the original arrangement of the data in the Euclidean space. To do it, the metric is derived for a simple 2-D classification problem, the Cushing's syndrome dataset [86]. It comprises 27 examples corresponding to patients with one of three types of this syndrome, each of which defines a class. The dataset contains measurements of urinary excretion rates of two steroid metabolites, tetrahydrocortisone and pregnanetriol, both measured in mg/24h. The fact that there are only two variables for each patient allows the direct visualisation of the data, as shown in Fig. 4.1.

Fig. 4.1 also displays the posterior probabilities $\hat{p}(c_j|\mathbf{x})$ estimated from the data with

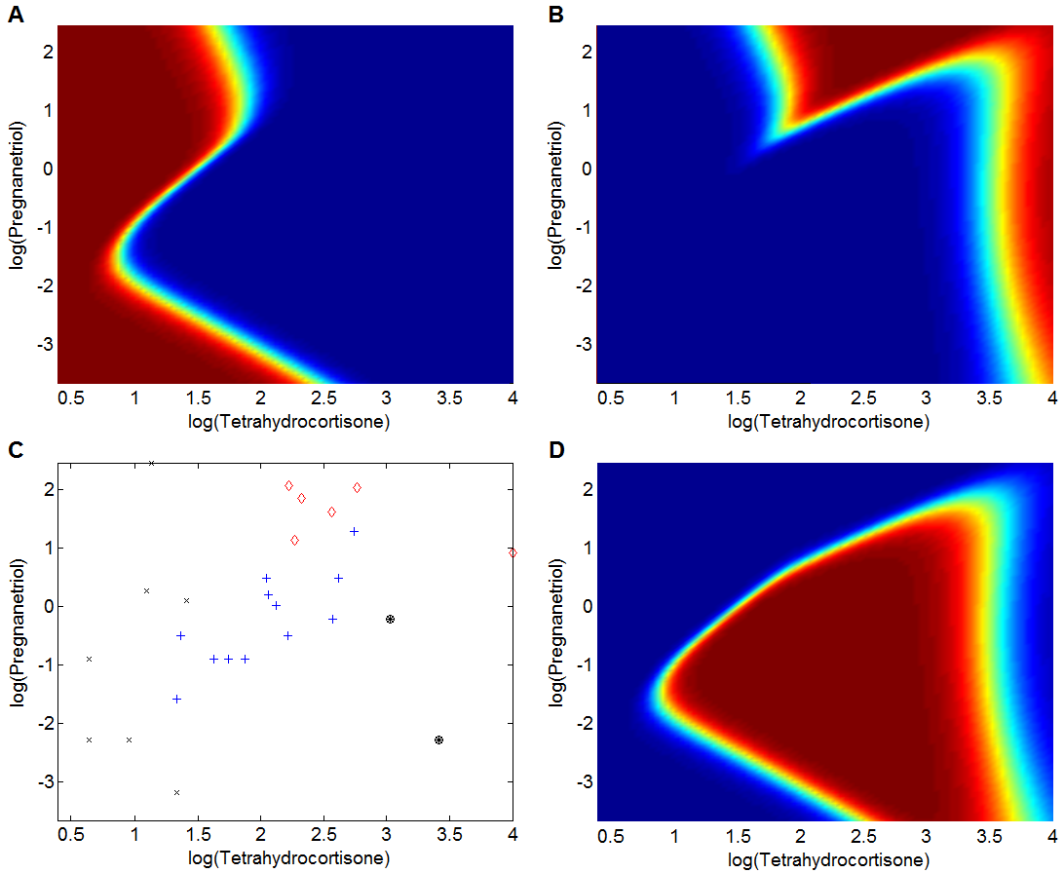


Figure 4.1: *Cushing's syndrome dataset and the estimated class membership probabilities. The dataset* **(C)** *contains three classes, plus two samples with unknown label, represented by black circles. The other three figures are colour plots of the conditional probabilities of the classes formed by* '×' **(A)**, '◊' **(B)** *and* '+' **(D)** *symbols, where red corresponds to values close to 1 and blue to values close to 0.*

an MLP, and used in Eq. (3.18) to obtain the FI matrix required to calculate distances with the metric. The free point approach introduced in Section 3.3.3 is then used to obtain the 27×27 pairwise Fisher distance matrix, and from it a 2-D Sammon mapping is produced, as explained in Section 3.6.1. In this case there are no edges involved, since the interest is to get an idea of the geometric structure of the data under the FI metric.

Fig. 4.2 compares the original dataset with the representation of the Sammon mapping. The effect of the metric on the arrangement of the data points is immediately apparent: points belonging to the same class are pulled together while pairs from different classes are pushed apart, which is exactly the objective of metrics learnt for classification problems, as seen in Chapter 2. The probability surfaces in Fig. 4.1 give an intuitive explanation as to how this is achieved: a path joining a pair of points in different classes must necessarily go through the slope of at least two of the three surfaces. Eq. (3.14) implies that the FI matrix will have relatively large components in locations of the data space where

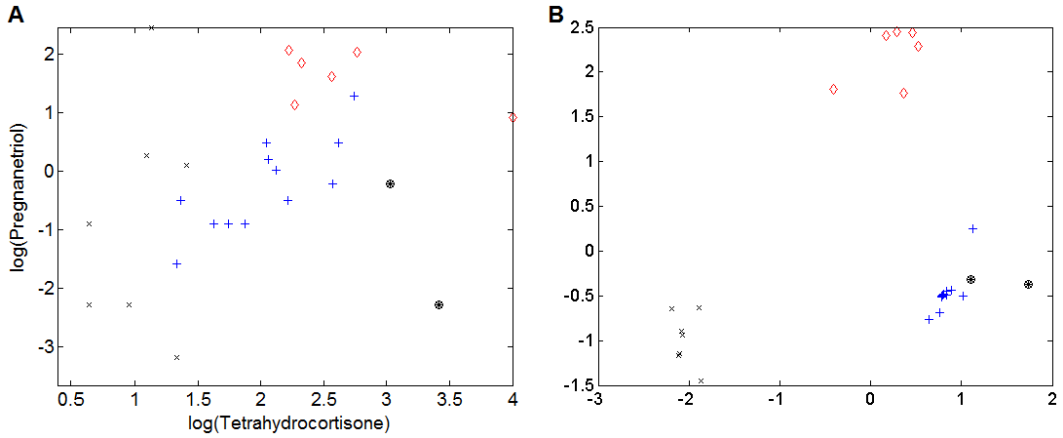


Figure 4.2: *Original Cushing's syndrome dataset* **(A)** *and Sammon mapping using Fisher distances* **(B)**. *Axes in Sammon mappings are unlabelled, as they do not represent any function of the input variables (only distances between points in the map are relevant, and not individual coordinates).*
The FI metric collapses the flat areas of the space and expands the relevant ones, where the conditional class probability changes. This results in compact communities that are well separated from each other.

the conditional class probabilities change, as is the case in the aforesaid slopes. The consequence is that the differential distances corresponding to the components of the global path that cross those high-variation areas will produce significant contributions to the final distance between the two points. On the other hand, paths connecting points from the same class will, in general, not have to leave the areas of the space where the surfaces are flat, resulting in small components of the matrix and therefore small distances.

This classification problem is simple, and it was possible for the MLP to attain a 100% accuracy. In cases where the class borders are less well-defined, as will be seen later in the chapter, the separation between the classes under the metric will not be as clear as in this example. However, even in difficult problems, the metric produces a better representation of the data where clear-cut and borderline cases are easy to identify and the bulk of each class is well separated from the rest of the groups.

4.2 Fisher metric and KNN classification

In this section, the Fisher metric is applied to a classification problem using synthetic data. Two versions of the standard K-nearest neighbour (KNN) classifier are compared: one that computes distances using the Euclidean metric (E-KNN) and another that uses the Fisher metric (F-KNN) derived from the class probabilities estimated with an MLP. The idea is to compare the two metrics in a simple classification context to study how

they perform for data of high dimensionality.

The dataset used consists of two classes generated by two Gaussian distributions centered in the origin with standard deviations $\sigma_0 = 0.9\mathbf{I}$ and $\sigma_1 = 2\mathbf{I}$. One distribution contains the other, giving rise to a non-linear border. This is a large dataset with 10^4 samples per class, which provide the MLP with enough training episodes to accurately estimate $p(c_0|\mathbf{x})$ and $p(c_1|\mathbf{x})$. After training, a validation dataset is produced using the original generating functions of the data. This smaller dataset (250 samples per class) contains the points to be classified by E-KNN and F-KNN.

Table 4.1: *KNN accuracy results for different values of the number of data dimensions, N, and the number of neighbours used, K.*

Dimensionality (N)	**Metric**	**Classification accuracy (%)**					
		$K = 3$	$K = 5$	$K = 7$	$K = 11$	$K = 15$	$K = 21$
2	Euclidean	69.4	72.4	74	73.6	75.4	75
	Fisher	72	74.8	74	75.6	76.8	76.6
5	Euclidean	87.4	88.2	88.2	88.8	88.2	88.6
	Fisher	73.6	72.8	74.4	76.6	79	79.4
10	Euclidean	88.8	89	88.8	87.2	86	85.8
	Fisher	92.2	92.2	92.2	93.2	93.4	93.8
15	Euclidean	81.6	80	77.6	74.8	73	71.4
	Fisher	93.2	93.4	93	93.2	93.8	93.2
25	Euclidean	66.4	64.4	61.8	57	56.2	54.4
	Fisher	94.6	95	95.2	95.2	95.2	95.2
40	Euclidean	51.6	50.6	50.4	50.2	50.2	50
	Fisher	94.2	94	94	94	94.2	94.4

Table 4.1 and Fig. 4.3 show the results of the simulations for Euclidean and Fisher metrics. In low dimensions, the two methods perform similarly. However, the accuracy of the Euclidean classifier increases until $N = 10$ and decreases from then on. To understand this behaviour, a histogram of the pairwise distances is plotted in Fig. 4.4 for different values of N, analysing the distribution of distances between pairs of points from the same class (intraclass distances) and from different classes (interclass distances).

Naturally, the performance of KNN is best when intraclass distances are small compared to interclass distances. In such case, the K nearest neighbours of a given validation point are likely to belong to the same class and therefore result in a correct classification. Fig. 4.4(A) shows all three distributions (intraclass 0, intraclass 1 and interclass) overlapping. Individual histograms show that intraclass distances have their peak more to the left than interclass distances, that is, they are generally smaller. In the next plot, Fig. 4.4(B), the shape shifts right and starts splitting into two humps, the left one corresponding to intraclass 0 distances and the other to intraclass 1 and interclass distances. The increase of the magnitude of distances with N is caused by the nature of the high-dimensional space.

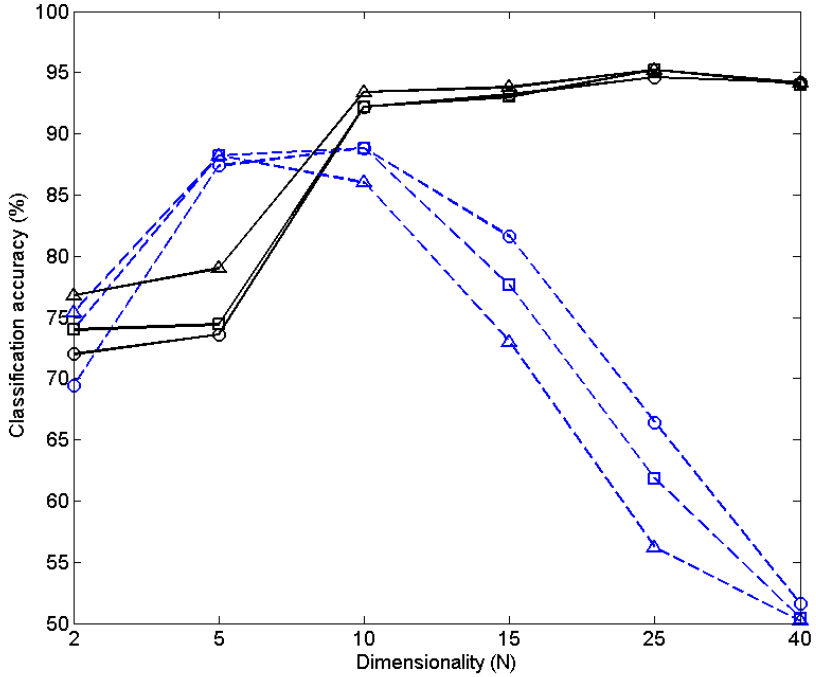


Figure 4.3: *Graphical representation of the classification accuracies in Table 4.1 for $K = 3$ (◯), $K = 7$ (□) and $K = 15$ (△) neighbours. Dashed blue lines and solid black lines correspond to Euclidean and Fisher metric results, respectively.*

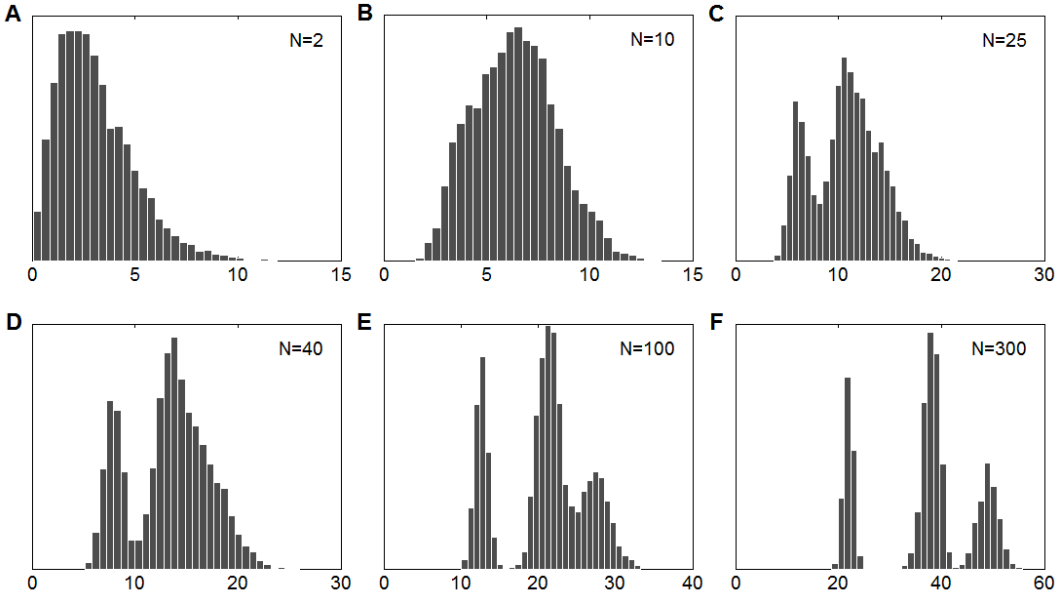


Figure 4.4: *Histograms of the pairwise Euclidean distances for different values of the number of dimensions. As N increases, the intraclass 1 and interclass distances grow faster than intraclass 0 distances, which results in an increase of the prediction accuracy of samples from class 0. However, intraclass 1 distances eventually become larger than interclass distances, which makes class 1 samples impossible to classify correctly.*

The reason why intraclass 0 distances shift more slowly is the smaller standard deviation of their generative function. At this point, where N is between 5 and 10, the classification of class 0 members becomes easier because their intraclass distances becomes even smaller with respect to interclass distances. For class 1, the situation is similar to the 2-dimensional case, so the overall result is an increase of the accuracy.

For larger values of N, distances keep growing. It is crucial to note that intraclass 1 distances increase faster than interclass ones. This results in a clear division of the three types of distances: intraclass 0 distances are the smallest ones, followed by interclass and then intraclass 1 distances. As a consequence, all points are classified as class 0 members: In the case of an actual class 0 sample, intraclass distances are much smaller than interclass ones, so the K chosen neighbours always belong to class 0. Conversely for class 1 members, the choice of neighbours inevitably causes a bad prediction.

The FI metric avoids this problem by relying on the variations of the probability surfaces, taking advantage of the fact that the MLP estimates $p(c_j|\mathbf{x})$ better in high dimensional spaces provided enough data is available. Fig. 4.5 shows histograms of Fisher distances for $N = 2$ and $N = 300$.

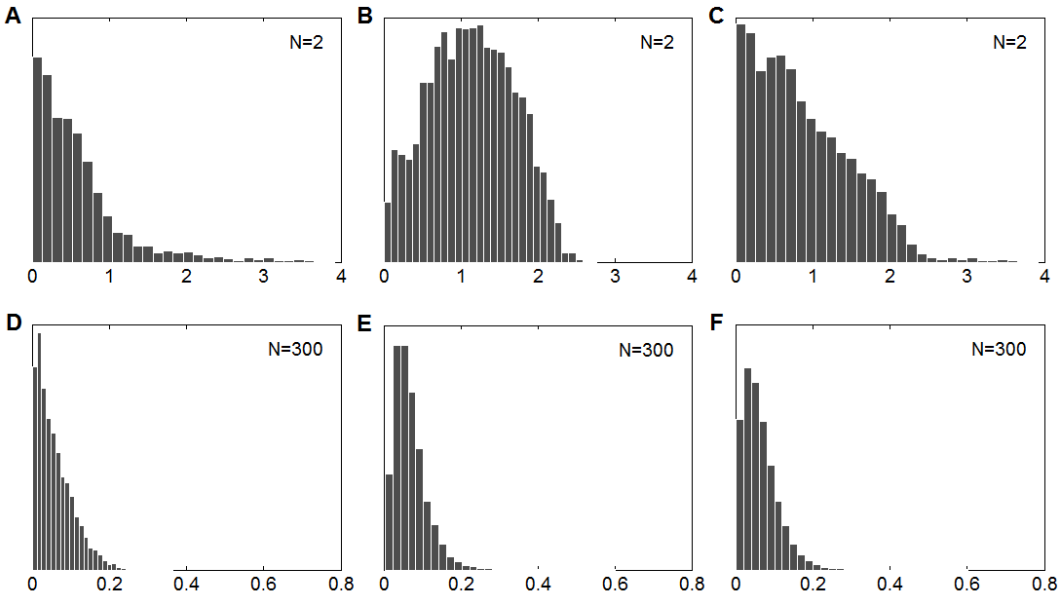


Figure 4.5: *Histograms of the pairwise Fisher distances for $N = 2$ and $N = 300$. Plots* **(A)** *and* **(D)** *correspond to intraclass distances,* **(B)** *and* **(E)** *to interclass distances and* **(C)** *and* **(F)** *to the combination of both. In contrast to the Euclidean case, the relative magnitude of intraclass distances with respect to the interclass ones is similar for any value of N, as depicted by their distributions for the extremes of the range of N considered.*

None of the variation seen in Fig. 4.4 appears in the Fisher case. Although there are also changes in the magnitude of the Fisher distances, intraclass distances remain in general smaller than interclass ones for both classes, regardless of the number of dimensions.

Interestingly, when N increases, Fisher distances decrease in magnitude. This is because Fisher distances can only be large when there is a lot of variation of the class-membership probabilities in the input space, and as the dimensionality of the space grows higher, steep boundaries become flatter and therefore distances become smaller. This also explains why the distribution of the interclass distances shifts left when going from Fig. 4.5(B) to Fig. 4.5(E).

The results indicate that the inclusion of the FI metric in the KNN classification process produces a clear outperformance of the Euclidean version for high-dimensional problems, even though they are both supervised methods. Furthermore, the choice of the parameter K for such problems seems unimportant, as shown by the stable accuracy plots in Fig. 4.3.

4.3 Comparison of geodesic calculation methods

This section includes two experiments: First, a collection of real-world datasets are used to compare the geodesic distance calculation methods described in Section 3.3.3, namely the free point approach, the straight path and the graph-based approximations. The second part of this section also compares these methods, this time studying their performance on synthetic datasets of different sample size. In both cases, performance is measured as the magnitude of the distances found (shorter distances are closer to the true geodesic) and the computational cost incurred in the process.

4.3.1 Performance comparison on real-world data

Five different datasets are used in the first experiment: 4 binary class datasets from the UCI Machine Learning Repository [100] and the Cushing's syndrome dataset introduced in Section 4.1; their main features are listed in Table 4.2.

Table 4.2: *Summary of the datasets used in Section 4.3.1.*

Dataset	Ionosphere	Liver Disorders	Sonar	Wisconsin Breast Cancer	Cushing's syndrome
Samples (L)	351	345	208	569	27
Variables (N)	32	6	60	30	2
Classes (J)	2	2	2	2	3

The same process is repeated for each of the datasets: an MLP is trained on the class labels, and the resulting synaptic weights are used to calculate the values of the FI matrix. Then, the matrix is used to compute local distances with Eq. (3.23) in the binary case or Eq. (3.19) directly in the multiclass problem. At this point, three methods

are used to obtain global distances: i) the straight-path or T-point approximation with $T = 100$ intervals in the numeric calculation of the path integral, ii) the graph-based approximation using the Floyd-Warshall algorithm and iii) the free point approach with 10 hyperplanes.

Regarding the free point approach, an initial path between the two points of interest must be defined in order to then apply the method and iteratively reduce its length. Two options are considered: The simplest possibility is to use the straight path connecting the points. Alternatively, one can use the path found by the graph-based approximation, which is by definition at least as short as the straight line between the points.

With the two initialisation possibilities, four different geodesic calculation methods result: the straight path approximation (TP), the graph-based approximation (G) and the free point approach with straight line (FPA) and graph-based (FPG) initialisations.

For each of the methods, a pairwise distance matrix is produced that contains every point-to-point Fisher distance. The values of these matrices are compared and the performance of a given method is evaluated in relation to another by calculating the relative distance reduction that the first algorithm brings over the second. Given two points $\mathbf{x}_A$ and $\mathbf{x}_B$, this reduction is computed as

$$\Delta d_{1-2} = \frac{d_1(\mathbf{x}_A, \mathbf{x}_B) - d_2(\mathbf{x}_A, \mathbf{x}_B)}{d_2(\mathbf{x}_A, \mathbf{x}_B)}, \tag{4.1}$$

where $d_1(\mathbf{x}_A, \mathbf{x}_B)$ is the distance between the points computed using the first method and $d_2(\mathbf{x}_A, \mathbf{x}_B)$ is the reference distance, given by the second algorithm. This measure is averaged over the $L(L-1)/2$ possible pairs, and it is used to measure the improvement that method 1 brings over method 2.

The computational performance of the different approaches is evaluated simply by measuring the time needed by each of them to calculate the pairwise distances. All the experiments in this section were run on an Intel Xeon X5355 2.66 GHz processor.

The left hand side of Table 4.3 evaluates the proposed geodesic estimation methods in terms of the distance reduction they achieve relative to the distances obtained by the alternatives, TP and G. The first two methods compared are FPA and TP. Since FPA initialises gradient descent using straight paths, this comparison is, in effect, assessing the magnitude of the reduction that the optimisation method obtains over the initial straight lines. This figure varies substantially across datasets, going from insignificant values of less than 0.5% to improvements of over 10%.

The next step is to compare the free point approach, again using initially straight lines, with the graph approximation. Although for the Sonar and Wisconsin datasets FPA manages to obtain a very slim improvement, in the other three cases G produces smaller

Table 4.3: *Performance results for the four methods tested: straight path approximation (TP); graph-based approximation (G), which performs TP first and then Floyd's algorithm; free point approach with straight line initialisation (FPA); and free point approach with Floyd's shortest path initialisation (FPG), which uses the shortest paths found by G to initialise the gradient descent optimisation (gd).*

Dataset	Distance variation (%)			Time elapsed (s)			
	FPA/TP	FPA/G	FPG/G	TP	G (TP+Floyd)	FPA	FPG (G+gd)
Ionosphere	-6.01	1.98	-1.18	426	426+0.72	5372	427+8439
Liver	-12.04	20.61	-2.68	366	366+0.75	5855	367+8364
Sonar	-3.16	-0.43	-1.10	132	132+0.15	3029	132+4330
Wisconsin	-0.33	-0.14	-0.18	1012	1012+3.07	8869	1015+14191
Cushing's	-4.33	2.38	-3.18	10	10+0.002	158	10+339

distances, especially for the Liver Disorders dataset.

The third column of this part of the table compares FPG and G. Similarly to the comparison FPA/TP, this evaluates how much distances are reduced by applying gradient descent on the paths found by G. As expected, there is a decrease in the distances for all datasets. It is, however, small in magnitude.

These last two comparisons, FPA/G and FPG/G, show how the free point approach benefits from the use of a better initialisation, in this case given by the Floyd-Warshall algorithm included in G. This highlights the well known limitation that characterises gradient descent methods, that is, the dependency of the solution obtained on the initial conditions.

The amount of time required to run each of the algorithms is shown in the right half of the table. The results are very clear: the use of the iterative approach adds a large computational load to the TP and G methods. FPA is slower than TP by a factor that ranges from around 9 to 23 in the Wisconsin and Sonar datasets respectively. For the graph methods, the difference is even bigger: FPG is 15 to 35 times slower than G in the Wisconsin and Cushing's datasets. This represents an important drawback of the algorithm and, given the marginal accuracy improvement that it provides, restricts its practical use to small datasets or cases where the computational cost is not a concern.

4.3.2 The effect of the sample size

The second experiment studies the effect of the number of data samples on the performance of the geodesic calculation methods. The data used in this example are 2-D synthetic datasets generated with the Gaussian distributions used in Section 4.2. Fig. 4.6 shows the conditional probability $\hat{p}(c_0|\mathbf{x})$ used to derive the Fisher metric. Different

sample sizes are tested to benchmark the performance of the different algorithms as the number of available points reduces.

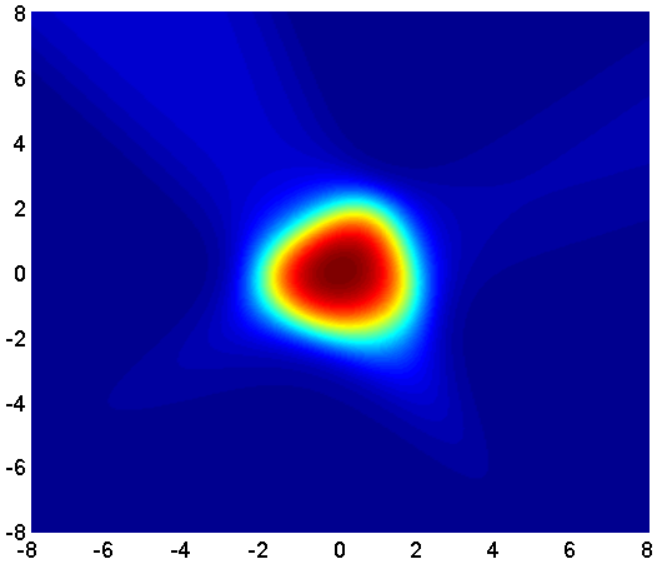


Figure 4.6: *Membership probability of class 0, the narrowest of the two Gaussians. In the plot, red corresponds to values of the probability close to 1 and blue to values close to 0. This surface is estimated using a large dataset containing 5000 samples per class to ensure a good estimation and, in turn, a metric that reflects similarity with respect to the true distributions accurately.*

Table 4.4: *Performance results for different sample sizes. For the smallest datasets, the process is repeated a number of times indicated by the second column and results are averaged to compensate for the higher dependency on the arrangement of the data points.*

Samples	Iterations	Distance variation (%)			
		FPA/TP	**FPA/G**	**FPG/FPA**	**FPG/G**
4	500	-11.3	-9.99	≈0	-10.13
10	50	-10.94	-6.94	-0.30	-7.90
20	20	-11.44	-4.07	-1.30	-6.51
50	5	-11.40	3.21	-4.64	-3.84
100	1	-11.06	9.45	-7.80	-2.71
200	1	-12.52	19.67	-12.24	-2.08
500	1	-11.73	24.49	-13.27	-1.40

Samples	Iterations	Time elapsed (s)			
		TP	**G (TP+Floyd)**	**FPA**	**FPG (G+gd)**
4	500	0.03	0.03+0.001	0.44	0.03+0.55
10	50	0.45	0.45+0.001	3.43	0.45+3.86
20	20	1.10	1.10+0.002	14.4	1.10+16.90
50	5	6.60	6.60+0.004	91.4	6.60+129.50
100	1	27.90	27.90+0.019	403.2	27.9+669.4
200	1	118.90	118.9+0.162	1616	119+3396
500	1	607.80	607.8+2.095	9121	610+24780

For each sample size, a dataset is generated of size ranging from 4 to 500 samples and the corresponding pairwise distance matrix is obtained using the different geodesic ap-

proximation methods. The length of the resulting paths is calculated using an accurate FI metric derived from a large sample of the distributions to ensure that the metric is faithful to the true distribution of the classes. The performance of the algorithms is evaluated in Table 4.4 with the same measures used in Table 4.3.

The first column, FPA/TP, shows how the methods based on straight line paths are not affected by the number of samples of the dataset. FPA improves consistently on TP reducing distances by an average of around 11 percent.

When FPA is compared to G, the influence of the sample size appears. For small datasets, FPA improves on G due to the small number of possible paths from which Floyd's algorithm can choose, resulting in a path quite different from the real geodesic. For that reason, gradient descent is able to reduce the length of that path even when using a far from optimal initialisation. As the sample size increases, the number of possible paths in the graph associated to the dataset increases, and before reaching a large number of samples G already obtains better distances than FPA.

This also explains the third column, FPG/FPA. With a small sample size, FPG and FPA are basically equivalent because most of the time FPG will be using straight lines as initial paths, but as the size of the dataset increases, the quality of the initialisations improve and so do the distances that FPG calculates.

The tendency in FPG/G confirms that, for small datasets, FPG behaves like FPA and therefore produces a significant improvement on G. However, as the number of samples increases, the path that Floyd's algorithm finds approaches a local minimum, reducing the room for FPG to improve on.

Regarding computational costs, the result is similar to what was seen in Section 4.3.1: the free point methods are much more expensive than TP or G, especially for the larger dataset, where using them is only slightly beneficial.

In short, the results in this section indicate that the free point approach is useful to fine tune the calculation of geodesic distances when accuracy is a priority. The additional precision of the distance approximation is achieved at the expense of a substantial increase of the computational cost of the method, which makes it prohibitive for large datasets.

4.4 Fisher networks for real-world data

This is the main section of the chapter, and it illustrates two crucial aspects in the process of constructing Fisher networks. First, Section 4.4.1 details the empirical training procedure of the MLP followed in the framework, describing the parameter selection criteria used and comparing the classification accuracies obtained with other methods

in the literature. The second part, Section 4.4.2, covers the practical details of the graph construction and shows examples of Fisher networks as well as their different representation possibilities.

4.4.1 MLP classification benchmark

The applications in this section focus on real-world datasets, again obtained from the UCI Machine Learning Repository. The six datasets included are listed in Table 4.5.

Table 4.5: *Summary of the datasets used in Section 4.4.*

Dataset	**Ionosphere**	**Liver Disorders**	**Pima Indians Diabetes**	**Sonar**	**Glass Identification**	**Wine**
Samples (L)	351	345	768	208	214	178
Variables (N)	32	6	8	60	9	13
Classes (J)	2	2	2	2	6	3

As discussed earlier, the first step in the Fisher network framework is to estimate the conditional class probabilities. From the two methods suggested to carry out that task, the LLR is straigthforward to obtain and does not require selection of any parameters to obtain the optimal model. However, in the case of the MLP, the default method of choice from the two due to its versatility, there are several parameters that must be fixed in order to produce the final estimator.

There are four parameters involved in the training of the MLP implemented in this framework: the number of nodes in the hidden layer, S; the learning rate of the back-propagation training, α; the number of iterations that the training algorithm runs, n_{iter}; and the control parameter of the weight decay regulariser, γ. To determine their values, the following steps are followed:

1. The dataset (L samples $\times$ N dimensions) is normalised so that the N variables all have mean zero and standard deviation one.

2. From the L labelled samples, 80% are used for the training of the MLP and 20% for validation.

3. Fixing $S = 10$ and $\gamma = 0.01$, α and n_{iter} are tuned to ensure full convergence of the training algorithm, i.e., classification rates in the training and test sets level out.

4. With those values of α and n_{iter}, a grid search is carried out for S and γ. In these experiments, the values tested are $S = \{2, 5, 10, 15, 20\}$ and $\gamma = \{0.001, 0.005, 0.01, 0.05, 0.1\}$. For each of the 25 possible combinations, the MLP is trained 100 times, each of which uses a different weight initialisation and assignment of the

dataset samples to training or validation; both of these factors are determined at random.

5. When the 2500 runs of the training algorithm have finished, the pair $[S,\gamma]$ is selected that provides the highest validation accuracy average over the 100 runs.

6. Finally, the MLP instance is chosen from the 100 runs corresponding to the best $[S,\gamma]$ combination. This can be done, for example, according to validation or global accuracy, this time in terms of individual runs instead of average values.

The parameter selection process favours estimators that generalise well, reducing the presence of overfitting as much as possible. Table 4.6 contains the results of the application of the selection procedure for the datasets considered.

Table 4.6: *Best performing parameters for the MLP estimator from the sets* $S = \{2, 5, 10, 15, 20\}$ *and* $\gamma = \{0.001, 0.005, 0.01, 0.05, 0.1\}$. *Validation and training accuracies are presented as mean values* (μ) *over the 100 runs of the backpropagation algorithm for the optimal set, along with their standard deviation* (σ). *The two rightmost columns show the best validation accuracy from those 100 runs and the corresponding total accuracy for that run (using the whole dataset).*

Dataset	Optimal parameters S	Optimal parameters γ	Validation accuracy (%) ($\mu \pm \sigma$)	Training accuracy (%) ($\mu \pm \sigma$)	Best validation accuracy (%)	Total accuracy (%)
Ionosphere	5	0.1	89.5±3.8	99.1±0.4	97.1	98.3
Liver	15	0.1	72.6±5.0	82.7±1.6	82.6	79.1
Diabetes	2	0.001	77.5±2.9	78.4±0.8	84.4	77.7
Sonar	20	0.01	80.5±5.4	100±0	95.2	99.0
Glass	20	0.01	71.1±6.8	90.6±1.9	88.4	89.7
Wine	10	0.1	98.3±2.0	100±0	100	100

The generalisation power of the MLP predictions is compared to other methods in the literature that use the same data. The results are shown in Table 4.7, and a brief description of the methods involved is provided in Table 4.8. Most of the references used in this comparison give validation accuracies from cross validation, providing a mean and a standard deviation that can be compared directly with the results in Table 4.6. However, some of the references in the benchmark use leave-one-out, others give only a single number or, in some cases, they do not specify whether the accuracies reported are training or validation figures. Ideally, this comparison would include full receiver operating characteristic (ROC) curves instead of a single figure, but they are seldom available in the literature. In any case, Table 4.7 serves as a rough reference of the performance of other published methods when applied to the same data.

The results show that the MLP classifiers used to derive the FI metric are competitive with other methods in the literature. That said, it is important to bear in mind that the Fisher network framework is not locked to this classifier. As long as it provides a

Table 4.7: *Comparison of validation accuracies obtained with the MLP used in the framework and other methods in the literature. Results are given in terms of mean (μ) and standard deviation (σ) when available. Trivial accuracies, provided for reference, are obtained by assigning all samples to the most prevalent class in the dataset.*

Dataset	MLP accuracy (%) ($\mu \pm \sigma$)	Trivial accuracy (%)	Literature accuracies (%) ($\mu \pm \sigma$)		
Ionosphere	89.5±3.8	64.1	87.2 [101] 94.2 [104]	91.1±5.0 [102] 94.7 [105]	94.0±3.3 [103]
Liver	72.6±5.0	58.0	65.5±7.8 [106] 72.5 [107]	70.5±4.2 [102] 76.5±13.4 [103]	70.9 [105]
Diabetes	77.5±2.9	65.1	74.4 [105] 76.9 [101]	75.3±3.9 [108] 78.6±4.5 [102]	75.6 [104]
Sonar	80.5±5.4	53.4	76.0±9.2 [106] 90.4±1.8 [109]	79.0 [101] 90.8±9.1 [103]	81.2±5.1 [102] 92.3 [107]
Glass	71.1±6.8	35.5	70.1±9.2 [108] 75.2 [107]	70.5±8.5 [103]	73.3 [110]
Wine	98.3±2.0	39.9	93.6±5.7 [106] 99.4 [110]	94.4±5.9 [108]	97.7±2.0 [102]

Table 4.8: *Description of the methods included in the comparison in Table 4.7.*

Ref.	Description
[101]	Semi-supervised clustering with genetic algorithm for the optimisation of the objective function.
[102]	NeC4.5: variant of C4.5 which combines traditional decision trees with an ensemble of neural networks used to preprocess the training data.
[103]	Multiobjective learning systems (MOLS) based on genetic algorithms (MOLS-GA) and evolution strategies (MOLS-ES).
[104]	Genetic programming applied to classification trees.
[105]	A-KFD: variant of the kernel Fisher discriminant classifier (KFD) where the kernel used is automatically selected as a linear combination of the elements of a predefined family of heterogeneous kernels.
[106]	Multivariate (oblique) classification trees inferred using the minimum message length (MML) principle.
[107]	WeightedIso: combination of Isomap and KNN classification (the latter is replaced by ADAMENN to produce the second proposed algorithm, Iso+Ada). However, the accuracy result displayed in Table 4.7 comes from the comparison of the presented algorithms with other existing methods; the best performing of them is DANN [42].
[108]	Classification using confidence-based association rule mining.
[109]	Feed-forward neural networks trained with a backpropagation learning algorithm.
[110]	Regularised variation of a K-class support vector classification-regression machine (K-SVCR), an adaptation of the SVM for multiclass assignment.

differentiable expression of the posterior probabilities $\hat{p}(c_i|\mathbf{x})$, any estimator can be used as the basis for the metric.

4.4.2 Construction of Fisher networks

Empirical selection of the network locality parameter

After checking that the MLP produces estimates comparable to those in the literature, the models are used to build Fisher networks. First, pairwise distances are calculated using the graph approximation, producing an $L \times L$ Fisher distance matrix. The Gaussian kernel of Eq. (3.31) is then applied to its elements, resulting in the adjacency matrix that defines the structure of the network.

Section 3.4.1 describes the three measures included in the framework to guide the selection of the width of the Gaussian kernel, σ_G. In this subsection, the measures are analysed for each of the six datasets considered.

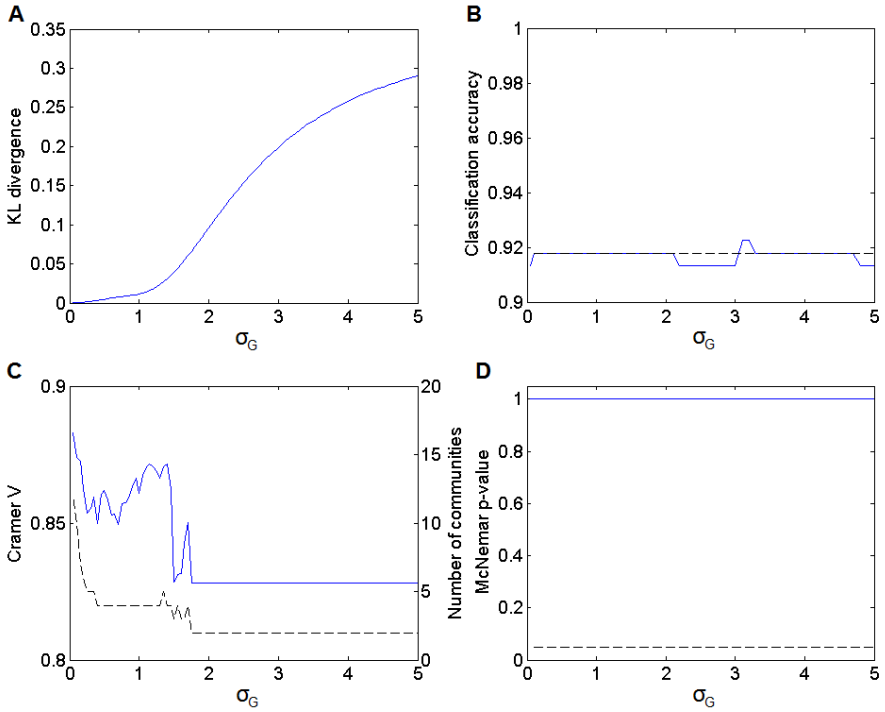


Figure 4.7: *Measures to aid the selection of σ_G, Sonar dataset.* **(A)** *KL divergence.* **(B)** *Classification accuracy (dashed line: MLP accuracy).* **(C)** *Cramer's V index (dashed line: number of communities).* **(D)** *McNemar's test p-value (dashed line: p=0.05).*

Fig. 4.7 shows the three measures for the Sonar dataset and values of σ_G between 0 and 5. Additionally, Fig. 4.7(B) includes the accuracy of the classifications used to calculate McNemar's test (network and MLP predictions), and Fig. 4.7(C) also displays the number of communities found using Newman's algorithm. The first plot shows the KL divergence increasing with σ_G. This is a general tendency for all datasets: small values of σ_G produce networks that predict $\hat{p}(c_i|\mathbf{x})$ better, which is an expected result since a small σ_G means the Gaussian kernel is narrow and therefore only the closest (and

most similar) references will have a significant weight in Eq. (3.32).

The second measure, Cramer's V index, indicates a high level of concordance between the communities and the classes for the whole range of σ_G, with the lowest values of the parameter producing the best results again. Finally, the third measure indicates that there is no statistically significant difference between the performances of the two classifiers, so the p-value by itself does not restrict the range of values of σ_G considered. This is supported by Fig. 4.7(B), which shows that the classification accuracy of the network is very stable and there are only a few minor deviations from that of the MLP. For this dataset, it is the KL divergence and Cramer's V that determine the best range of σ_G values, suggesting $\sigma_G < 1.5$.

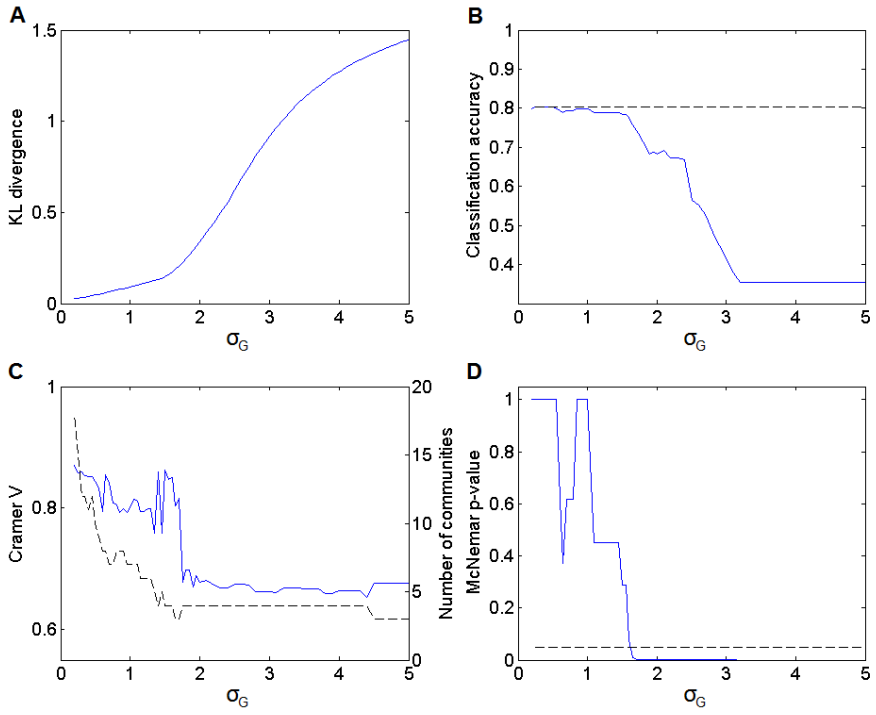


Figure 4.8: *Measures to aid the selection of σ_G, Glass dataset.* **(A)** *KL divergence.* **(B)** *Classification accuracy (dashed line: MLP accuracy).* **(C)** *Cramer's V index (dashed line: number of communities).* **(D)** *McNemar's test p-value (dashed line: p=0.05).*

Fig. 4.8 illustrates the same measures as Fig. 4.7, this time for the Glass dataset. This is a more difficult classification task, as seen in Fig. 4.8(B). The plots of the KL divergence and Cramer's V index are similar to those of the Sonar dataset, with the difference that, whereas in Fig. 4.7(C) the variations of the CV were relatively small, in Fig. 4.8(C) there is a larger difference between its maximum and minimum values. The reason for this is that the Sonar dataset is binary and its classes are relatively easy to separate, thus being well represented with large and small numbers of communities. The Glass dataset, on the other hand, contains six different classes with more mixing, and therefore suffers a bigger CV drop when the number of communities is small, that is, when σ_G is large.

In contrast to the results in Fig. 4.7(D), the p-value in Fig. 4.8(D) drops below the 0.05 threshold at around $\sigma_G = 1.5$, the point at which the classification accuracy of the network starts falling. All three performance measures appear to weaken as σ_G reaches 1.5 so, in practice, this would recommend $\sigma_G < 1.5$.

Figs. 4.9–4.12 contain the plots of the ancillary measures for the remaining datasets. The Ionosphere dataset figures are similar to the Sonar case, with the McNemar test also not providing any discriminating information. The KL divergence and Cramer's V suggest $\sigma_G < 1$, although, like before, the variations of Cramer's V are very small and large values of σ_G would not be detrimental in terms of classification accuracy.

The Liver dataset plots are not too far from the Sonar case either. Although Figs. 4.10(B) and 4.10(D) show some decrease in performance for large values of σ_G, it is a mild effect, and it is again the left hand side plots that help choose the threshold, $\sigma_G < 1$.

Fig. 4.11, corresponding to the Diabetes dataset, displays a considerable performance deterioration, with Figs. 4.11(B) and 4.11(D) indicating an irregular behaviour for $\sigma_G > 1$, where the classification rates of the network start to diverge from the MLP figure. This, along with the decrease of Cramer's V index also around that point, suggests selecting a value of the parameter below that threshold. Finally, for the Wine dataset, the most useful plots to determine σ_G are Figs. 4.12(B) and 4.12(D), which indicate that $\sigma_G < 2$ is to be preferred in order for the network to maintain the prediction power of the MLP.

These plots provide the information required to find suitable values of σ_G depending on the user preferences. For instance, in the experiments in the next subsection, an upper threshold is determined as the smallest σ_G for which the p-value is less than or equal to 0.05. Then, σ_G is simply selected as the smallest value in that range, since that will in general produce the best predictions (the smallest KL divergence) and a good CV value, if not the maximum available. This can serve as a sensible default selection procedure when there is not a given preference.

However, depending on the application, the user may prefer to focus solely on the coherence of the communities and select the top CV value regardless of the rest of the measures. Or maybe the interest is not to obtain too many (or too few) communities because large (or small) clusters are desired. Then, only values of σ_G that produce such result are considered. Furthermore, because any measure can be incorporated to the analysis, the selection process can be tailored to suit the needs of the user.

Table 4.9: *Upper thresholds and selected values of the parameter σ_G for the six datasets.*

Dataset	**Ionosphere**	**Liver Disorders**	**Pima Indians Diabetes**	**Sonar**	**Glass Identification**	**Wine**
Threshold	$\sigma_G < 1$	$\sigma_G < 1$	$\sigma_G < 1$	$\sigma_G < 1.5$	$\sigma_G < 1.5$	$\sigma_G < 2$
Selected value	0.10	0.15	0.05	0.05	0.20	0.10

Table 4.9 summarises the upper bounds of the range of accepted values of σ_G. In this case, σ_G is selected as the smallest value in the interval that does not cause numerical instabilities in Newman's community detection algorithm. Although this method always provides the same result by definition, when applied in practice issues can appear during the eigendecomposition calculation if the adjacency matrix contains weights that are very small compared to the largest values in the matrix (stability problems appear for components of the matrix $> 10^{100}$ times smaller than the largest ones). This happens when σ_G is very small, and it results in inaccurate eigenvectors and slightly different communities being found in repeated runs of the algorithm. To prevent this computational artifact, σ_G is selected large enough so that such extreme differences in the magnitude of the weights do not appear. This lower threshold value depends on the particular case, but it is easy to find a value of σ_G that approximates it by doing a quick line search.

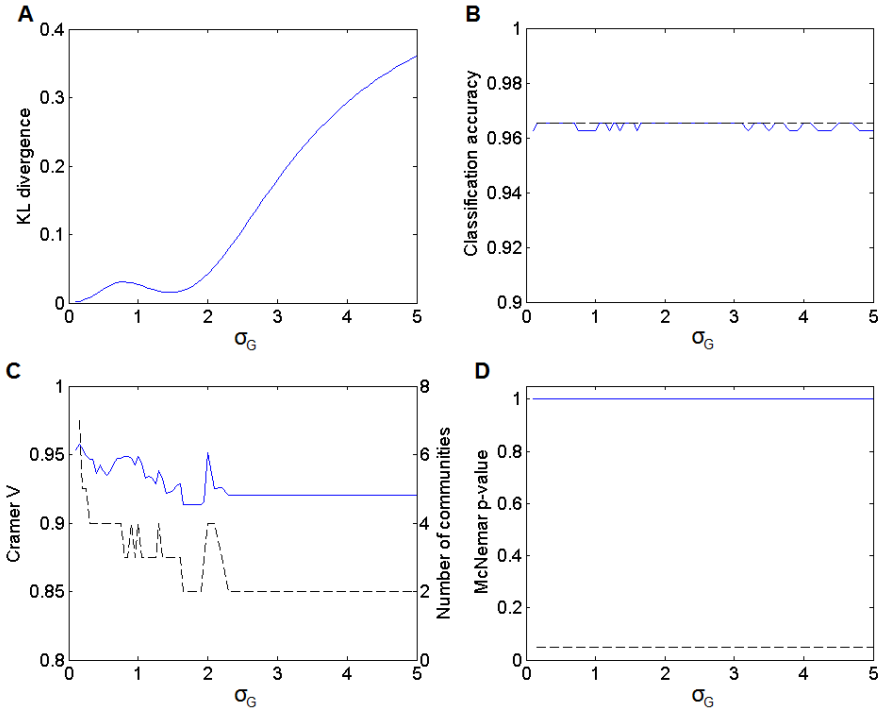


Figure 4.9: *Measures to aid the selection of σ_G, Ionosphere dataset.* **(A)** *KL divergence.* **(B)** *Classification accuracy (dashed line: MLP accuracy).* **(C)** *Cramer's V index (dashed line: number of communities).* **(D)** *McNemar's test p-value (dashed line: p=0.05).*

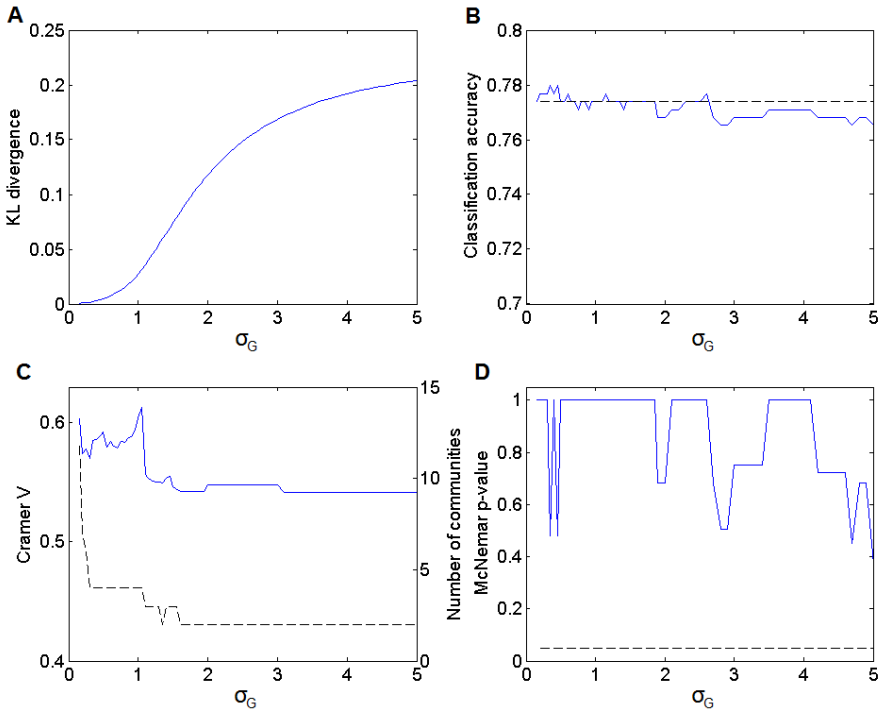


Figure 4.10: *Measures to aid the selection of σ_G, Liver dataset.* **(A)** *KL divergence.* **(B)** *Classification accuracy (dashed line: MLP accuracy).* **(C)** *Cramer's V index (dashed line: number of communities).* **(D)** *McNemar's test p-value (dashed line: p=0.05).*

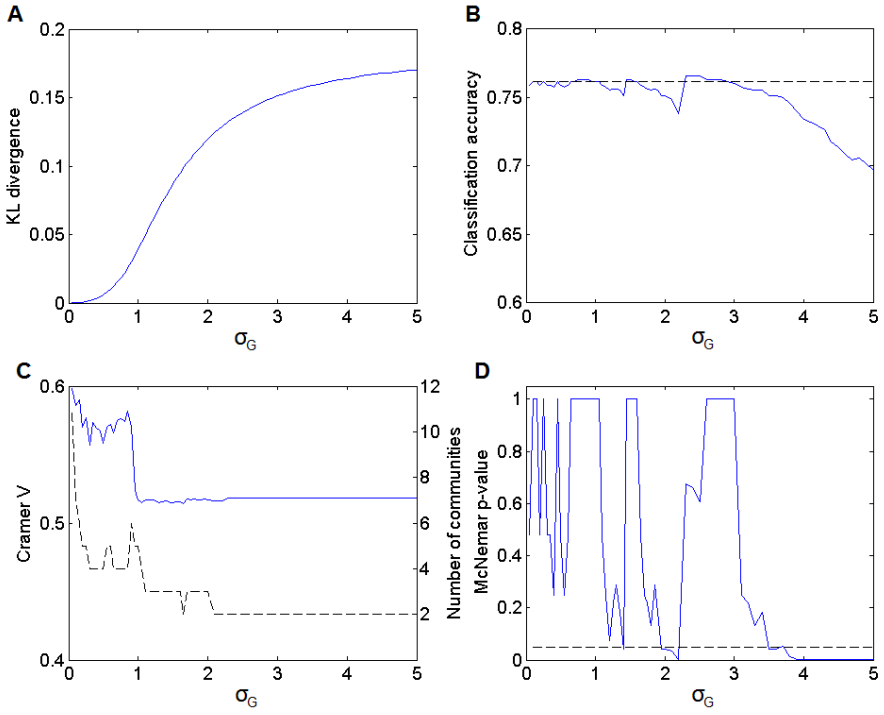


Figure 4.11: *Measures to aid the selection of σ_G, Diabetes dataset.* **(A)** *KL divergence.* **(B)** *Classification accuracy (dashed line: MLP accuracy).* **(C)** *Cramer's V index (dashed line: number of communities).* **(D)** *McNemar's test p-value (dashed line: p=0.05).*

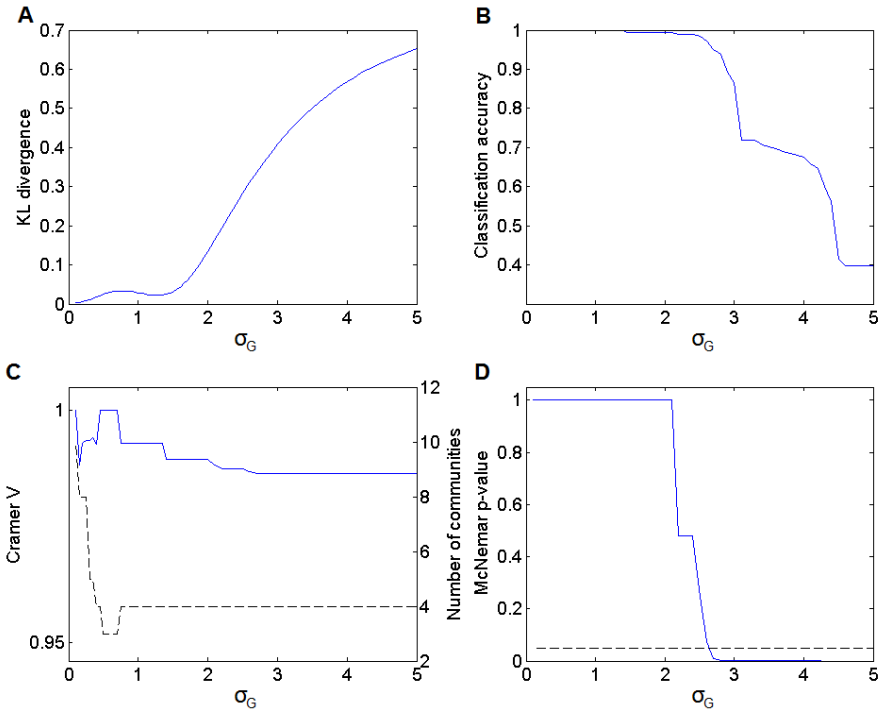


Figure 4.12: *Measures to aid the selection of σ_G, Wine dataset.* **(A)** *KL divergence.* **(B)** *Classification accuracy (dashed line: MLP accuracy).* **(C)** *Cramer's V index (dashed line: number of communities).* **(D)** *McNemar's test p-value (dashed line: p=0.05).*

Fisher network examples

This subsection presents a collection of Fisher networks built from the six datasets using the values of σ_G in Table 4.9. Additionally, these networks are compared to their Euclidean counterparts, which are graphs built following the same process, with the exception that the distances used to calculate the adjacency matrix of the graph are Euclidean. For all networks in this chapter, the arrangement of the nodes is given by a Sammon projection of the original dataset using Fisher or Euclidean distances, as discussed in Section 3.6.1. Axis values are not displayed in any of the Sammon mappings in Figs. 4.13–4.41 for the sake of clarity, since only the relative position of the points with respect to each other is relevant. In the representations, edges are displayed only between members of the same community to highlight cluster membership.

For each of the networks in Figs. 4.13–4.25, the set of communities found by Newman's algorithm is compared against the most widely-used clustering algorithm: k-means. After finding the communities for a particular case, k-means clustering is performed on the same dataset with k equal to the number of communities found by Newman's method. Then, both partitions of the data are compared against the true class labels using Cramer's V index with the same purpose as in the previous subsection: finding the degree of concordance between the partitions and the classes. This process is repeated 100 times for k-means and the maximum and mean values of the index are reported.

For those cases where the network is built using Fisher distances, the corresponding k-means is applied on the 2-D Sammon projection of the data. If, on the other hand, the construction of the network uses Euclidean distances, k-means is applied on the original dataset. This is to make the comparison fair in terms of the use of the class label information.

Fig. 4.13 is an example of a set of communities using the default σ_G selection procedure of the framework and the Sonar dataset. By design, the default criterion chooses a value of σ_G as small as possible, which in this case results in 12 communities that are quite compact, with little class mixing within the groups, as reflected by the high value of Cramer's V.

From a practical point of view, the first contribution of the Fisher network is that it provides a global view of the data which, thanks to the use of the Fisher metric, displays a meaningful structure that implicitly informs about the underlying class probability functions. The arrangement of the nodes makes it very easy to identify which areas of the network are clear in terms of class membership, located in both ends of the C shape; and which correspond to borderline cases, in the middle section of the curve. For the end user, this means that they can quickly get an idea of where their points of interest stand within the dataset in terms of the classification problem just by locating them in the plot,

without having to inspect the, in this case, 60 covariate values. Secondly, the division of the network into communities produces a stratification of the data, again according to the classification task at hand. This is interesting from the end user perspective because, given a case of interest, it provides a set of similar instances that can then be analysed in order to better understand the initial node.

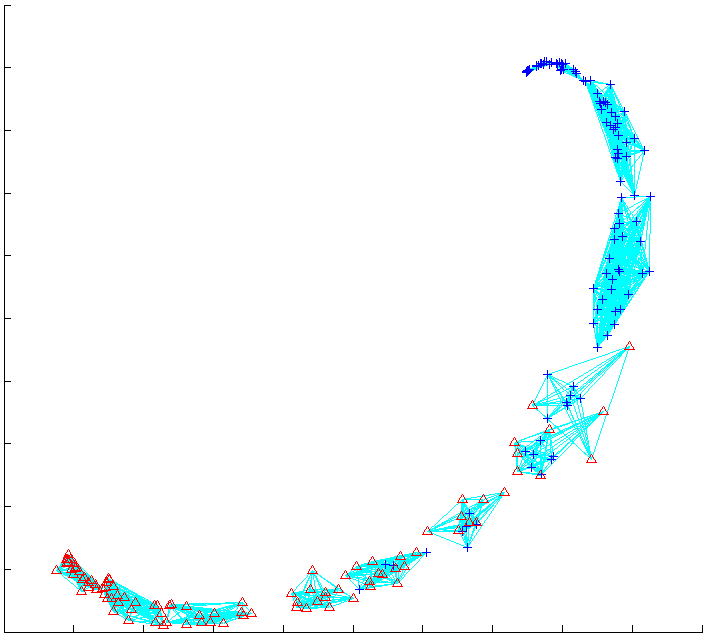


Figure 4.13: *Fisher network representation of the Sonar dataset,* $\sigma_G = 0.05$*. Twelve communities, 0 singletons, KL divergence:* $\varphi_{KL} = 0.0005$*. Cramer's V indices: Network communities:* $\varphi_{CV} = 0.8830$*. k-means clusters:* $\varphi_{CV} = 0.8747$ *(mean),* 0.8829 *(max).*

Fig. 4.14 replicates Fig. 4.13 with $\sigma_G = 0.20$. As one would expect, the outcome is the identification of fewer communities of larger size. Consequently, there are now only two communities covering the border area, with more class mixing within them than there was in the previous case, therefore producing a smaller concordance index. The coarser network granularity that this value of the locality parameter produces also means that the user receives more similar cases for each node. This exemplifies the kind of variation that can be obtained by tuning σ_G, which can help the user decide which value to choose from the suggested range.

The next graph, Fig. 4.15, contains a network built from the same dataset using Euclidean distances. The value of σ_G is selected, like in the Fisher case, as the smallest that provides stability of the eigenvector calculations. The reason why this value is bigger in general for Euclidean networks has to do with Euclidean distances reaching much larger values than Fisher ones (see Figs. 4.4 and 4.5 for an example), which means smaller weights and a stronger tendency for numerical issues to appear, as the previous subsection explained.

Going back to the figure, the communities found in this network are nowhere near their Fisher counterparts in terms of clarity of their structure. This is reflected not only in the visualisation of the clusters, but also in the KL divergence of the $\hat{p}(c_i|\mathbf{x})$ estimates, several orders of magnitude higher than the values obtained using the Fisher metric; and in Cramer's V index, which also deteriorates substantially.

Regarding the comparison between Newman's algorithm and k-means clustering, the φ_{CV} values indicate that they produce very similar partitions of the data in terms of coherence with the original class labels. For the six datasets studied, the values of this measure produced by the Fisher networks are very close to the mean φ_{CV} obtained by k-means, with the maximum value usually less than 0.01 above (with the exception of the Liver dataset, where the difference is slightly bigger). However, there are two significant advantages in the use of the network approach over k-means: the number of partitions is selected automatically and there are not any initialisation issues.

When using Euclidean distances, there is a clearly noticeable difference in performance regarding φ_{CV} in favour of k-means, as well as a larger divergence between the maximal and mean values. While the latter could be suggesting that the concordance of clusters and classes is more sensible to the initialisation of the centroids now that the metric does not draw similar points together, it is not clear why the two methods should differ from each other anymore than they do when using Fisher distances. This does not represent a concern, though, since the usefulness of the framework, as well a clearly improved performance, comes from the use of the FI metric.

Figs. 4.16–4.25 correspond to Fisher and Euclidean networks constructed from the remaining datasets. The results are consistent with the discussion above for the Sonar dataset: Fisher networks provide a clear visualisation of the data, with a tidier community structure and better probability predictions than in the Euclidean case, and they do so through an intuitive case-based approach.

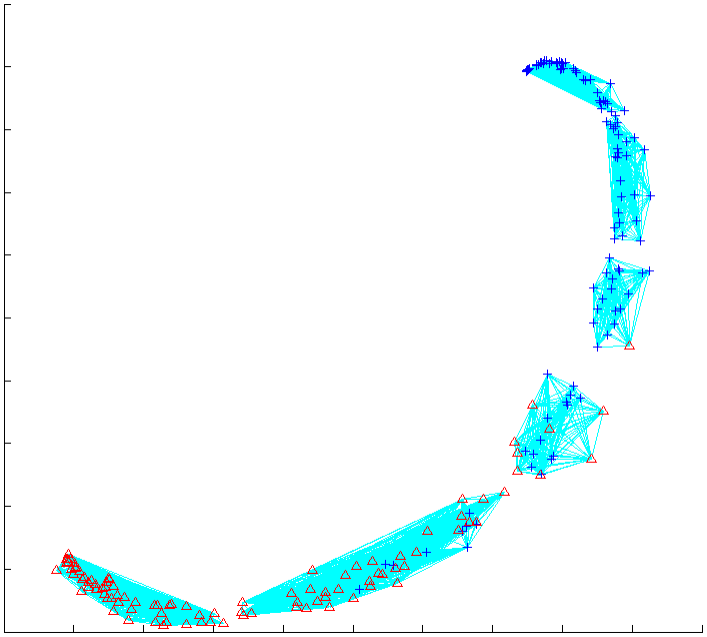


Figure 4.14: *Fisher network representation of the Sonar dataset,* $\sigma_G = 0.20$*. Six communities, 0 singletons, KL divergence:* $\varphi_{KL} = 0.001$*. Cramer's V indices: Network communities:* $\varphi_{CV} = 0.8627$*. k-means clusters:* $\varphi_{CV} = 0.8617$ *(mean),* 0.8730 *(max).*

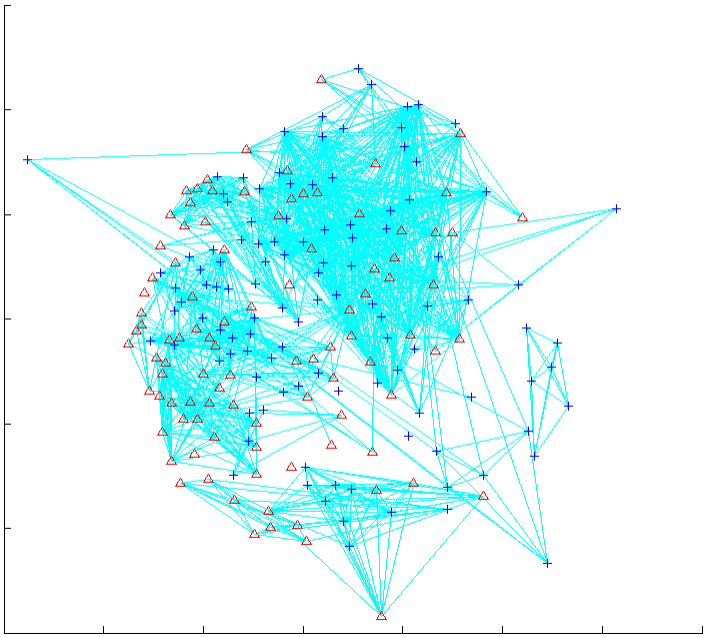


Figure 4.15: *Euclidean network representation of the Sonar dataset,* $\sigma_G = 2.30$*. Twenty-two communities, 7 singletons, KL divergence:* $\varphi_{KL} = 0.0922$*. Cramer's V indices: Network communities:* $\varphi_{CV} = 0.6235$*. k-means clusters:* $\varphi_{CV} = 0.7581$ *(mean),* 0.8261 *(max).*

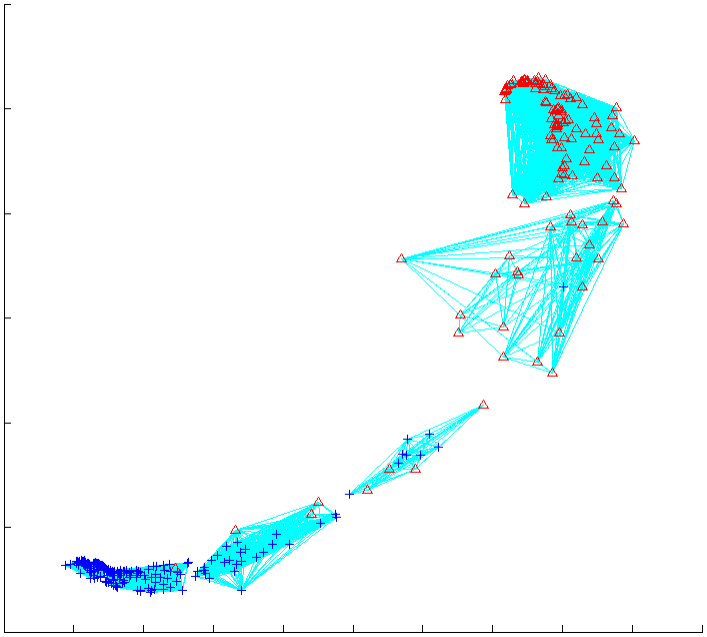


Figure 4.16: *Fisher network representation of the Ionosphere dataset,* $\sigma_G = 0.10$.
Seven communities, 0 singletons, KL divergence: $\varphi_{KL} = 0.0015$.
Cramer's V indices: Network communities: $\varphi_{CV} = 0.9538$.
k-means clusters: $\varphi_{CV} = 0.9565$ *(mean),* 0.9579 *(max).*

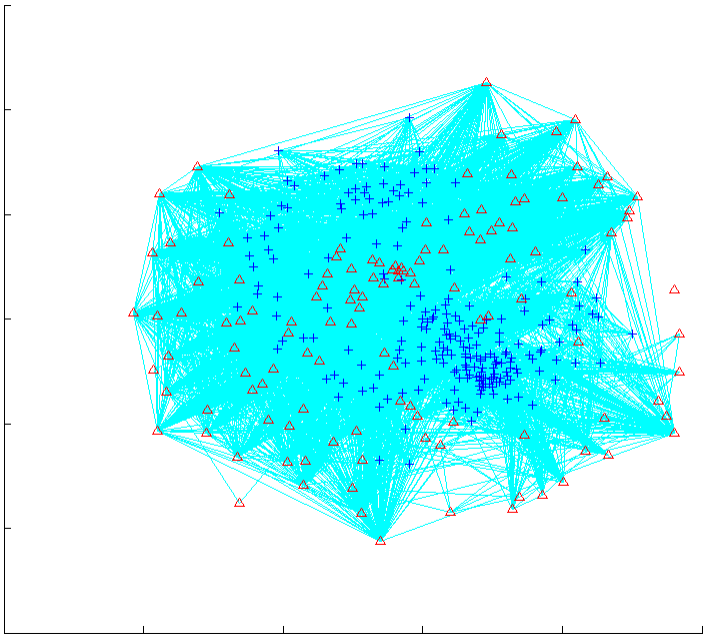


Figure 4.17: *Euclidean network representation of the Ionosphere dataset,* $\sigma_G = 1.80$.
Ten communities, 2 singletons, KL divergence: $\varphi_{KL} = 0.2282$.
Cramer's V indices: Network communities: $\varphi_{CV} = 0.7591$.
k-means clusters: $\varphi_{CV} = 0.7659$ *(mean),* 0.8287 *(max).*

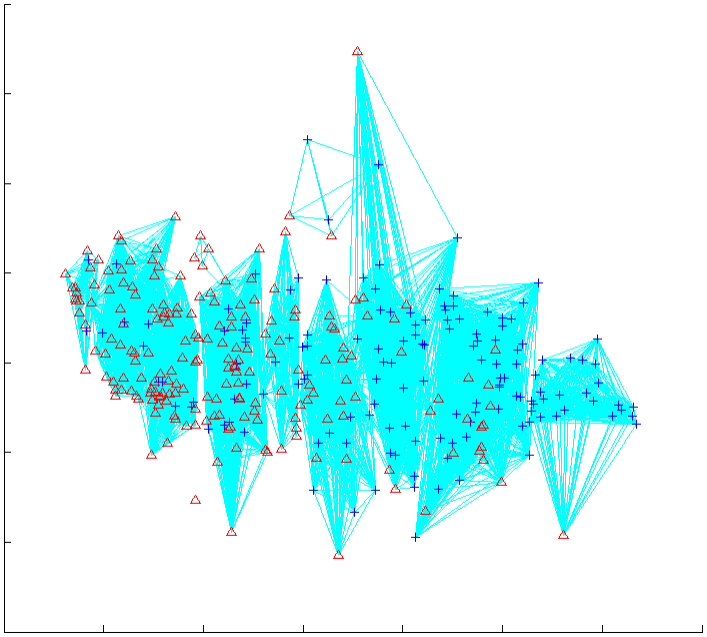


Figure 4.18: *Fisher network representation of the Liver dataset,* $\sigma_G = 0.15$*. Eleven communities, 1 singletons, KL divergence:* $\varphi_{KL} = 0.0008$*. Cramer's V indices: Network communities:* $\varphi_{CV} = 0.6033$*. k-means clusters:* $\varphi_{CV} = 0.6113$ *(mean),* 0.6404 *(max).*

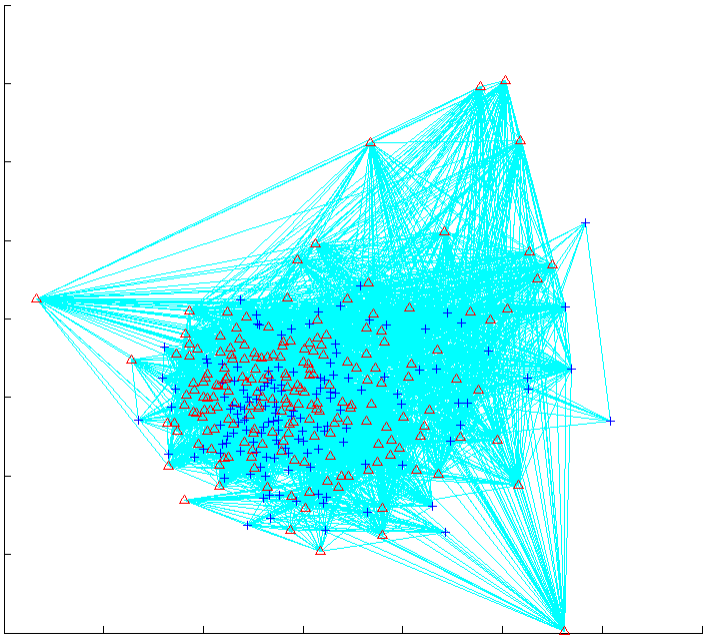


Figure 4.19: *Euclidean network representation of the Liver dataset,* $\sigma_G = 0.75$*. Twelve communities, 0 singletons, KL divergence:* $\varphi_{KL} = 0.1300$*. Cramer's V indices: Network communities:* $\varphi_{CV} = 0.2292$*. k-means clusters:* $\varphi_{CV} = 0.2626$ *(mean),* 0.3091 *(max).*

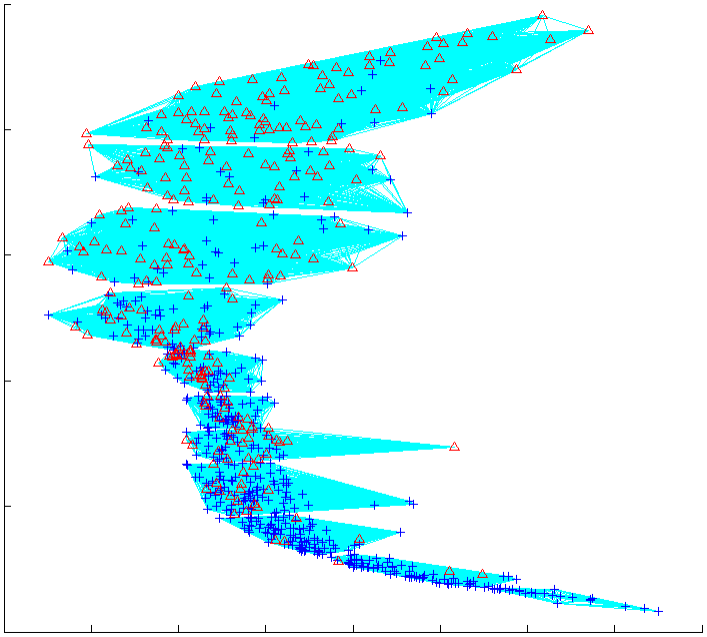


Figure 4.20: *Fisher network representation of the Diabetes dataset,* $\sigma_G = 0.05$*. Eleven communities, 0 singletons, KL divergence:* $\varphi_{KL} = 0.00004$*. Cramer's V indices: Network communities:* $\varphi_{CV} = 0.5981$*. k-means clusters:* $\varphi_{CV} = 0.5924$ *(mean),* 0.6027 *(max).*

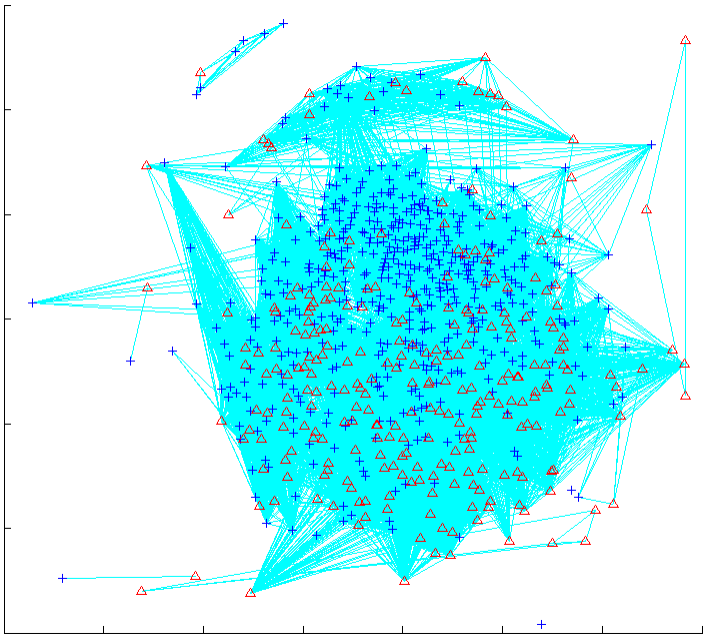


Figure 4.21: *Euclidean network representation of the Diabetes dataset,* $\sigma_G = 0.75$*. Twenty-three, 4 singletons, KL divergence:* $\varphi_{KL} = 0.016$*. Cramer's V indices: Network communities:* $\varphi_{CV} = 0.4464$*. k-means clusters:* $\varphi_{CV} = 0.5157$ *(mean),* 0.5417 *(max).*

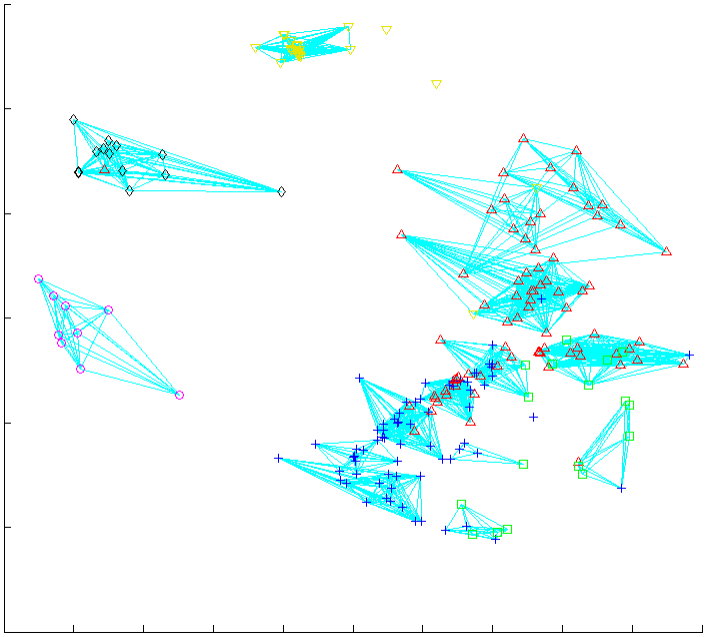


Figure 4.22: *Fisher network representation of the Glass dataset,* $\sigma_G = 0.20$*. Fifteen communities, 3 singletons, KL divergence:* $\varphi_{KL} = 0.0281$*. Cramer's V indices: Network communities:* $\varphi_{CV} = 0.8712$*. k-means clusters:* $\varphi_{CV} = 0.8529$ *(mean),* 0.8795 *(max).*

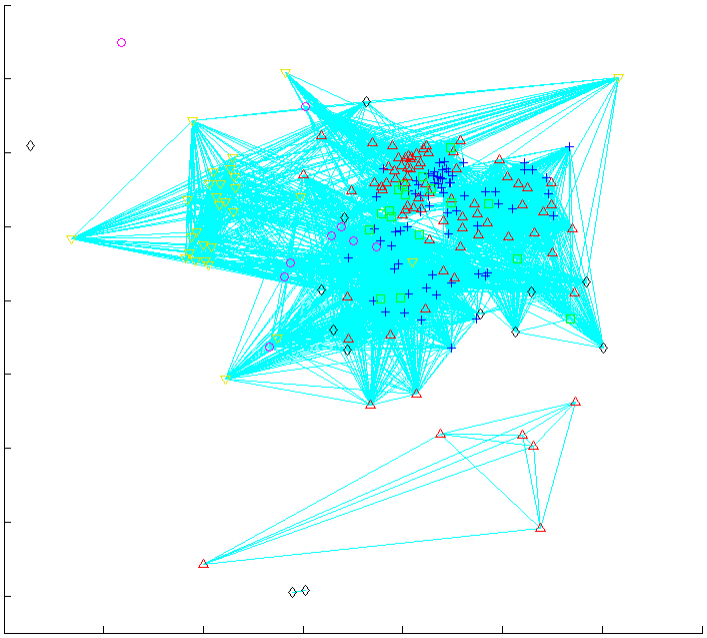


Figure 4.23: *Euclidean network representation of the Glass dataset,* $\sigma_G = 1.40$*. Seven communities, 2 singletons, KL divergence:* $\varphi_{KL} = 0.5311$*. Cramer's V indices: Network communities:* $\varphi_{CV} = 0.4915$*. k-means clusters:* $\varphi_{CV} = 0.6217$ *(mean),* 0.5574 *(max).*

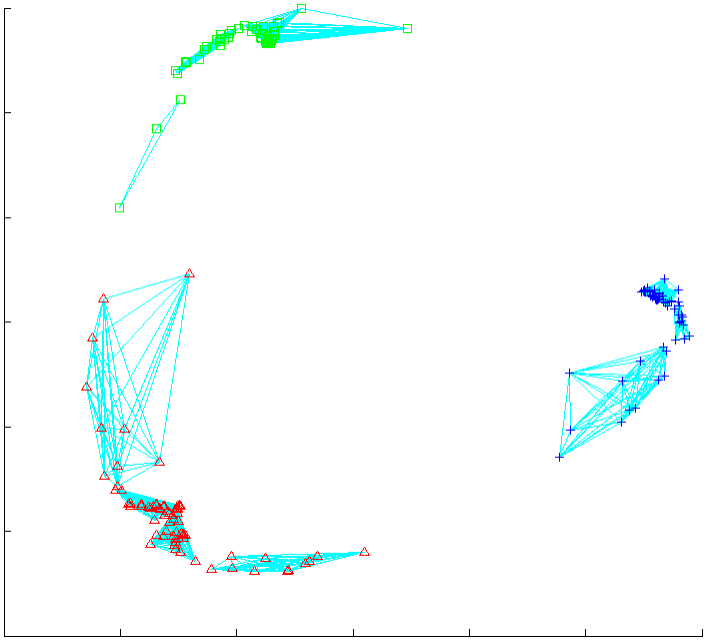


Figure 4.24: *Fisher network representation of the Wine dataset,* $\sigma_G = 0.10$*.*
Ten communities, 0 singletons, KL divergence: $\varphi_{KL} = 0.0031$*.*
Cramer's V indices: Network communities: $\varphi_{CV} = 1$*.*
k-means clusters: $\varphi_{CV} = 0.9940$ *(mean),* 1 *(max).*

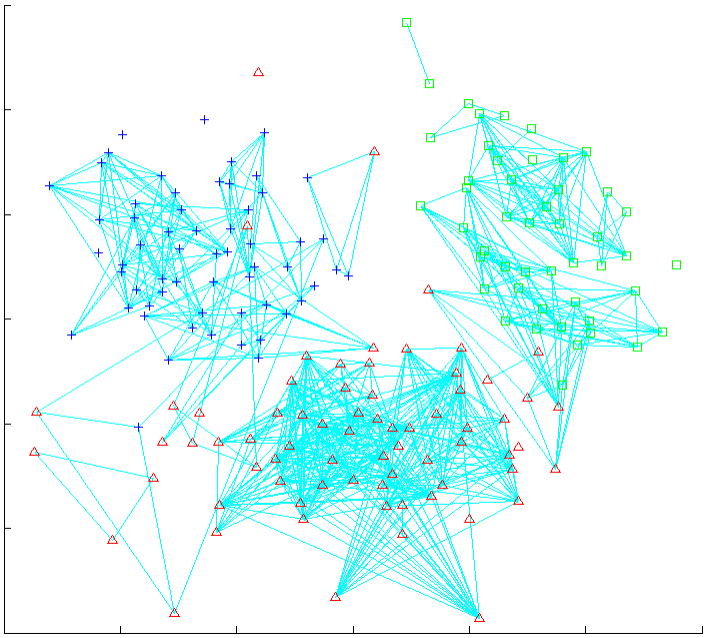


Figure 4.25: *Euclidean network representation of the Wine dataset,* $\sigma_G = 0.70$*.*
Twenty communities, 9 singletons, KL divergence: $\varphi_{KL} = 0.0670$*.*
Cramer's V indices: Network communities: $\varphi_{CV} = 0.9493$*.*
k-means clusters: $\varphi_{CV} = 0.9624$ *(mean),* 0.9840 *(max).*

Fisher networks with informative backgrounds

Figs. 4.13–4.25 show examples of Fisher network representations of the six datasets studied. Those are, however, only basic versions of the networks that the framework can produce. In this subsection, the informative backgrounds introduced in Section 3.6.1 are put into practice, replacing the plain white background of the standard graphs with a coloured image that graphically informs of the class membership probabilities of nodes and regions of the network.

Fig. 4.26 contains the same network in Fig. 4.13 with a coloured background. The first class (+) is represented by black, and white corresponds to the second (△). The distance-weighted contributions of the nodes combined produce the different tones of the background, visually encoding their class membership probabilities. This provides the user with an intuitive colour-coded measure of the class profile of the 2-D space, which they can use to quickly infer which class is predominant in the different areas of the representation, as well as how this predominance shifts when moving around the plot. In this case, the colour of the pixels goes from black in the top right corner to white in the bottom left along the C shape formed by the data, reflecting the local proportion of members from each class.

Regarding the colour transition in this figure, it is noticeable how sharp the changes in the tone are. This is caused by the small value of σ_G used: for low values of the parameter, the Gaussian kernel in Eq. (3.41) produces weights that focus mostly on the closest node, almost ignoring the rest. Let us look at a simple example to illustrate this: suppose the algorithm is going to calculate the colour of a certain pixel, and that the two closest nodes to that pixel in the network lie at a small distance. Let us define this *small distance* as, for example, the first decile of all pairwise distances, which for this particular dataset is 0.15. Back to the pixel, suppose the closest node is 0.15 apart and the second closest is slightly further away, say 0.165 (a 10% longer distance). For $\sigma_G = 0.05$, the weight corresponding to the first node is $0.12 \cdot 10^{-3}$, while the second node gets $0.02 \cdot 10^{-3}$, a contribution six times smaller, even though the distance difference is relatively small. This effect is stressed even more for longer distances, i.e., for pixels that are more distant from their closest node. The consequence is that, except for areas that are very close to a group of nodes, pixel colour is determined only by the closest node, hence the sharp variations.

It is not difficult, however, to obtain smoother backgrounds. Simply by selecting a bigger σ_G, the effect described above is alleviated. For example, if $\sigma_G = 0.20$, well below the threshold given by Table 4.9, the weights obtained in the example are 0.57 and 0.51. Slightly further away points are now taken more into consideration, which results in a more fluid change of the variation of the background colour. Fig. 4.27 represents the Fisher network of the Sonar dataset using this larger value of σ_G. The smoother colour

transition gives a sense of continuity in the 2-D representation and is also aesthetically more pleasing to the eye.

Figs. 4.28 and 4.29 display coloured Fisher networks for the Wine data. This is a three-class dataset, so more than two colours are needed to represent all classes. Red, green and blue are selected for this example. Like with the Sonar dataset, the default σ_G is small, and there are abrupt changes in the colours. Again, choosing a larger value of σ_G produces a softer and less distracting background texture that gives an overall nicer representation. The smoothness of the background is therefore another factor that can be taken into account when selecting the parameter from the range of values recommended by the framework. Figs. 4.30–4.33 contain examples for the rest of the datasets.

As discussed in Section 3.6.1, there is an alternative for the way the background displays the class memberhip probability information. For problems with many classes, where assigning a different colour to each of them could be confusing, or simply where a greyscale representation is desired, pixels can be coloured from white through black depending on the difference between the two largest $\hat{p}(c_i|\mathbf{x})$ values at that point. Although there are not any class-specific colours in the background, it still provides a useful visual indicator of the uncertainty of class predictions for the different regions of the network. Figs. 4.34–4.35 are examples of this colouring method for the two multiclass datasets studied in this section.

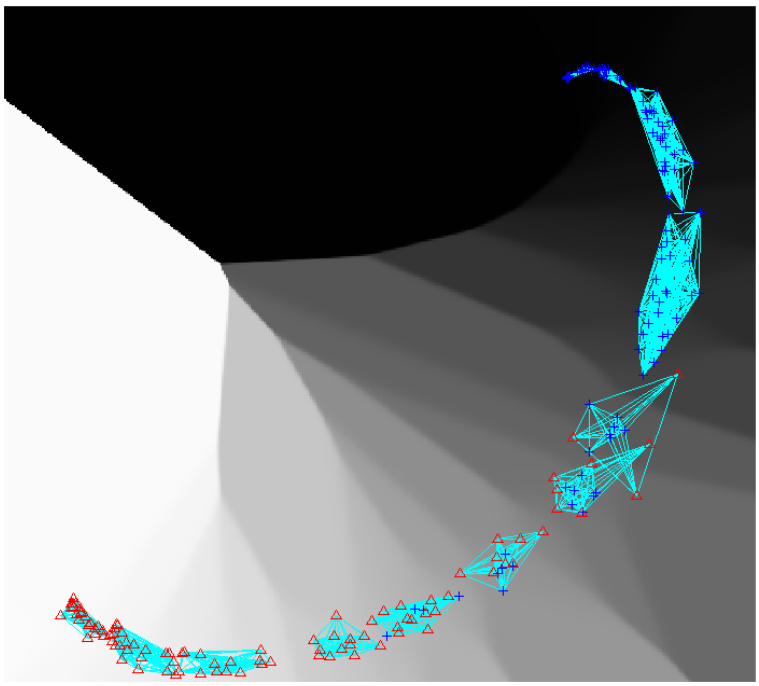


Figure 4.26: *Fisher network representation of the Sonar dataset with informative background,* $\sigma_G = 0.05$*. Black pixels represent high class + probabilities; white pixels correspond to large class* $\triangle$ *values.*

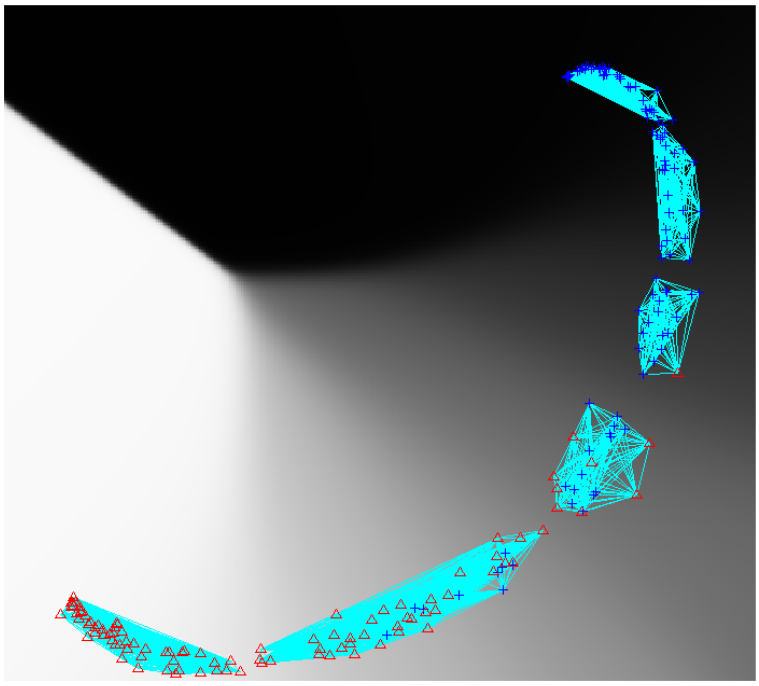


Figure 4.27: *Fisher network representation of the Sonar dataset with informative background,* $\sigma_G = 0.20$*. Black pixels represent high class + probabilities; white pixels correspond to large class* $\triangle$ *values.*

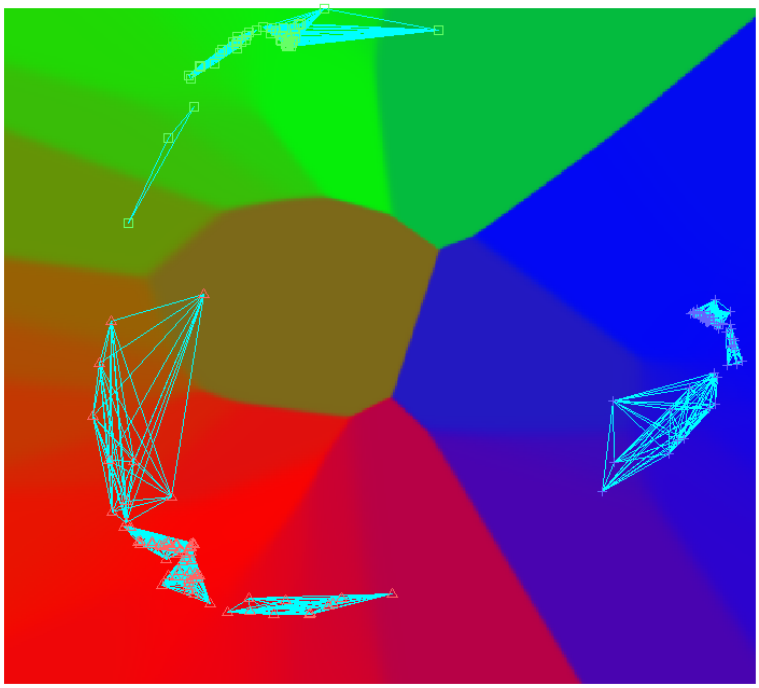


Figure 4.28: *Fisher network representation of the Wine dataset with informative background,* $\sigma_G = 0.10$. *Class* $\triangle$ *is represented by red, class* $+$ *by blue and class* $\square$ *by green.*

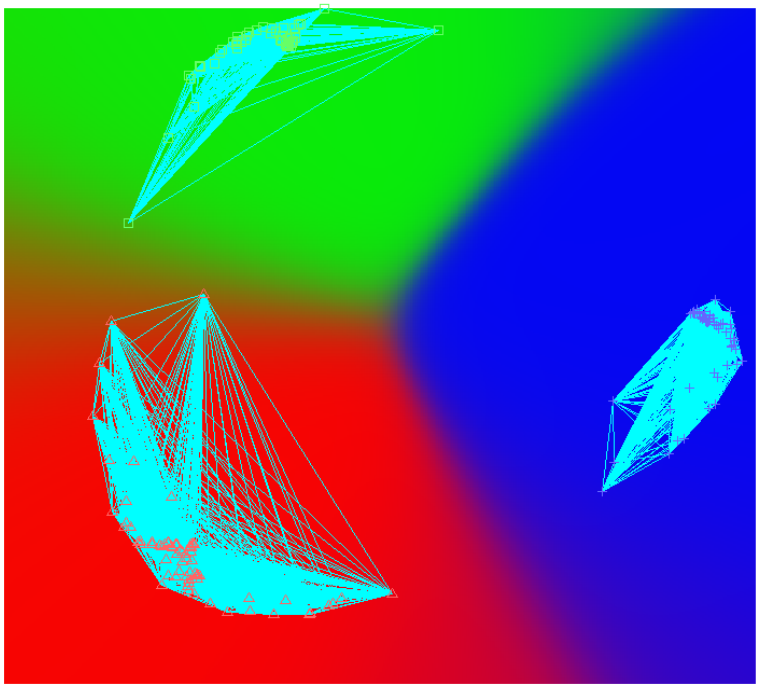


Figure 4.29: *Fisher network representation of the Wine dataset with informative background,* $\sigma_G = 0.50$. *Class* $\triangle$ *is represented by red, class* $+$ *by blue and class* $\square$ *by green.*

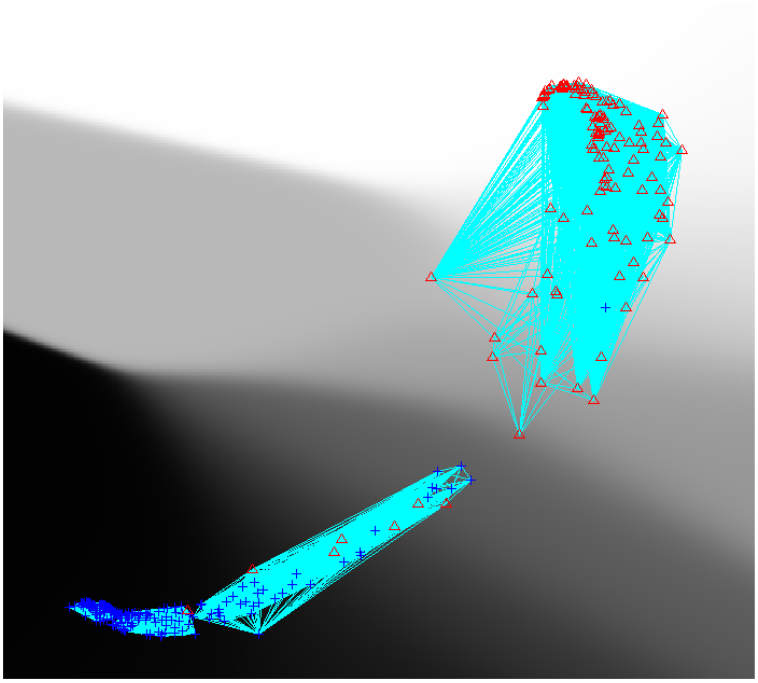


Figure 4.30: *Fisher network representation of the Ionosphere dataset with informative background, $\sigma_G = 0.20$. Black pixels represent high class + probabilities; white pixels correspond to large class $\triangle$ values.*

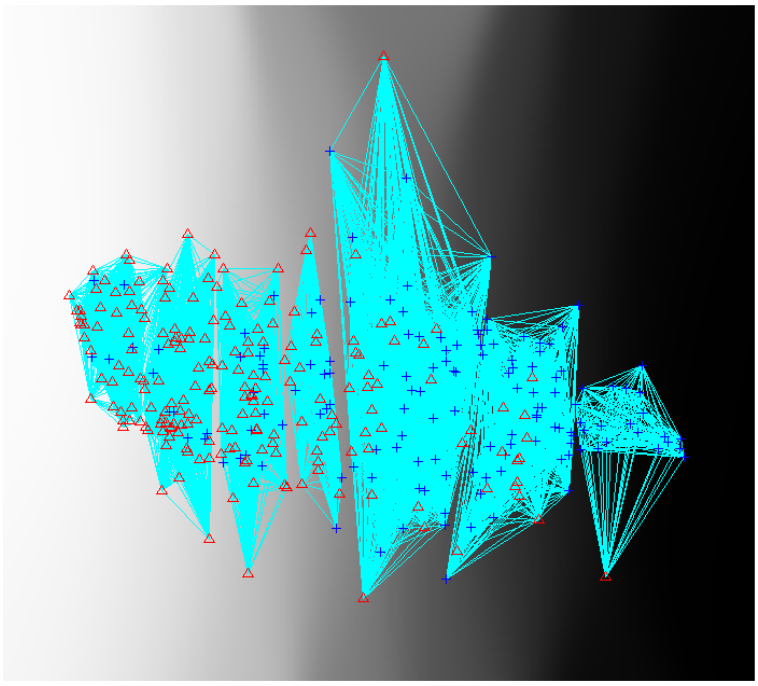


Figure 4.31: *Fisher network representation of the Liver dataset with informative background, $\sigma_G = 0.20$. Black pixels represent high class + probabilities; white pixels correspond to large class $\triangle$ values.*

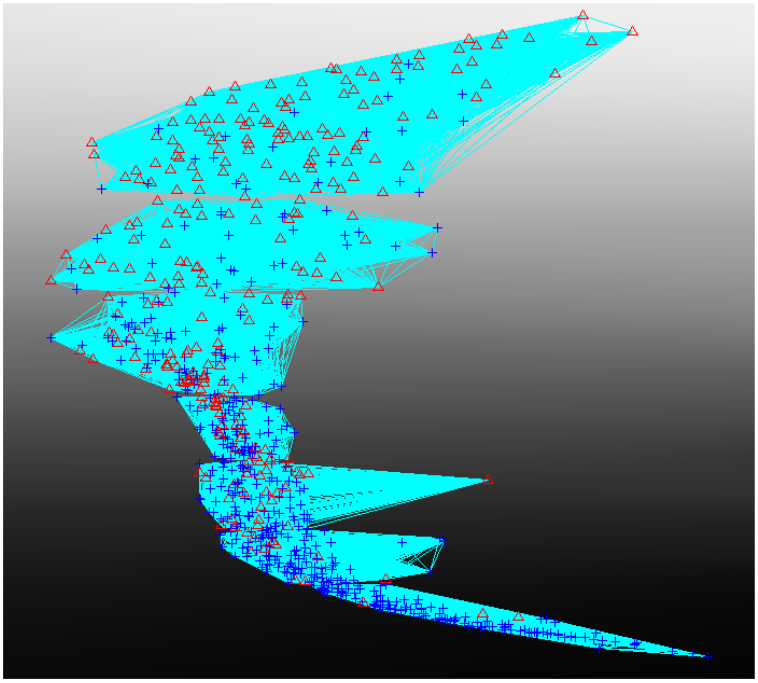


Figure 4.32: *Fisher network representation of the Diabetes dataset with informative background,* $\sigma_G = 0.10$. *Black pixels represent high class* $+$ *probabilities; white pixels correspond to large class* $\triangle$ *values.*

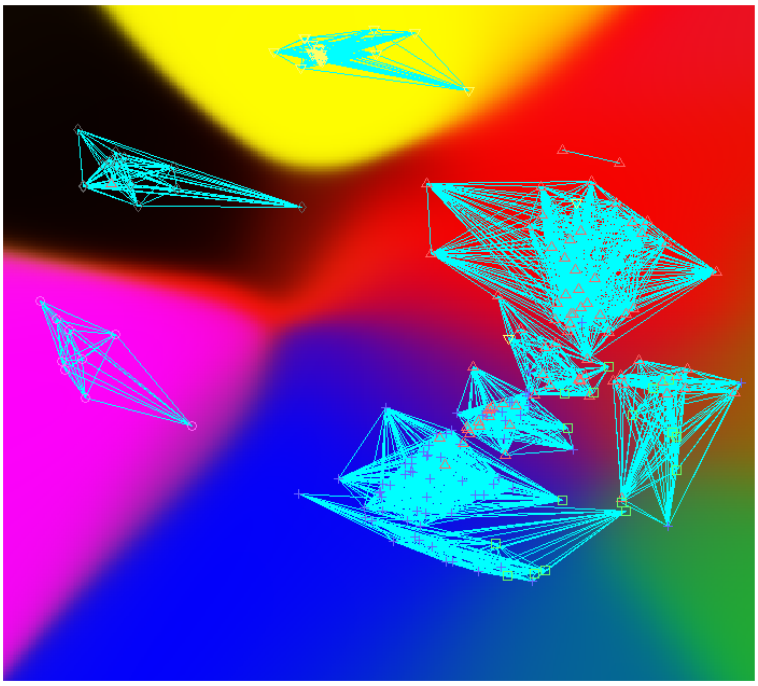


Figure 4.33: *Fisher network representation of the Glass dataset with informative background,* $\sigma_G = 0.50$. *Class* $\triangle$ *is represented by red, class* $+$ *by blue, class* $\square$ *by green, class* $\Diamond$ *by black, class* $\bigcirc$ *by pink and class* $\bigtriangledown$ *by yellow.*

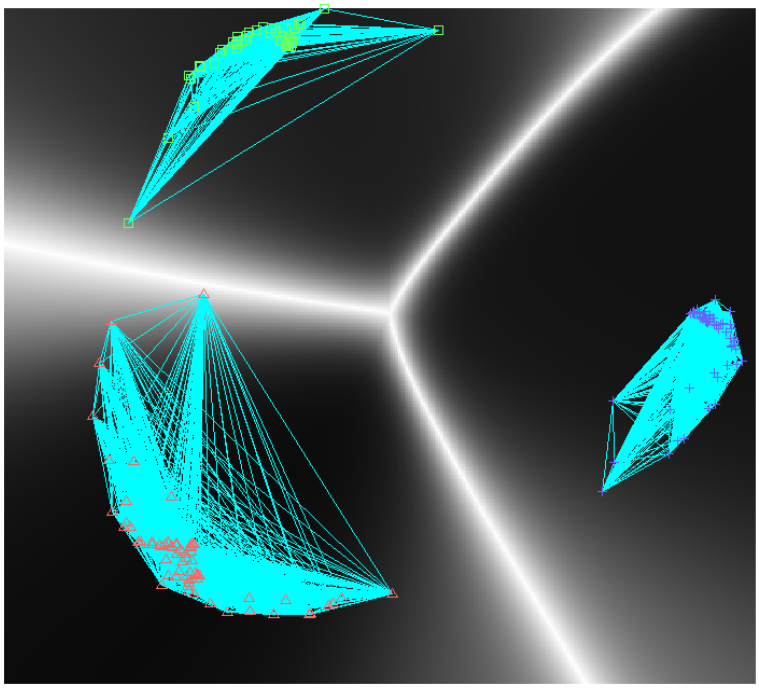


Figure 4.34: *Fisher network representation of the Wine dataset with informative background,* $\sigma_G = 0.50$*. Black pixels indicate a dominant class membership probability; white pixels indicate two equally likely classes.*

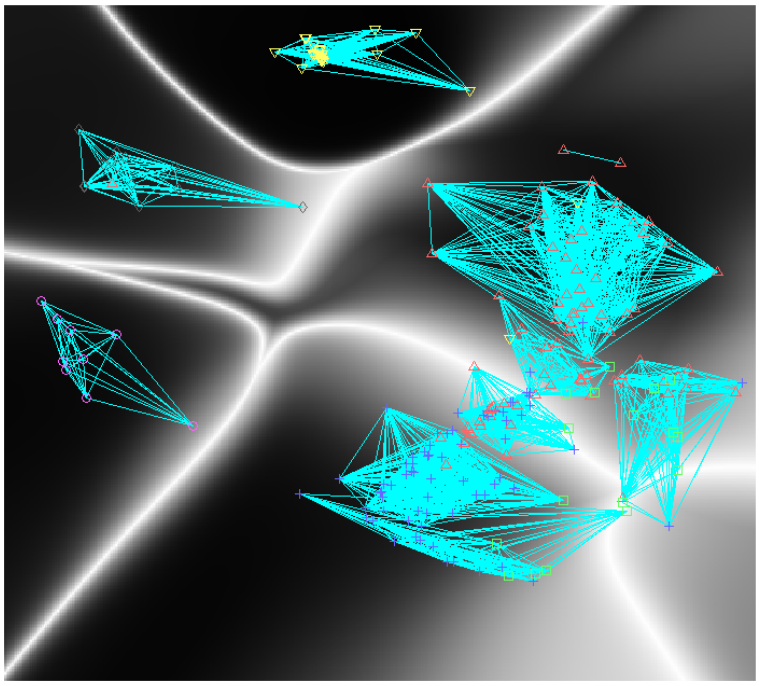


Figure 4.35: *Fisher network representation of the Glass dataset with informative background,* $\sigma_G = 0.50$*. Black pixels indicate a dominant class membership probability; white pixels indicate two equally likely classes.*

Reference cases in Fisher networks

The last part of this section studies the usefulness of reference cases in Fisher networks. As discussed in Section 3.6.2, the idea is to find the most central nodes in each community and use them as representatives of the clusters. For instance, in Fig. 4.39 a reference case is obtained for each of the six communities found in Fig. 4.27. These cases give the user a set of characteristic points that they can associate with each of the communities, making it easier to identify prototypical profiles for each of the clusters.

To assess how well the information contained in the structure of the network is represented by these central cases, the classification power of the network is compared to that of the base MLP for different numbers of references. First, a samples-to-reference ratio r is defined that gives the number of references used to represent each community as $\lceil n_g/r \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling function and n_g is the number of nodes in community g. The ratio r goes from 1 (taking every node as a reference case) to ∞ (using only one reference per community, regardless of how large the group is).

Once the set of reference cases $\mathcal{R}$ is identified, class membership probabilities for every node in the network are calculated using a weighted combination of the scores $a_k(\mathbf{x})$,

$$a'_k(\mathbf{x}_i) = \frac{1}{\sum_{\mathbf{x}_j \in \mathcal{R}} A_{ij}} \sum_{\mathbf{x}_j \in \mathcal{R}} A_{ij} a_k(\mathbf{x}_j), \tag{4.2}$$

which goes into a sigmoid or softmax function to give the estimated probabilities $\hat{p}'(c_k|\mathbf{x}_i)$. Then, class predictions are drawn from these estimates; Table 4.10 collects the results for different values of r. The values of σ_G used are taken from the previous subsection.

Table 4.10: *Reference case-based classification accuracies for different values of the samples-to-reference ratio. The accuracy of the MLP used to derive the Fisher metric for each dataset is provided for comparison.*

Dataset	σ_G	MLP accuracy (%)	Network accuracy (%)					
			r=1	r=5	r=10	r=25	r=50	r=∞
Ionosphere	0.20	97.72	97.44	97.72	97.72	97.72	97.44	97.44
Liver	0.20	77.97	78.26	78.26	78.55	77.97	77.97	77.97
Diabetes	0.10	79.17	79.04	79.17	79.17	79.17	79.17	79.04
Sonar	0.20	91.35	91.35	90.87	90.38	90.38	89.90	89.42
Glass	0.50	80.37	80.37	79.44	78.50	77.57	76.64	76.64
Wine	0.50	100	100	100	100	100	99.44	98.31

The first three rows of the table, corresponding to the Ionosphere, Liver and Diabetes datasets, do not suffer any significant performance variation when reducing the number of references used, even when using only one for each community. The Sonar dataset shows an accuracy loss when r increases, although it is very small in magnitude. Regarding

the multiclass problems, the Glass data suffers a slightly more noticeable deterioration when fewer references are considered, but even in the worst case, the resulting accuracy is not very far away from the base figure. The easiest classification task of the six, given by the Wine dataset, also experiences a small decrease in accuracy for the largest values of r. Figs. 4.36–4.41 represent the Fisher networks leading to the classification rates in the rightmost column of the table (one reference case per cluster), with central nodes highlighted by oversized markers. In these representations, edges are shown only between reference cases and their community neighbours.

Earlier in this chapter, Figs. 4.7–4.12 illustrate how Fisher networks can produce leave-one-out class predictions that result in classification accuracies as high as those obtained with the original MLP estimator. Further to that, the results in Table 4.10 suggest that a similar level of accuracy can be achieved using only a few central nodes. This is very interesting from the point of view of interpretability because it allows the explanation of the predictions made by the framework in terms of known data examples. This is also possible for the leave-one-out predictions, but by using reference cases the number of samples required to explain the assignments is reduced drastically. For these datasets, the predicted score in Eq. (4.2) is a simple linear combination with a number of terms that goes from 3 (Wine dataset) to 10 (Glass dataset). It is therefore possible to express compactly the exact individual contribution of each reference case $\mathbf{x}_j$ towards the classification of a sample $\mathbf{x}_i$ in terms of the weights A_{ij}.

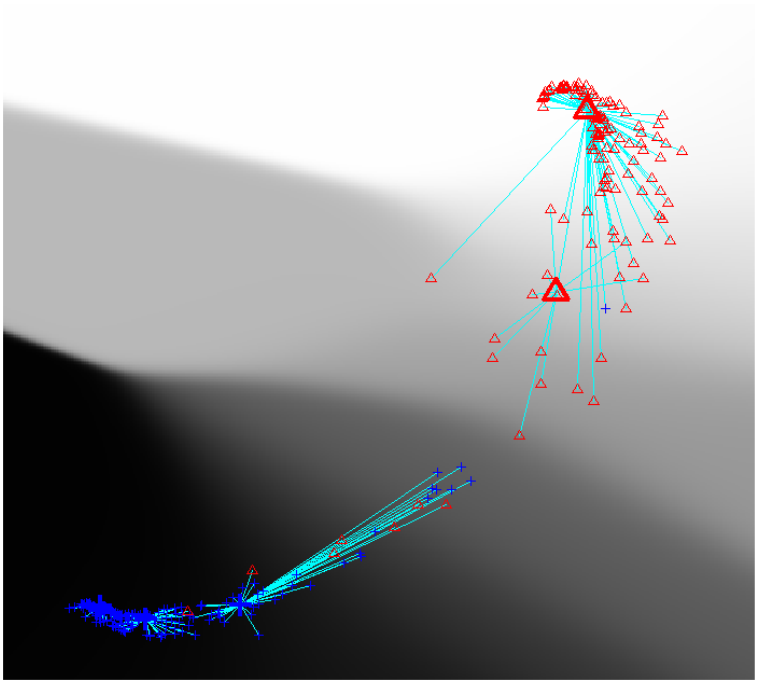


Figure 4.36: *Fisher network representation of the Ionosphere dataset with informative background and highlighted reference cases,* $\sigma_G = 0.20$.

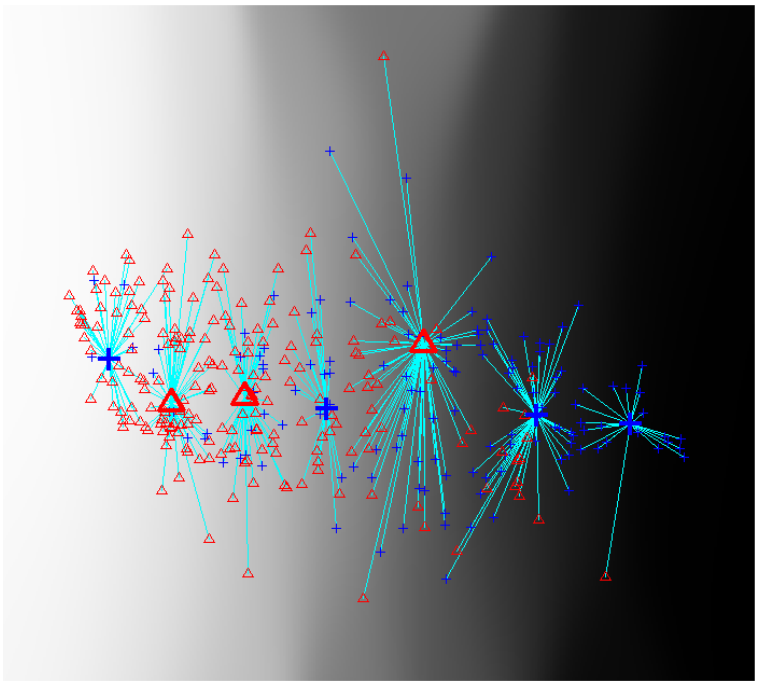


Figure 4.37: *Fisher network representation of the Liver dataset with informative background and highlighted reference cases,* $\sigma_G = 0.20$.

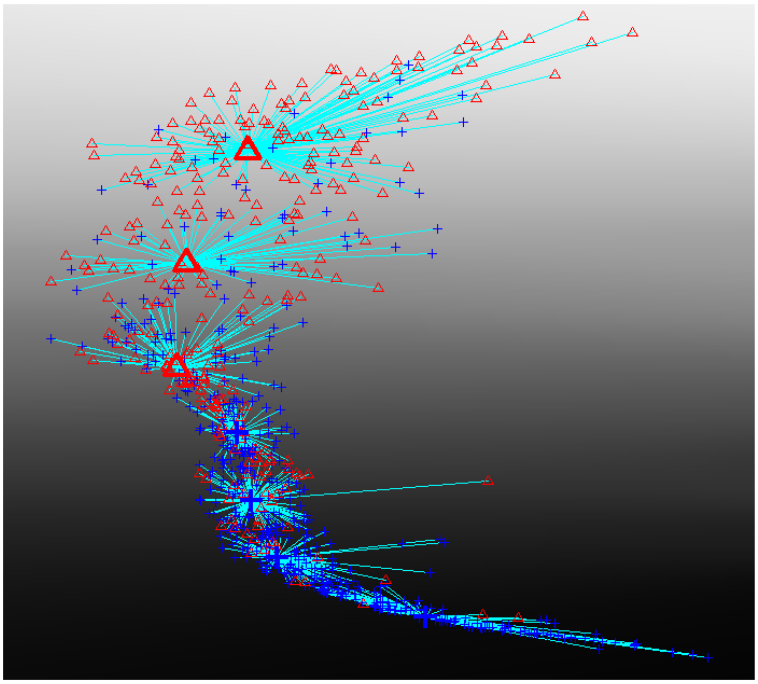


Figure 4.38: *Fisher network representation of the Diabetes dataset with informative background and highlighted reference cases,* $\sigma_G = 0.10$.

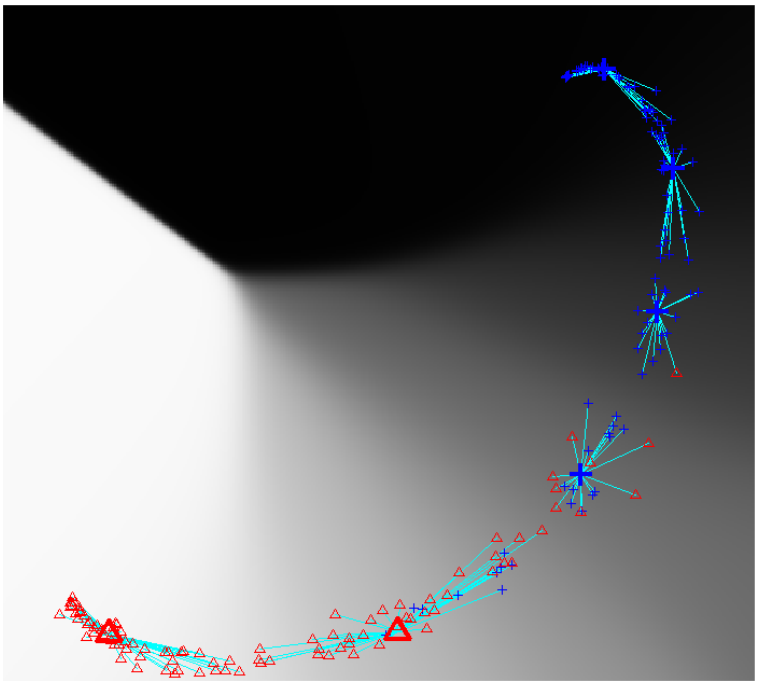


Figure 4.39: *Fisher network representation of the Sonar dataset with informative background and highlighted reference cases,* $\sigma_G = 0.20$.

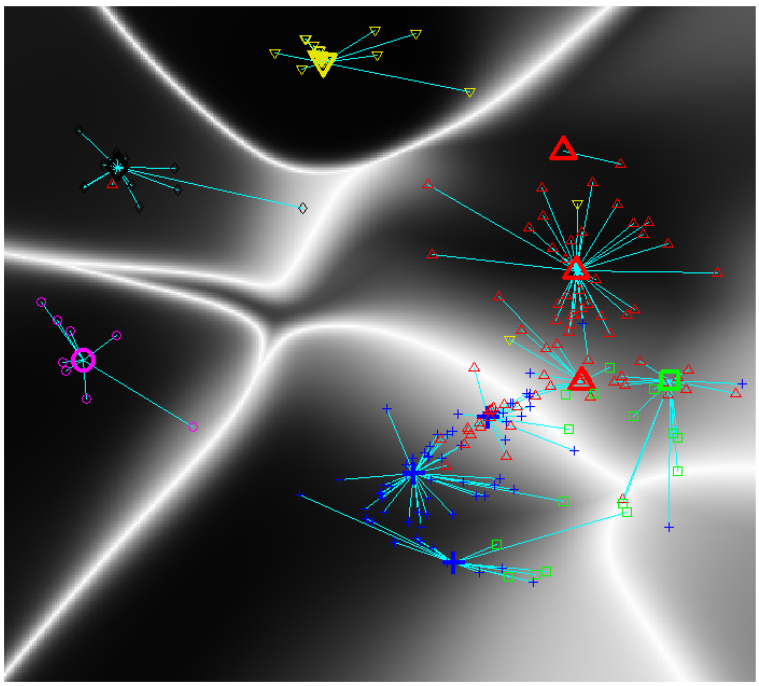


Figure 4.40: *Fisher network representation of the Glass dataset with informative background and highlighted reference cases,* $\sigma_G = 0.50$.

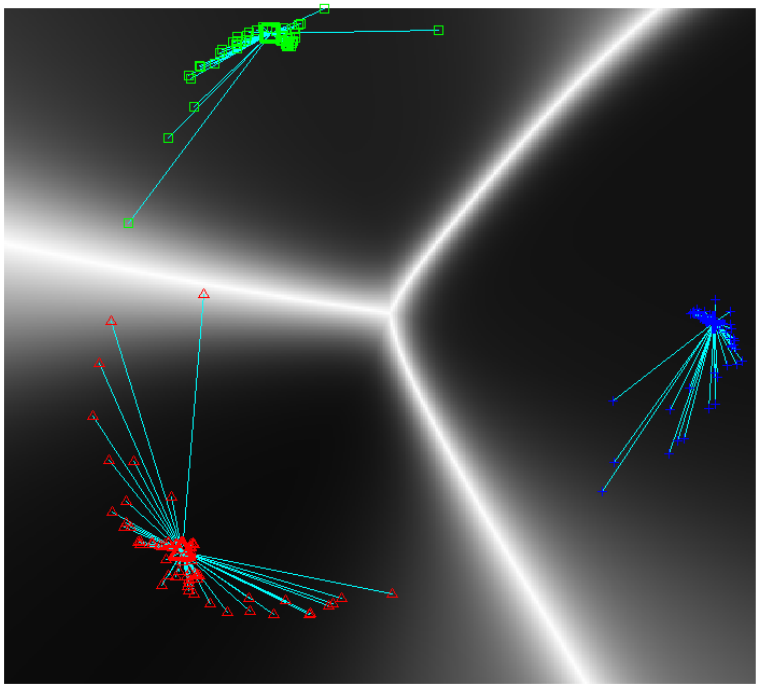


Figure 4.41: *Fisher network representation of the Wine dataset with informative background and highlighted reference cases,* $\sigma_G = 0.50$.

4.5 Supervised blind signal separation using FI metrics

The final section of the chapter presents another original application of the Fisher information metric [93]. In this case, the FI metric is combined with a blind signal separation (BSS) algorithm. BSS methods express complex signals as linear combinations of sources whose joint distribution is close to factorised into a product of independent univariate density functions. This approach is even more interpretable when it is applied in the convex space of positive semi-definite mixing and unmixing matrices [111], because then the sources and their partial contributions can be evaluated against prior knowledge.

In this application, synthetic data models are built from single-voxel magnetic resonance spectroscopy (MRS) signals corresponding to a neuro-oncology problem. The objective is that the sources obtained approximate prototypes for each brain tissue class and the maximal values in each row of the mixing matrix indicate the correct binary classification of the observations. In this context, the correct prototypes of the classes are taken to be the means of their generating distributions.

In a previous work [112], Ortega et al. investigate the use of non-negative matrix factorisation (NMF) methods [113, 114] for the extraction of tissue type-specific MRS signal sources in a fully unsupervised mode. This was applied to a multi-centre database that incorporates MRS data of several types of human brain tumours. The accuracies of the labels inferred for each patient case where comparable to those obtained by traditional supervised classifiers.

To try to improve the performance of the method, class label information is incorporated by deriving an FI metric from an MLP trained on the data and using it to define a Euclidean projective space where the BSS method is applied. The expectation is that the enhanced separation between classes provided by the Fisher metric translates into higher classification accuracies and sources that are closer to the true class prototypes.

Convex non-negative matrix factorisation

In this work, the Fisher metric is combined with a variant of the NMF family called Convex-NMF [111]. In conventional NMF methods, a non-negative data matrix $\mathbf{X}$ of size $N \times L$ (dimensions $\times$ samples) is approximately factorised into two non-negative matrices: the matrix of sources or data basis $\mathbf{S}$, of size $N \times K$, where K is the number of sources ($K < N$); and the mixing matrix $\mathbf{M}$, of size $K \times L$, whose columns provide the enconding of the data points in terms of the sources. The product of these matrices approximates the original data matrix, $\mathbf{X} \approx \mathbf{SM}$. In this study, each column of the data matrix corresponds to the spectrum of an observation and comprises radiation intensities at N different frequencies.

To achieve a more interpretable model, Convex-NMF imposes the constraint that the sources in $\mathbf{S}$ must lie within the column space of $\mathbf{X}$, that is, $\mathbf{S} = \mathbf{XU}$, where $\mathbf{U}$ is called the unmixing matrix. In this setting, $\mathbf{X} \approx \mathbf{XUM}$. By restricting $\mathbf{S}$ to convex combinations of the columns of $\mathbf{X}$ we can understand the sources as weighted sums of data points. This NMF variant applies to both non-negative and mixed-sign data matrices, since only $\mathbf{M}$ and $\mathbf{U}$ are required to be non-negative.

The mixing and unmixing matrices are updated with the following multiplicative algorithm,

$$\begin{aligned} \mathbf{M}^T &\leftarrow \mathbf{M}^T \sqrt{\frac{(\mathbf{X}^T\mathbf{X})^+ \mathbf{U} + \mathbf{M}^T\mathbf{U}^T (\mathbf{X}^T\mathbf{X})^- \mathbf{U}}{(\mathbf{X}^T\mathbf{X})^- \mathbf{U} + \mathbf{M}^T\mathbf{U}^T (\mathbf{X}^T\mathbf{X})^+ \mathbf{U}}} \\ \mathbf{U} &\leftarrow \mathbf{U} \sqrt{\frac{(\mathbf{X}^T\mathbf{X})^+ \mathbf{M}^T + (\mathbf{X}^T\mathbf{X})^- \mathbf{UMM}^T}{(\mathbf{X}^T\mathbf{X})^- \mathbf{M}^T + (\mathbf{X}^T\mathbf{X})^+ \mathbf{UMM}^T}}, \end{aligned} \tag{4.3}$$

where the function $(\cdot)^+$ keeps the positive elements of a matrix and replaces negative values with zeros and $(\cdot)^-$ operates conversely. $\mathbf{M}$ and $\mathbf{U}$ are initialised using k-means clustering, as proposed in [111]. This update process minimises the reconstruction error $||\mathbf{X} - \mathbf{XUM}||^2$ while ensuring that the elements of $\mathbf{M}$ and $\mathbf{U}$ remain non-negative.

Given that the MRS samples are of mixed sign, their sources should be too. Thus, if $\mathbf{S}$ is taken as the source spectra matrix, its columns will be readily interpretable and no pre-processing of the data will be required to ensure the non-negativity of $\mathbf{X}$, like it would be in a standard NMF method. The mixing matrix $\mathbf{M}$, non-negative by definition, can be understood as the concentration of the constituent signals in each data sample.

Description of the MRS data

The data analysed in this study are modelled from samples extracted from a database used in a previous publication [112]. Class (tumour type) labelling was used to generate posterior distributions of the data density using single multivariate normal models fitted to the mean and covariance matrices of class specific cohorts of single voxel proton MRS (SV-^{1}H-MRS) acquired at two different echo times (short, 20-32 ms (STE) and long, 135-144 ms (LTE)) from brain tumour patients.

The final dataset includes, at LTE, 20 samples of astrocytomas grade II (A2), 78 glioblastomas (GL) and 31 metastases (ME); and at STE it contains 22 A2 samples, 86 GL and 38 ME. The data dimensionality is 195, reflecting the clinically-relevant frequency intensity values measured in parts per million (ppm) that are typically sampled from each spectrum in the [4.24,0.50] ppm interval. A second dataset was generated to validate the method, comprising 50 samples drawn from the same distributions used for the training set.

Results

Three binary classification problems are tackled, one for each possible pair of classes. Performance is analysed in terms of classification accuracy and quality of the sources obtained, measured as the correlation between them and the true sources. Each classification problem is run 20 times to average the effect of the variable initialisations of the algorithm. Regarding the projection of the data, three methods are tested: Sammon mapping (Eq. (3.39)), standard MDS (Eq. (2.6)) and an iterative majorisation algorithm (IMA) [93].

Table 4.11: *Classification accuracy results for the original version of Convex-NMF and the three projection-based variations.*

Classes	Validation accuracy (%)							
		STE				LTE		
	Original	Sammon	MDS	IMA	Original	Sammon	MDS	IMA
A2 vs. GL	94	85.5	90	90	80	98	95	97
A2 vs. ME	97	93.8	99	99	88	99.9	99.9	100
GL vs. ME	67.7	72.8	75	74	62	81.8	82	83

Table 4.12: *Source correlations between the two sources obtained in each problem and the true mean spectra of the corresponding classes.*

Classes		Source correlation							
			STE				LTE		
		Original	Sammon	MDS	IMA	Original	Sammon	MDS	IMA
A2 vs. GL	A2	0.96	0.98	0.99	0.99	0.99	1	1	1
	GL	0.96	1	1	1	0.80	0.99	0.99	0.98
A2 vs. ME	A2	0.96	0.93	0.98	0.99	1	0.99	0.99	1
	ME	0.99	0.94	0.99	1	0.94	0.96	0.97	1
GL vs. ME	GL	0.94	1	0.98	0.98	0.68	0.99	0.99	0.99
	ME	0.98	1	1	1	0.89	0.99	0.99	0.99

The classification results on validation data (Table 4.11) show a general performance improvement over the original approach when the method is applied on the projected data using Fisher distances. There are some exceptions, however, where the standard Convex-NMF does better than the alternatives: In A2 vs. ME at STE it outperforms the Sammon version and in A2 vs. GL at STE it improves on all three mappings. Of the three supervised versions of the algorithm, MDS and IMA obtain better results, especially at STE.

Regarding the quality of the sources, Table 4.12 indicates that, in general, all four approaches produce very good approximations. The most interesting classification problem is GL against ME, for it is the most challenging from the three. These two classes have a very similar mean spectrum, which causes the low accuracies in the bottom row of Table 4.11. Despite that, the Fisher metric-based versions of Convex-NMF manage to

replicate the true sources remarkably well.

Fig. 4.42 represents the mean spectra of the true classes. As mentioned above, the resemblance between them is very high in terms of the position and height of the peaks. The only clear difference is the height of the second main peak. Figs. 4.43–4.46 contain the sources extracted by the four methods tested. Plain Convex-NMF does not get both main peaks right at the same time, and it instead identifies each class with one of them. On the other hand, the Fisher metric approaches retrieve sources that are almost identical to the original ones.

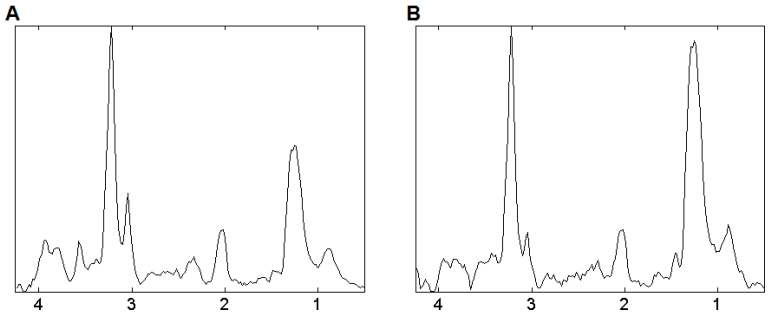


Figure 4.42: *True GL* **(A)** *and ME* **(B)** *mean spectra.*

The results confirm that an unsupervised BSS method like Convex-NMF can benefit from the use of known data labels and obtain higher classification rates and more accurate sources. Furthermore, this is achieved without deteriorating the interpretability of the results.

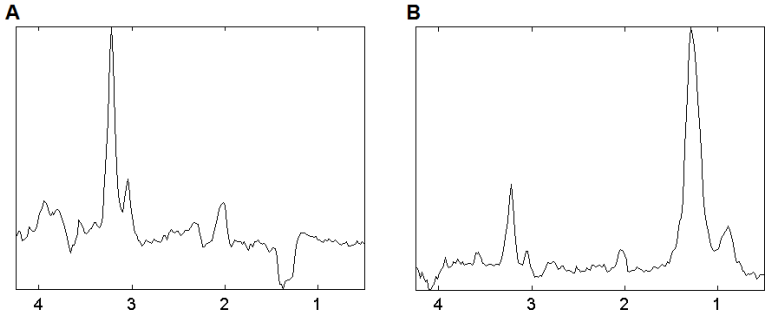


Figure 4.43: *Class GL* **(A)** *and ME* **(B)** *mean spectra obtained with plain Convex-NMF.*

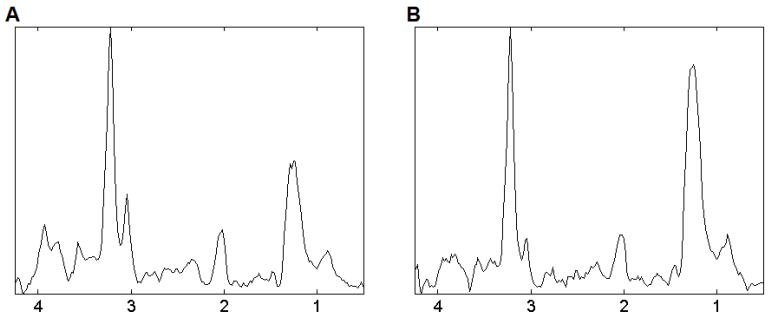


Figure 4.44: *Class GL* **(A)** *and ME* **(B)** *mean spectra obtained with Sammon mapping + Convex-NMF.*

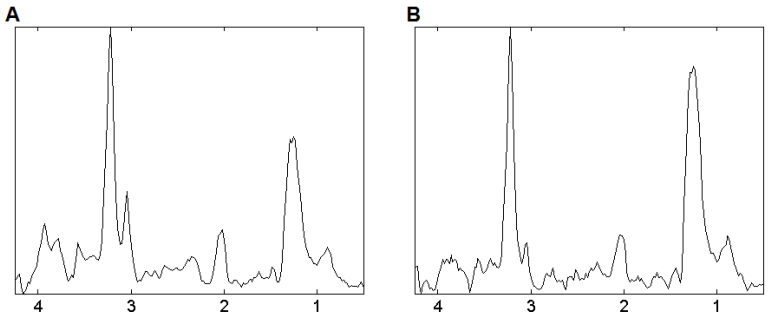


Figure 4.45: *Class GL* **(A)** *and ME* **(B)** *mean spectra obtained with MDS + Convex-NMF.*

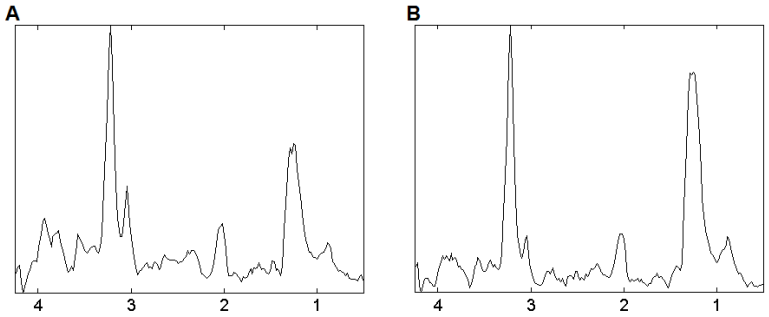


Figure 4.46: *Class GL* **(A)** *and ME* **(B)** *mean spectra obtained with IMA + Convex-NMF.*

4.6 Chapter summary

This chapter shows examples of the different modules that form the Fisher network framework put into practice. To start with, Section 4.1 illustrates the effect of the FI metric using a simple 2-D classification problem, showing that the metric rearranges data points precisely in the way it was desired to, as seen in Fig. 4.2. In Section 4.2, the metric is proven beneficial in high-dimensional KNN classification tasks using an artificial non-linear problem, where it outperforms the Euclidean version of the classifier. Section 4.3 then compares the proposed geodesic approximation algorithm with other methods in the literature. The performance figures indicate that, although the suggested approach obtains an improvement on the accuracy of existing methods, it is only a small gain that does not justify the added computational cost unless the dataset used is small.

The main set of experiments (Section 4.4) take the reader through the process of building Fisher networks using six real-world datasets. First, Section 4.4.1 benchmarks the classification performance of the MLP estimator used to derive the metric against other methods that have been applied to the same data previously in the literature, concluding that the accuracy of the chosen model is comparable to that of state-of-the-art methods. After that, Section 4.4.2 starts by applying the criteria described in Section 3.4.1 to find out suitable value ranges for the locality parameter σ_G. Using those values, examples of networks are built and described in the following subsection, where they are compared to analogous graphs generated using Euclidean distances. The latter show a significant deterioration in the quality of their predictions and in the concordance between the communities found and the true class labels, which highlights the benefits of including side information in the construction process.

Coloured networks are produced for the same datasets, providing a more informative visualisation of the data. Then, the last subsection finds reference cases for the six networks, and Table 4.10 shows that the classification accuracies of the MLP can be closely reproduced by using only a few central nodes in a case-based approach.

Finally, Section 4.5 discusses the inclusion of an element of supervision in the process of convex non-negative matrix factorisation. By applying CNMF on a mapped version of the data using Fisher distances, the performance of the plain algorithm is improved both in terms of classification accuracy and quality of the sources obtained.

Chapter 5

Review, conclusions and future work

The fifth and final chapter closes this thesis with a discussion of the work carried out and the degree to which it satisfies the objectives and functionalities initially planned for the framework. Then follows a list of interesting topics to investigate that could bring new features into the system and mitigate its limitations.

5.1 Review

The main motivation of this work was to develop a visualisation and classification methodology with a focus on interpretability. The initial idea was to use a Fisher information-based distance metric to define a pairwise similarity measure that would guide the construction of networks. Following that approach, a Fisher metric in the data space is derived from sigmoidal output discriminant estimators. The result is a Riemannian metric with a rigorous theoretical basis that calculates distances between points in terms of the class membership probability fluctuations present along the path followed, automatically filtering the local relevance of the different variables with respect to the problem at hand. Although the Fisher metric is only the first part of the framework, using it as the foundation of the system represents the first step towards interpretability, since it makes distances a reliable measure of how similar samples really are with regard to the underlying posterior class probabilities. Section 4.2 provides an example of the robustness of the metric, especially in high dimensional problems, the main application scope of the framework. Other methodologies can also benefit from the use of the metric, as seen in Section 4.5.

However, the calculation of Fisher distances is not as straightforward as it is, for instance, in a Euclidean metric. Capturing the differential relevance of the space comes at a cost:

the Fisher metric is Riemannian and, as such, distances must be computed as an infinite sum of infinitesimal displacements. Interestingly, Appendix C shows that distances under an FI metric derived from a linear estimator are given by a closed-form expression that is independent of the path followed. Unfortunately, this is only locally applicable to the non-linear case, and the challenge of approximating geodesic paths must be faced when calculating global distances. To do so, Section 3.3.3 presents the free point approach, a novel algorithm based on the iterative optimisation of an initial path. When it comes to performance, though, Section 4.3 shows that this method only provides a marginal improvement on other existing algorithms that are computationally much cheaper.

As planned, data are represented using similarity networks, which are built from pairwise Fisher distances using a Gaussian kernel that defines the structure of the graph. This process involves the selection of a parameter, σ_G, that controls the influence of locality in the generation of the weights of the network connections. The system presents the user with a number of measures to aid the selection of its value, or chooses it automatically if no external input is provided. Once the network is defined in the form of an adjacency matrix, a spectral community detection algorithm is applied to find the set of clusters that best represent the structure of the graph. The Fisher network is then ready to be represented by plotting its nodes and placing edges between those of them that belong to the same community, highlighting the structure of the data. Nodes are arranged in the 2-D plot using MDS techniques, producing a neat visualisation of the dataset where communities are automatically placed in an orderly pattern that reflects the class probabilities of their members.

A further interpretability enhancement is implemented with informative backgrounds, a variation of the standard representation that colours the background image of the network plot according to the local posterior probabilities. This gives the user an interpretable visualisation of the different parts of the network and how well defined they are in terms of class membership probability. In short, Fisher network representations are exactly what was looked for: an intuitive visualisation tool that can generate a highly informative identifying picture for a dataset regardless of its dimensionality.

The other main functionality that was expected from the framework is the ability to perform label prediction, again stressing the importance of obtaining interpretable results. Fisher networks achieve this through a simple weighted contribution of their nodes, which has proven to be able to reproduce the classification accuracies of the original probability estimator. Furthermore, the number of nodes that are taken into account can be reduced substantially without significantly deteriorating the accuracy of the predictions drawn: For the six real-world datasets studied in Section 4.4.2, using only the most central node in each community (less than a dozen in the busiest case) was enough to produce classification rates comparable to those given by an MLP, which were in line with other methods

in the literature.

5.2 Conclusions

Fisher networks constitute a novel mathematical framework for the analysis of categorised data that performs two main functions: Firstly, it offers a data visualisation tool that maps the original, often high-dimensional, space into a 2- or 3-D embedding where data can be observed directly. The use of a tailored distance metric provides the system with the ability to consider only relevant features of the space, making distances accurate measures of dissimilarity even when the number of covariates is large. Secondly, the framework produces interpretable probabilistic class predictions based on the estimates of a supervised model. These are explained to the user as a linear combination of the predictions of a reduced number of selected known cases in the dataset.

Although the derivation of the Fisher metric requires supervision in the form of class label information, once the training stage is over and the metric is defined, distances can be calculated between any pair of points. This means that new data samples with unknown membership can be incorporated into an existing Fisher network simply by calculating the strength of their connections to the rest of the points in the dataset. The community detection algorithm can then be used to assign the new nodes to the appropriate clusters.

To summarise, the Fisher network framework is a successful materialisation of the initially desired features, allowing the user to benefit from the performance of black box models while providing a case-based interpretation for their inferences and an informative representation of the data under study.

5.3 Future work

Even though the goals of the research project have been satisfied, there are several avenues that can be explored to improve the functionality of the framework.

- The estimation of the conditional class probabilities has been carried out using logistic regression in the linear case and an artificial neural network for non-linear problems, which is the default method of choice. The multilayer perceptron is a versatile and powerful estimator, but it does not need to be the only option available. It would be interesting to derive the Fisher metric for other probabilistic models e.g., radial basis function networks [86] or relevance vector machines [87], and compare the resulting visualisations and classification performance with those of the MLP-based approach.

- Although Section 3.6.3 explains how to add new data points to an existing Fisher network, Chapter 4 does not include any experiment that illustrates this using a validation dataset. Table 4.10 in Section 4.4.2 contains leave-one-out classification accuracies, but they come from data that was used for the training of the MLP, either as training or test cases, due to the reduced number of samples available.
  In the future, a more thorough experiment could be carried out where, by using a large enough dataset, a validation set can be selected and excluded from the training stage of the probability estimator without significantly affecting the quality of the $p(c_j|\mathbf{x})$ estimates. Then, once the Fisher network is built, these samples could be used as truly *new* cases, with the advantage that their class labels would be known. In a medical scenario, for example, this would allow the clinician to visualise new patients along with known ones in the same plot, also providing a selection of related previous patients to support the class assignment process.

- The representations in Figs. 4.13–4.41 display the community structure of the networks using edges to connect points from the same cluster. However, these edges do not provide the user with enough information to differentiate nodes that have a strong community membership from those whose cluster assignment is less influential in the modularity score of the partition. The arrangement of the nodes in the plot indirectly gives an indication of this, but the presence of edges is still distracting in some cases. A possible solution is to modulate the width of edges using their connection weights or, alternatively, the value of their elements in the corresponding eigenvector of the community identification algorithm.

- The Fisher network examples in Chapter 4 are built from datasets of moderate size, but even in those cases there are sometimes areas of the plots where a dense presence of data makes it difficult to recognise nodes and communities clearly. For example, in Fig. 4.13 there are two different communities at the very top of the C shape, but it is easy to confuse them with a single cluster. Similarly, some regions in Figs. 4.16, 4.20 and 4.24 are so crowded that even roughly estimating how many data points there are in them is impossible. These issues will be more frequent in large datasets, and for that reason it would be interesting to produce a hierarchical visualisation based on reference cases. At the top view level, only a few nodes would be displayed; the most central ones within their communities. Then, as the user zooms into the network, the number of nodes shown increases, again in an order given by their centrality. This would give a cleaner simplified view of the dataset without having to plot every single node, which would still be available to view if required.

- In regard to large datasets, the calculation of the full pairwise distance matrix could become computationally expensive when there are many examples involved, since

the time required is proportional to the sample size squared. This was not a concern in the experiments included in this thesis, so path integrals were approximated using a large number of divisions to ensure an accurate result. With bigger datasets, though, adjusting this parameter becomes of paramount importance. How to do this automatically depending on the particular circumstances (e.g., the separation between the points or their location in the space) is an important question that needs to be addressed in order for the framework to be computationally efficient in large problems.

- The results in Section 4.3 indicate that the performance of the proposed geodesic approximation method is heavily dependent on the initial path used and likely to get stuck in local minima. It may be worth investigating other initialisation procedures or alternatives to replace gradient descent in the optimisation of the objective function.

- Section 4.2 presented an analysis of the effect of the Fisher metric on a synthetic non-linear KNN classification problem for an increasing number of variables. The results provided interesting insights on the behaviour of both Fisher and Euclidean metrics. It would be interesting to research this topic in further detail using other artificially generated datasets with different data distributions.

- The study discussed above, as well as any functionality improvements, should also be made extendable to multiclass problems, since it is normally in such contexts where the advantages of interpretability and informative data visualisation are most needed.

Appendix A

Derivation of the FI matrix

This appendix contains the derivation of the Fisher information matrix expressions in Sections 3.2.1 and 3.2.2 from the first of the two definitions of the matrix in Eq. (3.14),

$$\mathbf{FI}(\mathbf{x}) = \sum_{j}^{J} (\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))(\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))^T \hat{p}(c_j|\mathbf{x}). \tag{A.1}$$

For the sake of clarity, the following abbreviations are used throughout this appendix:

$$\begin{aligned} \nabla = \nabla_{\mathbf{x}} &= \frac{d}{d\mathbf{x}} \\ a = a(\mathbf{x}) \quad &\text{(binary)} \\ a_j = a_j(\mathbf{x}) \quad &\text{(multiclass)} \\ \hat{p}_j &= \hat{p}(c_j|\mathbf{x}). \end{aligned} \tag{A.2}$$

The calculation of $\mathbf{FI}(\mathbf{x})$ is first carried out for binary classifiers with sigmoid output and then for multiclass estimators with softmax activation function.

A.1 FI matrix derivation for two classes

The output of the estimators considered in this subsection gives class-membership probabilities of the form

$$\hat{p}_1 = \frac{1}{1+\exp(-a)} \qquad \hat{p}_0 = \frac{\exp(-a)}{1+\exp(-a)}. \tag{A.3}$$

Eq. (A.1) requires the terms $\nabla \log(\hat{p}_j)$. The logarithms of the probabilities are given by

$$\begin{aligned} \log(\hat{p}_1) &= -\log(1+\exp(-a)) \\ \log(\hat{p}_0) &= -(a+\log(1+\exp(-a)), \end{aligned} \tag{A.4}$$

and their derivatives are

$$\begin{aligned} \nabla \log(\hat{p}_1) &= (1-\hat{p}_1)\nabla a \\ \nabla \log(\hat{p}_0) &= -\hat{p}_1 \nabla a. \end{aligned} \tag{A.5}$$

Inserting Eq. (A.5) in Eq. (A.1) and expanding the summation gives

$$\begin{aligned} \mathbf{FI}(\mathbf{x}) &= (\nabla \log(\hat{p}_1))(\nabla \log(\hat{p}_1))^T \hat{p}_1 + (\nabla \log(\hat{p}_0))(\nabla \log(\hat{p}_0))^T \hat{p}_0 \\ &= ((1-\hat{p}_1)\nabla a)((1-\hat{p}_1)\nabla a)^T \hat{p}_1 + (\hat{p}_1 \nabla a)(\hat{p}_1 \nabla a)^T (1-\hat{p}_1) \\ &= (\nabla a)(\nabla a)^T \hat{p}_1 (1-\hat{p}_1)^2 + (\nabla a)(\nabla a)^T \hat{p}_1^2 (1-\hat{p}_1) \\ &= (\nabla a)(\nabla a)^T \hat{p}_1 (1-\hat{p}_1). \end{aligned} \tag{A.6}$$

A.2 FI matrix derivation for multiple classes

The multiclass predictor used in Section 3.1.2 produces probability estimates given by

$$\hat{p}_j = \frac{\exp(a_j)}{\sum_{k=1}^{J} \exp(a_k)}, \tag{A.7}$$

and their logarithms are

$$\log(\hat{p}_j) = a_j - \log\left(\sum_{k=1}^{J} \exp(a_k)\right), \tag{A.8}$$

with derivatives

$$\nabla \log(\hat{p}_j) = \nabla a_j - \sum_{k=1}^{J} \hat{p}_k \nabla a_k. \tag{A.9}$$

Combining Eqs. (A.1) and (A.9),

$$\mathbf{FI}(\mathbf{x}) = \sum_{j}^{J} \left(\nabla a_j - \sum_{k=1}^{J} \hat{p}_k \nabla a_k\right) \left(\nabla a_j - \sum_{l=1}^{J} \hat{p}_l \nabla a_l\right)^T \hat{p}_j, \tag{A.10}$$

where the index l is introduced to avoid confusion. Expanding the product,

$$\begin{aligned}\mathbf{FI}(\mathbf{x}) = \sum_{j}^{J} \Bigg(& (\nabla a_j)(\nabla a_j)^T \\ & - \sum_{l}^{J} (\nabla a_j)(\nabla a_l)^T \hat{p}_l - \sum_{k}^{J} (\nabla a_k)(\nabla a_j)^T \hat{p}_k \\ & + \sum_{k}^{J} \sum_{l}^{J} (\nabla a_k)(\nabla a_l)^T \hat{p}_k \hat{p}_l \Bigg)^T \hat{p}_j.\end{aligned} \tag{A.11}$$

By noting that any quantity t can be expressed as $t \sum_i^J p_i = \sum_i^J t p_i$, Eq. (A.11) can be rearranged as

$$\begin{aligned}\mathbf{FI}(\mathbf{x}) = \sum_{j}^{J} \Bigg(& \sum_{k}^{J} \sum_{l}^{J} (\nabla a_j)(\nabla a_j)^T \hat{p}_k \hat{p}_l \\ & - \sum_{k}^{J} \sum_{l}^{J} (\nabla a_j)(\nabla a_l)^T \hat{p}_k \hat{p}_l - \sum_{k}^{J} \sum_{l}^{J} (\nabla a_k)(\nabla a_j)^T \hat{p}_k \hat{p}_l \\ & + \sum_{k}^{J} \sum_{l}^{J} (\nabla a_k)(\nabla a_l)^T \hat{p}_k \hat{p}_l \Bigg)^T \hat{p}_j.\end{aligned} \tag{A.12}$$

The summations can then be merged, resulting in the final expression,

$$\begin{aligned}\mathbf{FI}(\mathbf{x}) &= \sum_{j}^{J} \Bigg(\sum_{k}^{J} \sum_{l}^{J} \Big((\nabla a_j)(\nabla a_j)^T - (\nabla a_j)(\nabla a_l)^T \\ &\qquad - (\nabla a_k)(\nabla a_j)^T + (\nabla a_k)(\nabla a_l)^T \Big) \hat{p}_k \hat{p}_l \Bigg)^T \hat{p}_j \\ &= \sum_{j}^{J} \sum_{k}^{J} \sum_{l}^{J} (\nabla (a_j - a_k))(\nabla (a_j - a_l))^T \hat{p}_j \hat{p}_k \hat{p}_l.\end{aligned} \tag{A.13}$$

Appendix B

Interpretation of the FI metric in the context of information and divergence

The purpose of this appendix is to review the meaning and validity of the Fisher information metric in primary data space. The original definitions of information and divergence and their relationship with the metric [56] are discussed in the case where the role of the density function $p(\mathbf{x}|\boldsymbol{\theta})$ is taken by a posterior distribution $\hat{p}(c_j|\mathbf{x})$ that is assumed to have been fitted using empirical auxiliary data. The idea is to replicate the derivation of the traditional FI matrix expressions in the parameter space ensuring that the regularity assumptions hold, with the aim of gaining a rigorous insight into the meaning of the reversed formulation of the Fisher metric in the space of the covariates.

B.1 Information

The first step is to define the information of the likelihood of $\mathbf{x}_1$ given c_j, $\mathcal{L}(\mathbf{x}_1|c_j) = \hat{p}(c_j|\mathbf{x}_1)$, with respect to that of $\mathbf{x}_2$, $\mathcal{L}(\mathbf{x}_2|c_j) = \hat{p}(c_j|\mathbf{x}_2)$, for $\mathbf{x}_1$ and $\mathbf{x}_2$ samples of a multivariate random variable $\mathcal{X}$. The indicator c_j implies $y = j$, where y is the class label of $\mathbf{x}$ and j is one of the possible values of the discrete class variable $\mathcal{Y}$. Let $\mathcal{H}_i$, $i \in \{1, 2\}$, be the hypotheses that the points $\mathbf{x}_i$ were generated by class j. From Bayes' theorem,

$$\log \frac{\hat{p}(c_j|\mathbf{x}_1)}{\hat{p}(c_j|\mathbf{x}_2)} = \log \frac{\hat{p}(\mathbf{x}_1|c_j)}{\hat{p}(\mathbf{x}_2|c_j)} - \log \frac{\hat{p}(\mathbf{x}_1)}{\hat{p}(\mathbf{x}_2)}, \tag{B.1}$$

where $\hat{p}(\mathbf{x}_i)$ is the prior probability of data point $\mathbf{x}_i$ and $\hat{p}(\mathbf{x}_i|c_j)$ is the class conditional probability of $\mathbf{x}_i$ given c_j, that is, given membership to class j. Therefore, the right hand side of Eq. (B.1) is the difference of the divergence between the logarithm of the odds in favour of $\mathcal{H}_1$ before and after observing $y_1, y_2 = j$. This can be interpreted as

the information resulting from the observation of class membership, and the logarithm of the likelihood ratio, $\log(\hat{p}(c_j|\mathbf{x}_1)/\hat{p}(c_j|\mathbf{x}_2))$, is defined to be the information in $\mathcal{Y} = j$ for discrimination in favor of $\mathcal{H}_1$ against $\mathcal{H}_2$. The expectation of this information with respect to $\hat{p}(c_j|\mathbf{x}_1)$ over all possible classes gives what is normally known as the Kullback-Leibler (KL) divergence,

$$I_{KL}(\mathbf{x}_1 : \mathbf{x}_2) = \sum_j \log\left(\frac{\hat{p}(c_j|\mathbf{x}_1)}{\hat{p}(c_j|\mathbf{x}_2)}\right)\hat{p}(c_j|\mathbf{x}_1). \tag{B.2}$$

Combining Eqs. (B.1) and (B.2), the KL divergence can be expressed as

$$I_{KL}(\mathbf{x}_1 : \mathbf{x}_2) = \sum_j \log\left(\frac{\hat{p}(\mathbf{x}_1|c_j)}{\hat{p}(\mathbf{x}_2|c_j)}\right)\hat{p}(c_j|\mathbf{x}_1) - \log\frac{\hat{p}(\mathbf{x}_1)}{\hat{p}(\mathbf{x}_2)}. \tag{B.3}$$

This measure can therefore be seen as the expected value of the power of the class variable $\mathcal{Y}$ to distinguish between $\mathbf{x}_1$ and $\mathbf{x}_2$ or, in other words, the information that it contains about their likelihoods.

B.2 Divergence

Although $I_{KL}(\mathbf{x}_1 : \mathbf{x}_2)$ is what is commonly known as KL divergence, the original formulation of the divergence measure defined it as a symmetric measure, obtained by calculating the expected value of the information in Eq. (B.2) over both orders of $\mathcal{H}_1$ and $\mathcal{H}_2$,

$$\begin{aligned} J_{KL}(\mathbf{x}_1 : \mathbf{x}_2) &= I_{KL}(\mathbf{x}_1 : \mathbf{x}_2) + I_{KL}(\mathbf{x}_2 : \mathbf{x}_1) \\ &= \sum_j \log\left(\frac{\hat{p}(c_j|\mathbf{x}_1)}{\hat{p}(c_j|\mathbf{x}_2)}\right)(\hat{p}(c_j|\mathbf{x}_1) - \hat{p}(c_j|\mathbf{x}_2)), \end{aligned} \tag{B.4}$$

which, again, represents a measure of the difficulty to discriminate between the two hypotheses.

Both Eqs. (B.2) and (B.4) serve as starting points to calculate a local Taylor approximation that enables the derivation of the connection between these measures and the FI metric. However, the use of this approximation relies on the validity of three regularity conditions.

B.3 Regularity conditions

The first two regularity conditions used in the original formulation (see Chapter 2.6 in [56]) require the conditional class probabilities $\hat{p}(c_j|\mathbf{x})$ to be continuous, finite and differentiable everywhere in the input space. In the following, it is assumed that all vector derivatives with respect to the multivariate random variable $\mathcal{X}$ may be treated independently from one covariate to the others. This assumption applies in the context of posterior probability functions that may be linear in the parameters or semi-parametric, including generic non-linear models such as artificial neural networks.

The third regularity condition follows directly from the standard normalisation of conditional probability density functions (probability masses in the case of a discrete class variable),

$$\sum_j \hat{p}(c_j|\mathbf{x}) = 1 \longrightarrow \sum_j \nabla_{\mathbf{x}} \hat{p}(c_j|\mathbf{x}) = 0 \longrightarrow \sum_j \nabla^2_{\mathbf{x}} \hat{p}(c_j|\mathbf{x}) = 0. \tag{B.5}$$

Additionally, this last condition can be used to derive the dual definition of the FI matrix in Eq. (3.14). Expressing $\hat{p}(c_j|\mathbf{x})$ as $\exp(\log \hat{p}(c_j|\mathbf{x}))$ and taking derivatives of the left hand side summation in Eq. (B.5),

$$\nabla^2_{\mathbf{x}} \sum_j \exp(\log \hat{p}(c_j|\mathbf{x})) = \nabla_{\mathbf{x}} \sum_j (\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))\hat{p}(c_j|\mathbf{x}) = 0. \tag{B.6}$$

Expanding the derivatives on the right hand side, it follows that

$$\begin{aligned} &\sum_j (\nabla^2_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))\hat{p}(c_j|\mathbf{x}) \\ &+ \sum_j (\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))(\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))^T \hat{p}(c_j|\mathbf{x}) = 0, \end{aligned} \tag{B.7}$$

hence

$$\begin{aligned} &\sum_j (\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))(\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))^T \hat{p}(c_j|\mathbf{x}) \\ &\qquad = -\sum_j (\nabla^2_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))\hat{p}(c_j|\mathbf{x}), \end{aligned} \tag{B.8}$$

which shows the numerical equivalence of the two definitions.

B.4 Connection with the Fisher information

This subsection reproduces in the input space the relationship between the KL divergence and the Fisher metric described in [56]. The starting point is the information measure in Eq. (B.2) applied to a pair of adjacent points,

$$I_{KL}(\mathbf{x} : \mathbf{x} + d\mathbf{x}) = -\sum_j \log\left(\frac{\hat{p}(c_j|\mathbf{x} + d\mathbf{x})}{\hat{p}(c_j|\mathbf{x})}\right)\hat{p}(c_j|\mathbf{x}). \tag{B.9}$$

Taking a Taylor approximation of $\log(\hat{p}(c_j|\mathbf{x} + d\mathbf{x}))$ about $\mathbf{x}$ up to second order terms,

$$\begin{aligned} &\log(\hat{p}(c_j|\mathbf{x} + d\mathbf{x})) - \log(\hat{p}(c_j|\mathbf{x})) \\ &= d\mathbf{x}^T \nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})) + \frac{1}{2} d\mathbf{x}^T \Big(\nabla^2_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x}))\Big) d\mathbf{x}. \end{aligned} \tag{B.10}$$

Inserting this in Eq. (B.9),

$$\begin{aligned} I_{KL}(\mathbf{x} : \mathbf{x} + d\mathbf{x}) = &-\sum_j d\mathbf{x}^T (\nabla_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))\hat{p}(c_j|\mathbf{x}) \\ &-\frac{1}{2}\sum_j d\mathbf{x}^T \Big(\nabla^2_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x}))\Big) d\mathbf{x}\hat{p}(c_j|\mathbf{x}). \end{aligned} \tag{B.11}$$

According to Eq. (B.6), the first summation is zero, so

$$\begin{aligned} I_{KL}(\mathbf{x} : \mathbf{x} + d\mathbf{x}) &= -\frac{1}{2} d\mathbf{x}^T \left(\sum_j (\nabla^2_{\mathbf{x}} \log(\hat{p}(c_j|\mathbf{x})))\hat{p}(c_j|\mathbf{x})\right) d\mathbf{x} \\ &= \frac{1}{2} d\mathbf{x}^T \mathbf{FI}(\mathbf{x}) d\mathbf{x}. \end{aligned} \tag{B.12}$$

After a similar procedure, the symmetric divergence in Eq. (B.4) yields

$$J_{KL}(\mathbf{x} : \mathbf{x} + d\mathbf{x}) = d\mathbf{x}^T \mathbf{FI}(\mathbf{x}) d\mathbf{x}. \tag{B.13}$$

Based on Eqs. (B.12) and (B.13), the differential distance between a pair of nearby points in the input space can be understood as the KL divergence between their class membership probabilities. This is equivalent, going back to Section B.1, to the expected information provided by the class label to discern between the hypotheses $\mathcal{H}_1$ and $\mathcal{H}_2$ over all possible classes.

Appendix C

Derivation of the distance expression for linear estimators

This appendix covers the derivation of Eq. (3.23), which gives the distance between any pair of points under a Fisher metric defined using an LLR as the probability estimator. This is an important expression because it provides an analytical and efficient way to calculate global distances with a linear estimator in binary problems and it can also be used to approximate straight line distances for non-linear estimators by using a Taylor approximation.

Let us recall the expressions of the FI matrix

$$\mathbf{FI}(\mathbf{x}) = (\nabla_{\mathbf{x}} a(\mathbf{x}))(\nabla_{\mathbf{x}} a(\mathbf{x}))^T \hat{p}(c_1|\mathbf{x})(1 - \hat{p}(c_1|\mathbf{x})), \tag{C.1}$$

the generalised linear estimator,

$$a(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}, \tag{C.2}$$

and the path integral to be solved,

$$d(\mathbf{x}_A, \mathbf{x}_B) = \left| \int_{t_A}^{t_B} \sqrt{\dot{\mathbf{x}}(t)^T \mathbf{FI}(\mathbf{x}(t)) \dot{\mathbf{x}}(t)} dt \right|. \tag{C.3}$$

Combining Eqs. (C.1), (C.2) and (C.3),

$$\begin{aligned} d(\mathbf{x}_A, \mathbf{x}_B) &= \left| \int_{t_A}^{t_B} \sqrt{\dot{\mathbf{x}}(t)^T \boldsymbol{\beta}\boldsymbol{\beta}^T \dot{\mathbf{x}}(t) \hat{p}(c_1|\mathbf{x}(t))(1 - \hat{p}(c_1|\mathbf{x}(t)))} dt \right| \\ &= \left| \int_{t_A}^{t_B} \sqrt{(\boldsymbol{\beta}^T \dot{\mathbf{x}}(t))^2 \hat{p}(c_1|\mathbf{x}(t))(1 - \hat{p}(c_1|\mathbf{x}(t)))} dt \right|. \end{aligned} \tag{C.4}$$

The integration variable is now changed from t to $a(\mathbf{x}(\mathbf{t}))$, a for brevity,

$$a = \boldsymbol{\beta}^T \mathbf{x}(t) + \beta_0 \longrightarrow \frac{da}{dt} = \boldsymbol{\beta}^T \dot{\mathbf{x}}(t) \longrightarrow dt = \frac{1}{\boldsymbol{\beta}^T \dot{\mathbf{x}}(t)} da, \tag{C.5}$$

which leaves the integral in Eq. (C.4) as

$$\begin{aligned} d(\mathbf{x}_A, \mathbf{x}_B) &= \left| \int_{a(\mathbf{x}_A)}^{a(\mathbf{x}_B)} \sqrt{(\boldsymbol{\beta}^T \dot{\mathbf{x}}(t))^2 \hat{p}(c_1|\mathbf{x}(t))(1 - \hat{p}(c_1|\mathbf{x}(t)))} \frac{1}{\boldsymbol{\beta}^T \dot{\mathbf{x}}(t)} da \right| \\ &= \left| \int_{a(\mathbf{x}_A)}^{a(\mathbf{x}_B)} \sqrt{\hat{p}(c_1|\mathbf{x}(t))(1 - \hat{p}(c_1|\mathbf{x}(t)))} da \right|. \end{aligned} \tag{C.6}$$

Inserting the expression of the output of the linear estimator considered,

$$\hat{p}(c_1|\mathbf{x}) = \frac{1}{1 + \exp(-a(\mathbf{x}))}, \tag{C.7}$$

into Eq. (C.6) yields

$$d(\mathbf{x}_A, \mathbf{x}_B) = \left| \int_{a(\mathbf{x}_A)}^{a(\mathbf{x}_B)} \frac{\exp\left(-\frac{a}{2}\right)}{1 + \exp(-a)} da \right|. \tag{C.8}$$

A second variable change is carried out, this time it is b for a,

$$b = \exp\left(-\frac{a}{2}\right) \longrightarrow \frac{db}{da} = -\frac{1}{2} \exp\left(-\frac{a}{2}\right) = -\frac{1}{2} b \longrightarrow da = -\frac{2}{b} db, \tag{C.9}$$

which results in

$$d(\mathbf{x}_A, \mathbf{x}_B) = \left| -2 \int_{b(\mathbf{x}_A)}^{b(\mathbf{x}_B)} \frac{1}{1 + b^2} db \right| = 2 \left| [\arctan(b)]_{b(\mathbf{x}_A)}^{b(\mathbf{x}_B)} \right|. \tag{C.10}$$

The final step is to undo the last variable change, leaving the expression in terms of the generalised estimator $a(\mathbf{x})$,

$$d(\mathbf{x}_A, \mathbf{x}_B) = 2 \left| [\arctan(a(\mathbf{x}))]_{a(\mathbf{x}_A)}^{a(\mathbf{x}_B)} \right|. \tag{C.11}$$

References

[1] A. Vellido, J.D. Martín-Guerrero, and P.J.G. Lisboa. Making machine learning models interpretable. In *Proceedings of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 163–172, 2012.

[2] J. Bring. How to standardize regression coefficients. *The American Statistician*, 48(3):209–213, 1994.

[3] F. Mosteller and J.W. Tukey. *Data analysis and regression.* Addison-Wesley, 1977.

[4] M.W. Kattan, J.A. Eastham, A.M. Stapleton, T.M. Wheeler, and P.T. Scardino. A preoperative nomogram for disease recurrence following radical prostatectomy for prostate cancer. *Journal of the National Cancer Institute*, 90(10):766–71, 1998.

[5] M. d'Ocagne. *Traité de nomographie.* Gauthier-Villars, Paris, 1899.

[6] I. Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[7] A. Saltelli, K. Chan, and E.M. Scott. *Sensitivity analysis.* Wiley, 2000.

[8] V. Van Belle, B. Van Calster, D. Timmerman, T. Bourne, C. Bottomley, L. Valentin, P. Neven, S. Van Huffel, J.A.K. Suykens, and S. Boyd. A mathematical model for interpretable clinical decision support with applications in gynecology. *PLoS One*, 7(3), 2012.

[9] V. Vapnik. *Statistical learning theory.* Wiley, New York, 1998.

[10] J. Huysmans, B. Beasen, and J. Vanthienen. *Using rule extraction to improve the comprehensibility of predictive models.* Research report. Department of Decision Sciences and Information Management, K.U. Leuven, Belgium, 2006.

[11] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA, 1984.

[12] R. Andrews, J. Diederich, and A.B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, pages 373–389, 1995.

[13] T.A. Etchells and P.J.G. Lisboa. Orthogonal search-based rule extraction (OSRE) for trained neural networks: a practical and efficient approach. *IEEE Transactions on Neural Networks*, 17(2):374–384, 2006.

[14] T.S. Rögnvaldsson, T.A. Etchells, L. You, D. Garwicz, I.H. Jarman, and P.J.G. Lisboa. How to find simple and accurate rules for viral protease cleavage specificities. *BMC Bioinformatics*, 10, 2009.

[15] J.-S.R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[16] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[17] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15:116–132, 1985.

[18] J.-S.R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3):665–685, 1993.

[19] E.O. Madu, V. Stalbovskaya, B. Hamadicharef, E.C. Ifeachor, S. Van Huffel, and D. Timmerman. Preoperative ovarian cancer diagnosis using neuro-fuzzy approach. In *Proceedings of the European Conference on Emergent Aspects in Clinical Data Analysis*, pages 1–8, 2005.

[20] Z.Q. Gu and S.O. Oyadiji. Application of MR damper in structural control using ANFIS method. *Comput. Struct.*, 86(3-5):427–436, 2008.

[21] A. Khan, L. Sun, and E. Ifeachor. Content-based video quality prediction for MPEG4 video streaming over wireless networks. *Journal of Multimedia*, 4(4):228–239, 2009.

[22] S.L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[23] J. Whittaker. *Graphical models in applied multivariate statistics*. Wiley, Chichester, 1990.

[24] D. Bacciu, T.A. Etchells, P.J.G. Lisboa, and J. Whittaker. Efficient identification of independence networks using mutual information. *Computational Statistics*, pages 1–26, 2012.

[25] H. Carlin, I.H. Jarman, S. Chambers, P.J.G. Lisboa, S. Knuckey, C. Perkins, and M.A. Bellis. North West mental wellbeing survey - what influences wellbeing? *NHS North West commissioned report by the North West Public Health Observatory*, May 2011.

[26] J.H. Friedman. Flexible metric nearest neighbor classification. Technical report, 1994.

[27] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[28] K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[29] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 1994.

[30] W.S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[31] G. Young and A.S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3(1):19–22, 1938.

[32] J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 18(5):401–409, 1969.

[33] J. de Leeuw. Applications of convex analysis to multidimensional scaling. In *Recent Developments in Statistics*, pages 133–146. North Holland Publishing Company, Amsterdam, 1977.

[34] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[35] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[36] T. Cover and P. Hart. Nearest neighbor pattern classification. 13:21–27, 1967.

[37] E. Fix and J.L. Hodges. Discriminatory analysis, nonparametric discrimination: Consistency properties. *US Air Force School of Aviation Medicine*, Technical Report 4(3), 1951.

[38] K.Q. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, 2006.

[39] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, pages 513–520. MIT Press, 2004.

[40] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2002.

[41] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936.

[42] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616, 1996.

[43] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *Proceedings of the International Conference on Machine Learning*, 2003.

[44] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 776–792. Springer-Verlag, 2002.

[45] C. Domeniconi, J. Peng, and D. Gunopulos. Adaptive metric nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1281–1285, 2002.

[46] C. Domeniconi and D. Gunopulos. Adaptive nearest neighbor classification using support vector machines. In *Advances in Neural Information Processing Systems 14*, pages 665–672. MIT Press, 2001.

[47] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT'92)*, pages 144–152. ACM Press, 1992.

[48] C.J.C. Burges. Geometry and invariance in kernel based methods. In *Advances in Kernel Methods: Support Vector Learning*, pages 89–116. MIT Press, 1999.

[49] S. Amari and S. Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12:783–789, 1999.

[50] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, pages 367–373. MIT Press, 2001.

[51] J.T. Kwok and I.W. Tsang. Learning with idealized kernels. *Proceedings of the Twentieth International Conference on Machine Learning*, pages 400–407, 2003.

[52] T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11)*, pages 487–493. The MIT Press, 1998.

[53] S. Amari. *Differential-Geometrical Methods in Statistics (Lecture Notes in Statistics 28)*. Springer, 1985.

[54] C.R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bull. Calcutta Math. Soc.*, 1945.

[55] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[56] S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

[57] K.M. Carter, R. Raich, W.G. Finn, and A.O. Hero III. FINE: Fisher information nonparametric embedding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(11):2093–2098, 2009.

[58] S. Kaski and J. Sinkkonen. Metrics that learn relevance. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN-2000)*, volume 5, pages 547–552, 2000.

[59] S. Kaski, J Sinkkonen, and J. Peltonen. Bankruptcy analysis with self-organizing maps in learning metrics. *IEEE Transactions on Neural Networks*, 12:936–947, 2001.

[60] J. Peltonen, A. Klami, and S. Kaski. Improved learning of Riemannian metrics for exploratory analysis. *Neural Networks*, 17(8-9):1087–1100, 2004.

[61] Mark Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, 2010.

[62] M.E.J. Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[63] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(9), 2005.

[64] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[65] B.W. Kernighan and S. Lin. An eflicient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49:291–307, 1970.

[66] A. Pothen, H.D. Simon, and K.P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[67] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[68] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[69] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[70] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2), 2004.

[71] L.C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[72] M.E.J Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), 2004.

[73] J.R. Tyler, D.M. Wilkinson, and B.A. Huberman. Email as spectroscopy: automated discovery of community structure within organizations. In *Proceedings of the first international conference on communities and technologies*, 2003.

[74] M.J. Rattigan, M. Maier, and D. Jensen. Graph clustering with network structure indices. In *Proceedings of the 24th international conference on Machine learning*, pages 783–790. ACM, 2007.

[75] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[76] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6), 2004.

[77] K. Wakita and T. Tsurumi. Finding community structure in mega-scale social networks. *arXiv preprint cs/0702048*, 2007.

[78] J.M. Pujol, J. Béjar, and J. Delgado. Clustering algorithm for determining community structure in large networks. *Physical Review E*, 74(1), 2006.

[79] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008.

[80] R. Guimera and L.A.N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

[81] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[82] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical review E*, 72(2), 2005.

[83] M.E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[84] C.M. Bishop. *Pattern recognition and machine learning*. Springer New York, 2006.

[85] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[86] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[87] M.E. Tipping. Sparse Bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244, 2001.

[88] H. Ruiz, T.A. Etchells, I.H. Jarman, J.D. Martín-Guerrero, and P.J.G. Lisboa. A principled approach to network-based classification and data representation. *Neurocomputing*, 112(0):79–91, 2013.

[89] T. Jebara. *Machine Learning: Discriminative and Generative*. Kluwer, 2003.

[90] A.Y. Ng and M.I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. pages 841–848, 2001.

[91] H. Ruiz, S. Ortega-Martorell, I.H. Jarman, J.D. Martín-Guerrero, and P.J.G. Lisboa. Constructing similarity networks using the Fisher information metric. In *Proceedings of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2012.

[92] H. Ruiz, I.H. Jarman, J.D. Martín-Guerrero, and P.J.G. Lisboa. The role of Fisher information in primary data space for neighbourhood mapping. In *Proceedings of the 19th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2011.

[93] H. Ruiz, S. Ortega-Martorell, I.H. Jarman, A. Vellido, J.D. Martín-Guerrero, E. Romero, and P.J.G. Lisboa. Towards interpretable classifiers with blind signal separation. In *Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2012.

[94] A. Tsymbal, G. Rendes, M. Huber, and S.K. Zhou. HeC CaseReasoner: Neighborhood graph for clinical case retrieval and decision support. 2009.

[95] W.S. McCulloch and W.H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, 5(4):115–133, 1943.

[96] Y. LeCun. A learning scheme for asymmetric threshold networks. *Proceedings of Cognitiva*, 85:599–604, 1985.

[97] D.E. Rumelhart, G.E. Hintont, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[98] P. Werbos. Beyond regression: new fools for prediction and analysis in the behavioral sciences. *PhD thesis, Harvard University*, 1974.

[99] H. Cramér. *Mathematical methods of statistics*, volume 9. Princeton University Press, 1946.

[100] K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.

[101] A. Demiriz, K. Bennett, and M.J. Embrechts. Semi-supervised clustering using genetic algorithms. In *In Artificial Neural Networks in Engineering (ANNIE-99)*, pages 809–814. ASME Press, 1999.

[102] Z.H. Zhou and Y. Jiang. NeC4.5: neural ensemble based C4.5. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):770–773, 2004.

[103] X. Llora, D.E. Goldberg, I. Traus, and E. Bernadó. Accuracy, parsimony, and generality in evolutionary learning systems via multiobjective selection. In *Learning Classifier Systems*, pages 118–142. Springer, 2003.

[104] J. Eggermont, J.N. Kok, and W.A. Kosters. Genetic programming for data classification: Partitioning the search space. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1001–1005. ACM, 2004.

[105] G. Fung, M. Dundar, J. Bi, and B. Rao. A fast iterative algorithm for fisher discriminant using heterogeneous kernels. In *Proceedings of the 21st international conference on Machine learning*. ACM, 2004.

[106] P.J. Tan and D.L. Dowe. MML inference of oblique decision trees. In *Proceedings of the 17th Australian joint conference on Advances in Artificial Intelligence*, pages 1082–1088. Springer-Verlag, 2004.

[107] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, and N. Koudas. Non-linear dimensionality reduction techniques for classification and visualization. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 645–651. ACM, 2002.

[108] S. Mutter, M. Hall, and E. Frank. Using classification to evaluate the output of confidence-based association rule mining. In *AI 2004: Advances in Artificial Intelligence*, pages 538–549. Springer, 2005.

[109] R.P. Gorman and T.J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural networks*, 1(1):75–89, 1988.

[110] P. Zhong and M. Fukushima. Regularized nonsmooth Newton method for multi-class support vector machines. *Optimisation Methods and Software*, 22(1):225–236, 2007.

[111] C. Ding, T. Li, and M.I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55, 2010.

[112] S. Ortega-Martorell, P.J.G. Lisboa, A. Vellido, M. Julià-Sapé, and C. Arús. Non-negative matrix factorisation methods for the spectral decomposition of MRS data from human brain tumours. *BMC bioinformatics*, 13(1), 2012.

[113] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[114] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

Publications

This section contains a list of the works published during the research period leading to this thesis, displayed in inverse chronological order. A copy of the articles is included.

- **A principled approach to network-based classification and data representation** [88].
  H. Ruiz, T.A. Etchells, I.H. Jarman, J.D. Martín-Guerrero, and P.J.G. Lisboa.
  Neurocomputing, 2013.

- **Towards interpretable classifiers with blind signal separation** [93].
  H. Ruiz, S. Ortega-Martorell, I.H. Jarman, A. Vellido, J.D. Martín-Guerrero, E. Romero, and P.J.G. Lisboa.
  Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN), 2012.

- **Constructing similarity networks using the Fisher information metric** [91].
  H. Ruiz, S. Ortega-Martorell, I.H. Jarman, J.D. Martín-Guerrero, and P.J.G. Lisboa.
  Proceedings of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 2012.

- **The role of Fisher information in primary data space for neighbourhood mapping** [92].
  H. Ruiz, I.H. Jarman, J.D. Martín-Guerrero, and P.J.G. Lisboa.
  Proceedings of the 19th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 2011.

Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A principled approach to network-based classification and data representation

Héctor Ruiz [a,*], Terence A. Etchells [a], Ian H. Jarman [a], José D. Martín [b], Paulo J.G. Lisboa [a]

[a] Department of Mathematics and Statistics, School of Computing and Mathematical Sciences, Liverpool John Moores University, James Parsons Building, L3 3AF Liverpool, United Kingdom

[b] Department of Electronic Engineering, University of Valencia, Campus de Burjassot-Paterna, 46100 Burjassot, Valencia, Spain

ARTICLE INFO

Keywords:
Dataset visualisation
Interpretable model
Fisher information
Riemannian metric
Similarity network
Community detection

ABSTRACT

Measures of similarity are fundamental in pattern recognition and data mining. Typically the Euclidean metric is used in this context, weighting all variables equally and therefore assuming equal relevance, which is very rare in real applications. In contrast, given an estimate of a conditional density function, the Fisher information calculated in primary data space implicitly measures the relevance of variables in a principled way by reference to auxiliary data such as class labels. This paper proposes a framework that uses a distance metric based on Fisher information to construct similarity networks that achieve a more informative and principled representation of data. The framework enables efficient retrieval of reference cases from which a weighted nearest neighbour classifier closely approximates the original density function. Importantly, the identification of nearby data points permits the retrieval of further information with potential relevance to the assessment of a new case. The practical application of the method is illustrated for six benchmark datasets.

1. Introduction

One of the obstacles found in data mining applications is the fact that high-dimensional data cannot be visualised directly, making it difficult to observe data structure and to interpret derived models. This lack of interpretability limits insights into the data structure and will likely reduce the practical usefulness of the methods proposed. This aspect is of paramount importance in many practical applications, in particular in medicine and commerce, where the end user is often unfamiliar with the details of data analysis methodologies and therefore will be more prone to accept recommendations if they are expressed using their own language of expertise.

We propose a framework that, from a dataset with indicator labels, produces an intuitively interpretable representation of it in the form of a similarity network informed by a given query about binary or multiclass assignment. Data points are represented by nodes connected by edges whose strength depends on pairwise similarity with respect to a conditional density function. The underlying structure of the network reflects the statistical geometry of the original data space as determined by the density function estimates. It is then straightforward to visualise similarity even for high dimensional data. Furthermore, this enables an informative access to the dataset, or case-based retrieval, showing each observation in the context of the most similar instances and globally through its position within the network as a whole.

However, if a network is to be built on the basis of similarity between data points, a rigorous definition of similarity is required. Measures of similarity are central to pattern recognition and data mining methodologies, although they are not always explicitly calculated. For instance, using a distance function to measure similarity between pairs of elements of a space is an intuitive way to understand their relationship in the context of a particular problem domain. Where explicit metrics are used in practice, the Euclidean distance is a common choice because of its simplicity and cheap computational cost, often overlooking the equal weighting of each dimension that it implies. However, this can be very misleading, especially in practical high-dimensional applications, where it is usual to find variables that are partially or totally irrelevant to the topic of interest. An approach to deal with this difficulty is the widespread application of feature selection, but this represents a binary filter which does not show the relative similarity between individual observations.

Fisher information (FI) is a well-known measure of the relevance of a parameter to a probability function. It can be used to derive a local Fisher metric in either parameter [1] or primary data space [2]. Previous work defined the Fisher kernel with reference to generative models using a Fisher score obtained by differentiating the logarithm of the density function $p(x|\theta)$ with

* Corresponding author.
E-mail addresses: H.Ruiz@2010.ljmu.ac.uk (H. Ruiz), T.A.Etchells@ljmu.ac.uk (T.A. Etchells), I.H.Jarman@ljmu.ac.uk (I.H. Jarman), Jose.D.Martin@uv.es (J.D. Martín), P.J.Lisboa@ljmu.ac.uk (P.J.G. Lisboa).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.12.050

respect to the model parameters [2]. This was then inserted into discriminative models including regularised logistic regression classifiers.

We show in Appendix A that the assumptions implied in the definition of the Fisher Information apply also in the space of the covariates. This defines a similarity measure between point pairs by the symmetric divergence between the posterior distributions $p(c|x)$ of classifiers fitted to the class labels, which are categorised by a discrete random variable C. The concept of a metric defined by differentiating a posterior distribution $p(c|x)$ with respect to the coordinates is reported in [3] as a natural extension of the metric defined in parameter space e.g. in [4]. This paper also showed that geodesic distances between infinitesimal proximal points are invariant under differentiable, invertible transformations of the coordinate space. However, there is no discussion of the validity of the assumptions made in [4] regarding the connection of the Kullback–Leibler divergence with the Fisher metric. Similarly, FI in primary data space is applied to Bayesian logistic regression in [5], again assuming the standard form in [4] but switching from the space of model parameters to that of coordinates of the data.

This paper proposes a local metric using the FI to measure the information that an infinitesimal disturbance about a point in primary data space carries about the posterior distribution fitted to the auxiliary data. In effect, this metric rescales each covariate dimension, expanding those corresponding to informative features for classification and compressing the rest. By using this metric to measure similarity between data pairs, a network can be built that maps the arrangement of the dataset in the high-dimensional Riemannian space defined by the Fisher metric.

The structure of these "Fisher networks" is analysed in this work with Newman's algorithm for community detection in networks [6]. This technique uses the spectral properties of the network to identify the set of partitions that best divide the structure according to a well-defined measure of modularity, resulting in nodes grouped in communities in a similar way that cluster analysis groups data samples together in the original data space. However, two important benefits arise from the use of networks: first, the visualisation of the data is straightforward regardless of the number of covariates and, second, the result of the algorithm is stable and not dependent on initialisation.

Several real-world datasets are used to illustrate the application of the framework. The result is a visualisation tool with a novel perspective that automatically structures the data using only the relevant contributions of the covariates to the class membership probability and discards any unnecessary information.

2. Methodology

The first half of this section describes the calculation of distances between points in the data space under the Fisher metric, highlighting the most important expressions along the process. Density estimation is carried out using a discriminative approach. Previous work on the Fisher metric in the data space used generative models in conjunction with Bayes' theorem to estimate the posterior probability function $p(c|x)$ [3,7,8]. By using a linear logistic regressor (LLR) or a multilayer perceptron (MLP), direct density estimation is avoided as an intermediate step rendering metric learning more computationally efficient.

The intention of this work is not to contribute to the debate of whether generative or discriminative models are to be preferred in machine learning applications. There is plenty of literature available on that topic, e.g. [9,10]. This section shows how the metric can be efficiently constructed from discriminative estimators, providing an alternative to the generative approach used elsewhere.

The second part of the section discusses the process followed to map a distance measure onto a similarity network of data points. The method used to extract the community structure of the resulting networks is briefly described, explaining the principles that make this algorithm useful in practice.

The main original contribution of this paper is the integration of an FI metric and a network community extraction algorithm to produce a framework for an intelligent visualisation of data. Additional novelties are the derivation of the FI metric in the space of the data using discriminative estimators like the MLP and LLR (for which an analytical expression of the distances is presented), a detailed interpretation of the meaning of the FI in primary data space in the context of classification and the use of a simple 2D Sammon projection based on Fisher distances to provide a tidy representation of the community structure of a given dataset.

2.1. The Fisher metric

There exists a well-known relationship between the FI metric and the Kullback–Leibler (KL) divergence, detailed in [4] when the parameter vectors θ_1 and θ_2 are neighbouring points in the parameter space. Starting from the definition of the divergence and applying a Taylor expansion, it is shown that the divergence is related to a Riemannian metric in the space of the parameters determined by the FI matrix.

An analogous derivation of that relationship can be carried out in the input space, now using neighbouring points $x_1=x$ and $x_2=x+\Delta x$. Given a posterior probability $p(c|x)$, assumed to have been fitted using empirical data, the KL divergence is given by

$$I(x : x+\Delta x) = -\sum_c p(c|x)\log\frac{p(c|x+\Delta x)}{p(c|x)}\,. \tag{1}$$

The reader is referred to Appendix A for a thorough description of the process followed to establish the relationship between Eq. (1) and the differential form of the Fisher metric

$$I(x : x+\Delta x) = \tfrac{1}{2}\, d(x,x+\Delta x)^2 = \tfrac{1}{2}\Delta x^T G(x)\Delta x, \tag{2}$$

where $G(x)$ is the FI matrix, defined in the context of the present work as

$$\begin{aligned} G(x) &= \sum_c (\nabla_x \log p(c|x))(\nabla_x \log p(c|x))^T p(c|x) \\ &= -\sum_c \nabla_x^2 \log p(c|x) p(c|x). \end{aligned} \tag{3}$$

Again, detail on the meaning of these expressions and their derivation can be found in the appendix. As a summary, Eq. (2) shows that the distance between two nearby points x_1 and x_2 under the Fisher metric is equivalent to the divergence between the corresponding probabilities $p(c|x_1)$ and $p(c|x_2)$. Therefore, directions of the space along which infinitesimal displacements produce large variations of the posterior $p(c|x)$ will have large differential Fisher distances, indicating high relevance with respect to the external class label and vice versa.

Table 1
Summary of the datasets used in the experiments.

Dataset	Sonar	Ionosphere	Liver	Diabetes	Wine	Glass
Samples	208	351	345	768	178	214
Dimensions	60	32	6	8	13	9
Classes	2	2	2	2	3	6

2.1.1. FI matrix for binary classifiers

The definition of the FI matrix in Eq. (3) assumes an analytical expression for the posterior probability function $p(c|x)$. This function must be estimated from the data and must be everywhere differentiable. The derivation in this section applies to estimators with sigmoid link functions, thus deriving conditional density functions parameterised by

$$p(c|x) = \frac{c+(1-c)e^{-a(x)}}{1+e^{-a(x)}}, \quad c=\{0,1\}. \tag{4}$$

Taking derivatives of the logarithm of the probability distribution in Eq. (4) and inserting them into either of the definitions of the FI matrix in Eq. (3) yields

$$G(x) = (\nabla_x a(x))(\nabla_x a(x))^T p(c=1|x)(1-p(c=1|x)), \tag{5}$$

where $a(x)$ denotes the generalised linear estimator a evaluated at $X=x$, namely

$$a(x) = \beta^T x + \beta_0 \text{ (LLR)} \tag{6}$$

$$a(x) = W2 \cdot \Phi(W1 \cdot x + B1) + B2 \text{ (MLP)}, \tag{7}$$

where β and β_0 are the regression coefficients of the LLR; $W1$, $W2$, $B1$ and $B2$ are the synaptic weights of the MLP and Φ is the sigmoid function.

2.1.2. FI matrix for multinomial classifiers

The derivation extends naturally to multiple classes, $N_C \geq 2$, by representing the posterior probability function $p(c_i|x)$ for each class by the multinomial function

$$p(c_i|x) = \frac{e^{a_i(x)}}{\sum_j^{N_C} e^{a_j(x)}}. \tag{8}$$

As in Section 2.1.1, taking logarithms and derivatives leads to the expression of the FI matrix, resulting in the equivalent of Eq. (5) for more than two classes

$$G(x) = \sum_i^{N_C}\sum_j^{N_C}\sum_k^{N_C} \nabla_x(a_i-a_j)\cdot\nabla_x(a_i-a_k)^T p(c_i|x)p(c_j|x)p(c_k|x). \tag{9}$$

2.1.3. Distance calculation with linear estimators

Given the FI matrix, the infinitesimal distance between a pair of neighbouring points in the data space is given by the quadratic differential form in Eq. (2), which can be integrated to calculate the path integral distance between any pair of points x_A and x_B

$$d(x_A,x_B) = \left| \int_0^1 \sqrt{\dot{x}(t)^T G(x(t))\dot{x}(t)}dt \right|, \tag{10}$$

where $x(t)$ represents a path in the space that goes from $x_A=x(t=0)$ to $x_B=x(t=1)$. When estimating $p(c|x)$ using an LLR, the integral can be solved analytically (see Appendix B) resulting in the following closed-form expression:

$$d(x_A,x_B) = \left| 2\left[\arctan\left(e^{-a(x(t))/2}\right)\right]_{a(x_A)}^{a(x_B)} \right|, \tag{11}$$

which is independent of the particular path from x_A to x_B. In other words, any path that joins x_A and x_B have the same length for the Riemannian metric defined by the FI.

Although this may seem counter-intuitive, it becomes clear when the detail of the method is scrutinised more closely. When the probability function is estimated using a binary model from the exponential family with a linear score function, the whole data space in effect collapses into a straight line. This is because, in the linear case, the score $a(x)$ only takes into account displacements along a single direction in the space. For the particular case of LLR, this direction is given by the vector of regression coefficients β due to the dot product $\beta^T x$ present in Eq. (6). The result is that the posterior probability can only vary along that direction. The Fisher metric focuses on variations of this probability, so for any given pair of points it is only the distance between their projections onto the vector β what determines the distance between them in the Riemannian space. In other words, because only one direction of the space is relevant, every path that connects the points will have the same length regardless of its shape.

In the case of the MLP, the activation $a(x)$ is a non-linear function of x, and consequently its first derivative depends on x. This greatly complicates the integral in Eq. (10), making it non-analytic. The same happens in the multinomial case, where the FI matrix given by Eq. (9) does not allow for an analytical solution for the path integral. Distances in these cases depend on the choice of $x(t)$, and a method is required to find the length of the shortest path i.e. the geodesic distance.

2.1.4. Efficient estimation of the geodesic distance

The estimation of the geodesic distance can be efficiently tackled using the graph approximation proposed in [7]. Although the way the authors derive the Fisher metric in that work is different from the discriminative approach followed here, both alternatives end up facing the same problem of approximating the shortest path. The method they propose is simple but effective: first, all pairwise Fisher distances between points in the dataset are obtained using straight paths and calculating numerically the integral in Eq. (10). Then, these distances are used as edges of a graph, each connecting the corresponding pair of nodes. Finally, the geodesic distance between a given pair of points in the dataset is estimated as the minimal path between their respective vertices. This can be found using the implementation of Floyd's graph search algorithm in [11].

By definition, the result of the graph approximation depends on the particular set of data points available to use as vertices. The iterative approach in [12] removes this dependence and uses only the definition of the metric to estimate geodesic distances. However, when put into practice it only provides a marginal improvement on the distances found, yet the heavy computational burden that it adds makes it only viable for small datasets.

2.2. The Fisher network

After defining the Fisher metric and the process to calculate distances under it, the last step in the construction of similarity networks is to determine the relationship between the pairwise distances between the data points and the strength of the connections between the nodes. Then, the community structure can be analysed with the spectral algorithm described in Section 2.2.2.

2.2.1. From distances to connections

The methods in Section 2.1 provide the tools to calculate an objective measure of dissimilarity between point pairs. A further step is therefore required to transform each distance value into a quantitative measure of the similarity between pairs of data points, with respect to the original estimate of the conditional density function. Assuming that zero distance maps to maximum similarity and recognising that the anisotropy of the projective space of covariates has been captured by the Fisher metric, it is natural to transform distances into similarity indicators which represent the edge weights in the network using a Gaussian radial kernel

$$A_{ij} = e^{-d(x_i,x_j)^2/\sigma^2}, \tag{12}$$

where $d(x_i,x_j)$ is the distance between the points and σ is the parameter that determines the width of the kernel. This parameter adjusts the degree of locality in the connections of the network. It is in fact a useful parameter for practical applications, as shown in Section 3.

The pairwise weights A_{ij} are grouped together in the adjacency matrix A, which is the starting point for the application of Newman's algorithm to find the community structure of the network.

2.2.2. Newman's algorithm for community detection

This subsection is a brief overview of the community structure detection algorithm proposed in [6]. The method finds a natural division of a network into subgroups, of a number and size automatically determined by the intrinsic structure of the network and the edge strengths. The objective is to partition nodes into groups in such a way that there are fewer than expected edges between the resulting groups. Accordingly, a measure of modularity is defined as the number of edges falling within groups less the expected number in an equivalent network with edges placed at random. The objective is to maximise the overall modularity of the network. In order to perform this maximisation, the modularity measure is expressed as a matrix with elements

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}, \tag{13}$$

where A_{ij} is the pairwise weight of the connection between nodes i and j given by Eq. (12), k_i is the degree of node i and m is the total number of edges in the network. The modularity of a group of nodes within the network is defined as

$$Q = \frac{1}{2m}\sum_{ij} B_{ij}, \tag{14}$$

where nodes i and j both belong to that group. Now suppose the whole network is divided into two groups in such a way that node i receives a label $s_i=1$ if it belongs to the first group and $s_i=-1$ if it falls in the second. The modularity of that division is, from Eq. (14)

$$Q = \frac{1}{2m}\sum_{ij} B_{ij}\frac{(s_i s_j+1)}{2}, \tag{15}$$

which corresponds to the following matrix form

$$Q = \frac{1}{4m} s^T B s, \tag{16}$$

where s is a column vector of the labels defined above. Therefore, the problem is to find the set of labels s (i.e. a particular binary division of the network) that maximises Eq. (16). The solution to this (see [6] for details) is given by taking $s=\mathrm{sign}(u_1)$, where u_1 is the eigenvector of B corresponding to the largest positive eigenvalue.

The process is repeated for each of the resulting partitions in a hierarchical manner. Every further split will add a quantity to the modularity score resulting from the first partition, given by Eq. (16). If this contribution is positive, the proposed partition is accepted, otherwise leaving the corresponding subgroup as it is. When a point is reached where dividing any of the current partitions deteriorates the overall modularity, the process stops.

The benefits of this algorithm are threefold: first, no specification of the number or size of the communities is needed, as they arise naturally from the network; second, the resulting communities are well-defined as indivisible subgraphs, making the stopping condition of the algorithm completely clear; and third, there is no initialisation involved in the process, so the results are completely repeatable.

2.2.3. Construction of the Fisher network

After the explanation of the methods involved in the framework, this section provides a step-by-step description of the process of building a Fisher network starting from a dataset:

(1) The first step is to train the model that is going to estimate $p(c|x)$. This means finding the coefficients β and β_0 in the case of the LLR and the weights $W1$, $W2$, $B1$ and $B2$ for the MLP (see Eqs. (6) and (7)).
(2) The FI matrix can now be calculated at any point of the data space using Eqs. (5) and (9), and therefore distances between points can be obtained by approximating Eq. (10) using the graph approximation [7] discussed in Section 2.1.4 (or using Eq. (11) directly if the estimator used is the LLR). The distance between every pair of points is calculated, and the results are put together in a square pairwise distance matrix of size the number of samples in the dataset.
(3) Using Eq. (12), each element in the distance matrix is used to generate a weight that represents the strength of the connection between a pair of points. These weights are also grouped into a matrix, and they define the structure of a fully connected network where each node corresponds to a data point in the original dataset. See Section 3.1 regarding the choice of the parameter σ in Eq. (12).
(4) As an optional step, the weakest connections in the matrix can be pruned out to avoid their influence on the resulting communities, although as Section 3.2 suggests this does not have a significant effect.
(5) The pairwise weight matrix is processed using Newman's algorithm, finding the set of communities that best modularise the network.
(6) The Fisher network is obtained from the weight matrix by pruning the edges that connect nodes from different communities and keeping those that connect nodes from the same partition.

3. Experimental results and discussion

This section reviews the construction of the similarity network from a practical point of view, applying the methods described in Section 2. Six known real-world datasets are studied, all of which are available from the UCI machine learning repository (http://archive.ics.uci.edu/ml/). These include binary classification problems (Pima Indians Diabetes, Ionosphere, Liver Disorders and Sonar datasets) as well as multinomial (Wine and Glass Identification datasets). Table 1 contains the main characteristics of the six datasets.

For the estimation of the posterior probability $p(c|x)$, an MLP regularised with weight decay is trained on a normalised version of the dataset. The network parameters are determined by the calibration of the posterior probability function, empirically estimated for test data comprising a random sample of approximately a third of the samples in each dataset. Once the posterior distribution is available, the FI matrix is calculated using Eqs. (5) and (9) and pairwise geodesic distances are estimated with the graph approximation suggested in [7] and described in Section 2.1.4. This method approximates the integral in Eq. (10) by dividing it into T pieces, and the authors show that $T=10$ is sufficient in terms of performance in their experiments. The datasets used in our work are relatively small, so we set $T=100$ for all the experiments as it is computationally affordable.

3.1. Selection of σ

The width σ of the Gaussian kernel in Eq. (12) needs to be adjusted. In order to study the effect of changing the value of this

parameter, we analyse three different measures obtained from networks built using a range of values of σ.

The first measure arises from performing a leave-one-out prediction of the posterior $p(c|x)$ for each node in the network. First, the score $a(x_i)$ of each node is estimated as a weighted sum of the rest of the nodes' scores

$$\hat{a}(x_i) = \frac{1}{\sum_j A_{ij}} \sum_j A_{ij} a(x_j). \tag{17}$$

The estimate $\hat{a}(x_i)$ forms the activation of a sigmoid function to obtain the predicted probability $\hat{p}(c|x_i)$. This is repeated for the whole dataset, and the quality of the predictions is measured by the discrete KL divergence between the estimates and the MLP probabilities used as reference

$$KL(p,\hat{p}) = -\frac{1}{L} \sum_i \sum_c p(c|x_i) \log \frac{\hat{p}(c|x_i)}{p(c|x_i)}. \tag{18}$$

The divergence is normalised using the number of samples L to make it comparable across datasets of different sizes.

The second measure we use is Cramer's V statistic (CV), which provides an evaluation of how coherent the communities found are with respect to the class labels of the data. It is calculated from the contingency table of the vector of community memberships given by Newman's algorithm and the vector of true labels, as

$$CV = \sqrt{\frac{\chi^2}{L \cdot min(C_1 - 1, C_2 - 1)}}, \tag{19}$$

where χ is the chi-squared statistic, L is the number of data points (or nodes in the network), C_1 is the number of communities and C_2 is the number of classes in the dataset. This index ranges from 0 to 1 indicating the concordance between the community allocation of the data points and the original class membership, with CV=0 meaning no association and CV=1 corresponding to complete concordance.

The last measure has to do with the classification accuracy associated to the predictions given by Eq. (17) and that of the MLP, which is used as a reference. Because the two classifiers are applied to the same data, the extent to which the differences between the errors they make can be attributed to chance is quantified with McNemar's test [13]:

$$z = \frac{|n_A - n_B| - 1}{\sqrt{(n_A + n_B)}}. \tag{20}$$

This can be approximated by a normal $N(0,1)$ distribution, so it is possible to calculate a p-value from the test and therefore define a threshold for significance, which is of interest as we would like a σ that provides a network that can retain the classification ability of the MLP.

Fig. 1 shows the three measures in Eq. (18)–(20) for the Sonar dataset and values of σ up to 5. Additionally, Fig. 1(b) includes the accuracy of the classifications used to calculate McNemar's test (network predictions and MLP) and Fig. 1(c) displays the number of communities found using Newman's algorithm. The first plot shows the KL divergence increasing with σ. This is a general tendency for all datasets: small values of σ produce networks that predict $p(c|x)$ better, which is an expected result since a small σ means the Gaussian kernel is narrow and therefore only the closest (and most similar) references will have a significant weight in Eq. (17). The second measure, Cramer's V, indicates a high level of concordance between the communities and the classes for the whole range of σ, with the lowest values producing the best results again. Finally, the third measure is indicating that there is no statistically significant difference between the performances of the two classifiers, so the p-value by itself would not

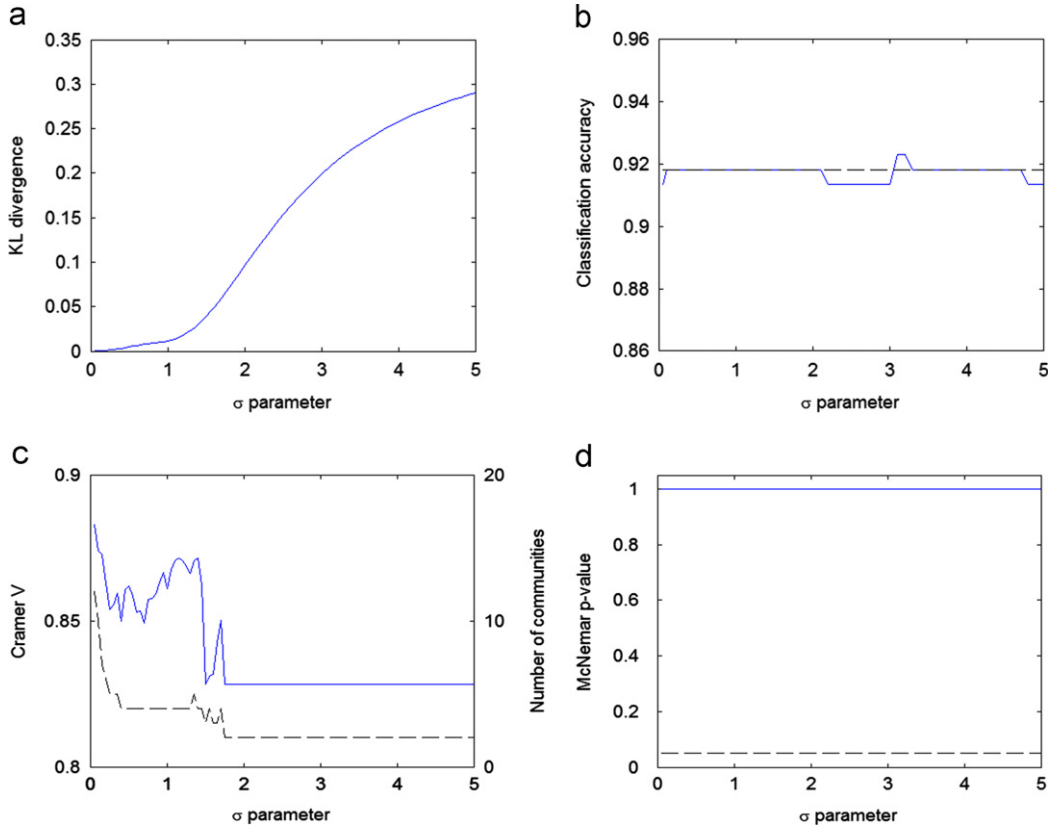


Fig. 1. Measures to aid the selection of σ, Sonar dataset. (a) KL divergence. (b) Classification accuracy (dashed line: MLP accuracy). (c) Cramer's V index (dashed line: number of communities). (d) McNemar's test p-value (dashed line: $p=0.05$).

restrict the range of values from which to choose. This is explained by Fig. 1(b), where we can see that the classification accuracy of the network is very stable and there are only a few minor deviations from that of the MLP.

Fig. 2 covers the same measures as Fig. 1, now for the Glass dataset. This is a more difficult classification task, as Fig. 2(b) depicts. The plots of the KL divergence and Cramer's V index are similar to those of the Sonar dataset, with the difference that, whereas in Fig. 1(c) the variations of the CV were relatively small, in Fig. 2(c) there is a larger difference between its maximum and minimum values. The reason for this is that the Sonar dataset is binary and its classes are easy to separate, thus being well represented with large and small numbers of communities. The Glass dataset, on the other hand, contains six different classes with more mixing, and therefore suffers a bigger CV drop when the number of communities is small, i.e. when σ is large.

In contrast to the result in Fig. 1(d), the p-value in Fig. 2(d) suffers a drop that takes it below the 0.05 threshold at around $\sigma=1.5$, the point at which the classification accuracy of the network starts falling. All three performance measures appear to weaken as σ reaches 1.5, so in practice, we take this as a restriction on the selection of its value, in this case $\sigma<1.5$.

These plots provide the user with the information required to find suitable values of σ depending on their preferences. For instance, for the experiments in next section we start by finding the smallest value of σ (if any) for which the p-value obtained from McNemar's test was less than or equal to 0.05, and use it as an upper bound. Then, we simply select the smallest σ in that range, as that will in general produce the best predictions (the smallest KL divergence) and a good CV value (if not the maximum).

This can serve as a sensible default criterion when there is no preference. However, depending on the application, the user may prefer to focus solely on coherence of the communities and select the top CV value regardless of the rest of the measures. Or the interest may be to obtain a certain number of communities, so only values of σ that produce that many partitions are considered, and then the default criterion is applied. If a good calibration of the predictions is a priority, a standard measure such as the Hosmer–Lemeshow test [14] could be applied. Whatever the interest is, a suitable measure can be incorporated to the analysis, so the end user can always obtain a solution that satisfies their needs by appropriately selecting σ.

3.2. Network examples: the Sonar dataset

In this section, the Sonar dataset is used to illustrate examples of communities extracted from different networks built from the same dataset. Four values of σ are studied: 0.05, 0.10, 0.15 and 0.20. Additionally, a second parameter is introduced: the proportion of pruned edges. After calculating the edge weights, a certain percentage of the connections in the adjacency matrix is removed, starting from those with the smallest weight. The idea is to study how the community structure of the network changes depending on the amount of edges removed. Fig. 3 contains a summary of the results.

The KL divergence in Fig. 3(a) is calculated the same way as Eq. (18), only in this case the prediction of each node's posterior is carried out using only nodes that belong to the same community. It can be understood as a measure of the prediction power of the communities. Fig. 3(b) plots the number of communities found by

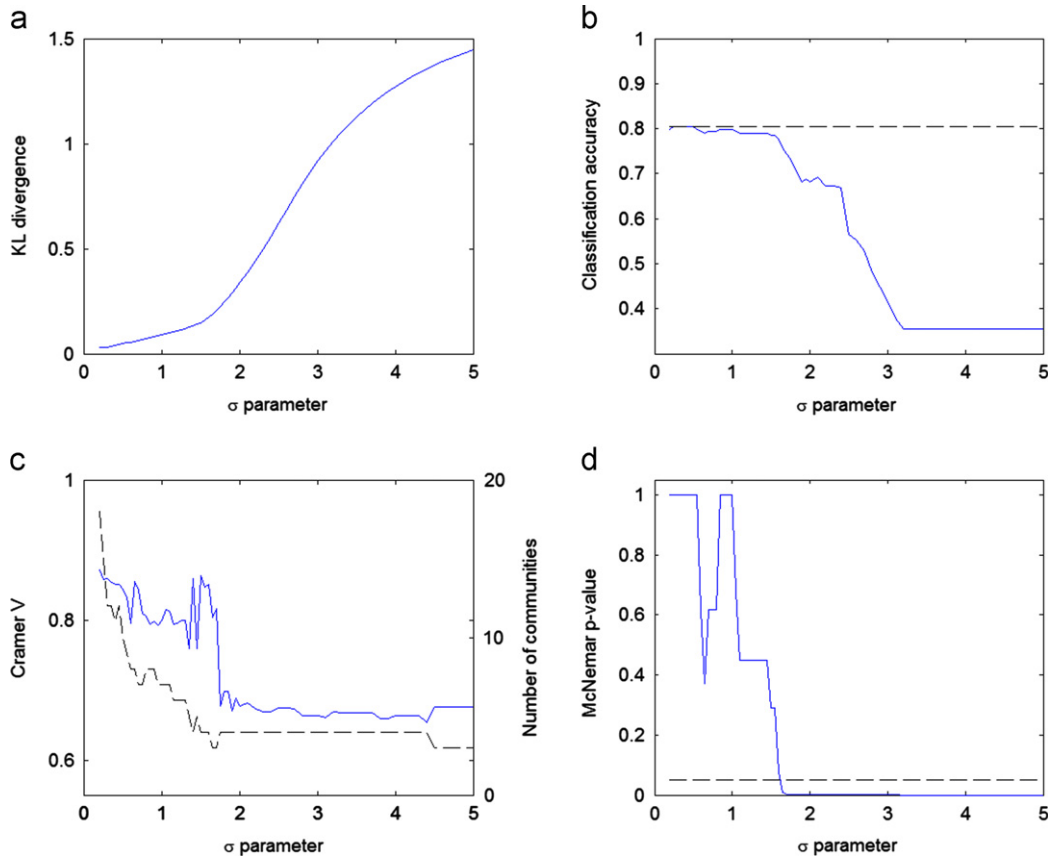


Fig. 2. Measures to aid the selection of σ, Glass dataset. (a) KL divergence. (b) Classification accuracy (dashed line: MLP accuracy). (c) Cramer's V index (dashed line: number of communities). (d) McNemar's test p-value (dashed line: $p=0.05$).

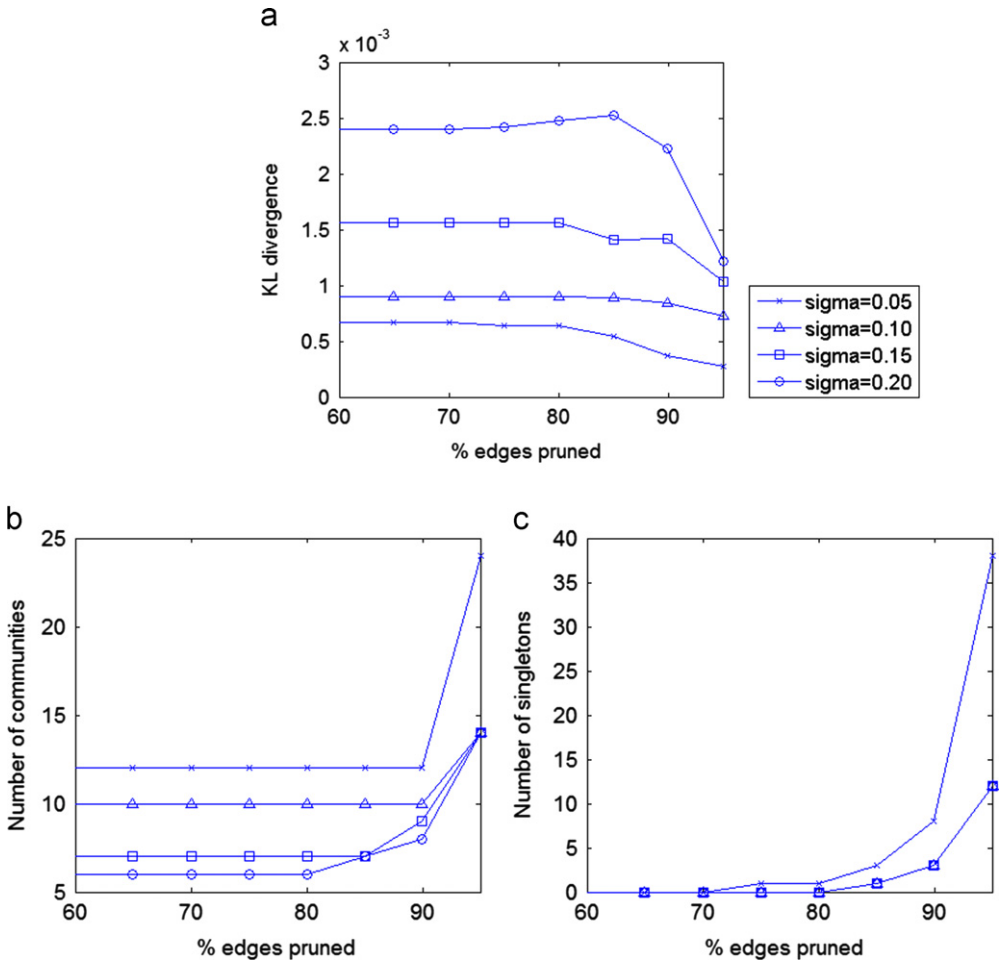


Fig. 3. Characteristics of the communities found against percentage of edges pruned, from 60% to 95%; there is no variation in any of the plots for $<60\%$ edges pruned. (a) KL divergence. (b) Number of communities. (c) Number of singletons.

the algorithm, and Fig. 3(c) shows the number of singletons, i.e. isolated nodes that are not members of any community.

The influence of σ is as expected: small values produce smaller but more numerous communities due to local connections being favoured over longer ones. As σ increases, this effect is mitigated and the number of communities decreases, resulting in larger subgraphs. Regarding the KL divergence, the observed values are in line with the plots in Figs. 1 and 2(a), with the measure increasing as σ does.

The conclusion from Fig. 3 is that edge pruning does not have a big impact on the community structure, as the outcomes remain stable for most of the pruning range. Only when the percentage of removed edges becomes very significant, around 80% in this particular dataset, it affects the resulting communities. It does so by increasing the number of communities and singletons present in the network. This is, again, a result of enforcing locality, this time by directly removing weak connections corresponding to far away points. Fig. 3(a) also shows a decrease in the KL divergence when pruning increases, but most of this reduction is explained by the appearance of singletons, as these are not taken into account by the measure.

Figs. 4–6 show three examples of the communities found in the Sonar dataset for different values of the parameters. Each node in the adjacency matrix is represented as a triangle or a cross depending on its class label and edges connect nodes that belong to the same community. To obtain a representation of the communities as tidy as possible, nodes are arranged in the plot according to a 2D Sammon projection of the dataset [15]. The

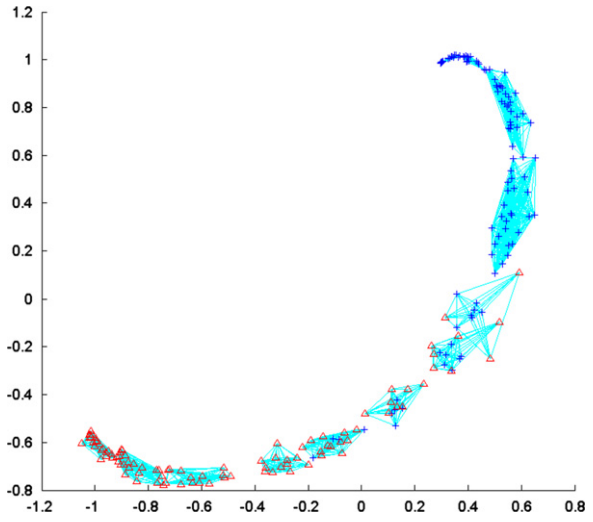


Fig. 4. Fisher network representation of the Sonar dataset. $\sigma=0.05$, no pruning. Twelve communities, 0 singletons, KL divergence$=0.00067$ (over 208 nodes). Network communities: CV$=0.8830$. k-means (mapped dataset): max CV$=0.8829$, mean CV$=0.8747$.

Sammon mapping algorithm takes a matrix of pairwise distances between points in the dataset and projects the points into a lower dimensional space. The objective is a mapped version of the dataset where pairwise distances are as close as possible to those

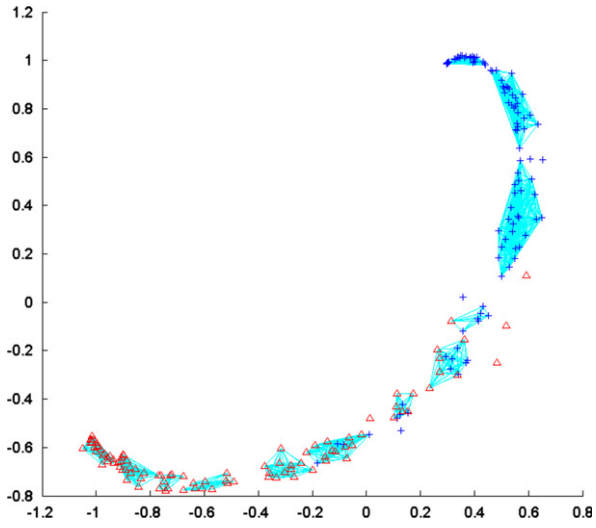


Fig. 5. Fisher network representation of the Sonar dataset. $\sigma=0.05$, 90% pruning. Twelve communities, 8 singletons, KL divergence=0.00037 (over 200 nodes). Network communities: CV=0.8996. *k*-means (mapped dataset): max CV=0.9058, mean CV=0.8966.

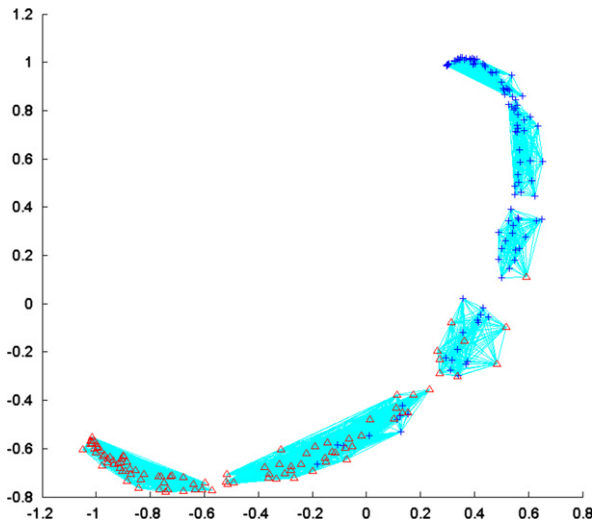


Fig. 6. Fisher network representation of the Sonar dataset. $\sigma=0.20$, no pruning. Six communities, 0 singletons, KL divergence=0.00240 (over 208 nodes). Network communities: CV=0.8627. *k*-means (mapped dataset): max CV=0.8730, mean CV=0.8617.

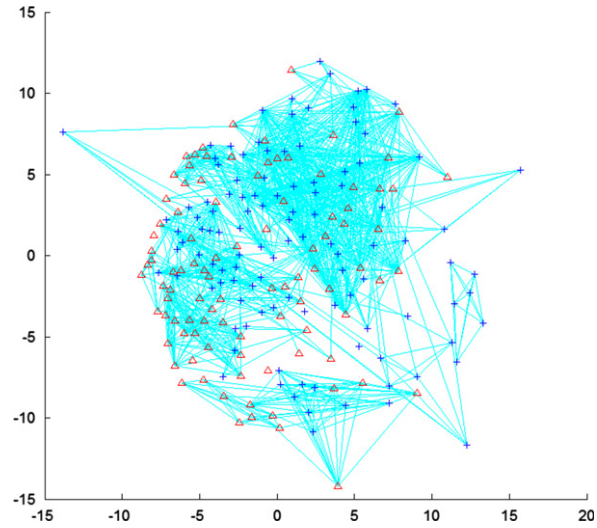


Fig. 7. Euclidean network representation of the Sonar dataset. $\sigma=2.30$, no pruning. Twenty-two communities, 7 singletons, KL divergence=0.17792 (over 201 nodes). Network communities: CV=0.6235. *k*-means (mapped dataset): max CV=0.8261, mean CV=0.7581.

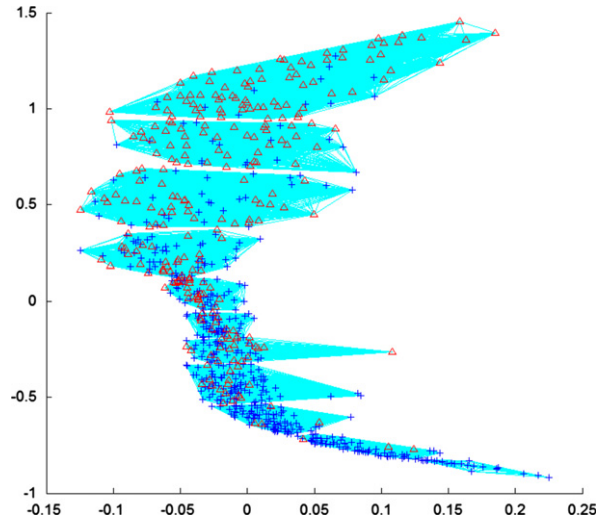


Fig. 8. Fisher network representation of the Diabetes dataset. $\sigma=0.05$. Eleven communities, 0 singletons, KL divergence=0.00012 (over 768 nodes). Network communities: CV=0.5981. *k*-means (mapped dataset): max CV=0.6027, mean CV=0.5924.

in the original high-dimensional space. Note that this algorithm is used only to determine the position of the nodes in the visualisation of the communities, and has no influence whatsoever on the resulting structure of the networks.

For each of the networks in Figs. 4–17, the set of communities found by Newman's algorithm is compared against the most widely-used clustering algorithm: *k*-means. After finding the communities for a particular case, *k*-means clustering is performed on the same dataset with k equal to the number of communities found by Newman's method. Then, both partitions of the data are compared against the true class labels using the CV index in Eq. (19) with the same objective as in Section 3.1: finding the degree of concordance between the partitions and the classes. This process is repeated 100 times for *k*-means and the maximum and mean values of the CV are reported.

For those cases where the network is built using Fisher distances, *k*-means clustering is applied on the 2D Sammon projection of the data. If the construction of the network uses Euclidean distances, then *k*-means uses the original dataset. This is to make the comparison fair in terms of the use of the class label information.

Fig. 4 is an example of a set of communities found using a small value of σ. The resulting 12 communities are quite compact, and there is little class mixing within the groups, as reflected by the high CV value. The arrangement of the nodes makes it very easy to identify which areas of the network are clear in terms of class membership, located in both ends of the "C" shape, and which correspond to the borderline cases, in the middle section of the curve.

In generating Fig. 5, 90% of the weakest connections are removed prior to running Newman's algorithm. This has a bigger impact on the border area between the classes than anywhere else in the network, as this is where points tend to be more separated between each other due to the higher variations of $p(c|x)$. As a result of that, some of the nodes there become singletons and the borderline communities decrease in size. Some

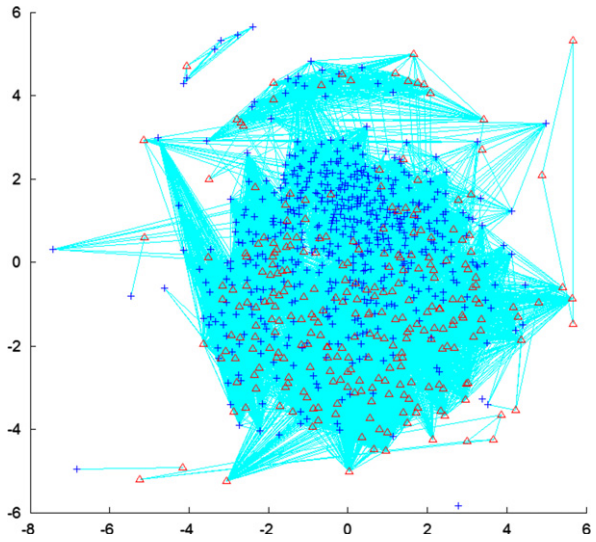


Fig. 9. Euclidean network representation of the Diabetes dataset. $\sigma=0.75$. Twenty-three communities, 4 singletons, KL divergence=0.02155 (over 764 nodes). Network communities: CV=0.4464. *k*-means (mapped dataset): max CV=0.5417, mean CV=0.5157.

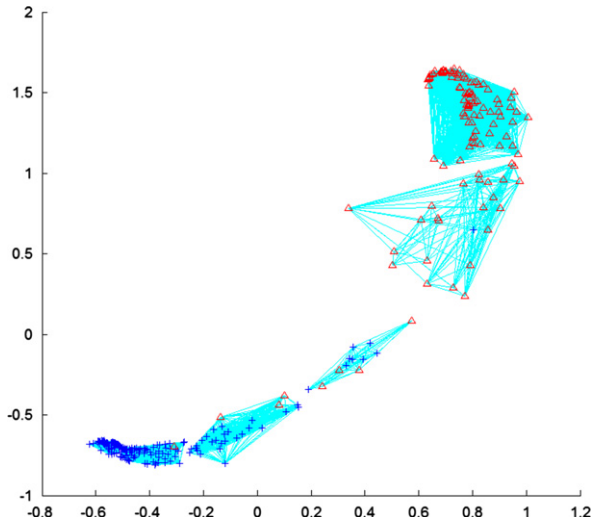


Fig. 10. Fisher network representation of the Ionosphere dataset. $\sigma=0.10$. Seven communities, 0 singletons, KL divergence=0.00147 (over 351 nodes). Network communities: CV=0.9538. *k*-means (mapped dataset): max CV=0.9579, mean CV=0.9565.

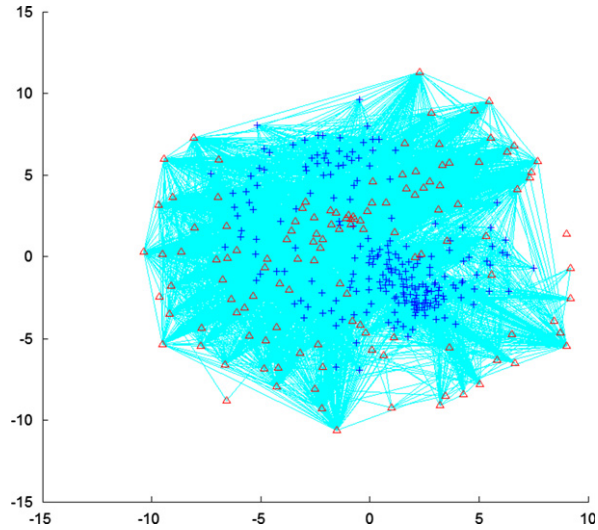


Fig. 11. Euclidean network representation of the Ionosphere dataset. $\sigma=1.80$. Ten communities, 2 singletons, KL divergence=0.22294 (over 349 nodes). Network communities: CV=0.7591. *k*-means (mapped dataset): max CV=0.8287, mean CV=0.7659.

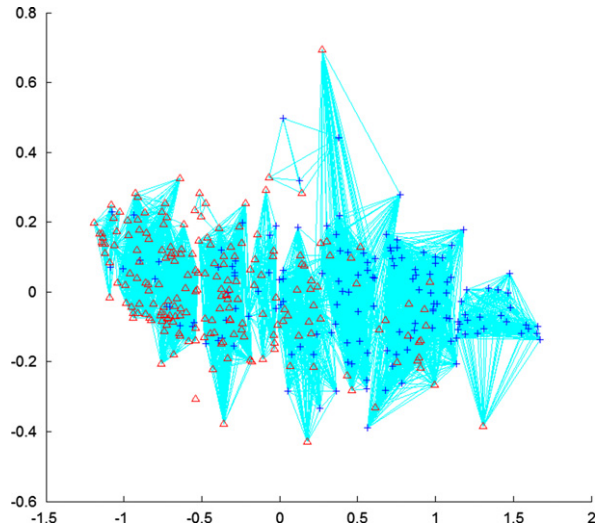


Fig. 12. Fisher network representation of the Liver dataset. $\sigma=0.15$. Eleven communities, 1 singletons, KL divergence=0.00259 (over 344 nodes). Network communities: CV=0.6033. *k*-means (mapped dataset): max CV=0.6404, mean CV=0.6113.

of the communities in the border become more homogeneous after the formation of the singletons, and this results in a slightly higher value of the CV compared to that of Fig. 4.

Fig. 6 replicates Fig. 4 with $\sigma=0.2$. As expected, the outcome is fewer communities of larger size. As a consequence, there are now only two communities covering the border area, with more class mixing within them than there was in the border communities in Figs. 4 and 5, therefore producing a smaller CV. These three figures exemplify the variation that can be obtained in the community structure of the same dataset by tuning the available parameters.

Finally, Fig. 7 is an example of a network built from the same dataset following the same construction process but using Euclidean instead of Fisher distances. The value of σ is selected so that it is as small as possible without causing numerical instabilities in Newman's algorithm. Although this method always provides the same result by definition, when applied in practice, issues can appear during the calculation of the eigenvectors and eigenvalues if the adjacency matrix contains weights that are very small compared to the largest values in the matrix (e.g. $>10^{100}$ times smaller), and this can result in inaccurate eigenvectors and consequently slightly different communities being found in consecutive runs of the algorithm. To prevent this from happening, σ is selected large enough so that there are not such extreme differences in the magnitude of the weights. The threshold value depends on the particular case, and is easy to find with an acceptable precision. In practice, this value acts as a lower limit of the range of σ resulting from Section 3.1. In this work, this minimum value goes from 0.05 to 0.20 for Fisher networks and from 0.7 to 2.30 for Euclidean networks.

Going back to Fig. 7, the communities found in the Euclidean network are nowhere near their Fisher counterparts in terms of clarity of their structure. This is reflected not only in the visualisation of the communities, but also in the KL divergence of the $p(c|x)$ estimates, several orders of magnitude bigger than the values obtained using the Fisher metric, and in the CV index, which also deteriorates substantially.

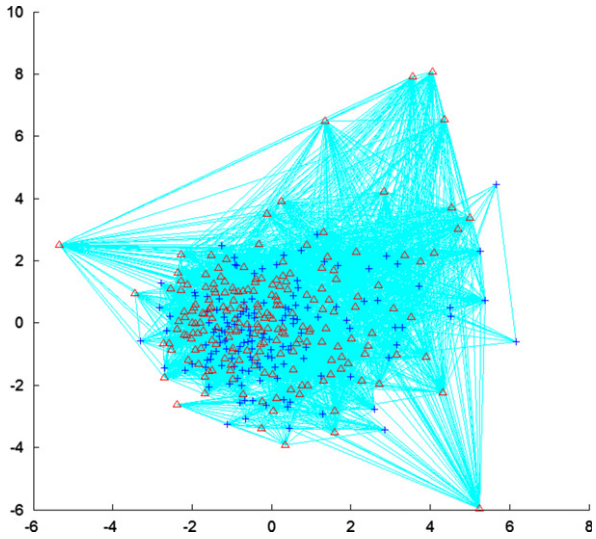


Fig. 13. Euclidean network representation of the Liver dataset. $\sigma=0.75$. Twelve communities, 0 singletons, KL divergence=0.15042 (over 345 nodes). Network communities: CV=0.2292. *k*-means (mapped dataset): max CV=0.3091, mean CV=0.2626.

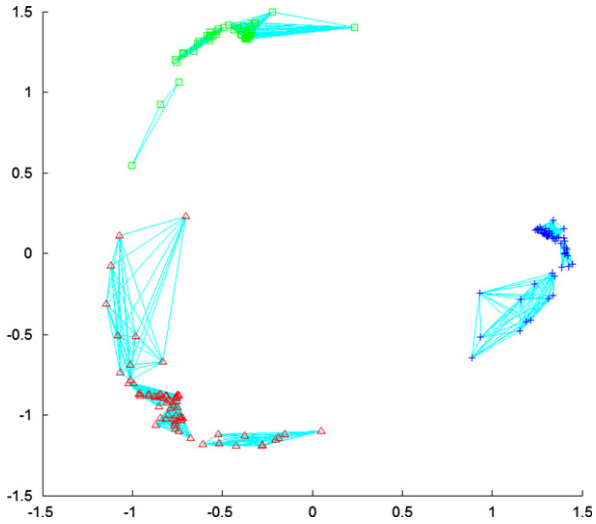


Fig. 14. Fisher network representation of the Wine dataset. $\sigma=0.10$. Ten communities, 0 singletons, KL divergence=0.00344 (over 178 nodes). Network communities: CV=1. *k*-means (mapped dataset): max CV=1, mean CV=0.9940.

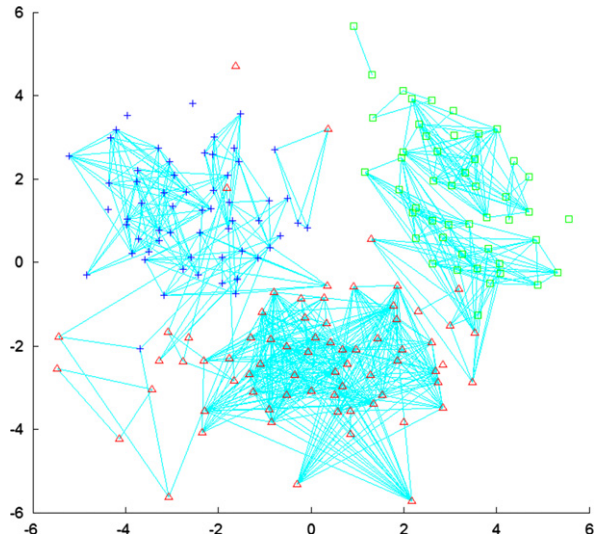


Fig. 15. Euclidean network representation of the Wine dataset. $\sigma=0.7$. Twenty communities, 9 singletons, KL divergence=0.06453 (over 169 nodes). Network communities: CV=0.9493. *k*-means (mapped dataset): max CV=0.9840, mean CV=0.9624.

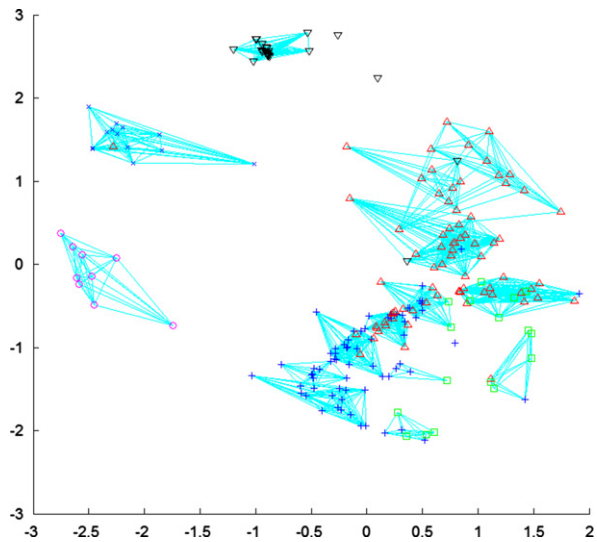


Fig. 16. Fisher network representation of the Glass dataset. $\sigma=0.20$. Fifteen communities, 3 singletons, KL divergence=0.03033 (over 211 nodes). Network communities: CV=0.8712. *k*-means (mapped dataset): max CV=0.8795, mean CV=0.8529.

Regarding the comparison of Newman's algorithm and *k*-means clustering, the CV values indicate that they produce very similar partitions of the data in terms of coherence with the original class labels. For the six datasets studied, the CV values produced by the Fisher networks are very close to the mean CV value obtained from *k*-means, with the maximum value usually less than 0.01 above (with the exception of the Liver dataset, where the difference is slightly bigger). However, there are two significant advantages in the use of the network approach over *k*-means: the number of partitions is selected automatically and there is no sensibility to the initialisation of the algorithm.

When using Euclidean distances, there is a clearly noticeable difference in performance in terms of the CV in favour of *k*-means, as well as a bigger divergence between its max and mean CV values. While the latter could be indicating that the concordance of clusters and classes is more sensible to the initialisation of the centroids now that the metric does not draw similar points together, we do not see a clear explanation as to why the two methods should differ from each other now anymore than they did before. However, since the usefulness of the framework is based on the use of Fisher metric, this does not represent a critical issue.

3.3. Other datasets

Figs. 8–17 correspond to networks built using the other five datasets listed before. In all cases, a range of suitable values of σ is calculated following the guidelines in Section 3.1, and σ is set to the smallest value within that range that provides stability of the eigenvector calculation. No pruning of edges is performed in any of the cases.

The results are consistent with those in Section 3.2. Fisher networks allow for a clear visualisation of the data, with a tidier community structure and better predictions of the posterior probability than in the Euclidean case, as is reflected by the CV index and KL divergence.

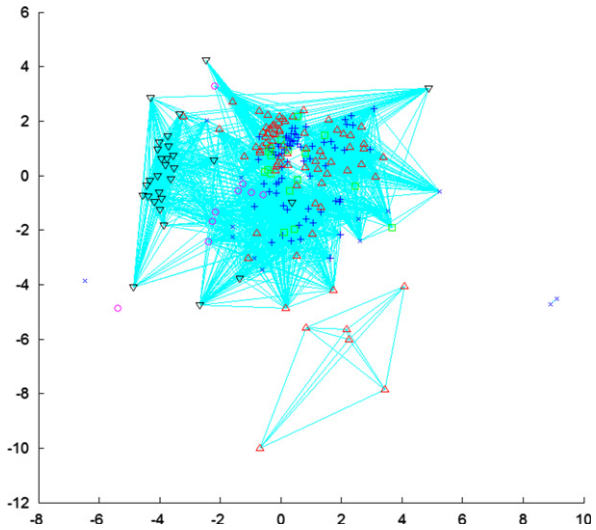


Fig. 17. Euclidean network representation of the Glass dataset. $\sigma=1.40$. Seven communities, 2 singletons, KL divergence=0.48365 (over 212 nodes). Network communities: CV=0.4915. *k*-means (mapped dataset): max CV=0.6217, mean CV=0.5574.

4. Conclusion

A framework is proposed that combines a Fisher information based similarity measure with network structure analysis algorithms to provide a rigorously principled visualisation tool for high-dimensional data. Regardless of the data domain, a global view of the dataset is produced that helps to interpret how the data instances are distributed with respect to each other and what is the underlying structure, all of that without losing sight of what the question of interest is, built into the framework by the posterior probability $p(c|x)$. The benefits of using the Fisher metric are obvious when compared to a conventional choice like the Euclidean metric.

This methodology could be potentially very useful as a semi-supervised classifier/visualisation tool, using the labelled portion of the data to estimate the probability and derive the Fisher metric in order to later build a relational network containing both labelled and unlabelled samples. For example, in a medical application, this would provide doctors not only the basic information about class membership prediction of an unlabelled case, but also its location within the network, the set of known cases it is more similar to, the subdivision of the network that it belongs to, etc.

Future lines of work worth considering include the study of alternatives for the visualisation of the network community structure in 2D and 3D and the use of graph centrality measures to identify the most relevant nodes in the networks for use as reference cases in a collaborative approach. This could be useful to reduce the size of large networks by using one or several reference nodes as prototypes that represent whole communities, producing a simplified and more tractable representation of the data.

On a related note, in applications with large datasets, the minimal path approximation used in Section 3 may become unfeasible, as it uses Floyd's algorithm (complexity $O(N^3)$, where N is the number of samples). In that case, other possibilities like Dijkstra's algorithm with Fibonacci heaps would need to be investigated to reduce the computational cost, as it is shown e.g. in [16].

Appendix A

This section reviews the meaning and validity of the FI metric in primary data space. The definitions of information and divergence and their relationship with the Fisher information metric are discussed in the case where the role of density function $p(x|\theta)$ is taken by a posterior distribution $p(c|x)$ that is assumed to have been fitted using empirical auxiliary data. The idea is to replicate the original derivation of the FI expressions in [4], ensuring that the regularity assumptions hold, with the ultimate aim of gaining a rigorous insight into what exactly the meaning of the reversed definition of the Fisher metric is in the space of covariates.

A.1Information

The first step is to define the information of a likelihood function, $p(x_1|c)$, with respect to another, $p(x_2|c)$, when x_1 and x_2 are points in the data space X and c is a value of the discrete class variable C. Let H_i, $i=1,2$, be the hypotheses that the points x_i were generated by class c. By Bayes' theorem

$$\log\frac{p(c|x_1)}{p(c|x_2)}=\log\frac{p(x_1|c)}{p(x_2|c)}-\log\frac{p(x_1)}{p(x_2)}, \tag{A.1}$$

where $p(x_i)$ is the prior probability of the data point x_i and $p(x_i|c)$ is the class conditional probability of x_i given $C=c$. Therefore, the right side of Eq. (A.1) is the difference of the divergence between the logarithm of the odds in favour of H_1 before and after observing $C=c$. This can be interpreted as the information resulting from the observation of class membership, and the logarithm of the likelihood ratio, $\log(p(c|x_1)/p(c|x_2))$, is defined to be the information in $C=c$ for discrimination in favour of H_1 against H_2. The expectation of this information with respect to $p(c|x_1)$ for the whole set of class labels C gives what is normally known as the Kullback–Leibler (KL) divergence

$$I(x_1:x_2)=\sum_c p(c|x_1)\log\frac{p(c|x_1)}{p(c|x_2)}. \tag{A.2}$$

Using Eq. (A.1), the KL divergence between distributions $p(c|x_1)$ and $p(c|x_2)$ can be expressed as

$$I(x_1:x_2)=\sum_c p(c|x_1)\log\frac{p(x_1|c)}{p(x_2|c)}-\log\frac{p(x_1)}{p(x_2)}. \tag{A.3}$$

The information $I(x_1{:}x_2)$, or non-symmetric KL divergence, is a measure that gives the mean value with respect to $p(c|x_1)$ of the logarithm of the odds of the class conditional probabilities of the hypotheses, $p(x_i|c)$, and the logarithm of the odds of the prior density functions, $p(x_i)$.

A.2 Divergence

The original formulation of the KL divergence [4] defines it as a symmetric measure, obtained by calculating the expected value of the information in Eq. (A.3) over both orders of H_1 and H_2, and represents a measure of the difficulty of discriminating between the two hypotheses:

$$J(x_1{:}\,x_2)=I(x_1:x_2)+I(x_1:x_2)=\sum_c(p(c|x_2)-p(c|x_1))\log\frac{p(c|x_2)}{p(c|x_1)}. \tag{A.4}$$

Both Eqs. (A.4) and (A.2) can be used as the starting point for calculating a Taylor approximation that applies locally for infinitesimal disturbances about a given point in primary data space (see Section A.4). However, the use of this approximation relies on the validity of three regularity conditions.

A.3 Regularity conditions

The first two regularity conditions used in [4, Chapter 2.6] require the probability density $p(c|x)$ to be continuous, finite and everywhere differentiable. In the following, it is assumed that all vector derivatives with respect to the multivariate random variable X may be treated independently from one covariate to the others. This assumption applies in the context of posterior probability functions that may be linear in the parameters or semi-parametric, including generic non-linear models such as artificial neural networks.

The third regularity condition follows directly from the standard normalisation of conditional probability density functions, or probability masses in the case of a discrete class variable

$$\sum_c p(c|x) = 1 \rightarrow \sum_c \nabla_x p(c|x) = 0 \rightarrow \sum_c \nabla_x^2 p(c|x) = 0. \tag{A.5}$$

Expressing $p(c|x)$ as $\exp(\log p(c|x))$ and taking derivatives of the first summation in Eq. (A.5)

$$\nabla_x^2 \sum_c e^{\log p(c|x)} = \nabla_x \sum_c \nabla_x \log p(c|x) \cdot p(c|x) = 0. \tag{A.6}$$

it follows that

$$\sum_c \nabla_x^2 \log p(c|x) p(c|x) + \sum_c (\nabla_x \log p(c|x))(\nabla_x \log p(c|x))^T p(c|x) = 0, \tag{A.7}$$

hence

$$\sum_c (\nabla_x \log p(c|x))(\nabla_x \log p(c|x))^T p(c|x) = -\sum_c \nabla_x^2 \log p(c|x) p(c|x) = G(x), \tag{A.8}$$

which shows the numerical equivalence between the two definitions of the FI matrix.

A.4 Fisher information

This subsection reproduces the relationship between the KL divergence and the Fisher metric as described in [4] but in the space of the data. The starting point is Eq. (1) in Section 2.1, the KL divergence between the posterior probabilities of two neighbouring points x and $x+\Delta x$:

$$I(x : x+\Delta x) = -\sum_c p(c|x) \log \frac{p(c|x+\Delta x)}{p(c|x)}. \tag{A.9}$$

Taking the Taylor expansion of $\log p(c|x+\Delta x))$ about x up to second-order terms

$$\log p(c|x+\Delta x) - \log p(c|x) = \Delta x \nabla_x \log p(c|x) + \frac{1}{2!} \Delta x^T \nabla_x^2 \log p(c|x) \Delta x. \tag{A.10}$$

This is inserted in Eq. (A.9), yielding

$$I(x : x+\Delta x) = -\sum_c \Delta x \nabla_x p(c|x) - \frac{1}{2!} \sum_c p(c|x) \Delta x^T \nabla_x^2 \log p(c|x) \Delta x. \tag{A.11}$$

Since the first term is zero according to Eq. (A.5)

$$I(x : x+\Delta x) = -\frac{1}{2!} \Delta x^T \left\{ \sum_c \nabla_x^2 \log p(c|x) p(c|x) \right\} \Delta x = \frac{1}{2} \Delta x^T G(x) \Delta x, \tag{A.12}$$

where $G(x)$ is the Fisher information matrix defined as any of the two forms in Eq. (A.8).

A similar procedure using Eq. (A.4) shows that

$$J(x : x+\Delta x) = \Delta x^T G(x) \Delta x. \tag{A.13}$$

Based on Eqs. (A.12) and (A.13), the differential distance between a pair of points $x_1 = x$ and $x_2 = x+\Delta x$ can be understood as the KL divergence between their class membership probabilities. This is equivalent, going back to Section A.1, to the information available to discern between the hypotheses H_1 and H_2 given a particular class label and averaged over all possible label values, where hypothesis H_i states that the point x_i was generated by the distribution of class label c.

Appendix B

This appendix covers the derivation of Eq. (11), which calculates the distance between any pair of points under the Fisher metric when using an LLR as the probability estimator. This is an important expression because it provides an analytical way to calculate global distances with a linear estimator in binary problems and can also be used to approximate straight-line distances with non-linear estimators using a Taylor approximation.

Inserting Eq. (5) into Eq. (10) gives

$$d(x_A, x_B) = \left| \int_0^1 \sqrt{\dot{x}(t)^T (\nabla_x a(x(t)))(\nabla_x a(x(t)))^T \dot{x}(t) \cdot p(c=1|x(t))(1-p(c=1|x(t)))} dt \right|, \tag{B.1}$$

where $x(t)$ is a path that goes from $x(t=0)=x_A$ to $x(t=1)=x_B$. The predictor $a(x)$ is given by Eq. (6), resulting in

$$d(x_A, x_B) = \left| \int_0^1 \sqrt{\dot{x}(t)^T \beta \beta^T \dot{x}(t) \cdot p(c=1|x(t))(1-p(c=1|x(t)))}\, dt \right| = \left| \int_0^1 \sqrt{(\beta^T \dot{x}(t))^2 \cdot p(c=1|x(t))(1-p(c=1|x(t)))}\, dt \right|. \tag{B.2}$$

We now change the integration variable to $a(x(t))$, a for brevity

$$a = \beta^T x(t) + \beta_0 \rightarrow \frac{da}{dt} = \beta^T \dot{x}(t) \rightarrow dt = \frac{1}{\beta^T \dot{x}(t)} da, \tag{B.3}$$

which leaves the integral as

$$d(x_A, x_B) = \left| \int_{a(x_A)}^{a(x_B)} \sqrt{(\beta^T \dot{x}(t))^2 \cdot p(c=1|x(t))(1-p(c=1|x(t)))} \frac{1}{\beta^T \dot{x}(t)} da \right| = \left| \int_{a(x_A)}^{a(x_B)} \sqrt{p(c=1|x(t))(1-p(c=1|x(t)))}\, da \right|. \tag{B.4}$$

Using the expression in Eq. (4) and simplifying

$$d(x_A, x_B) = \left| \int_{a(x_A)}^{a(x_B)} \frac{e^{-a(x(t))/2}}{1+e^{-a(x(t))}} da \right|. \tag{B.5}$$

And another variable change

$$b = e^{-a(x(t))/2} \rightarrow \frac{db}{da} = -\frac{1}{2} e^{-a(x(t))/2} = -\frac{1}{2} b \rightarrow da = -\frac{2}{b} db. \tag{B.6}$$

This results in

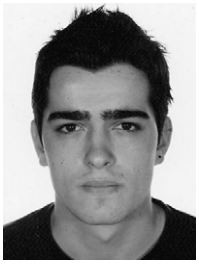
$$d(x_A, x_B) = \left| -2 \int_{b(x_A)}^{b(x_B)} \frac{1}{1+b^2} db \right| = \left| 2 \int_{b(x_A)}^{b(x_B)} \frac{1}{1+b^2} db \right| = \left| 2[\arctan(b)]_{b(x_A)}^{b(x_B)} \right|. \tag{B.7}$$

The final step is to undo the last variable change, returning to $a(x(t))$

$$d(x_A, x_B) = \left| 2 \left[\arctan\left(e^{-a(x(t))/2} \right) \right]_{a(x_A)}^{a(x_B)} \right|. \tag{B.8}$$

References

[1] S. Amari, Information geometry on hierarchy of probability distributions, IEEE Trans. Inf. Theory 47 (5) (2001) 1701–1711.
[2] T.S. Jaakola, D. Haussler, Exploiting generative models in discriminative classifiers, Proc. Adv. Neural Inf. Process. Syst. 11 (1998) 487–493.
[3] S. Kaski, J. Sinkkonen, Metrics that learn relevance, in: Proceedings of the International Joint Conference on Neural Networks, vol. 5, 2000, pp. 547–552.
[4] S. Kullback, Information Theory and Statistics, Wiley, New York, 1959.
[5] M. Girolami, B. Calderhead, S.A. Chin, Riemann manifold Langevin and Hamiltonian Monte Carlo methods, J. R. Statist. Soc. B 73 (2) (2011) 1–37.
[6] M.E.J. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. USA 103 (23) (2006) 8577–8582.
[7] J. Peltonen, A. Klami, S. Kaski, Improved learning of Riemannian metrics for exploratory analysis, Neural Networks 17 (2004) 1087–1100.
[8] S. Kaski, J. Sinkkonen, J. Peltonen, Bankruptcy analysis with self-organizing maps in learning metrics, IEEE Trans. Neural Networks 12 (4) (2001) 936–947.
[9] A.Y. Ng, M.I. Jordan, On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes, Proc. Adv. Neural Inf. Process. Syst. 14 (2001) 841–848.
[10] T. Jebara, Machine Learning: Discriminative and Generative, Kluwer, Norwell, 2003.
[11] D.F. Gleich, MatlabBGL Package, Version 4.0, 2008.
[12] H. Ruiz, I.H. Jarman, J.D. Martín, P.J.G. Lisboa, The role of Fisher information in primary data space for neighbourhood mapping, in: Proceedings of the European Symposium on Artificial Neural Networks, 2011, pp. 381–386.
[13] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, 1996.
[14] D.W. Hosmer, S. Lemeshow, Goodness of fit tests for the multiple logistic regression model, Commun. Stat. Theory Methods 9 (10) (1980) 1043–1069.
[15] J.W. Sammon, A nonlinear mapping for data structure analysis, IEEE Trans. Comput. 5 (1969) (1969) 401–40918 5 (1969) (1969) 401–409.
[16] R. Cruz-Barbosa, D. Bautista-Villavicencio, A. Vellido, On the computation of the geodesic distance with an application to dimensionality reduction in a neuro-oncology problem, in: Proceedings of the Iberoamerican Congress on Pattern Recognition, LNCS 7042, 2011, pp. 483–490.



Héctor Ruiz received a BEng degree in Telecommunications Engineering (2008) and an MEng degree in Electronic Engineering (2010) from the University of Valencia, Spain. He is currently pursuing the PhD degree in the School of Computing and Mathematics at Liverpool John Moores University, UK. His current research interests include data mining, machine learning, network analysis and distance metric learning.

Terence A. Etchells is a research fellow in the School of Computing and Mathematics at Liverpool John Moores University, UK.

His research interests are: Rule Extraction from Smooth Decision Surfaces; Scalable Algorithms for Clustering, Association Maps, Network Analysis and Graphical Models.

He completed his first degree in Mathematics at York University (UK) in 1983, gained an M.Sc. in Mathematics and Mathematics Education at Leeds University (UK) in 1992 and completed his Ph.D. in Rule Extraction at Liverpool John Moores University (UK) in 2004.

Ian H. Jarman is a senior researcher in interpretive and predictive modelling, Department of Mathematics and Statistics, Liverpool John Moores University (LJMU). Ian graduated from LJMU with a first class honours degree in Mathematics, Statistics and Computing in 2002, continuing to complete an EPSRC funded PhD in Bioinformatics and then research assistant until 2006. In 2007 he became a research associate with the School of Psychological Sciences, Manchester University undertaking collaborate research with Unilever Corporate R&D. Ian returned to LJMU in 2009 as senior researcher and manager of a team of data analysts/researchers, transferring to his current position in 2011.

José D. Martín–Guerrero received a B.Sc. degree in Physics (1997), a B.Sc. degree in Electronic Engineering (1999), an M.Sc. Degree in Electronic Engineering (2001) and a Ph.D. degree in Machine Learning (2004) from the University of Valencia, Spain. He is currently a Senior Lecturer at the Department of Electronic Engineering, University of Valencia. His research interests include Machine Learning and Computational Intelligence, with special emphasis in Reinforcement Learning and visualization methods. He is a Member of the Medical Data Analysis Task Force (Data Mining Technical Committee, IEEE Computational Intelligence Society).

Paulo Lisboa chairs the Department of Mathematics and Statistics at Liverpool John Moores University. He has published on computational data analysis for medical decision support and machine learning, focusing on interpretation by end-users. This includes automatic rule generation, clustering & visualisation of high-dimensional data, and conditional independence maps, with over 200 refereed publications and awards for downloads and citations of journal papers. He chairs the Medical Data Analysis Task Force in the Data Mining Technical Committee of the IEEE-CIS and is associate editor for several journals including Neural Networks, IET Science Measurement & Technology, Neural Computing Applications and Applied Soft Computing.

WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IJCNN

Towards interpretable classifiers with blind signal separation

Héctor Ruiz
Department of Mathematics and Statistics
Liverpool John Moores University
Liverpool, United Kingdom
H.Ruiz@2010.ljmu.ac.uk

Sandra Ortega-Martorell
Departament de Bioquímica i Biología Molecular
Universitat Autònoma de Barcelona
Cerdanyola del Vallès, Spain
Sandra.Ortega@uab.cat

Ian H. Jarman
Department of Mathematics and Statistics
Liverpool John Moores University
Liverpool, United Kingdom
I.H.Jarman@ljmu.ac.uk

Alfredo Vellido
Department of Computer Languages and Systems
Universitat Politècnica de Catalunya
Barcelona, Spain
avellido@lsi.upc.edu

José D. Martín
Departamento de Ingeniería Electrónica
Universidad de Valencia
Burjassot, Spain
jose.d.martin@uv.es

Enrique Romero
Department of Computer Languages and Systems
Universitat Politècnica de Catalunya
Barcelona, Spain
eromero@lsi.upc.edu

Paulo J.G. Lisboa
Department of Mathematics and Statistics
Liverpool John Moores University
Liverpool, United Kingdom
P.J.Lisboa@ljmu.ac.uk

***Abstract*—Blind signal separation (BSS) is a powerful tool to open-up complex signals into component sources that are often interpretable. However, BSS methods are generally unsupervised, therefore the assignment of class membership from the elements of the mixing matrix may be sub-optimal. This paper proposes a three-stage approach using Fisher information metric to define a natural metric for the data, from which a Euclidean approximation can then be used to drive BSS. Results with synthetic data models of real-world high-dimensional data show that the classification accuracy of the method is good for challenging problems, while retaining interpretability.**

Blind signal separation; non-negative matrix factorisation; Fisher information; Riemannian metric; data mapping; magnetic resonance spectroscopy; brain tumour

I. INTRODUCTION

Blind signal separation (BSS) is a well-known family of tools to separate complex signals into linear combinations of sources whose joint distribution is close to factorised into a product of independent univariate density functions for the individual sources. This approach is rendered even more interpretable when it is applied in the convex space of positive semi-definite mixing and unmixing matrices [1]. Both the sources themselves and the partial membership of each source class can then be evaluated against prior knowledge.

In our example, synthetic data models are built from single voxel magnetic resonance spectroscopy (MRS) signal corresponding to a neuro-oncology problem. The sources will ideally approximate prototypes for each brain tissue class and the maximal values in each row of the mixing matrix will correspond to the correct binary classification of that observation. In this data set the correct prototype is taken to be the mean of the generating distribution.

In a previous work [2], the authors investigated the application of non-negative matrix factorisation (NMF) methods [3,4] for the extraction of tissue type-specific MRS

signal sources in a fully unsupervised mode, to the analysis of an international, multi-centre database that incorporates MRS data corresponding to several types of human brain tumours. The accuracies of the labels inferred for each patient case where comparable to traditional supervised classifiers.

However, there is some instability in the classification arising from mixing in data space, especially for challenging differential assignments such as the discrimination of low astrocytic tumours from high grade and from metastatic growths. This limitation arises because the method is fully unsupervised. This is reflected in the generating modes by wide standard deviations for each class in addition to high dimensionality of the data.

Recently, the Fisher information matrix has been proposed as an effective way to build a meaningful, well determined metric in primary data space with which to disaggregate class-labelled data [5]. This metric defines the natural geometry of data space taking into account both the position and class labelling of each data point. However, the Fisher information metric is non-Euclidean, i.e. it does not respect the triangle inequality, with the consequence that projective methods such as BSS cannot be applied directly in this space.

The natural way to bridge the gap between the non-Euclidean metric and projective spaces is to map the data onto an approximate, rigorous Euclidean metric space. This may be done using data projection methods such as the Sammon mapping, multidimensional scaling (MDS) or the iterative majorisation algorithm (IMA).

In this paper we use synthetic data to study the hypothesis that the three-stage approach consisting on first defining a Fisher Information metric, then approximating the empirical data distribution with a Euclidean projective space onto which, subsequently, Convex-NMF [1] can be applied, results in a natural decomposition of the data with sources that are closer to the true class prototypes and in higher classification accuracies than those obtained using a purely unsupervised implementation of NMF.

II. Methodology

A. Convex non-negative matrix factorisation

In this study, we use a variant of the non-negative matrix factorisation (NMF) family [3,4], namely Convex-NMF [1]. In conventional NMF methods a non-negative data matrix V (of dimensions d-by-n, where d is the data dimensionality and n is the number of observations) is approximately factorised into two non-negative matrices: the matrix of sources or data basis W (of dimensions d-by-k, where k is the number of sources, and $k < d$) and the mixing matrix H (of dimensions k-by-n, each of whose columns provides the encoding of a data point: the spectrum of an observation in this study). The product of these two matrices provides a good approximation to the original data matrix, in the form $V \approx WH$.

To achieve interpretability, Convex-NMF imposes a constraint that the vectors (columns) defining W must lie within the column space of V, i.e. $W=VA$ (where A is an auxiliary adaptative weight matrix that fully determines W), so that $V \approx VAH$. By restricting W to convex combinations of the columns of V we can, in fact, understand each of the basis or sources as weighted sums of data points. This NMF variant applies to both non-negative and mixed-sign data matrices, and only H and A are constrained to be non-negative. These factors are updated as follows:

$$H^T \leftarrow H^T \sqrt{\frac{(V^TV)^+A \;+\; H^TA^T(V^TV)^-A}{(V^TV)^-A \;+\; H^TA^T(V^TV)^+A}}$$
$$A \leftarrow A \sqrt{\frac{(V^TV)^+H^T \;+\; (V^TV)^-AHH^T}{(V^TV)^-H^T \;+\; (V^TV)^+AHH^T}} \quad , \qquad (1)$$

where $(\cdot)^+$ is the positive part of the matrix, where all negative values become zeros, and $(\cdot)^-$ is the negative part of the matrix, where all positive values become zeros. H and A are initialised using K-means clustering, as proposed in [1]. This multiplicative algorithm minimises the reconstruction error given by $\|V-VAH\|^2$, while ensuring that the elements of the matrices H and A remain nonnegative.

B. Interpretation of Convex-NMF in the context of MRS data

Given that the observed MRS data are of mixed sign, their sources should also be of mixed sign. Thus, understanding W as the source spectra matrix, the sources will be intuitively interpretable and no pre-processing of the spectra will be required in order to make them non-negative, thus preventing any unnecessary loss of information (in the case of our data, losing the information in the negative peaks of the LTE MRS spectra). Due to the fact that it is non-negative by definition, the mixing matrix H can be understood as estimates of the concentration/abundance of the constituent signals.

C. Fisher information metric

The Fisher information (FI) is a measure of the amount of information that a variable x carries about a magnitude θ upon which its probability depends [6]. This is obtained by deriving the logarithm of the conditional probability $p(x|\theta)$ with respect to θ and then calculating the conditional expectation over x with respect to the said probability, which results in the measure being independent on x.

In this work, an alternative definition is used where the roles of x and θ are swapped, and therefore the derivative is of $p(\theta|x)$ with respect to x [7]. This FI is now a function of x and takes the form of a square matrix of the same dimensionality as x, that is, the dimensionality of the data space:

$$FI(x) = E_{p(\theta|x)}\{(\nabla_x \log p(\theta|x))(\nabla_x \log p(\theta|x))^T\}$$
$$= -E_{p(\theta|x)}\{\nabla_x^2 \log p(\theta|x)\} \quad , \qquad (2)$$

where $E_{p(\theta|x)}$ denotes the expectation over the values of θ with respect to $p(\theta|x)$ and ∇_x is the gradient with respect to x.

The motivation behind this modification of the original FI is to use it in data mining applications, where x would be a point in the space of the data and θ would be an auxiliary class variable c. In this scenario, it is easy to define a differential metric using the FI matrix:

$$d(x, x+\Delta x)^2 = \Delta x^T FI(x) \Delta x \; . \qquad (3)$$

This gives the distance between two neighbouring points x and $x+\Delta x$ under the metric defined by the FI matrix.

The interesting property of this metric is that it automatically scales each dimension of the data space according to its degree of relevancy with respect to class membership, expanding directions along which p($c|x$) changes rapidly and compressing those where the variation is little. The result is a Riemannian space where the posterior class membership probability changes evenly in all directions.

D. Multi-layer perceptron

A crucial stage in the development of the FI metric is the estimation of p($c|x$). The ability of the metric to precisely reflect similarity between data points into distances is conditioned by how accurate the probability function on which the FI is based is.

Our choice for an estimator is a multi-layer perceptron (MLP), a feedforward artificial neural network whose versatility makes it ideal for highly non-linear data distributions. The MLP is present in the initial learning step of the process, where its internal weights are trained with the labelled dataset. The perceptron can then estimate probabilities for unlabelled instances in this semi-supervised manner.

E. Dataset projection

After estimating the class membership probability it is possible to compute distances between any two points x_A and x_B in the data space by solving the following path integral along the geodesic path:

$$d(x_A, x_B) = \left| \int_{x_A}^{x_B} \sqrt{\dot{x}(t)^T FI(x(t))\dot{x}(t)}\, dt \right|, \qquad (4)$$

where $x(t)$ is the shortest path that goes from x_A to x_B in the space defined by the Fisher metric.

This integral is not directly solvable for the non-linear case; one way around this is to approximate it by dividing the geodesic path into a number of segments whose distance can be obtained using the linear solution to the integral [5].

At this point, a pairwise distance matrix is produced that contains the Fisher distances between every pair of points in the dataset. This will be used to map the dataset from the original primary data space into a Euclidean feature space of the desired dimensionality. We compare several projection methods, namely the Sammon mapping, MDS and IMA.

1) Sammon mapping: This algorithm is used to analyse multivariate data by mapping the data points from an original high dimensional space to a space of lower dimensionality [8]. This non-linear mapping is based on the preservation of the original pairwise distances between data points when moving to the new data space.

The algorithm considers the situation of having N points in a space of dimensionality L that we want to map to another space of dimensionality D, and defines another set of N vectors in this D-space. The distance between points x_i and x_j in the original space is given by d_{ij}*, and the distance between their corresponding maps in the D-space is denoted by d_{ij}. An initial set of mapped points is generated, which result in a value for the following error function, also known as Sammon's stress:

$$E = \frac{1}{\sum_{i<j} d_{ij}} \sum_{i<j}^{N} \frac{\left(d_{ij}^{*} - d_{ij}\right)^2}{d_{ij}^{*}} . \qquad (5)$$

The position of the points in the D-space is then iteratively adjusted to reduce the error. In most cases, the method used to minimize this error function is gradient descent.

The distance measure for d_{ij}* is usually Euclidean. However, since we want to take into account the prior information that we have about the data in the form of class labels, we use the Fisher metric to compute these distances.

2) Metric multidimensional scaling: The idea on which metric MDS is based, as in Sammon mapping, is the preservation of the original pairwise distances in the projected space. The error function usually has a similar shape as (5) [9]. In this work, we use

$$E = \frac{1}{N^2} \sum_{i<j}^{N} \left(d_{ij}^{*} - d_{ij}\right)^2 . \qquad (6)$$

The main difference with (5) lies in the absence of normalisation of the squared differences of the distances. Similar to Sammon mapping, the position of the set of mapped points is iteratively adjusted by gradient descent so as to minimise the error function. Again, the distances d_{ij}* are computed with the Fisher metric.

3) Iterative majorisation algorithm: The last algorithm in this section expresses the mapping from an original L-space to a D-space as a function $f(x;W)=W^T \cdot \Phi(x)$, where W is a P-*by*-D matrix containing the free parameters and $\Phi(x)=(\Phi_1(x), \ldots, \Phi_P(x))^T$ contains the values of the P basis functions $\Phi_i(x)$. The mapping $f(x;W)$ is a linear combination of these basis functions, which can be linear or non-linear. In this work, we have used $P=N$ with $\Phi(x_i)=(d_{i1}^{*}, d_{i2}^{*}, \ldots, d_{iN}^{*})^T$, where d_{ij}^{*} is the Fisher distance between points x_i and x_j. The method tries to minimise the error function

$$E = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(d_{ij}^{*} - q_{ij}(W)\right)^2 , \qquad (7)$$

where $q_{ij}(W)=\|W^T(\Phi(x_i)-\Phi(x_j))\|$. This is minimised with respect to the weights W using the iterative majorisation algorithm. More detail on this can be found in [10].

III. Experimental Results

A. Description of the data

The data analysed in this study are modelled from samples extracted from a database used in a previous publication [2]. Class (tumour type) labelling was used to generate posterior distributions of the data density, i.e. p(data|class) using single multivariate normal models fitted to the mean and variance/covariance matrices of class specific cohorts of single-voxel proton MRS (SV-^{1}H-MRS) acquired at two different

echo times (short, 20-32 ms (STE) and long, 135-144 ms (LTE)) from brain tumour patients.

The analysed data set included, at LTE, samples of the generated data for 20 astrocytomas grade II (A2), 78 glioblastomas (GL), and 31 metastases (ME); at STE, it included 22 A2, 86 GL, and 38 ME. The data dimensionality is 195 reflecting the clinically-relevant frequency intensity values measured in parts per million (ppm) that are typically sampled from each spectrum in the [4.24,0.50] ppm interval.

A second dataset was generated for the validation of the methods. In this dataset, each class has 50 samples generated using the same means and covariance matrices used for the training set. Note that, during the experiments, the metric is derived using the training dataset labels only, and then it is applied to calculate pairwise distance matrices of the training and validation datasets, so no usage of the validation class labels is made, as is expected from a semi-supervised learning method.

All parameters mirror the actual data as closely as possible. The aim of using generated data is to be able to test the proposed methodology against known ground truth.

B. Empirical results

Pairwise classification between types of brain tissue were performed, paying attention to the accuracy of the results and the quality of the sources obtained.

Accuracy is measured as the ratio of correctly classified cases out of the total number of instances, and the quality of the sources is determined in terms of how similar they are compared to the mean spectrum of the corresponding class. Similarity is assessed using the correlation between the resulting sources and mean spectra. Each classification problem is run 20 times to average the effect of the variation that the initialisation of Convex-NMF causes on the results.

The classification results on training data (Table I) show a general improvement on the performance of the original approach when we use the Fisher metric pre-processing before applying Convex-NMF, whether it is using the Sammon mapping, MDS or IMA. The increase of the accuracy on the validation dataset (Table III) is smaller, as one would expect, but still significant.

There are some exceptions where Convex-NMF performs better than the alternatives, as in GL vs. ME at STE, when using MDS with training data; in A2 vs. ME at STE, when using Sammon with validation data; and in A2 vs. GL at STE, where Convex-NMF outperforms all three mappings with validation data. Despite that, there is an overall tendency of accuracy increase.

Regarding the quality of the sources (Tables II and IV), all four approaches yield very good sources in general. The most interesting case is GL against ME both at STE and LTE, where plain Convex-NMF does not perform as well as in the other classifications. This is because GL and ME types have a very similar spectral pattern (their mean spectra have correlations of 0.9891 at STE and 0.9211 at LTE between each other), which also explains why the classification accuracies are so low for those two cases. The Fisher metric alternatives manage to obtain very high source correlations and accuracies even for the GL vs. ME problem, due to the additional information that they bring into Convex-NMF from the auxiliary data labels through the Fisher metric.

In the accuracy tables I and III, each cell contains the amount of well classified samples in percentage and also in fraction form in brackets. The column *Original* refers to the results obtained applying plain Convex-NMF, and the other subheaders identify the projection method used to map the data. In the source correlation tables, the two scores within each cell correspond to the correlation between the source of the corresponding tissue type and the true mean spectrum of that class.

TABLE I. CLASSIFICATION ACCURACIES FOR THE TRAINING DATASET

		STE				**LTE**			
		Original	***Sammon***	***MDS***	***IMA***	***Original***	***Sammon***	***MDS***	***IMA***
A2 vs. GL	total	83.3% (90/108)	96.8% (104.6/108)	96.3% (104/108)	99.1% (107/108)	60.2% (59/98)	99% (97/98)	99% (97/98)	99% (97/98)
	A2	100% (22/22)	90.5% (19.9/22)	100% (22/22)	100% (22/22)	100% (20/20)	100% (20/20)	100% (20/20)	100% (20/20)
	GL	79.1% (68/86)	98.4% (84.7/86)	95.3% (82/86)	98.8% (85/86)	50% (39/78)	98.7% (77/78)	98.7% (77/78)	98.7% (77/78)
A2 vs. ME	total	98.3% (59/60)	99.8% (59.9/60)	100% (60/60)	100% (60/60)	86.3% (44/51)	100% (51/51)	99.9% (50.9/51)	100% (51/51)
	A2	100% (22/22)	99.5% (21.9/22)	100% (22/22)	100% (22/22)	100% (20/20)	100% (20/20)	100% (20/20)	100% (20/20)
	ME	97.4% (37/38)	100% (38/38)	100% (38/38)	100% (38/38)	77.4% (24/31)	100% (31/31)	99.8% (30.9/31)	100% (31/31)
GL vs. ME	total	69.8% (86.6/124)	82.2% (102/124)	69.6% (86.4/124)	88.7% (110/124)	60.6% (66/109)	95.6% (104.2/109)	94.5% (103/109)	97.2% (106/109)
	GL	61.1% (52.6/86)	77.3% (66.5/86)	57.4% (49.4/86)	86% (74/86)	55.1% (43/78)	95.6% (74.6/78)	92.3% (72/78)	96.2% (75/78)
	ME	89.5% (34/38)	93.4% (35.5/38)	97.4% (37/38)	94.7% (36/38)	74.2% (23/31)	95.5% (29.6/31)	100% (31/31)	100% (31/31)

TABLE II. SOURCE CORRELATIONS FOR THE TRAINING DATASET

		STE				LTE			
		Original	***Sammon***	***MDS***	***IMA***	***Original***	***Sammon***	***MDS***	***IMA***
A2 vs. GL	A2	0.98	0.89	0.99	0.94	0.98	0.98	0.97	0.98
	GL	0.96	1	1	1	0.70	1	1	1
A2 vs. ME	A2	0.98	0.93	0.94	0.99	0.99	0.99	0.98	0.99
	ME	0.99	1	1	1	0.88	1	0.99	1
GL vs. ME	GL	0.95	0.99	0.97	0.97	0.73	0.99	0.99	0.99
	ME	0.98	1	1	1	0.86	0.99	1	1

TABLE III. CLASSIFICATION ACCURACIES FOR THE VALIDATION DATASET

		STE				LTE			
		Original	***Sammon***	***MDS***	***IMA***	***Original***	***Sammon***	***MDS***	***IMA***
A2 vs. GL	total	94% (94/100)	85.5% (85.5/100)	90% (90/100)	90% (90/100)	80% (80/100)	98% (98/100)	95% (95/100)	97% (97/100)
	A2	98% (49/50)	82.8% (41.4/50)	88% (44/50)	84% (42/50)	100% (50/50)	100% (50/50)	92% (46/50)	100% (50/50)
	GL	90% (45/50)	88.1% (44.1/50)	92% (46/50)	96% (48/50)	60% (30/50)	96% (48/50)	98% (49/50)	94% (47/50)
A2 vs. ME	total	97% (97/100)	93.8% (93.8/100)	99% (99/100)	99% (99/100)	88% (88/100)	99.9% (99.9/100)	99.9% (99.9/100)	100% (100/100)
	A2	98% (49/50)	92.9% (46.5/50)	98% (49/50)	98% (49/50)	100% (50/50)	100% (50/50)	100% (50/50)	100% (50/50)
	ME	96% (48/50)	94.6% (47.3/50)	100% (50/50)	100% (50/50)	76% (38/50)	99.9% (49.9/50)	99.9% (49.9/50)	100% (50/50)
GL vs. ME	total	67.7% (67.7/100)	72.8% (72.8/100)	75% (75/100)	74% (74/100)	62% (62/100)	81.8% (81.8/100)	82% (82/100)	83% (83/100)
	GL	64% (32/50)	71.3% (35.7/50)	58.1% (29.1/50)	64% (32/50)	52% (26/50)	84.8% (42.4/50)	92% (46/50)	86% (43/50)
	ME	71.4% (35.7/50)	74.2% (37.1/50)	91.9% (45.9/50)	84% (42/50)	72% (36/50)	78.7% (39.4/50)	72% (36/50)	80% (40/50)

TABLE IV. SOURCE CORRELATIONS FOR THE VALIDATION DATASET

		STE				LTE			
		Original	***Sammon***	***MDS***	***IMA***	***Original***	***Sammon***	***MDS***	***IMA***
A2 vs. GL	A2	0.96	0.98	0.99	0.99	0.99	1	1	1
	GL	0.96	1	1	1	0.80	0.99	0.99	0.98
A2 vs. ME	A2	0.96	0.93	0.98	0.99	1	0.99	0.99	1
	ME	0.99	0.94	0.99	1	0.94	0.96	0.97	1
GL vs. ME	GL	0.94	1	0.98	0.98	0.68	0.99	0.99	0.99
	ME	0.98	1	1	1	0.89	0.99	0.99	0.99

Figures 1 to 5 represent the sources involved in a particular classification problem, GL against ME at LTE for the validation dataset. We have chosen to illustrate this case because it is where Convex-NMF struggles most to achieve good classification rates and sources due to the prototypical spectra of the two tissue types being very similar, as can be seen in figure 1. It is therefore for a problem like this that the Fisher metric approaches will show a more significant performance improvement.

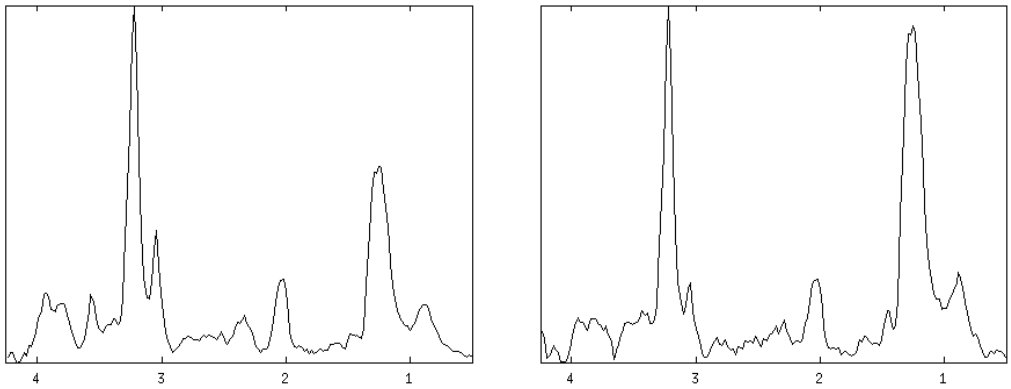


Figure 1. True GL (left) and ME (right) prototype spectra. X-axis: frequencies in ppm scale. Y-axis: Intensities normalised to unit length (UL2)

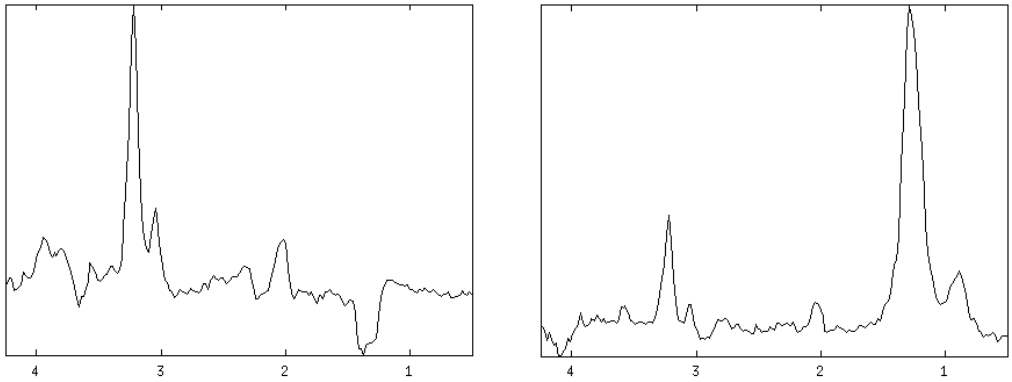


Figure 2. GL (left) and ME (right) sources retrieved using Convex-NMF. X-axis: frequencies in ppm scale. Y-axis: Intensities normalised to unit length (UL2)

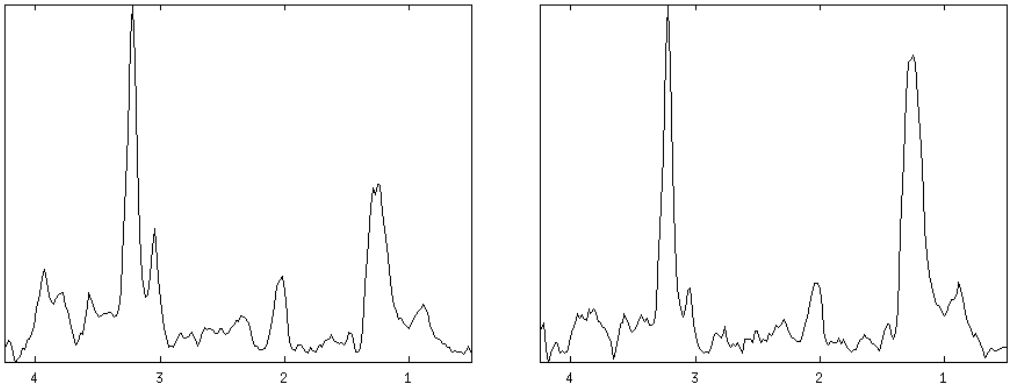


Figure 3. GL (left) and ME (right) sources retrieved using Sammon map. X-axis: frequencies in ppm scale. Y-axis: Intensities normalised to unit length (UL2)

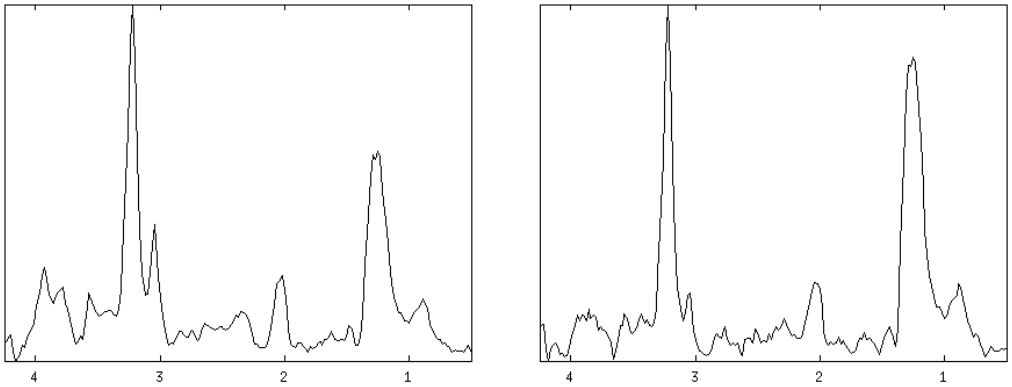


Figure 4. GL (left) and ME (right) sources retrieved using MDS. X-axis: frequencies in ppm scale. Y-axis: Intensities normalised to unit length (UL2)

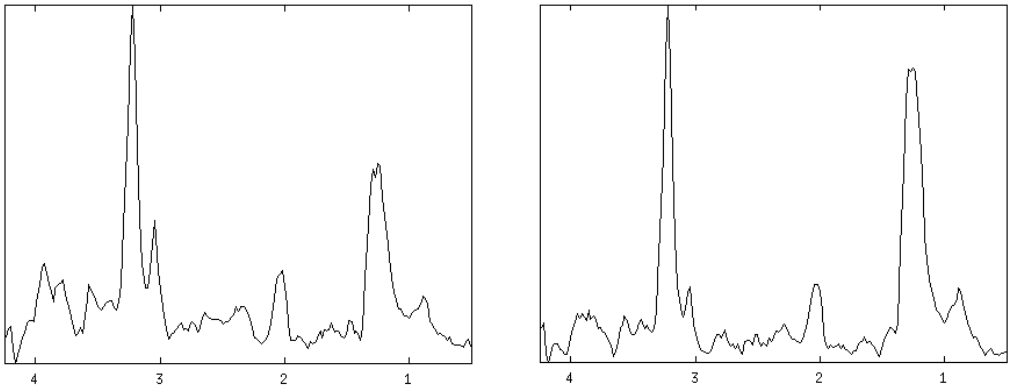


Figure 5. GL (left) and ME (right) sources retrieved using IMA. X-axis: frequencies in ppm scale. Y-axis: Intensities normalised to unit length (UL2)

The resemblance between the true spectra of the two classes is very high in terms of the position and height of the peaks. The only clear differences between them are the height ratio of the two main peaks and the height of the small one immediately to the right of the left main peak. We can see in figure 2 how the basic Convex-NMF does not manage to get both peaks right at the same time. Instead, it identifies each class with one of the two main peaks. The rest of the methods, however, pick up correctly the proportion of the height of the peaks for both classes and produce sources very similar to the originals.

IV. Discussion

The results of the experiments performed confirm the hypothesis that an unsupervised interpretable method for blind classification/signal separation, namely as Convex-NMF, can benefit from the use of known data labels and result in a more accurate classifier without any loss in the interpretability of the results.

Moreover, a mechanism is provided that achieves blind signal separation with a semi-supervised approach, by first finding a natural metric to describe the class assignments, followed by a mapping of the data into an approximate distribution in a Euclidean space where the blind signal separation can be applied with standard projective methods.

Furthermore, for the data analysed in this work, not only was the accuracy of the classification of the samples generally better, but the sources extracted were also of higher quality than those obtained using the original unsupervised method, both in the training and validation stages. In our opinion, the improvement on these two aspects, especially in complex classification problems, justifies the additional pre-processing steps that precede the original approach.

Future work is to replicate this methodology on the original medical data.

References

[1] C. Ding, T. Li, and M.I. Jordan, "Convex and semi-nonnegative matrix factorizations," IEEE Transactions on pattern analysis and machine intelligence, vol. 32, no. 1, pp. 45-55, 2010.

[2] S. Ortega-Martorell, P.J.G. Lisboa, A. Vellido, M. Julià-Sapé, and C. Arús, "Non-negative Matrix Factorisation methods for the spectral decomposition of MRS data from human brain tumours," BMC Bioinformatics, vol. 13, no. 38, 2012

[3] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," Environmetrics, vol. 5, no. 2, pp. 111-126, 1994.

[4] D. Lee and H. Seung, "Learning the parts of objects by non-negative matrix factorization," Nature, vol. 401, no. 6755, pp. 788-791, 1999.

[5] H. Ruiz, I.H. Jarman, J.D. Martín, and P.J.G. Lisboa, "The role of Fisher information in primary data space for neighbourhood mapping," European Symposium on Artificial Neural Networks (ESANN) 2011 proceedings.

[6] S. Amari, "Information geometry on hierarchy of probability distributions," IEEE Information theory, vol. 47, no. 5, pp. 1701-1711, 2001.

[7] S. Kaski and J. Sinkkonen, "Metrics that learn relevance," International Joint Conference on Neural Networks (IJCNN) 2000 proceedings, vol. 5, pp. 547-552, 2000.

[8] J.W. Sammon, "A nonlinear mapping for data structure analysis," IEEE Transactions on computers, vol. C-18, no. 5, pp. 401-409, 1969.

[9] J.A. Lee and M. Verleysen, "Nonlinear Dimensionality Reduction", Springer-Verlag, NY, 2007.

[10] Z. Zhang, "Learning metrics via discriminant kernels and multidimensional scaling: Toward expected Euclidean representation", International Conference on Machine Learning (ICML) 2003 proceedings, pp. 872-879, 2003.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Constructing similarity networks using the Fisher information metric

H. Ruiz[1], S. Ortega-Martorell[2], I. H. Jarman[1], J. D. Martín[3], P. J.G. Lisboa[1]

1 - School of Computing and Mathematical Sciences - Department of Mathematics and Statistics - LJMU, Liverpool L3 3AF - UK
2 - Departament de Bioquímica i Biología Molecular – Universitat Autònoma de Barcelona, Cerdanyola del Vallés (Barcelona) - Spain
3 - Escuela Técnica Superior de Ingeniería - Departamento de Ingeniería Electrónica - Universidad de Valencia, Burjassot (Valencia) - Spain

Abstract. The Fisher information metric defines a Riemannian space where distances reflect similarity with respect to a given probability distribution. This metric can be used during the process of building a relational network, resulting in a structure that is informed about the similarity criterion. Furthermore, the relational nature of this network allows for an intuitive interpretation of the data through their location within the network and the way it relates to the most representative cases or prototypes.

1 Introduction

Measures of similarity between data points are central to pattern recognition and data mining methodologies, although they are not always explicitly calculated. Nevertheless, using a distance function to measure similarity between pairs of elements of a space is an intuitive way to understand their relationship in the context of a particular problem domain. The Euclidean distance is a common choice because of its simplicity and little computational cost, even though the equal weighting of each dimension, for instance in clustering, leads to results that can be heavily dependent on the choice of data representation.

The Fisher information (FI) is a natural choice of metric in the space of probabilistic density functions [1]. In the case of the space of the covariates, a natural similarity measure between points is provided by the symmetric divergence between the posterior distributions $p(c|\boldsymbol{x})$ of classifiers fitted to the class labels, which are categorized by a discrete random variable C. The concept of a metric defined by differentiating a posterior distribution $p(c|\boldsymbol{x})$ with respect to the coordinates is reported in [2] as a natural extension of the metric defined in parameter space, e.g. in [3]. In a recent publication [4], we explained in detail the process of deriving a metric from the Fisher information using linear and non-linear models and presented a novel approach to the problem of finding geodesic distances in non-Euclidean metrics. The idea in the present paper is to follow the same process and use the FI metric to go from the dataset in the original high-dimensional space to a network where data points are nodes connected to each other by edges based on their similarity.

The resulting networks are analysed in terms of prediction accuracy and structure and the closing section discusses the interpretability of the classifier by identifying relevant reference cases.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

2 Methodology

This section provides a brief description of the FI metric derivation and discusses the choice of the method used to build the networks.

2.1 Derivation of the Fisher information metric

The FI is a local measure of the variation that an infinitesimal displacement of a point produces on the value of a probability distribution when evaluated at that point. Traditionally, the space where this displacement takes place is that of some parameter vector $\boldsymbol{\theta}$ upon which the probability function depends. We are, however, more interested in the approach introduced in [2], where the space of interest is the primary data space, i.e. the space where the dataset under study lies. The data is assumed to be divided into classes, with $p(c|\boldsymbol{x})$ representing the posterior probability of the class variable given a point in the data space.

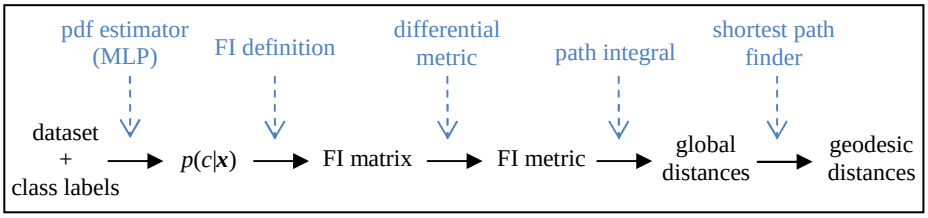


Fig. 1: Derivation of the FI metric

Figure 1 outlines the process of deriving the FI metric. Initially we have a dataset along with the class membership of each of the points. Using a density estimator, we obtain the posterior distribution $p(c|\boldsymbol{x})$. This estimate completely determines the metric in the sense that only a good model will produce a metric that reflects similarity accurately with respect to the true probability density. We use a multilayer perceptron (MLP) for this purpose because of its versatile architecture, which makes it ideal for non-linear data distributions.

The FI takes the form of a square matrix of the same dimensionality as the data. It is obtained from the estimated density according to either of the two equivalent definitions in (1).

$$\boldsymbol{FI}(\boldsymbol{x})_{p(c|\boldsymbol{x})} = \begin{cases} E_{p(c|\boldsymbol{x})}\{(\nabla \ln p(c|\boldsymbol{x}))^2\} \\ -E_{p(c|\boldsymbol{x})}\{(\nabla^2 \ln p(c|\boldsymbol{x}))\} \end{cases} \quad (1)$$

where $E_{p(c|\boldsymbol{x})}$ is the conditional expectation over the values of the class label c with respect to $p(c|\boldsymbol{x})$. The matrix defines a differential metric for the calculation of infinitesimal distances:

$$d(\boldsymbol{x}, \boldsymbol{x} + \Delta\boldsymbol{x})^2 = \Delta\boldsymbol{x}^T \boldsymbol{FI}(\boldsymbol{x}) \Delta\boldsymbol{x} \quad (2)$$

This can be integrated to calculate the distance between any pair of points by using the path integral

$$d(\boldsymbol{x_A}, \boldsymbol{x_B}) = \left| \int_0^1 \sqrt{\dot{\boldsymbol{x}}(t)^T \boldsymbol{FI}\big(\boldsymbol{x}(t)\big)\dot{\boldsymbol{x}}(t)}\, dt \right| \quad (3)$$

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

where $\boldsymbol{x}(t)$ is a path that goes from $\boldsymbol{x_A} = \boldsymbol{x}(t = 0)$ to $\boldsymbol{x_B} = \boldsymbol{x}(t = 1)$. At this point we can compute global distances in the space along a given path. The last part of the process is to find geodesic paths between points and to calculate their length. To do so, we use the free points approach. The reader is referred to [4] for an explanatory section on this algorithm.

2.2 Construction of the networks

Usual methods to build networks are k-nearest neighbours (kNN), where each data point is connected to the k nearest points, and ϵ-neighbourhood, where a connection is present when points are closer than a constant distance ϵ. kNN is preferred over ϵ-neighbourhood because it is adaptive to scale and density, while the use of the latter can result in disconnected graphs.
During the experiments carried out for this work, we applied kNN and b-matching. The b-matching method [5] is more rigorous than kNN in that it ensures that the final number of neighbours of each node is always the same. However, it only guarantees to converge if the linear programming relaxation of the formulation of the b-matching problem is tight [6]. In practice, when applying the algorithm to our data we found that it did not converge most of the time. For this reason, we only use kNN in this work.

3 Experimental results

In this section, we study the implications of the use of the FI metric in the construction of networks. Three aspects are discussed: the visualization power of networks, classification accuracies using kNN and the presence of network substructure.
The synthetic data analysed in this study are modelled from samples extracted from a data base used in a previous publication [7]. Class (tumour type) labelling was used to generate posterior distributions of the data density, using single multivariate normal models fitted to the mean and variance/covariance matrices of class specific cohorts of single-voxel proton Magnetic Resonance Spectroscopy (SV ^{1}H-MRS) from brain tumour patients.
This synthetic set included samples of the generated data for 78 glioblastoma-like (GL), and 31 metastasis-like (ME) cases. The data dimensionality is 195 reflecting the clinically-relevant frequency intensity values measured in parts per million (ppm) that are typically sampled from each spectrum in the [4.24,0.50] ppm interval. A second dataset was generated for the validation of the methods. In this dataset, each class has 50 samples generated using the same means and covariance matrices used for the training set. The discrimination between GL and ME, on the basis of SV ^{1}H MRS information, is a very challenging problem due to their radiological similarities. The appearance of both pathologies is often dominated by large peak intensities corresponding to neutral lipids, a byproduct of necrosis [8].

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

3.1 **Visualizing data using networks**

Using networks to represent data is a powerful visualization tool, especially when the data space is high-dimensional and direct examination is not possible. Similarity between points in the original space is captured by the connections in the network, enabling the viewer to see how the data looks like in terms of structure and clustering. Figures 2 and 3 show the kNN networks built from the dataset using Euclidean and Fisher distances respectively, with black nodes representing GL cases and white corresponding to ME. It is immediate to see a much clearer structure in the Fisher network (Fig.3), with edges connected in a very local manner. The Euclidean network (Fig.2), on the other side, presents a very fuzzy arrangement of the edges, and the grouping of the nodes is quite weak in terms of connectivity within/between groups.

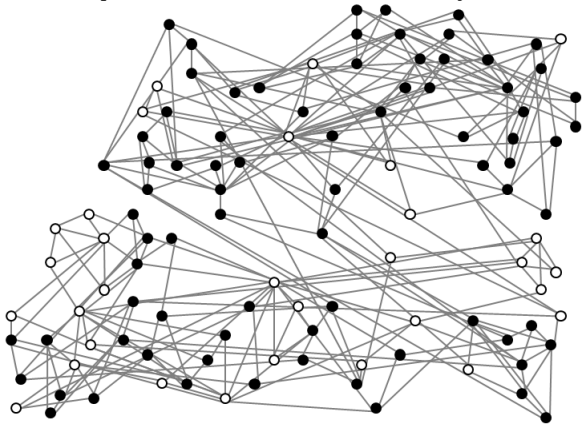


Fig. 2: kNN network (k=3) using Euclidean distances. Black = GL, white = ME.

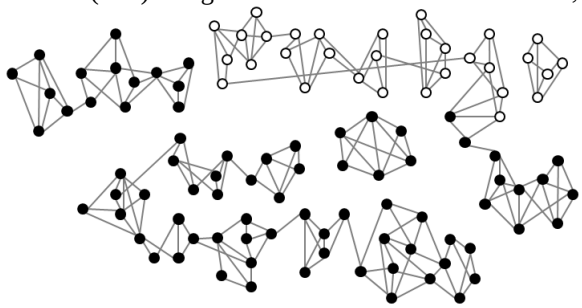


Fig. 3: kNN network (k=3) using Fisher distances. Black = GL, white = ME.

The information contained in the class labels is put in the form of a distance measure by the FI metric and is captured in the network, producing an informative and intuitive visualization of the data that otherwise would be difficult to interpret.

3.2 **Classification rates**

Tables 1 and 2 contain the classification accuracies using Euclidean and Fisher kNN classification (E-kNN and F-kNN, respectively) for different values of k. Table 2 corresponds to the results with the validation dataset. Fisher kNN obtains very good

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

accuracies as reported in the first table because these are the training samples of the MLP. The second table provides a more realistic impression.

	1	3	5	7	9	11	13	15
E-kNN	0.83	0.81	0.81	0.83	0.76	0.74	0.77	0.74
F-kNN	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.98

Table 1: Classification rates on the training dataset

	1	3	5	7	9	11	13	15
E-kNN	0.72	0.79	0.77	0.78	0.74	0.69	0.69	0.68
F-kNN	0.77	0.80	0.80	0.81	0.82	0.80	0.79	0.80

Table 2: Classification rates on the validation dataset

During the validation stage, the use of the FI metric brings little improvement on the Euclidean kNN results for small values of k, but the difference becomes more significant when the size of the neighbourhood increases. When this happens, the performance of E-kNN deteriorates, so points correctly classified for a small number of neighbours are misclassified when more neighbours are taken into account, reflecting heterogeneity in the local structure of the network. The stability of F-kNN under this variation is caused by the FI metric moving the areas of the space with a high density of points from the same class away from the border regions between classes. This means that points away from the border form very compact and homogeneous groups that are far away from the areas of mixed membership, therefore having more stable neighbourhoods with respect to k.

3.3 **Class substructure**

Going back to Fig.3, we stated that a clear structure in the network is easy to see, not only because the classes are well separated, but also because within each class, nodes are arranged forming small groups or clusters. In this section, we briefly look into some of these clusters to find out the differences between them.
The plots in Fig.4 are the mean spectra of the points in each cluster, the first plot corresponding to the real GL spectrum. The four clusters are part of the GL class, and are circled in red in the miniature view of the network.

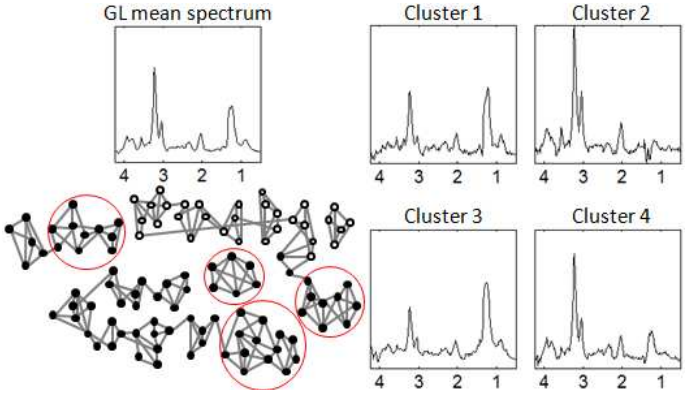


Figure 4: Mean spectra of the different clusters.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

The plots show how clusters from the same class are different in terms of the height of the two main peaks of the spectrum, which vary from group to group giving rise to a different "prototype" in each of them.
The shape of each spectrum has a specific meaning and corresponds to a different medical condition, so within the same GL class we can find different subtypes of brain tissue. In other words, a sample classified as GL could be further subclassified depending on where it lies within the network.

4 Conclusions

The FI can work as a measure of how similar points in the space are with respect to some class membership probability distribution. To do so, we derive a metric from the FI matrix, and therefore a distance measure. This can be used to build a relational network that captures similarity in the original data space and translates it into node to node connections, resulting in a more interpretable representation of the data, especially when the original data space is of high dimensionality.
The structure of the network contains useful information on how the data is distributed in the space. Section 3.3 presented a very simple analysis of some of the substructures found in the dataset. Our motivation for the use of networks is to develop a way of interpreting new data by mapping it into the base network and relating it to the reference cases in it. By doing so, we go from just a scalar that represents the probability of a point belonging to a certain class to a much more informative tool that not only predicts a category for the data, but also puts it into context by telling how it relates to the most representative cases.
It is important to bear in mind the small sample size of the data (109 points in a 195-dimensional space). We chose to keep the original size in the synthetic dataset because it is not crucial for our aim of showing the interpretability of the methodology. However, if the estimation of the probability surfaces was required to be very precise, a larger dataset would be necessary.

References

[1] S. Amari (2001). Information geometry on hierarchy of probability distributions. IEEE Information Theory, 47, vol. 5: 1701-1711.
[2] S. Kaski, & J. Sinkkonen (2000). Metrics that learn relevance. IJCNN 2000 proceedings, vol 5:547-552.
[3] S. Kullback (1959). Information theory and statistics. Wiley. New York, 1959.
[4] H. Ruiz, I. H. Jarman, J. D. Martín, & P. J. Lisboa (2011). The role of Fisher information in primary data space for neighbourhood mapping. ESANN 2011 proceedings.
[5] T. Jebara, J. Wang, & S. F. Chang (2009). Graph construction and b-matching for semi-supervised learning. ICML 2009 proceedings.
[6] B. Huang, & T. Jebara (2011). Fast b-matching via sufficient selection Belief Propagation. AISTATS 2011 proceedings.
[7] S. Ortega-Martorell, A. Vellido, P.J.G. Lisboa, M. Julia-Sape, & C. Arus (2011). Spectral decomposition methods for the analysis of MRS information from human brain tumours. IJCNN 2011 proceedings.
[8] A. Vellido, E. Romero, M. Julia-Sape, C. Majos, A. Moreno-Torres, J. Pujol, & C. Arus (2011). Robust discrimination of glioblastomas from metastatic brain tumors on the basis of single-voxel ^{1}H MRS. NMR in Biomedicine. In press. DOI: 10.1002/nbm.1797

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

The role of Fisher information in primary data space for neighbourhood mapping

H. Ruiz[1], I. H. Jarman[2], J. D. Martín[3], P. J. Lisboa[1]

1 - School of Computing and Mathematical Sciences - Department of Mathematics and Statistics - LJMU, Liverpool L3 3AF - UK
2 - Centre for Public Health - LJMU, Liverpool L3 2ET - UK
3 - Escuela Técnica Superior de Ingeniería - Departamento de Ingeniería Electrónica - Universidad de Valencia, Burjassot (Valencia) - Spain

Abstract. Clustering methods and nearest neighbour classifiers typically compute distances between data points as a measure of similarity, with nearby pairs of points considered more like each other than remote pairs. The distance measure of choice is often Euclidean, implicitly treating all directions in space as equally relevant. This paper reviews the application of Fisher information to derive a metric in primary data space. The aim is to provide a natural coordinate space to represent pairwise distances with respect to a probability distribution $p(c|x)$, defined by an external label c, and use it to compute more informative distances.

1 Introduction

The primary purpose of this work is to define framework to calculate the similarity between data points, in primary data space, using an auxiliary variable which is a class label. This will enable networks of data points to be arranged in a way that is informed about this variable. For the sake of illustration, we measure classification rate using k-NN to evaluate the near neighbour homogeneity of the data with respect to the auxiliary variable.

In the standard formulation, each observation consists of a set of N variables, and therefore represents a point in the N-dimensional space. A very intuitive and widely used way to compute distances between data points is to use the Euclidean metric. This distance assigns equal relevance to all directions and, by extension, to all variables, but in reality each attribute will have a different degree of influence over the auxiliary label c. In this work, similarity between data points is defined with respect to some auxiliary data comprising observations of a dichotomous variable c which divides the dataset into two classes. Data points are considered close to each other if they have similar class membership probabilities, and this definition also applies to groups of points and areas of the dataspace. The goal of this definition of similarity is to form clusters or divide the data into classes that are homogeneous with respect to the label c. That is precisely what a learning metric does. Effectively, such a metric resizes each dimension in space expanding those corresponding to relevant features and compressing those related with less important ones.

While the bulk of statistical work on Fisher information is focused on the space of model parameters, there also is some previous work on learning metrics defined in primary data space [1,2,3] with successful applications to self-organizing maps and standard clustering algorithms such as k-means. In common with the literature, the

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

work presented in this paper shares the objective of developing an intelligent metric that improves the performance of the algorithms, but differs in three key aspects.

Firstly, the way the Fisher metric is obtained. By definition, the metric is derived from the probability density *p(c|x)*, which must therefore be estimated [1,2,3]. The estimations used in this work are drawn directly from the posterior distributions of class membership, with either generalised linear models or generic (semi-parametric) non-linear inference models, namely a linear logistic regressor and a multilayer perceptron (MLP).

The second main difference is the approach used to compute distances. In the non-Euclidean space resulting from the application of the Fisher metric, the shortest distance requires the explicit optimisation of the distance measured across a geodesic path. This is discussed in [3,4,5], but the two most related solutions [3] make strong simplifying assumptions, one using a single straight line between distant points and the other depending on the particular layout of the data points. We propose a new method which is efficient in iteratively adjusting the path towards the shortest distance, as explained in section 2.2.

Finally, the motivation for the construction of the Fisher metric in this work is different than that of the existing literature. In previous work, the concept of learning metrics is included into existing clustering and classification methods to improve their performance. That is not the case here. The methodology that this paper presents has been developed with the intention of applying it to the construction of graphs from datasets with auxiliary variables.

2 Methodology

This section describes the different concepts involved in the metric building process. First the Fisher information is introduced and derived for linear and non-linear estimators assuming a logistic regression transfer function of the output. Second, the problem of finding geodesics is addressed and introduces the proposed generic approach for distance estimation in primary data space with non-linear metrics.

2.1 Fisher information in the primary data space

The FI value [6] at a particular data point *x* in the space is the difference between the information that the probability distributions *p(c|x)* and *p(c|x+dx)* carry, where *dx* is infinitesimally small. In other words, a large FI value at a certain point means that a slight change in the position of that point strongly influences the posterior density function and thus that area of the space is very relevant with respect to the auxiliary data *c*.

The metric is defined by the matrix *G(x)* in the well-known quadratic differential form [7]:

$$ds^2 = dx^T G(x) dx = \sum_{i=1}^{N} \sum_{j=1}^{N} g_{ij}(x)\, dx_i dx_j \qquad (1)$$

The Fisher information matrix in primary data space is defined equivalently by:

$$FI(x)_{p(c|x)} = \begin{cases} E_{p(c|x)}\{(\nabla \ln p(c|x))^2\} \\ -E_{p(c|x)}\{(\nabla^2 \ln p(c|x))\} \end{cases}$$

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

The calculation involves the conditional expectation over the values of the external label *c* with respect to the probability function *p(c|x)*. Limiting *c* to a discrete variable simplifies the calculation because the expectation, which would be computed as an integral in the continuous case, becomes a summation. We further assume a specific structure for the form of the posterior probability *p(c|x)*, namely

$$p_c = p(c|x) = \frac{c + (1-c)e^{-a(x)}}{1 + e^{-a(x)}} \quad , \quad c = \{0,1\} \tag{2}$$

The dependence on *x* is contained in the activation variable *a*, which defines the complexity of the estimator. In the logistic regressor, *a* is just a linear combination of the input vector *x* and the coefficient vector β, while in the case of the MLP it is given by a non-linear, but differentiable function of the inputs. Once *p(c|x)* is defined, the FI can be expressed in matrix form, assuming column vectors, as follows:

$$FI(x) = \big(\nabla a(x)\big)\big(\nabla a(x)\big)^T p_{c=1}(1 - p_{c=1})$$

Returning to (1) yields the distance between infinitely close points. A general formula for the distance between two points is obtained by solving the path integral:

$$d(x_A, x_B) = \left| \int_0^1 \sqrt{\dot{x}(t)^T FI\big(x(t)\big)\dot{x}(t)}\, dt \right| \tag{3}$$

The next section provides a solution of integral in (3) in closed form for dichotomous classifiers of the assumed form, whether linear or not.

2.1.1 Fisher distance with a linear estimator

We start with the logistic regression, with $a=\beta^T x+\beta_0$. Since *a* is linearly dependent on *x*, its first derivative is constant, resulting in the following expression for the integral:

$$d(x_A, x_B) = \left| \int_0^1 \sqrt{\dot{x}(t)^T \beta\beta^T \dot{x}(t) \cdot p_{c=1}(1 - p_{c=1})}\, dt \right| \tag{4}$$

which is readily solved by substituting *a* as the integration variable giving:

$$d(x_A, x_B) = \left| 2\left[arctg\left(e^{-\frac{a(t)}{2}} \right) \right]_{a(t=0)}^{a(t=1)} \right| \tag{5}$$

It is important to note that distance is independent of the particular path from x_A to x_B., in effect collapsing the data space onto the projections along the vector defined by the weights β.

2.1.2 Extension to a non-linear estimator

The natural next step is to develop an expression for the distance when using a non-linear estimator of the posterior density distribution by solving equation (3) as in the previous section, but with *a* as a non-linear function of *x.* Using the first two terms of the Taylor expansion of *a* produces a linear approximation for which the distance expression (5) applies. Equation (5) is thus globally applicable for the linear case, but only locally valid for a non-linear estimator. In this work, the so-called free points approach is used to overcome this limitation.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

2.2 Geodesic distances. The free points approach

The algorithm described in this section performs two important functions: it ensures that the Taylor approximation used previously holds and it finds the geodesic path between x_A and x_B. Figure 1 shows an illustrative sketch of the approach.

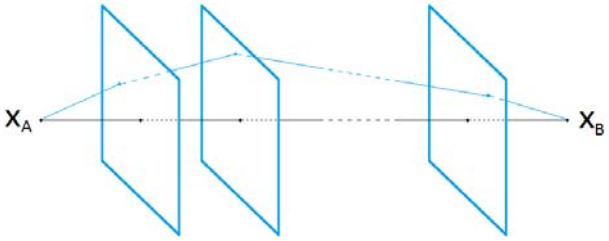


Fig. 1: The free points approach for 3-dimensional data.

The method starts by dividing the straight line joining x_A and x_B into segments. A hyperplane is then defined between each pair of consecutive segments. All these hyperplanes are parallel between each other and orthogonal with respect to the straight line path. Then, a point is defined in each hyperplane, with the idea of forming a path from x_A to x_B by joining all the points as shown in fig. 1. Since the points can move freely within their respective hyperplanes, any path can be formed by appropriately choosing the number of hyperplanes and the position of the points.

The points move as a result of the minimization of the objective function, defined as the overall length of the path computed as the sum of each segment's length. Each of these individual distances is calculated using (5). Since this applies locally for the non-linear case, every free point must be close to its two neighbours, and that is guaranteed by choosing a large enough number of hyperplanes.

3 Experimental results

In this section, the Fisher metric is put into practice in a classification problem using synthetic data. Two versions of the standard k-nearest neighbours (kNN) classifier are compared: one computes pairwise distances using the Euclidean metric (E-kNN) and the other uses the Fisher distance (F-kNN) derived from a MLP.

The method is benchmarked using a kNN classifier to assess the homogeneity of the resulting network with respect to the external label, not because the classifier itself brings any originality. Fisher metric based classifiers can be found in the literature, the most important being the SVM-Fisher kernel methods [8].

The dataset consists of two classes generated by two Gaussian distributions with same means but different standard deviations (0.9 and 2). One distribution contains the other, creating a non-linear border. This is a large dataset (10^4 samples/class) that provides the MLP with enough training episodes to accurately estimate $p(c|x)$, which is critical for the estimation of the Fisher information. A validation dataset is generated using the original generating functions of the data. This smaller dataset (250 samples/class) contains the points to be classified, calculating distances using either the Euclidean or Fisher metric and taking a majority vote in the usual way.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

Table 1 shows the results of the simulations. The first row in each cell shows the percentage of correctly classified points using E-kNN and F-kNN in that order. The relative increase of accuracy when using the Fisher metric appears in the second row.

k \ N	2	5	10	15	25	40
3	69.4/72 +3.75%	87.4/73.6 -15.79%	88.8/92.2 +3.83%	81.6/93.2 +14.21%	66.4/94.6 +42.47%	51.6/94.2 +82.56%
5	72.4/74.8 +3.31%	88.2/72.8 -17.46%	89/92.2 +3.6%	80/93.4 +16.75%	64.4/95 +47.52%	50.6/94 +85.77%
7	74/74 +0%	88.2/74.4 -15.65%	88.8/92.2 +3.83%	77.6/93 +19.85%	61.8/95.2 54.05%	50.4/94 +86.51%
11	73.6/75.6 +2.72%	88.8/76.6 -13.74%	87.2/93.2 +6.88%	74.8/93.2 +24.6%	57/95.2 +67.02%	50.2/94 87.25%
15	75.4/76.8 +1.86%	88.2/79 -10.43%	86/93.4 +8.6%	73/93.8 +28.49%	56.2/95.2 +69.4%	50.2/94.2 +87.65%
21	75/76.6 +2.13%	88.6/79.4 -10.38%	85.8/93.8 +9.32%	71.4/93.2 +30.53%	54.4/95.2 +75%	50/94.4 +88.8%

Table 1: Simulation results for input dimensionality N and k neighbours.

In low dimensions, the two methods perform similarly. However, the accuracy of the Euclidean classifier increases until N=10 and decreases from then on. To understand this behaviour, a histogram of the pairwise distances is plotted in fig. 2 for different values of N, comparing interclass and intraclass distances. The performance of kNN is best when intraclass distances are small compared to interclass distances.

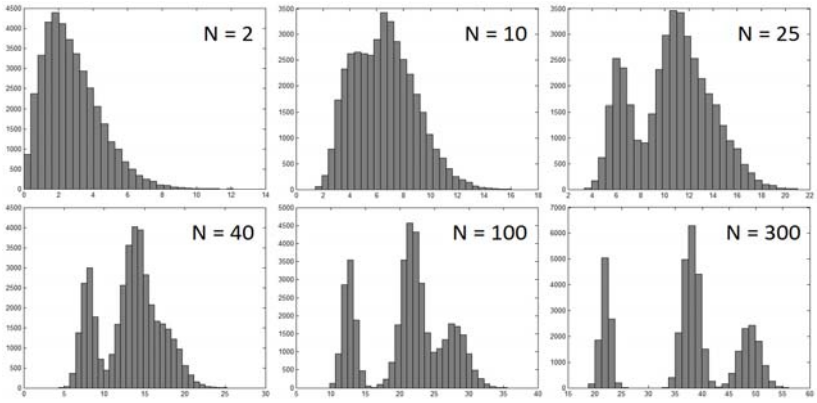


Fig. 2: Histograms of the pairwise distances for different values of N.

The 2-dimensional case in Figure 2 shows all three distributions overlapping. Individual histograms show that intraclass distances have their peak slightly more to the left than interclass distances. In the next plot, N=10, the shape shifts right and starts splitting up into two humps, the left one corresponding to class 1 intraclass distances and the other formed by class 2 intraclass distances and interclass distances. The increase of the distances is related to the increase of the diagonal of a hypercube when N grows and is caused by the nature of the high dimensional space.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

The reason for class 1 distances to shift more slowly is their smaller standard deviation. At this point, the classification of class 1 members becomes easier because their intraclass distances remain small with respect to interclass ones. For class 2 the situation is similar as when N=2, so the overall result is an increase of the accuracy.

In the last four cases the distances keep growing as mentioned. Very important is the fact that class 2 distances increase faster than interclass distances. This results in a clear division of the three groups with increasing width and mean when going from class 1 to class 2, causing the classification of all points as class 1 members. In the case of an actual class 1 member, intraclass distances are much smaller than interclass ones, so the k chosen neighbours always belong to class 1. For class 2, interclass distances are smaller, resulting in a wrong choice of neighbours and a bad prediction.

The Fisher metric compensates this effect by resizing the space dimensions. On top of that, the MLP estimates $p(c|x)$ much better in high dimensions, so the result obtained is a very accurate classification. Also notice the stability of the percentages achieved with F-kNN when the parameter k varies for large values of N.

4 Conclusions

This paper outlines the construction process of the Fisher metric from the choice of the probability estimator to the development of a distance expression. Unique from any previous work, the Fisher information is derived from sigmoidal output estimators, and from this an analytical expression is obtained for the Fisher matrix.

In addition, a closed form expression for the geodesic distance is obtained by solving the path integral for a linear estimator. This opens the door to a distance expression for the general non-linear case by local linearization of the response surface of the MLP. Then the free points approach is introduced to find the geodesic between points with the new metric and also to ensure that the approximations hold.

References

[1] S. Kaski, J. Sinkkonen and J. Peltonen, Bankruptcy analysis with self-organizing maps in learning metrics, *IEEE Transactions on Neural Networks*, 12(4):936-947, 2001.

[2] J. Salojärvi, S. Kaski and J. Sinkkonen, Discriminative clustering in Fisher metrics. In *Artificial Neural Networks and Neural Information Processing – Supplementary proceedings* (ICANN/ICONIP 2003), June 26-29, Istanbul (Turkey), 2003.

[3] J. Peltonen, A. Klami and S. Kaski, Improved learning of Riemannian metrics for exploratory analysis, *Neural Networks*, 17:1087-1100, 2004.

[4] R. Kimmel and J. A. Sethian, Computing geodesic paths on manifolds, *Proceedings of the National Academy of Sciences of the USA*, 95(15):8431-8435, 1998.

[5] J. Baek, A. Deopurkar and K. Redfield, Finding geodesics on surfaces, unpublished, 2007.

[6] S. Kullback. *Information theory and statistics,* John Willey and Sons, New York, 1959.

[7] M. H. J. Gruber, Some applications of the Rao distance to shrinkage estimators, *Communications in Statistics – Theory and Methods*, 37:180-193, 2008.

[8] T. Jaakkola, M. Diekhans and D. Haussler, Using the Fisher kernel method to detect remote protein homologies, *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology* (ISMB-99), 149-158, August 6-10, Heidelberg (Germany), 1999.