TECHNIQUES FOR DATA PATTERN SELECTION AND ABSTRACTION

"Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by Konstantinos Nikolaidis."

September 2012

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Dr Goulermas for his invaluable and constructive help regarding this research. His suggestions, support and more importantly his tolerance made everything look easier and enabled me to complete this work, which would have been impossible otherwise. After all these years I spent next to him I consider him more a friend than a supervisor.

I would also like to thank my parents and brother for their encouragement and love throughout my studies. Their support made everything possible. I am also grateful to my grandmothers for their constant support. I also wish to acknowledge the help and support of Helen.

I would also like to express my appreciation to Myrto Pavlini for her love, tolerance and understanding.

Finally, I my special thanks are extended to the following people:

- Eduardo Rodriguez for his valuable advice on Maths and Genetic programming as well as his assistance with the Matlab tool.
- Tingting Miu for her assistance and suggestions on my last piece of work for Chapter 6 of this thesis.
- Elena Marchiori for providing her code of the CCIS algorithm.
- The entire staff of the Electrical and Electronics Engineering Department of Liverpool University for their excellent disposition.
- Last but not least, Del, Gino, Biliou and Kornel for the endless hours we spent together.

ABSTRACT

This thesis concerns the problem of prototype reduction in instance-based learning. In order to deal with problems such as storage requirements, sensitivity to noise and computational complexity, various algorithms have been presented that condense the number of stored prototypes, while maintaining competent classification accuracy. Instance selection, which recovers a smaller subset of the original training set, is the most widely used technique for instance reduction. But, prototype abstraction that generates new prototypes to replace the initial ones has also gained a lot of interest recently. The major contribution of this work is the proposal of four novel frameworks for performing prototype reduction, the Class Boundary Preserving algorithm (CBP), a hybrid method that uses both selection and generation of prototypes, Instance Seriation for Prototype Abstraction (ISPA), which is an abstraction algorithm, and two selective techniques, Spectral Instance Reduction (SIR) and Direct Weight Optimization (DWO).

CBP is a multi-stage method based on a simple heuristic that is very effective in identifying samples close to class borders. Using a noise filter harmful instances are removed, while the powerful heuristic determines the geometrical distribution of patterns around every instance. Together with the concepts of nearest enemy pairs and mean shift clustering this algorithm decides on the final set of retained prototypes.

DWO is a selection model whose output set of prototypes is decided by a set of binary weights. These weights are computed according to an objective function composed of the ratio between the nearest friend and nearest enemy of every sample. In order to obtain good quality results DWO is optimized using a genetic algorithm.

ISPA is an abstraction technique that employs the concept of data seriation to organize instances in an arrangement that favours merging between them. As a result, a new set of prototypes is created.

Results show that CBP, SIR and DWO, the three major algorithms presented in this thesis, are competent and efficient in terms of at least one of the two basic objectives, classification accuracy and condensation ratio. The comparison against other successful condensation algorithms illustrates the competitiveness of the proposed models.

The SIR algorithm presents a set of border discriminating features (BDFs) that depicts the local distribution of friends and enemies of all samples. These are then used along with spectral graph theory to partition the training set in to border and internal instances.

TABLE OF CONTENTS

CHAPTER 1:

MACHINE LEARNING

The purpose of this chapter is to provide a broad overview of the field of Machine learning. This section emphasizes on instance-based learning, and more specifically on non-parametric methods for classification using the nearest neighbour rule. Section 1.1 briefly describes the concept of Machine learning and its applications. Section 1.2 provides a small introduction to the pattern recognition problem in machine learning. Section 1.3 is a brief summary of the various types of classifiers that exist and introduces the Nearest Neighbour (NN) classifier, which is used throughout this work. Finally, section 1.4 presents the problems that arise in instance-based learning and are related with the use of NN, and explains the rationale behind this PhD study.

1.1 Introduction

Machine learning is a scientific field that has its origins in computer science. It is a subfield of Artificial Intelligence and it involves the development of models that can mimic intelligent abilities of humans. These adaptive models are trained in order to "learn", which can be understood as the ability of a machine to automatically make decisions. The learning process of computers is achieved by the design of algorithms and techniques that train their parameters using past experience enabling them to adapt to the environment. The constantly increasing amount of information available has brought a lot of attention to information theory and data analysis, with substantial developments in the last two decades. Machine learning has grown from really simple tasks that only concerned the few (specialists), to more of a mainstream concerned with highly complicated statistical and computational principles. The evolution of Machine learning has transformed it into an interdisciplinary field that makes use of some basic tools such

as statistics, probability theory, information theory, optimization and control theory, but also it expands to other scientific areas like philosophy and archaeology [Miu08].

Machine learning appears in many aspects of modern day life as it has a very wide range of applications. From simple examples such as web page ranking in search engines and automatic translations to rather complex applications such as medical diagnosis and bio-informatics, machine learning is successfully applied. It is actively used the last few years for security purposes, i.e. face recognition or verification, fingerprint recognition or credit card fraud [Smo10]. Other applications that involve machine learning are financial such as stock market analysis and direct marketing, robotics, computer games, image processing, speech or handwriting recognition and failure detection.

1.2 Pattern Recognition

The problem of pattern recognition has been the focus of research for many years, and while initially it was mostly on a theoretical basis the development of machine learning algorithms enabled the use of it on cutting edge practical applications. Bishop [Bis06] described machine learning algorithms as a function $\psi(\boldsymbol{x})$, which takes an input vector $\boldsymbol{x}$ and generates an output vector $\boldsymbol{y}$ of the same form as the target vector. Machine learning algorithms consist of two distinct stages, the first one being the training phase. During this stage, the model uses the input vectors to train its parameters according to the learning function ψ. The key objective of the learning process is the ability of the model to generalize, meaning to extract general information from the inputs that enables it to correctly treat unknown data patterns. Once the learning process ends, the model proceeds to the testing phase. The algorithm is then tested on new unknown samples that determine the generalization capability of the model.

The taxonomy of machine learning problems largely varies as the characterization of algorithms can be based on different elements. For example, depending on the type of training data used, algorithms can be categorized to *supervised* or *unsupervised* learning methods, the former being the one mainly addressed in this work. Supervised learning algorithms receive as input vectors pairs of objects consisting of the vectors and their respective target values. Unsupervised learning methods, on the contrary, are characterized by the absence of a priori information. Other types of algorithms include semi-supervised, active learning and reinforcement learning methods. Another grouping can be made depending on the desired output of each model. If the output of the model is a continuous variable it is the answer to a regression problem. On

the other hand, the output of a classification problem is a class label that represents the category of the pattern.

More precisely, pattern classification is a formulation of supervised learning whose goal is to accurately predict the class labels of unseen patterns. The decision making of the model is driven by the training data supplied to the algorithm. Given a set of n labeled training samples $X = \{x_1, x_2, ..., x_n\} \in R^d$, where R^d is the d-dimensional real feature space, and each sample is associated with a unique class label $\psi(x) \in L = \{l_1, l_2, ..., l_c\}$, with c being the number of classes, the objective of a classification algorithm is to construct a functional mapping $\psi : R^n \rightarrow L$ so that any unseen sample x_i is correctly assigned to a class label l_i. Pattern classification can be sorted in binary and multi-class classification. Binary classification is the task of classifying the input samples into one of two possible sets, whereas the latter can assign a pattern to one of multiple classes. In order to simplify the multi-class problem, in some cases, one can consider it as a series of binary problems. Although many consider them two different tasks, no distinction between binary and multi-class classification is made in the experiments and implementations of this study.

1.3 Nearest Neighbour Classifier

Despite the fact that pattern recognition is a relatively new science, various classifiers have been introduced. A large group of classifiers, namely linear classifiers, are designed to classify data regardless of the underlying distribution of the training patterns. In this case, the decision surface is considered to be a linear function of the unknown pattern x. Linear classifiers are known for being relatively simple and computationally inexpensive [The99]; such models are linear discriminant functions like [Fis36], [Zha10$_a$], which are used in various applications [Yu08], and the perceptron algorithm [Hay99]. For more complicated case where classes are non-linearly separable the use of non-linear classifiers is required. Some examples of such algorithms are the multi-layered perceptron methods [Hay99] and the radial basis function network [Hay99] or the decision trees [Sug06]. Another approach involves the classification of patterns based on the probability of it belonging to a certain class. These classifiers depend on the probability distributions of the training patterns; some representative algorithms are the maximum likelihood parameter estimator [Kay93], the parzen windows approach [Bab96] and the nearest neighbour classifier [And02].

Arguably the simplest method for pattern classification is the k-NN classifier, which is based on the Nearest Neighbour (NN) rule, one of the better-known instance

based learning algorithms to perform supervised non-parametric classification. It is widely used in machine learning because of its simplicity and the fact that its error probability is bounded by twice the Bayes error rate. All instances of the training set are represented by position vectors in a multidimensional feature space, and the *k*-Nearest Neighbour rule (*k*-NN) classifies unseen samples based on their closest *k* instances and requires *k* being a positive integer. In its simplest form, where *k*=1, the output value is simply the class of the nearest neighbour. Otherwise, the pattern is assigned to the class of the majority of its *k* nearest neighbours. Hence, in order to avoid ties between classes, *k* is usually chosen as an odd number. Despite the fact that *k*-NN is a learning method that can be used for regression as well, it is utilized only as a non-parametric classifier in this PhD study.

1.4 Problem and Scope

Algorithms that use the NN rule, and instance-based learning methods more generally, suffer from two principal issues. Firstly, a major concern is storage requirement because of the need to store the entire dataset in some type of memory. Secondly, the increased time complexity from having to search large portions of the stored prototypes, in order to predict new queries. The larger the dataset used the higher the response time of the algorithm. Apart from these, a third concern is the noisy instances present in the database. Along with the entire training set noisy instances are also stored, thus degrading accuracy and overall performance of the algorithm.

In order to tackle these drawbacks, rapid advances have been made in the field of data condensation, with the development of numerous methods that target in reducing the training set size, while keeping the error rate as low as possible. Hence, the problem in instance reduction is to determine a set of $m << n = |X|$ prototypes using the original training set $X \in R^d$ that can accurately describe the original distribution. Therefore, the resultant prototype set will allow not only high classification accuracy but also minimal cardinality, and as a result computational efficiency. So, data reduction methods seek the minimum number of instances that can provide the maximum possible classification accuracy, and as [Gar10] and [Tri11] explain in their review articles, can be categorized to *instance selection* algorithms, which select a small representative subset of the initial training set, and *instance abstraction* algorithms that generate a new set of prototypes to replace the initial ones. The latter type of methods can often result in higher condensation, due to the freedom of replacing instances, but this may lose track of the contribution of the original instances. On the other hand, the former type of methods is only allowed to select instances from the original ones, leading thus to lower

condensation, in an attempt to optimise both objectives of accuracy and condensation together. Which type is used depends on the focus of the application. If the creation of new samples to fill regions in the domain of the problem to improve weak representative samples in the original dataset is prioritised, the latter type is preferred. Otherwise, if the preservation of the geometric and discriminative characteristics of the original instances is prioritised, the former type is preferred. Also, instance selection methods are usually much faster.

The wide range of algorithms developed to deal with the issues related to instance-based learning show how significant the problem is. As a result, many works including [Jan04$_a$, Jan04$_b$, Wil00], have analysed and compared various instance reduction techniques. A clear distinction should be made between instance reduction, with which this thesis is concerned, and dimensionality reduction. Considering a dataset as a matrix, instance reduction decreases the number of the rows of the matrix (attributes), while dimensionality reduction deals with the columns (features) of the matrix.

The aim of this thesis is two-fold. Firstly, to solve the problem of instance-based learning using new alternatives. This is achieved by introducing novel techniques for instance reduction. Secondly, to contribute in the field of machine learning not only theoretically, but also practically. In order for the developed algorithms to be successful, they should involve innovative aspects, but they should also be effective. Therefore, the proposed techniques should account for improvements and enhancements in terms of the required objectives, when compared to already known methods in the literature.

In chapters 2 and 3, an investigation of previous work done on the field of instance selection and abstraction is presented and a thorough analysis of each method is performed. This thesis identifies the important aspects of data condensation, based on which it proposes some novel techniques for prototype reduction. These techniques are presented in chapters 4, 5 and 6. The epilogue recapitulates the contributions and advantages of the proposed algorithm, while it discusses possible improvements along with new topics for research.

1.5 Practical Development

In order to test the methods proposed in this thesis various experiments were performed on both synthetic and real datasets. Synthetic examples used were created by Dr Goulerma's group in Liverpool University, while the real datasets were selected from the UCI Machine Learning Repository [Bla98]. In order to evaluate the performance and the capabilities of these methods, a comparison was made against other well-known instance

reduction techniques. Some of these algorithms were implemented by the author of this thesis, while their respective authors provided others. The software tool used to develop the algorithms and obtain all experimental results was Matlab.

1.6 Publications

- Nikolaidis, K., Rodriguez, M.E., Goulermas, J.Y., and Wu, Q.H., 2010. "Instance seriation for prototype abstraction." In *Proc. of IEEE 5th BICTA International Conference*, Liverpool, pp. 1351-1355.
- Rodriguez, M.E., Nikolaidis, K., Goulermas, J.Y., Ralph, J.F., and Miu, T., 2010. "Collaborative projection pursuit for face recognition." In *Proc. of IEEE 5th BICTA International Conference*, Liverpool, UK.
- Nikolaidis, K., Goulermas, J.Y., and Wu, Q.H., 2011. "A class boundary preserving algorithm for data condensation." *Pattern Recognition*, vol.44, pp. 704-715.
- Nikolaidis, K., Rodriguez, M.E., Goulermas, J.Y., and Wu, Q.H., 2012. "Spectral graph optimization for instance reduction." *IEEE Trans. Neural Networks*, vol. 23, pp. 1169-1175.
- Rodriguez, M.E., Nikolaidis, K., Miu, T., Ralph, J.F., and Goulermas, J.Y., 2012. "Towards collaborative feature extraction for face recognition." *Natural Computing*, vol.11(3), pp. 395-404.
- Nikolaidis, K., Miu, T., and Goulermas, J.Y., 2013. "Prototype reduction based on direct weighted pruning." *Pattern Recognition*. (submitted for publication)

CHAPTER 2:

INSTANCE SELECTION

The aim of this chapter is to accurately describe the field of prototype reduction, more specifically instance selection algorithms, and provide a thorough analysis of various existing methods in the literature. Section 2.1 briefly describes the concept of instance selection and its processes. Section 2.2 describes instance selection algorithms that are based on the use of the nearest neighbour concept. In section 2.3 methods that define the relative neighbourhood of prototypes are presented. Section 2.4 is an extensive analysis of various graph methods used for prototype reduction. In 2.5 algorithms using instance weight learning for instance selection are described, while section 2.6 demonstrates the developments on nearest enemy-based techniques for instances reduction. Section 2.7 investigates instance-based techniques that use density estimation as the main tool for condensation. Finally, section 2.8 presents some novel prototype selection algorithms that employ unusual means, such as evolutionary computation or projection of samples to new dissimilarity spaces.

2.1 Introduction

In order to tackle all problems that arise with the use of the nearest neighbour rule in classification, numerous methods have been developed and exist in the literature, that intend to prune the number of prototypes and simultaneously keep the error rate as low as possible. These data condensation methods can be categorized in to two subgroups, instance selection algorithms, which select a small representative subset of the initial training set, and prototype abstraction algorithms that generate a new set of prototypes to replace the initial ones. The former type of methods has been widely used since it has been the subject of research for nearly 50 years, since the first selective algorithms were introduced [Har68] and [Wil72]. By simply selecting a subset of the initial training set, selection algorithms have the advantage of maintaining the majority of the information

existing in the training set. Prototype selection methods can be further subcategorized to editing algorithms, that aim to improve classification accuracy by removing harmful instances, condensation techniques that focus on discarding superfluous instances, and hybrid methods, which are a combination of the other two, and demonstrate highly competitive performances since they deal not only with noisy but redundant prototypes as well [Gar10].

Classification involves the use of a training set X of preclassified instances, and a testing set of unseen samples. In instance selection algorithms, during the training process, a small subset of X is selected and applying the k-NN classifier it is used to predict the class labels of all samples in the testing set. As already mentioned no artificial prototypes are generated; Hence, having an initial set $X = \left\{ x \in \Re^d \right\}$ of n d-dimensional instances, where each sample is associated with a unique class label $\psi(x) \in L = \left\{ l_1, ..., l_c \right\}$, the problem in instance selection is to determine a set of m representative prototypes from X (where $m << n$) that best describes the initial distribution.

2.2 Editing and Condensing Nearest Neighbour

One of the simplest editing rules is the Editing Nearest Neighbour (ENN), proposed by Wilson in 1972 [Wil72]. Given a set of n labeled instances, Wilson used the k nearest neighbour rule to reach a decision for every instance and filter the original training set. His method selects an instance x_i from X and its k nearest neighbours are computed. The class of x_i is determined by the class of the majority of its k nearest neighbours, and whenever a tie occurs, random selection is used to assign the class. In case of misclassification, x_i is removed from the original set X. Hence, ENN is an iterative algorithm and the final subset contains only instances that are correctly classified by their k-NN. As a result, noisy samples are removed resulting to the improvement of the classification accuracy.

The Condensed Nearest Neighbour (CNN) rule, introduced by Peter Hart in 1968 [Har68], is one of the first techniques of supervised instance selection. It is an additive algorithm that concentrates on reducing a training set to the smallest possible subset S that can classify all instances of the initial set correctly. Initially, a random instance x_i of X is selected and inserted in S. Then, another instance x_j is chosen and using the NN rule is classified according to S. In case of miss-classification it is inserted in S, thus the additive nature of CNN. This repetitive process continues until all samples are classified correctly. In contrast to ENN that enhances classification accuracy by discarding noise,

CNN aims to preserve the classification accuracy already achieved by selecting instances of the training set that correctly classify the rest.

CNN has the disadvantage that much depends on random selection. Therefore, many algorithms were prompted by it, in search of the optimum subset S. One such algorithm is the Reduced Nearest Neighbour (RNN), which enhances data condensation by removing redundant instances [Gat72]. After the application of CNN every prototype x_i in S is tested and if its removal results in no miss-classifications in X, x_i is considered superfluous and permanently discarded. Although RNN highly reduces the size of the original set, it does not guarantee a minimal output set.

Another instance selection technique is the Selective Nearest Neighbour (SNN) proposed in [Rit75], which retains instances close to the class boundaries. SNN rule states that every instance x_i of the original training set has to be closer to a same class (friend) instance of the output subset than to any other enemy instance. In order to achieve this, a binary $n \times n$ matrix A is constructed, such that:

$$A_{ji} = \begin{cases} 1 & if\ x_j \in Y_i \\ 0 & if\ x_j \notin Y_i \end{cases} \tag{2.1}$$

where Y_i is the set of all friend instances of x_i that lie closer than its nearest enemy. Some rules for deletion of rows and columns of A are then applied to obtain the final subset. Although this method can display competitive results, the use of A largely increases its complexity compared to methods such as CNN and RNN.

Based on the SNN algorithm, another method was developed to decrease the computational complexity of the nearest neighbour classification. The Modified Selective Subset (MSS) introduced in [Bar05]. The proposed algorithm is similar to SNN with a slight modification on the Y_i set, which drives MSS to select instances that lie closer to the class boundaries. Consequently, the main purpose of MSS is not a minimal consistent subset like SNN, but rather a more accurate representation of the initial class borders.

In [Tom76] another extension of CNN was introduced. The rule of Ordered CNN aims by discarding centre instances, to extract a small subset of X that has high classification accuracy. To achieve this, OCNN randomly selects an instance x_i along with its nearest enemy x_j, which by definition is going to be a boundary sample. The nearest enemy of x_j that classifies x_i correctly is then computed, x_k, and added to an initially empty set S (Fig. 2.I). All patterns in the original training set are then classified according to S and every time a misclassification occurs the same process takes place. When all patterns are accurately classified and the algorithm terminates, the output set S contains instances that lie close to the decision boundaries.

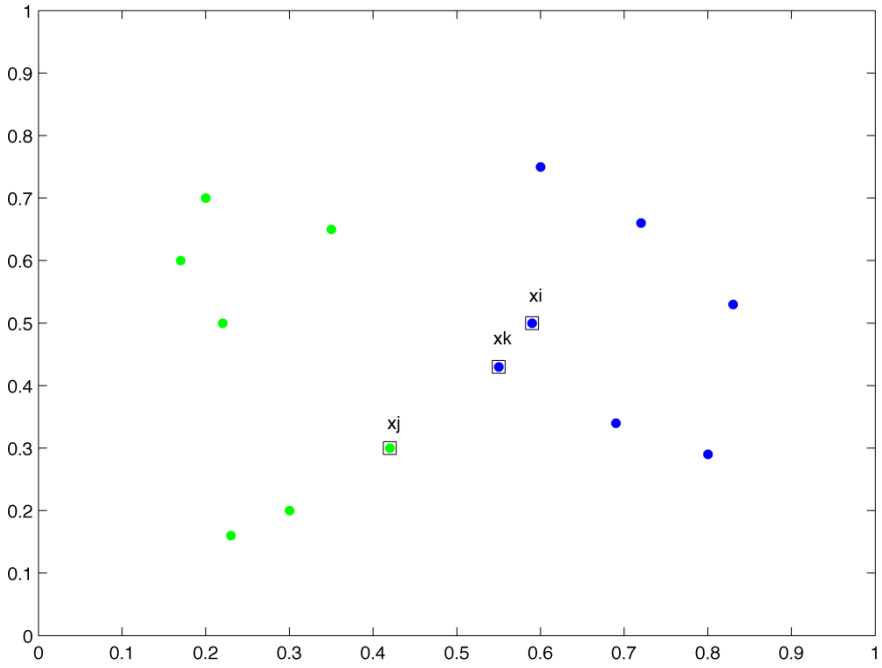


Figure 2.I- A two class two dimensional example. For an instance x_i the OCNN algorithm successively computes its nearest enemy x_j and the same-class border prototype x_k.

Chidananda introduced another method for instance condensation using the concept of mutual neighbourhood [Chi79]. The proposed technique is based on the CNN algorithm with the major difference being its initialization. The Mutual Neighbourhood Value (MNV) of every sample is determined, which is computed with respect to the nearest enemy of every instance. The initial prototype that is selected and moved to an initially empty set S is the one with the lowest MNV. Each sample remaining in the original set is then classified according to S, and only misclassified instances are maintained. This process, similar to CNN, ends only when the final output set can classify all instances successfully. During the second stage of the algorithm an evaluation of the remaining instances takes place. Hence, to deal with redundant samples, an instance x_i is eliminated from S if its removal results in no misclassifications. When all instances are treated, the algorithm terminates and the final subset contains highly informative samples that are close to the class boundaries.

In [Aha91] three methods for instance-based learning were introduced, the Growth (additive) algorithm, Shrink (subtractive) algorithm and Instance Based 3 algorithm (IB3). The Growth algorithm is a very similar technique to CNN since it is initialized with an empty set S and a randomly selected instance x_i. Every instance of the original training set is then classified with respect to S and in case of misclassification is inserted in S. In contrast to CNN only one pass through the training set occurs; hence, the Growth algorithm is not very efficient, nor robust to noise. On the contrary, the Shrink algorithm is initialized with S being equal to the original training set X. Then every

instance is checked and, if it can be classified correctly by the remaining prototypes of S is discarded. According to Wilson [Wil00], this algorithm is extremely sensitive to noise compared to RNN. From the three methods proposed, IB3, which is also an additive algorithm, is the most effective. Only instances that can be characterized as acceptable are moved in to *S*. An instance x_i is retained if it is not classified correctly by its nearest acceptable instance. The confidence interval that determines acceptability is defined as:

$$\frac{p+\frac{z^2}{2r} \pm z\sqrt{\frac{p(p-1)}{r}+\frac{z^2}{4r^2}}}{1+\frac{z^2}{r}} \tag{2.2}$$

where z is the confidence factor[1], r is the number of classification attempts of the given instance of *S*, and p is the classification accuracy based on n. An instance is considered acceptable and retained by IB3 if its accuracy is higher than the upper bound of the confidence interval. While every instance that has lower acceptability than the lower limit is removed immediately from *S*, if an instance is within the confidence interval it is not discarded until the very end of the process.

Another algorithm, called Fast Condensed Nearest Neighbour, was recently proposed in [Ang07a], which discards redundant and harmful instances to largely reduce the size of the training set *X*. FCNN uses a subset *S*, which in the initial state holds the centroids of the classes of *X*. For every instance x_i of *S*, FCNN denotes two sets *A* and *B*, where *A* contains all Voronoi neighbours of x_i, i.e. instances that are closer to x_i than to any other prototype in *S*, while *B* holds the Voronoi enemies of x_i. During each iteration a representative instance from *B*, with respect to x_i, is inserted in *S* and sets *A* and *B* for all instances are updated. This procedure continues until all elements of *S* have no Voronoi enemies, meaning *B* is empty for every x_i belonging to *S*. In the specific work, two different rules to update *S* are analysed, depending on the way of selecting the representative sample. In the first case, FCNN1 selects the nearest neighbour of x_i in *B*, while in the other case the centroids of *B* are selected. The latter algorithm is called FCNN2. Additionally, two rules, the triangle inequality and *k*-Nearest Neighbour, are used, in order to further reduce the computation time and error rate of the algorithm respectively. FCNN is another instance selection method that discards centre instances; hence, *S* consists mainly of instances close to the decision boundary. In order to further improve the performance of Fast Condensation Nearest Neighbour, a new method of condensed nearest neighbour was introduced in [Ang07$_b$], the Parallel FCNN. In this case, the entire training set is divided in *k* subsets $X_1, X_2 \ldots X_k$ that are then assigned to one of *k* parallel nodes $P_1, \ldots, P_k$. Each node uses the FCNN algorithm described before to

[1] 0.9 is used for acceptance and 0.7 to reject.

reduce the number of instances. Computational time required is then reduced as all nodes can communicate with each other to compute the final training set S.

The modified condensed nearest neighbour (MCNN), which is also an instance reduction method based on Hart's algorithm, was proposed in [Sus02]. MCNN initializes a subset S of the original set X, holding just one instance as a representative prototype for each class. Two different cases have been proposed for selecting the initial representatives, either by computing the sample mean of each class and then selecting the closest prototype to the computed value or by selecting the centroids of every class (as seen in Fig. 2.II). The entire set of instances is then classified according to the representative prototypes and all misclassified instances are moved to another set, from which new representative are selected using the same method as before. S is then updated and the training set is once more evaluated in a similar manner until all samples in the training set are classified correctly. The MCNN algorithm also includes a deletion operator to further improve its performance. So every prototype in the final subset S is evaluated and the ones found superfluous are discarded, leading to an even smaller output subset.

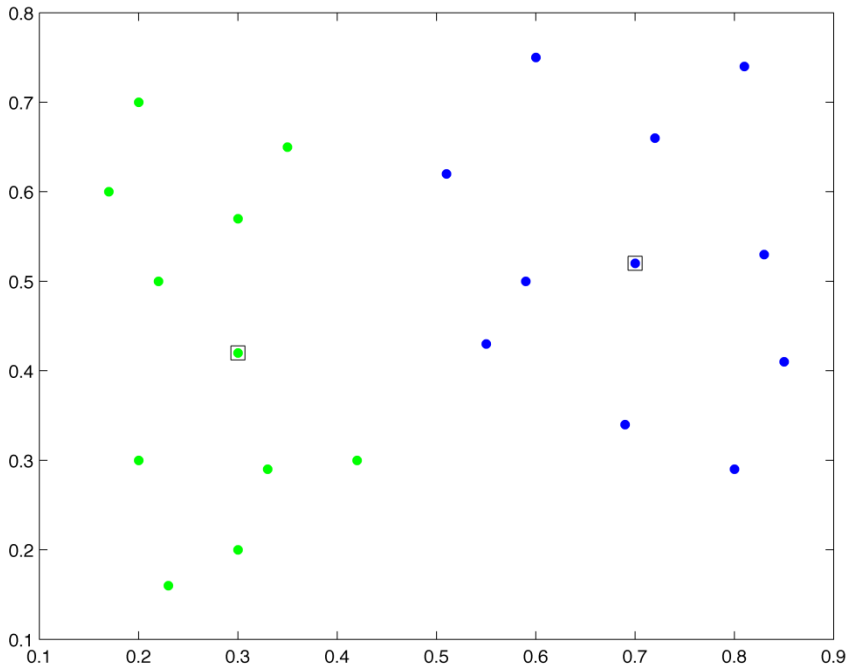


Figure 2.II- A two class two dimensional example similar to the one used in Fig. 2.I. The representative prototypes of the two classes are indicated by "□".

In [Ull74] two more methods for obtaining a reduced subset S are introduced. The first method developed is very similar to the CNN algorithm with the only difference being the use of a 'dead-zone' threshold δ. For an instance x_i of class label $\psi(x_i)$ the decision whether to insert x_i to S is taken according to the following criterion:

$$\left\|x_i - x_z\right\| + \delta < \left\|x_i - x_j\right\| \tag{2.3}$$

where x_z is a same class instance and x_j represents instances of other classes $\psi(x_j) \neq \psi(x_i)$. This method is simplified to CNN if $\delta=0$, and is further enhanced in [Cho06]. The second method described, involves a matrix manipulation similar to Gates proposal in [Gat72] with the addition of the 'dead-zone' threshold.

In 2006 another data reduction method based on condensed nearest neighbour was presented, Generalised CNN [Cho06]. GCNN aimed to select a smaller subset *S* of the initial training set using an implementation very similar to the one introduced in [Har68]. In Condensed Nearest Neighbour an instance x_i is retained as long as:

$$\left\|x_i - q\right\| - \left\|x_i - p\right\| > 0 \tag{2.4}$$

where *p* is the nearest same-class prototype and *q* is the nearest different-class prototype of x_i. In the case of GCNN the rule to retain a prototype is modified as follows:

$$\left\|x_i - q\right\| - \left\|x_i - p\right\| > \rho\delta_n \text{ for } \rho \in [0,1) \tag{2.5}$$

where δ_n is the minimum distance between prototypes of different classes. It is obvious that CNN is a special case of GCNN for $\varrho=0$. The algorithm randomly selects a prototype from each class in order to initialize *S*, and then the process is similar to CNN as all samples are checked individually with the misclassified ones being moved to a new set. From this set a new prototype for each class is selected and added to *S*. The process, as explained, is iterative and terminates when all prototypes of the original set are correctly classified. It should be mentioned that the selection of the prototype to be moved in *S* can be done either randomly [Cho06], or according to a specific rule (such as centroids or mean of samples) [Sus02].

A density-based reduction algorithm for identifying and removing outliers was proposed in [Cao08]. This method defines a score for every instance in the training set, the density-similarity-neighbor based outlier factor (DSNOF), which is a measure of how much an instance is considered an outlier. According to this value computed, harmful instances are recognized and discarded. During the first step of the algorithm the *k*-distance[2] neighbours of every instance $x_i \in X$ and its density are determined.

$$D(x_i) = \frac{\left|N_k(x_i)\right|}{kd(x_i)} \tag{2.6}$$

Where *D* is the density, *kd* is the *k*-distance of x_i and $\left|N_k(x_i)\right|$ is the size of the *k* distance neighbourhood. DSNOF constructs the similar density series (SDS) of x_i in a matrix that consists of its similarity neighbours in descending order. In order to compute the DSNOF of an instance x_i, the average series cost of x_i has to be calculated:

---

[2] K-distance of a point $x \in X$ is the distance *d(x,o)* between x and an instance in *X* such that:

1) For at least k points $o' \in X \setminus \{x\}$ it holds that $d(x,o') \leq d(x,o)$
2) For at most k-1 instances $o' \in X \setminus \{x\}$ it holds that $d(x,o') < d(x,o)$

$$ASC(x_i) = \sum_{k=1}^{n} \frac{d(ao_k)}{k} \tag{2.7}$$

where $d(ao_k)$ is the distance between two adjacent instances in SDS. The final step is the calculation of DSNOF, which is expressed as:

$$DSNOF(x_i) = \frac{|N_k(x_i)|ASC(x_i)}{\sum_{o \in N_k(x_i)} ASC(o)} \tag{2.8}$$

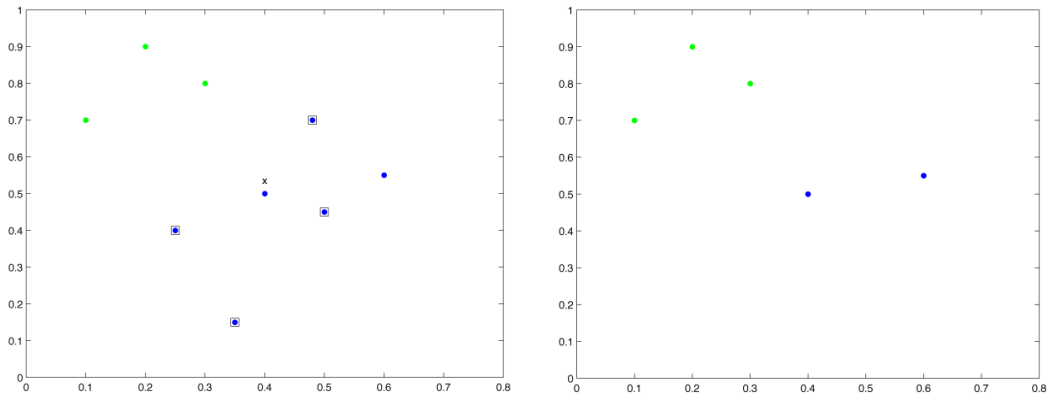
DSNOF is the probability of a data point being an outlier, hence, instances with high such values, which exceed a certain threshold set by the user, are considered outliers and can finally be removed.

2.3 Neighbourhood based Approaches

In the nearest neighbour approach only the distance between instances is taken into consideration to define the relative neighbourhood. On the contrary, two new methods introduced in [Cha96], the Nearest Centroid Neighbourhood (NCN) and Nearest Mean Neighbourhood (NMN), use not only closeness but also symmetry to define the neighbourhood of a prototype. During the first step of the process the nearest neighbour of x_i is determined, x_j, and the centroids M_{jk} of x_j and every other instance x_k of the remaining training set $X - \{x_i, x_j\}$ are computed. The instance x_z, which produces the closest centroid to x_i, in terms of Euclidean distance, is the one selected to define the Neighbourhood. In order to resolve ties, the instance that lies the farthest from the neighbour found previously is selected. As a result, the algorithm determines the neighbours that lie symmetrically around an instance, and spread in all directions as illustrated in Figure 2.III(a). In a similar manner to NCN, instead of computing the centroids between instances, NMN chooses the median to determine the neighbourhood of each instance in the training set.

This neighbourhood-based technique can be efficiently employed for data reduction as proposed in [Loz03], where the geometrical distribution of every instance is determined by computing its Nearest Centroid Neighbourhood. The neighbourhood of every prototype includes all instances computed by the NCN algorithm until its nearest enemy is reached. Then for every region of same class instances, a sample $x_i \in X$ is used as the representative prototype. x_i has to be chosen carefully as it should cover the largest area possible, hence, the instance with the largest number of neighbours in its NCN is selected. Condensing then is achieved by removing the rest of the instances in each neighbourhood, as can be observed in Fig. 2.III(b). After a group of samples is discarded, the algorithm proceeds by checking and updating all remaining

neighbourhoods, in case any of the removed instances contribute to more than one neighbourhood. The algorithm terminates when all instances of the entire training set are treated. But, in order to further improve the classification capability of the model and accurately deal with the problem of border instances being removed, some additional steps were introduced. At the end of the process the condensed training set is used to classify the initial training set, and all misclassified instances are added to the final subset. There is also a second variation of the described method, which instead of selecting a representative prototype for each neighbourhood computes its centroid. As a result, the output of this method is a reduced set of newly generated prototypes.



(a) (b)

Figure 2.III- A two class two dimensional example. **(a)** NCN of a sample pattern x defined by the same-class instances indicated by "□". **(b)** Neighbourhood of instance x_i after Lozano's method has been applied.

In [Guo05] a method for prototype reduction using a global neighbourhood to represent all instances that lie within it is introduced. The similarity matrix of the entire training set is computed and each instance has an extra variable that can take two different values, 'grouped' or 'ungrouped'. Initially this is set to ungrouped for all samples. The local neighbourhood of every instance x_i is defined as the maximum number of instances that belong to the same class with an additional error rate ε defined by the user. Based on the local neighbourhoods the global neighbourhood, N_i, can be determined, along with a representative vector R that holds all the required information of the neighbourhood; the class label of the entire region, the similarity of the furthest instance inside the neighbourhood to the central instance x_i, the number of all the covered instances and the representative prototype x_i. The variable of all samples included in N_i is then set to 'grouped' and the algorithm terminates when all samples are assigned to a neighbourhood. The resulting vectors can accurately characterize every different region of the data space and all new instances x_j are classified with respect to their similarity to the representatives R using the following formula:

$$\sigma(x_i, x_j) = HVDM(x_i, x_j) - s(x_i) \quad (2.9)$$

where $s(x_i)$ is the similarity of the representative of N_i and HVDM is the Value Difference Metric described in [Sta86], and x_j is assigned to class of the representative prototype that provides minimum score value σ. So space is partitioned into neighbourhoods and search is performed only within the representatives and not through the entire training X.

A nearest neighbour method that computes local neighbourhoods for classification was introduced in [Dom02]. The adaptive metric nearest neighbour algorithm (ADAMENN) proposed a new distance metric, the Chi-squared distance, which is capable of producing more homogeneous neighbourhoods. ADAMENN is an adaptive algorithm for pattern classification. Similarly, another adaptive metric technique for producing neighbourhoods was presented in [Has96]. In this case linear discriminant analysis was used to determine the suitable metric. In general, neighbourhoods and neighbourhood relations are very important concepts in the field of machine learning. Initially, topological algorithms that deal with neighbourhood spaces were designed to operate merely as classifiers, such as [Owe84] and [Sal91], or distance metrics [Wei09], in order to exceed the performance of other local classifiers as the k-NN rule. However, neighbourhood based techniques are now considered a very powerful tool that is widely used in various applications of machine learning, such as instance reduction, which has already been presented in this section, or for feature selection [Hu08].

2.4 Graph based Approaches

Analysis of the metric space can also be applied to data reduction and particularly instance removal. In [Tou79], Toussaint used Voronoi-based condensing to reduce the original data set. Voronoi decomposition divides the vector space into cells, with each cell containing one instance x_i and all the points that are closer to x_i than to any other instance. As a result, an instance is discarded if all of its neighbouring cells (polygons), the ones that have a common side with the instance, belong to the same class. Therefore, center instances are removed while class boundaries are maintained, which is the reason why Voronoi diagrams are widely used in machine learning. Toussaint also provided a very thorough review of graph methods and prototype reduction in [Tou80], while he also proposed the concept of Relative Neighbourhood Graph that computes the geometrical neighbourhood of every instance. This method has not been used directly for instance selection but has been employed by other graph-based algorithms [Muh03] and [Jar92].

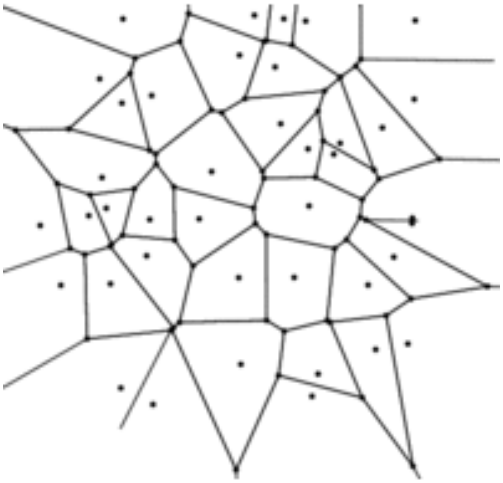


Figure 2.IV- Voronoi polygons [cs.sunysb.edu]

Another method equivalent to Voronoi diagrams is Gabriel graphs, which are formed when two Gabriel neighbours are connected with an edge. Two points x_i and x_j, are Gabriel neighbours when the disk having line segment x_ix_j as its diameter contains no other points. This method is used in the Hybrid Gabriel-Graph algorithm introduced in [Bha05]. In order to compute the Gabriel neighbours of the samples in the training set, a framework called GSASH, Gabriel spatial approximation sample hierarchy, is proposed by Bhattacharya. GSASH is a graph, at which every node corresponds to a data item and nodes that are Gabriel neighbours are connected by an edge. The original training set is edited in the beginning by using Gabriel neighbours, so instances that are misclassified by their Gabriel neighbours are permanently discarded. This is a technique based on Wilson's editing method [Wil72]. After filtering, in order to remove redundant instances from the training set, all samples that their Gabriel neighbours belong to the same class are removed. During the third and final step of the algorithm, the iterative case filtering (ICF) algorithm [Bri02] is used on the reduced training set to obtain the final output subset. So, instead of the *k*-NN method GSASH uses Gabriel neighbours to classify all new samples.

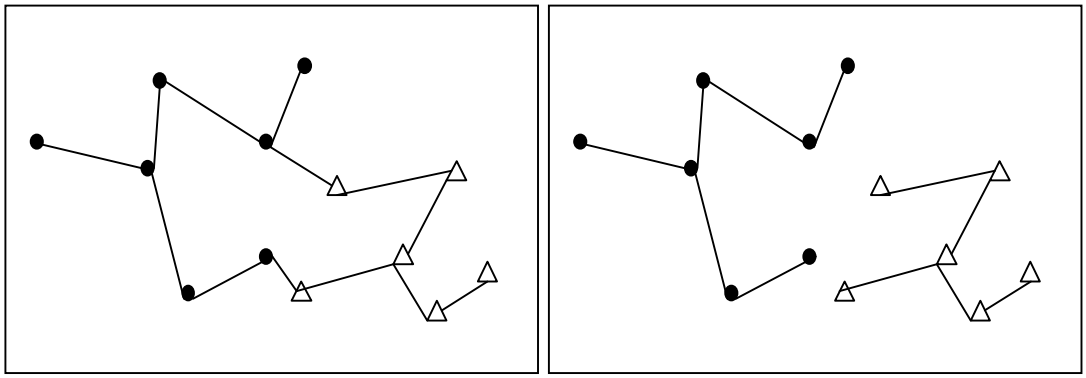
In [Muh03], the geometrical neighbourhood of every instance x_i of X is determined using the Relative Neighbourhood Graph method [Tou80] and instances belonging to different classes are either discarded or relabeled. This editing technique is very efficient for noise and outliers removal, because it uses cutting edge weights to decide whether an edge between two instances should be cut or not (Fig. 2.V). For an instance x_i belonging to a class l_i the matching null hypothesis, which is the probability of an instance in the neighbourhood of x_i not belonging to the same class, is defined as:

$$H_o = 1 - p(l_i) \tag{2.10}$$

where p is the global proportion of class l_i in the entire training set. An absolute weight of cutting edges is computed for every instance x_i and given by:

$$J_i = \sum_{j=1}^{N_i} w_{ij} I_i(j) \tag{2.11}$$

where k is the number of instances in the neighbourhood of x_i, w_{ij} is the weight of the edge between x_i and x_j and I are independent and identically distributed random variables, according to Bernouilli law [Muh03]. Based on the value of J under the null hypothesis samples can be optimized as good, doubtful or bad and thresholds are set in order to distinguish between them. When all instances of the training set are evaluated, bad and doubtful ones are selected and checked individually. From these instances, the ones that have good instances lying within their neighbourhood are optimized according to the majority of their k-nearest neighbours, while samples with no good instances within their k-NN are discarded. This method tries to condense the training set and simultaneously increase class separability.



(a) (b)

Figure 2.V- A two class two dimensional example. **(a)** The Relative Neighbourhood graph is depicted. The edges between the two classes are cut off.

Another graph-based algorithm is Hit Miss Networks (HMN). These 'networks' are directed graphs of instances in the training set [Mar08]. For every sample, the nearest neighbours from all classes are determined and an edge between the sample and its neighbours is defined; a hit edge is an edge between instances of the same class, while a miss edge is defined if instances belong to different classes. As a result, every instance of X has a number of outgoing edges equal to the total number of classes. A hit and a miss degree are computed for every node of the training set, as can be observed in Fig. 2.VI. Based on the hit and miss degrees computed by the HMN algorithm, the following deletion criterion is applied:

$$w_c \left| M(x_i) \right| + \varepsilon > (1 - w_c) \left| H(x_i) \right| \tag{2.12}$$

where w is the weight of each class l, H and M are the hit and miss degrees respectively, and ε is the error coefficient (ε<1).

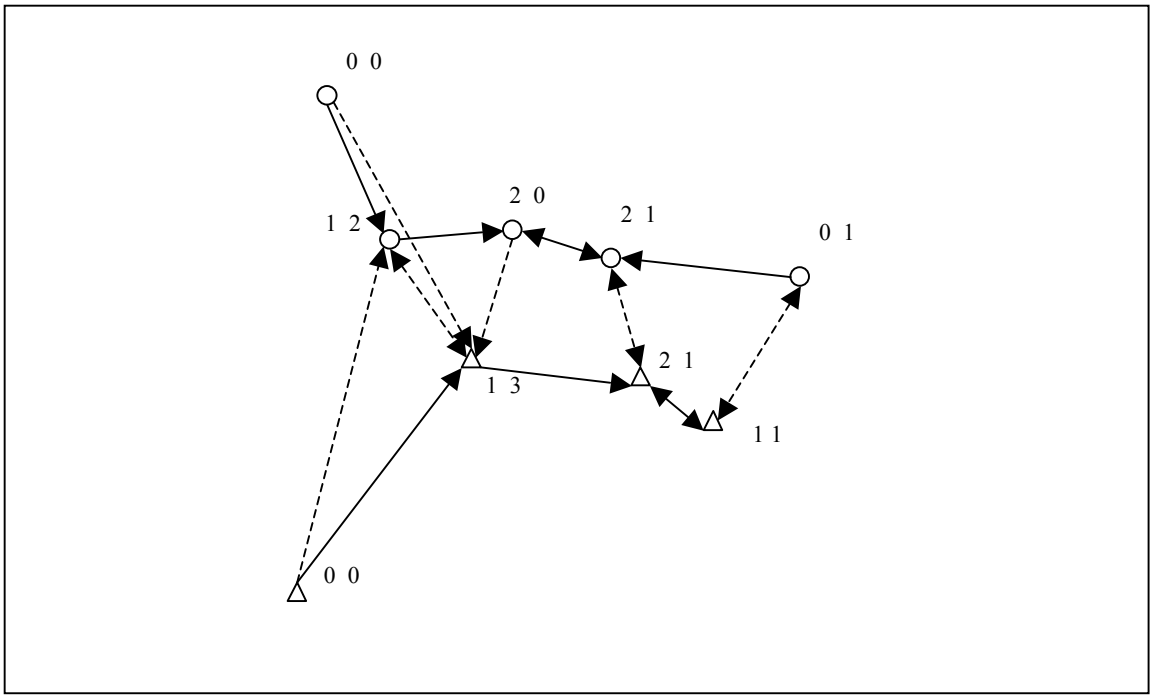


Figure 2.VI- Hit Miss Network of a two class two dimensional data set. The two numbers next to each sample pattern indicate its respective hit (solid lines) and miss (dash lines) degrees.

This rule tends to discard isolated instances with zero hit degree, along with noisy instances. HMN can lead to the removal of a very large number of instances that may cause a drop in the classification accuracy of the algorithm. Thus, to avoid large losses of important information, three different heuristics are employed. Firstly, a threshold is set for the number of vectors belonging to each class. If this number falls below the threshold set, all discarded instances of the specific class with hit degrees greater than zero are restored. A second heuristic involves multi-class training sets (more than 3 classes). All instances that have non-zero hit degree along with a small valued miss degree are restored. The threshold proposed in [Mar08] is $|L|/2$. Finally, a threshold is set for every class and instances with a greater number of hit edges than the 25% of their own class are retained. These instances are likely very close to the class centers contributing to cluster identification.

Based on HMN two other algorithms, Class Conditional Instance Selection (CCIS) [Mar10] and Laplace Instance Filtering (LIF) [Mar09] have been introduced. The first method is an enhanced version of HMN that makes use of two directed graphs, the within-class graph $G_{wc} = (V, E_{wc})$, which is for same class instances and the edge matrix for a vertex *v* is defined as:

$$E_{wc} = \{u \in X : \psi(u) = \psi(v), u \in 1NN(v)\} \tag{2.13}$$

and for enemy instances the between-class graph $G_{bc} = (V, E_{bc})$ that is defined in a similar manner with an edge matrix:

$$E_{bc} = \left\{ u \in X : \psi(u) \neq \psi(v), u \in 1NE(v) \right\} \tag{2.14}$$

where *NE(.)* defines the nearest enemy of *v*. Using the Kullback-Leibler divergence [Lin91],

$$K(p_1, p_2)(v) = p_1(v) \log \frac{p_1(v)}{1/2\left(p_1(v) + p_2(v)\right)} \tag{2.15}$$

where $p_1(.)$ and $p_2(.)$ are two discrete probability distributions over *X*. Denoting by $p_w(.)$ and $p_b(.)$ the within and between-class degrees of a vertex divided by the total in degree of G_{wc} and G_{bc} respectively, the final class conditional score is given by:

$$CCS(v) = K(p_w, p_b)(v) - K(p_b, p_w)(v) \tag{2.16}$$

So, CCIS displays negative scores for instances that contribute more to the between-class divergence and retains the ones with higher scores. This algorithm demonstrates considerable improvement in the condensation capability compared to its predecessor HMN.

On the other hand, the latter algorithm, LIF, is of significant importance as it is the first method to introduce Laplacian graphs to instance selection. So, the Laplace score is defined as the discrete Laplace operator for the between-class graph acting on the degree function of the within-class graph. For a vertex *u* in the between-class graph connected to the set of vertices *v*, the Laplace score is defined as

$$L(g)(u) = \frac{1}{\sqrt{d(u)}} \sum_{u \sim v} \left(\frac{g(u)}{\sqrt{d(u)}} - \frac{g(v)}{\sqrt{d(v)}} \right) \tag{2.17}$$

where $d(u)$ is the degree function in the between-class graph and $g(\cdot)$ is the degree of the given vertex in the within-class graph. The above equation can be rewritten in matrix notation as

$$L(g) = \mathbf{L}_{BC} \mathbf{W}_{WC} \mathbf{1}_{n \times 1} \tag{2.18}$$

where $\mathbf{L}_{BC}$ is the normalized Laplacian matrix for the between-class graph, $\mathbf{W}_{WC}$ is affinity matrix for the within-class graph, and $\mathbf{1}_{n \times 1}$ is a column vector with all its elements equal to one. The normalization of the Laplacian matrix is performed so that the rows of the affinity matrix sum to one. Therefore,

$$\mathbf{L}_{BC} = \mathbf{I} - \mathbf{D}_{BC}^{-1/2} \mathbf{W}_{BC} \mathbf{D}_{BC}^{-1/2} = \begin{cases} 1 & \text{if } u = v \\ \dfrac{-1}{\sqrt{d(u)d(v)}} & \text{if } u \sim v \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

Based on the computed score, similar to HMN, a deletion criterion is employed to make a decision for every instance, with samples displaying negative values being removed from the training set. The basic application of the LIF algorithm is to identify and remove outliers since it operates as a noise filter.

A *k*-nearest neighbour model that uses a weighted sum of the influence of different classes on instances is proposed in [Hua07]. This method determines the neighbourhoods N that accurately represent the entire training set $X = \{x_1,...x_i,...x_n\} \in R^d$. After normalising all the input samples, a threshold α and a density control value p are set, as well as a '0' tag that is assigned to every instance. This threshold represents the maximum acceptable value of noise allowed in a neighbourhood, as can be seen in Figure 2.VII. The next step of the algorithm is the computation of the density of every class l_i:

$$p_i = \frac{m_i}{2\sqrt{n}} \tag{2.20}$$

where m_i is the number of instances belonging to the specific class. Choosing an instance x_i with a '0' tag, the algorithm performs a search of its nearest neighbour that does not satisfy the following condition:

$$\frac{k(l_i)}{r} \geq \frac{p_i}{p} \tag{2.21}$$

where $k(l_i)$ is the number of instances belonging to class of x_i, r is the radius from point x_i and p is the density of the region determined. The neighbourhoods N that satisfy condition (2.22) are chosen as representative candidates, otherwise the tag of x_i becomes '-1'.

$$\frac{\sum_{x_j \notin c_i \wedge x_j \in N_i \wedge x_j \in X} \left(1 - \frac{d(x_i, x_j)}{r}\right)}{\sum_{j=1, j \neq i}^{n} \left(1 - \frac{d(x_i, x_j)}{r}\right)} \tag{2.22}$$

The candidates found are then scanned to determine the one with the largest number of instances that is set as the class representative[3], while all instances covered by it are tagged as '1'. The algorithm terminates when all instances are treated, while new samples are classified with respect to the computed representatives instead of the original set of instances.

[3] In case of ties between candidates the one with the smaller radius r is chosen.

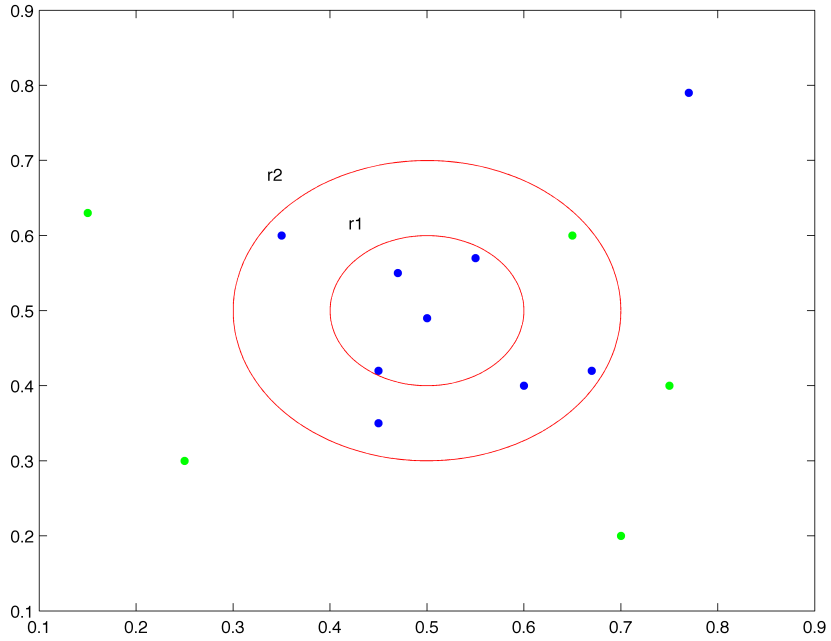


Figure 2.VII- The different radii define different error zones. r_1 indicates a zero error allowance. r_2 defines an area where a certain amount of noise is permitted.

2.5 Instance Weight Learning

Instance weight learning (IWL) is a form of lazy learning, which has been used not only for regression, dimensionality reduction and various other applications, but also for noise and outlier removal, as well as for prototype reduction [Atk97]. In IWL different weights w with $w_i \in R^d$, are assigned to every instance x_i of the original set $X = \{x_1,...x_i,...x_n\} \in R^d$, and as explained in [Kan08] these weights show if and how important an instance is. Although the majority of these methods employ IWL on the distance metric in order to improve the classification error of the *k*-NN algorithm like [Fer07], [Par06$_a$], there exist some cases where IWL has been used directly for prototype reduction. An example of such a method is the adaptive distance metric proposed in [Ric99], where misclassified prototypes are moved towards the right class. An asymmetric weight is assigned to each prototype of the subset and nearest neighbour classification is applied using a local asymmetrically weighted similarity metric (LASM):

$$\delta(x,y) = \sqrt{\left(\sum_{i=1}^{d} w_i(x,y)\|x_i - y\|^2\right)} \tag{2.23}$$

A subset of the original set is selected and two functions, reinforcement R and punishment P, are defined. Weights are assigned and can be defined as:

$$w_i(x,y) = \begin{cases} w_i^0(x) & \text{if } y \le x_i \\ w_i^1(x) & \text{otherwise} \end{cases} \tag{2.24}$$

where w is the weight, x_i is the prototype and y is the query. Weights are initialized in the beginning and a test sample y is selected. Then the nearest neighbour prototype x_i of sample y is determined and if both belong to the same class *R(.)* is used, otherwise *P(.)* is applied. The reinforcement and punishment steps are expressed as:

$$R_i^0\left(w(x),x,y\right)=\begin{cases} w_i^0(x)-aw_i^0(x)\left|x_i-y\right| & if\ y<x_i \\ w_i^0(x) & \text{if } y\geq x_i \end{cases} \tag{2.25}$$

$$R_i^1\left(w(x),x,y\right)=\begin{cases} w_i^1(x)-aw_i^1(x)\left|x_i-y\right| & \text{if } y\geq x_i \\ w_i^1(x) & \text{if } y<x_i \end{cases} \tag{2.26}$$

$$P_i^0\left(w(x),x,y\right)=\begin{cases} w_i^0(x)+\dfrac{\beta}{2}\left(1-\left|2w_i^0(x)-1\right|\right)\left|x_i-y\right| & \text{if } y<x_i \\ w_i^0(x) & \text{if } y\geq x_i \end{cases} \tag{2.27}$$

$$P_i^1\left(w(x),x,y\right)=\begin{cases} w_i^1(x)+\dfrac{\beta}{2}\left(1-\left|2w_i^1(x)-1\right|\right)\left|x_i-y\right| & \text{if } y\geq x_i \\ w_i^1(x) & \text{if } y<x_i \end{cases} \tag{2.28}$$

where $\alpha\in[0,1]\ and\ \beta\in[0,1]$ are the reinforcement and punishment rates respectively. The algorithm terminates when all training samples are classified correctly.

On the other hand, condensation methods that employ IWL use various heuristics to compute the different weights. As a result, the processes each algorithm employs in order to determine which instances should be discarded and which retained vary largely. Paredes and Vidal in [Par00$_b$] proposed such a reduction technique that assigns weights to prototypes and finally discards the ones displaying the highest weight values. Each prototype x_i is assigned a weight w_I and the weighted prototype dissimilarity is defined as:

$$\left\|y-x_i\right\|_{wp}=w_i\left\|y-x_i\right\| \quad where\ \ w_i\in[0,\infty] \tag{2.29}$$

In the next step f and e, which are the same and different class nearest prototypes of x_i, are determined and then the ratio $\dfrac{\left\|x_i-f\right\|_{wp}}{\left\|x_i-e\right\|_{wp}}$ is minimized, so that small weights are assigned to prototypes that are close to the acceptance region of their own class. Minimization is provided using the following update equations:

$$w_f=w_f-\frac{\mu\left\|x-f\right\|_{wp}}{w_e\left\|x-e\right\|_{wp}} \tag{2.30}$$

$$w_e=w_e-\frac{\mu w_{x_i}\left\|x-f\right\|_{wp}}{w_e^2\left\|x-e\right\|_{wp}} \tag{2.31}$$

where μ is a user defined learning factor. The same process is repeated for all prototypes of the training set, and editing occurs, when all prototypes are assigned a weight, by discarding the ones that have a weight over a certain threshold. This algorithm is an enhanced version of the original concept, which was initially analysed in [Par00$_a$], and

exhibits a great improvement in terms of the performance of the method. To conclude, the final version of the specific condensing algorithm is a method that combines the concepts of IWL for error minimization as proposed in [Par06$_a$] and IWL for prototype reduction [Par00$_b$], which was introduced in [Par06$_b$].

A recently proposed method for supervised instance-weight learning [Deh07] uses a similarity metric $\mu(X)$ to obtain the optimal weights for every instance x_i. This method achieves condensing of the original training set by optimizing the weights of misclassified instances, which are discarded at the end of the process. The similarity metric between x_i and instance x_j is defined as:

$$\mu\left(x_j, x_i\right) = \frac{1 - \left\|x_j - x_i\right\|}{\left\|z - y\right\|} \tag{2.32}$$

where z and y are the two instances that provide the maximum possible distance that can occur in the training set. The weights w are incorporated indirectly in the similarity metric in the search of the nearest neighbour of every instance x_i.

$$w = \max\left\{\mu\left(x_j, x_i\right) w_i \mid i = 1,...,n\right\} \tag{2.33}$$

In the initial state, all samples are retained, hence, all weights are set to 1, and an instance x_i belonging to a class l_i is randomly selected from X for removal by setting $w_i = 0$. Then, all same class instances that are correctly classified along with enemy instances that are misclassified are also removed. All these samples remain unaffected by the change in the weight of x_i. For every remaining instance in the reduced set a score is computed using function (2.21) and compared to a threshold τ, such that $x_j \in l_i \; iff \; Sc(x_j) < \tau$.

$$Sc(x_j) = \frac{\max_j \left\{\mu(x_j, x_k) w_k \middle| k \neq j\right\}}{\mu(x_j, x_i)} \tag{2.34}$$

The purpose of this method is to optimize the classification accuracy by optimizing the threshold. So the threshold that leads to the minimum misclassifications is determined and chosen to train the algorithm. Ultimately, samples that do not contribute to the improvement of classification accuracy are discarded since a weight of zero is assigned to them.

A locally adaptive nearest neighbour technique for classification was presented in [Dom05]. Although this method is not a direct condensation method since no deletion of instances occurs, it uses a weighting scheme to define a neighbourhood around a new query to speed up the classification process.

2.6 Nearest Enemy based Techniques

A prototype reduction technique called Minimal Consistent Set, MCS, which selects a smaller subset of the original instances in the training set X, was introduced in [Das94]. Its basic concept is that for an instance x_i to be correctly classified, an instance x_j of the same class should be closer to x_i than its nearest unlike neighbour. For this condition to be true all samples that lie within a radius of influence defined by the vector and its nearest enemy are determined and receive a vote. So an instance x_i of the training set gets a vote from every sample it can correctly classify. MCS is a subtractive algorithm as all instances, after the process of voting is complete, are initially included in the output set. Therefore, samples are removed in an iterative way by discarding the voters of the most voted instance. Votes are then updated according to the remaining samples, and the algorithm terminates when all instances have been processed and no further removals can occur.

The MCS algorithm is one of the first nearest enemy based techniques in literature; hence, it presented a number of drawbacks, as noisy instances and outliers could get few or no votes at all and, as a result, make it into the reduced set. In order to avoid this effect that could lead to misclassifications and low accuracy, an extension was proposed in [Zha08]. The proposed method is very similar to MCS as once more for every instance x_i its neighbours within the distance r are determined and voted for. But, in this case the algorithm is additive since the output set is initially empty. Again the most voted instance is selected during each iteration and added to the reduced set, while the votes are updated. The concept behind this procedure is that the instance that guarantees the most correct classifications is retained. The other modification of this method is the existence of an error threshold, which when exceeded the algorithm terminates. Allowing for an error in the training set means that not all instances will be accounted for in the resulting subset, hence, noisy instances with few votes will not be considered at the later stages of the algorithm. As a result the classification accuracy displayed by MCS is improved as this algorithm deals not only with redundant instances but also with harmful ones.

Wilson and Martinez presented a series of subtractive algorithms called Decremental Reduction Optimization Procedure (DROP 1-5) and Decremental Encoding Length (DEL) [Wil00]. DROP 1 is the basic reduction model, while DROP 2-5 and DEL are expansions that enhance the performance of the algorithm via noise filters and other extensions. These condensation procedures make use of the concept of *associate instances* of a sample x_i, which are instances that have x_i as one of their k nearest neighbours. Then, the deletion of a sample depends on the effect it will have on the classification of its associates. More specifically, DROP 1 is a subtractive algorithm that

discards an instance x_i if the majority of its associates are classified correctly in the absence of x_i. In order to tackle some deficiencies of this method, DROP 2 was introduced that sorts instances in descending order of distance from their nearest enemy. This way samples near the class boundaries, which account for the most important information of the underlying distribution, are processed last; hence, retained by the algorithm. But, the most efficient of the algorithms is DROP 3, which best addresses the problem of noisy instances. Compared to DROP 2 a filter is added as a preprocessing step to remove samples that are misclassified by their k nearest neighbours. The next method, DROP 4, extends the filtering process, so that instances are removed only if they are not classified correctly by their k nearest neighbours and their removal does not affect classification accuracy. This method is proposed in order to avoid the removal of a very large number of instances that may occur by DROP 3, but as a drawback it displays increased computational complexity. DROP 5 is another extension of DROP 2, where instances are processed in ascending order of nearest enemy distance, in order to obtain better filtering of noisy instances. Finally, DEL is proposed, which is a modification of DROP 4. In this method the filtering criterion is altered and an instance is considered noisy and discarded if it is misclassified by its k nearest neighbours and its deletion does not lead to an increase of the encoding length cost.

A case-based algorithm that uses a deletion criterion to reduce the size of the original set of samples is introduced in [Smy95]. For every instance x_i, the Reachable set, which contains all same class instances that lie within a hypersphere centred at x_i with a radius equal to the distance between x_i and its nearest enemy sample, and the Coverage set are computed. The coverage set contains all instances that include x_i in their reachable set. According to these sets all instances are assigned to one of the four following categories. Pivotal cases, which are instances that can be solved only by themselves; hence, their reachable set size is one. Pivotal instances are basically outliers. Auxiliary cases that are surplus samples and do not affect the consistency of the algorithm. Their coverage set is a subset of the coverage set of another instance. There are also spanning instances, which are samples that their presence is not immediately effective. These instances may become essential to the competence of the method if some other samples are removed. And lastly, support cases that come in groups and solve each other. The deletion of such instances affects the ability of the algorithm only when the whole group is removed. So, the Footprint Deletion algorithm organizes instances according to the effect their deletion will have on the stability of the algorithm and auxiliary cases, which are considered redundant, are treated first; the support cases follow up, spanning cases and then finally pivotal instances. The problem with this method is that it does not take into consideration the performance of the algorithm as it focuses only on competence.

Another enemy based technique that uses lazy learning to train the algorithm is the Iterative Case Filtering algorithm (ICF), which was initially introduced in [Bri99] and was optimized in [Bri02]. The main scope of ICF is to discard instances close to class centers and retain the ones that surround the class boundaries. Similar to DROP 3, ICF uses Wilson's editing algorithm [Wil72] as a preprocessing step to handle noisy samples. Based on the concepts of coverage and reachability that already have been described, ICF removes an instance if it can be described by more samples than it can classify correctly. This means that instances that display a larger reachable set than the size of the coverage set are discarded. The result of this algorithm is a decline in computational complexity, since search is performed in a very small set of training instances compared to the entire training set T.

A very recent reduction algorithm that uses the concept of nearest enemy is described in [Fay09]. The Template Reduction for *k*-NN (TRKNN) is an iterative procedure for removing redundant samples. Using the one-against-all concept the algorithm starts from internal samples and constructs chains of nearest enemies to determine instances close to class boundaries. The chain is stopped when the successive distances converge and the change is less than an error-bounded threshold.

2.7 Density based Approaches

Astrahan [Ast70] was one of the first to use a prototype reduction algorithm that used density estimation of a point for clustering. A disc of radius *r* is used in order to estimate the density around an instance and using these estimations the sample with the highest density is selected. Another disc of different radius centered at the densest sample is then computed and all instances that lie within it are discarded.

Based on Astrahan's method another algorithm using density estimation for efficient data reduction was proposed in [Mit02]. Density estimation is performed for every instance x_i of the training set $X = \{x_1, x_2 \ldots x_i \ldots x_n\} \in R^d$:

$$\hat{f}_N(x_i) = \frac{k}{n} \times \frac{1}{A_r} \tag{2.35}$$

where A_r is the volume of the hypersphere around x_i, n the size of X and k is the number of nearest neighbours within the hypersphere, which is a user defined variable. This estimation is then used for data reduction, as for every sample, its distance $\|x_k - x_i\|$ to its k^{th} nearest neighbour is computed and the instance that displays the lowest distance is going to be the one with the largest density f_N (Fig. 2.VIII). The selected prototype x_i is then inserted in the new reduced subset, which is initially empty, while all samples in X

that lie within a radius of $2\|x_k - x_i\|$ from x_i are permanently removed. The proposed algorithm is an iterative procedure and the described process is repeated until all samples in X are processed. It should be noted that the final output of the algorithm is a subset of the original training set, where the denser regions are represented by a larger number of points, since the radius r that defines the hypersphere is inversely proportional to the computed density f_N.

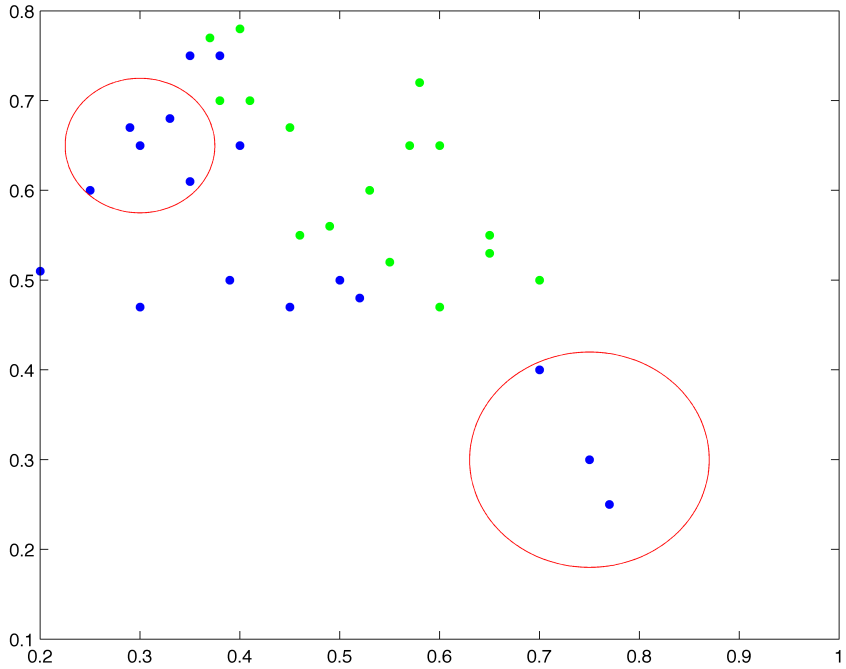


Figure 2.VIII- A smaller radius is defined by high density instances compared to samples laying in more sparse regions. All instances laying within the illustrated discs are discarded.

Another selection technique for removing superfluous samples from the original training set was proposed in [Wu02]. The basic concept of this algorithm is to remove samples from dense regions of same class instances in the feature space. Thinning of such regions does not affect the competence of the method as long as the number of remaining prototypes is larger than k, the number of instances used by the k-NN classifier. In order to resolve between relevant and redundant samples, a deletion score for every instance is computed, namely its attractive capacity, with high values of this score indicating instances that will be discarded.

A successful density-based method for data condensation using weighting of instances is introduced in [Gir03]. Having an initial training set $X \in R^d$, Parzen windows is used to obtain an estimate of the density of every instance Equation (2.36) and the maximum likelihood estimator criterion (MLE) is used to determine the weighting coefficients according to Equation (2.37).

$$\hat{p}(x;h,\gamma) = \sum_{n=1}^{N} \gamma_n K_h(x,x_n) \tag{2.36}$$

where h is the width of the kernel and γ a weighting coefficient.

$$\hat{\gamma}_{MLE} = \arg\max_{\gamma} \frac{1}{N} \sum_{m=1}^{N} \log \sum_{n=1}^{N} \gamma_n K_h(x_m, x_n) \tag{2.37}$$

Subject to two constraints $\sum_n \gamma_n = 1 \; and \; \gamma_n \geq 0 \forall n$. The Integrated Squared Error, which for a density estimate with a parameter θ is expressed by Equation (2.38), is then minimized to achieve data reduction.

$$\hat{\theta} = \arg\min \int \hat{p}^2(x;\theta)dx - 2E_{p(x)}\{\hat{p}(x;\theta)\} \tag{2.38}$$

where $E_{p(x)}\{\hat{p}(x)\}$ is the expectation of $\hat{p}(x)$ with respect to $p(x)$. Equation (2.31) can be further investigated:

$$E_{p(x)}\{\hat{p}(x;\theta)\} \approx \sum_{i=1}^{N} \gamma_i \hat{p}_h(x_i) \tag{2.39}$$

$$\int \hat{p}^2(x;\theta)dx = \sum_{i,j=1}^{N} \gamma_i \gamma_j \int K_h(x,x_i)K_h(x,x_j)dx = \sum_{i,j=1}^{N} \gamma_i \gamma_j C(x_i,x_j) \tag{2.40}$$

Combining (2.38), (2.39), (2.40) and applying a Gaussian window $G_h(x,x_i)$ the argument that has to be optimized for estimating ISE becomes:

$$\arg\min_{\gamma} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \gamma_i \gamma_j G_{2h}(x_i,x_j) - \sum_{i=1}^{N} \gamma_i \hat{p}_h(x_i) \tag{2.41}$$

As a result, the proposed algorithm, using smooth density estimation, retains only instances of high density regions as the weights of low density samples are driven to 0 and are discarded from the training set.

As already mentioned reduction algorithms are widely used in machine learning to improve the performance of classifiers and tackle problems such as storage, noise and time requirements that make the operation of classifiers inefficient. In [Ber00] a new adaptive *k*-NN classifier was introduced that classifies samples based on the local kernel estimation computed by the *k*-nearest neighbours of every unseen pattern. In order to simplify the complexity of this method, and enable its use on real applications, a reduction technique is applied as an adaptive mixture model is used instead of the local Parzen window estimator. As a result, the computation requirements of the model are highly reduced.

In [Hua06] another density based algorithm with two variations was proposed that uses entropy to determine appropriate representative prototypes from the original set of instances in *X*. Initially, the algorithm randomly selects *k* representative samples, where *k* is the desired number of prototypes in the output subset, and computes the representative entropy or the weighted representative entropy of every instance x_i in *X*. The two methods are called Representative Entropy Data Reduction (REDR) and

weighted representative Data Reduction (WREDR) respectively. At every iteration a search process selects the sample that displays the minimum value of RE/WRE in order to replace the worst representative prototype (the one with the maximum RE/WRE value). The closest samples to the new prototype are then removed from the training set to avoid overlapping representatives, and the method terminates when no change in selected prototypes occurs. The proposed algorithm identifies prototypes that are uniformly scattered in space in order to avoid the problem of overlapping, while simultaneously describing the entire distribution as accurately as possible.

2.8 Miscellaneous Approaches

In [Pek06] various existing instance reduction algorithms, such as ModeSeek, which is a selection algorithm and *k*-Centers that is an abstraction method, are discussed and compared. The *k*-Centers technique is based on local neighbourhoods and selects a subset of k instances per class that are retained. Initially k representatives for every class are randomly selected, and inserted in a new set of representatives. For every instance x_i of the training set, its nearest neighbour x_j in the representative set is determined, and x_i is added to a set A_j of instances x_j can correctly solve. Finally, the center of every set A_j is used as the representative. But, the novelty of the described algorithm is the dissimilarity metric used, which ensures that these instances are going to be evenly scattered in the prototype space. But the novelty of the reduction technique proposed lies in the fact that all prototypes are initially projected to a dissimilarity space and using the new representation condensation and classification of instances is performed. The designing of a dissimilarity space can be very complicated as it is equivalent to designing new features for the training set X. But as explained also in [Nik12], if the measure is efficient and well designed it can largely improve the discriminatory capabilities of the algorithm. For example in the new representation space weights can be optimized based on the importance of every sample with surplus instances having small valued weights. In overall, methods that project instances in a (dis)similarity space prior to any reduction process can achieve better generalization. In [Pek06] it is proven that a different representation on a new dissimilarity space can successfully capture information that gives a more accurate description of the underlying distribution. Therefore, Bayesian classifiers perform better. Using a representation set Z, for a simple two-class problem, the linear decision function is defined as:

$$f\left(D(x,Z)\right) = \left[D(x,Z) - \frac{1}{2}\left(m_1 + m_2\right)\right]^T \times C^{-1}\left(m_1 - m_2\right) + \log\frac{p_1}{p_2} \qquad (2.42)$$

where $D(.,Z)$ is the dissimilarity mapping, m_1 and m_2 are the mean vectors of the respective prototypes, p_1 and p_2 are the class prior probabilities and C the sample covariance matrix of the dissimilarity space. Similarly, the quadratic decision surface is given by

$$f\left(D(x,Z)\right) = \sum_{i=1}^{2}(-1)^i\left(D(x,Z)-m_i\right)^T \times C_i^{-1}\left(D(x,Z)-m_i\right) + 2\log\frac{p_1}{p_2} + \log\frac{|C_1|}{|C_2|} \tag{2.43}$$

where C_1 and C_2 are the estimated class covariance matrices in the representation space of the two instances respectively. The basic concept of creating a representation space using dissimilarity functions was a promising direction on instance selection, thus, since its proposal by Pekalska in 2006, it has been used for learning in various other techniques such as [Dui08] and [Pek08].

Another such method is introduced in [Rie09] where a graph representation is designed and different selection techniques, already introduced in literature, are employed to determine the final reduced set of instances. So, the proposed algorithm uses the reduction methods as a tool on the graph embedding. Initially, a graph $G = (V,E)$ is defined, where V is the finite set of nodes and E is the set of edges. The distance metric used in order to construct the dissimilarity graph is chosen to be the graph edit distance, which represents the minimum cost needed to transform one graph into another. As explained in [Zen09] graph edit distance is a widely used similarity metric, employed in various applications such as computer vision and pattern analysis, despite its large computational complexity. So the graph edit distance is given by

$$d(g_1,g_2) = \min_{(e_1,\ldots,e_k)\in Y(g_1,g_2)} \sum_{i=1}^{k} J(e_i) \tag{2.44}$$

where g_1 and g_2 are the source and target graphs respectively, J the edit cost function, e is the edit operation and $Y(.)$ represents the group of edit paths required to transform the source graph g_1 into the target graph g_2. Using the above equation the graph embedding proposed by this method defines the following mapping:

$$\varphi_n^P(g) \mapsto \left(d(g,p_1),\ldots,d(g,p_n)\right) \tag{2.45}$$

where $d(.)$ is the dissimilarity measure used, which is the edit graph distance in the particular case, between the selected graph g and p_j that is the *j-th* prototype. The above embedding is of exponential complexity due to the computation of the graph edit distance. Then the algorithm proceeds with the selection of prototypes employing different reduction algorithms. Using the dissimilarities between instances as features, the first attempt to determine a smaller subset makes use of the CNN algorithm introduced in [Har68]. After the condensing technique, modified condensing proposed in [Sus02] is employed. Other methods used for reduction of instances involve the reduced and

selective nearest neighbours, RNN [Gat72] and SNN [Rit75] respectively, that have already described, and Chang's merging algorithm [Cha74]. Finally, an editing approach is also tested [Dev80], that is a technique based on ENN and focuses on removing outliers that are misclassified by the 3-NN classifier. As a conclusion, the major contribution of the specific method is not the reduction process but rather the representation of instances as graphs with each one being defined as a set of dissimilarities.

Despite the continuous advances in the field of instance-based learning, the application of evolutionary algorithms to instance selection has only recently been discussed. In general, evolutionary methods use chromosome modeling, where each chromosome represents one plausible solution to the problem of instance selection, to perform a genetic search for the best possible subset of prototypes. The major drawback of evolutionary computation for instance reduction is the increased complexity and the high computational requirements of large datasets. In [Can05, Can06] a model combining the genetic algorithm CHC with a stratification strategy to tackle this problem is introduced. Initially, *n* individuals are generated from a parent population. Then, these are randomly paired to create offspring. By selecting the best chromosomes between the parents and the offspring for further reproduction, the population gradually converges to the better optimum.

Another evolutionary method for instance selection has been presented in [Ped08] and expanded in [Gar09], where a recursive divide and conquer technique is used to reduce the computational complexity of the algorithm. The proposed method divides the training set in subsets and the instance selection algorithm is applied on each subset independently. The retained prototypes are then re-joined and the partitioning procedure along with the application of the selection algorithm is repeated.

2.9 Conclusion

In this chapter, an extensive survey of instance selection algorithms has been provided, along with their main characteristics. Experimental analysis on selection techniques has shown that no ideal method exists, but which algorithm is used depends on the focus of the application at hand. Instance selection algorithms display high speeds and very competitive accuracies because they preserve the geometric and discriminative characteristics of the original instances. On the other hand, the fact that these methods are only allowed to select instances from the original ones, leads to relatively low condensation. To conclude, the user has to understand the main advantages and disadvantages of every algorithm in order to determine which one to choose.

CHAPTER 3:

INSTANCE ABSTRACTION

In this chapter an overview of the abstraction algorithms that exist in the literature is provided. The structure of this chapter is as follows. Section 3.1 presents the process of prototype generation, which substitutes the original training set with a set of newly generated instances. Section 3.2 thoroughly describes various prototype-condensing methods that merge instances based on some fusion criterion. Section 3.3 is an extensive analysis of methods that use the Learning Vector Quantization (LVQ) algorithm for training purposes. Finally, section 3.4 is a brief description of the concept of clustering, while existing methods employing it for prototype reduction are presented.

3.1 Introduction

As explained by Bezdek and Kuncheva in [Bez98$_a$], prototype reduction algorithms can be categorized according to the type of reduction into instance selection and prototype abstraction methods. In contrast to selection algorithms that choose a smaller subset of relevant prototypes from the original data, abstraction methods generate a new set of vectors, which do not coincide with any of the original instances, to replace the entire training set. Generating new prototypes can largely reduce the size of the initial training set; and, in some cases, their performance can even surpass the performance of selection algorithms, since prototypes can be fitted in the data space so as to address the needs of the underlying distribution. For example, prototypes can be generated near the decision surface to improve the separability between classes and clearly define class borders. Hence, abstraction techniques can effectively deal with the problem of data sparsity. It is also possible to use a combination of abstraction and selection methods in order to achieve adequate and satisfactory results, as performed by the algorithm suggested in chapter 4.

With the constant increase in information, prototype reduction has become an essential preprocessing step of nearest neighbour algorithms, and instance abstraction has developed in a very promising technique. Depending on their different properties, such as generation mechanisms or the type of the resulting prototypes, a distinction between them can be established. The following three categories are defined.

- *Rule-based prototype merging*. This group includes all algorithms that use a set of rules to merge instances of the original training set and create a set of new prototypes. Instances are selected, evaluated and if allowed, meaning if the merging criterion of the algorithm is met, are replaced by a new prototype that is defined according to some rule of the algorithm.
- *Learning vector quantization methods*. It is a competitive learning technique that is widely used in prototype generation. Prototypes are moved in the data space according to some rewarding or punishment rules, in order to fit the needs of the underlying distribution.
- *Clustering algorithms*. It involves all unsupervised algorithms that separate samples in different groups of instances that share some certain characteristic. Each group, or cluster, is then considered a different prototype. Although this is a field of machine learning that is differentiated from instance reduction, there is a close relation between clustering and prototype abstraction methods.

3.2 Rule-based Prototype Merging

In [Cha74], Chang presented a method for supervised learning that generated a new set of prototypes. The objective of this algorithm is to reduce the number of prototypes while maintaining the highest possible accuracy. Having an initial training set X; Chang's algorithm combines all pairs of closest prototypes, as long as they belong to the same class and their merging does not lead to an increase of classification error. Initially, a sample x_i is randomly selected from the training set, and during the next step of the algorithm its nearest neighbour x_j is determined. The pair is evaluated and if merging is allowed, the resulting prototype is of the same class as the original ones and is computed in terms of their weighted average; hence, the resulting prototype x' is computed using the following formula:

$$x' = \frac{w_i x_i + w_j x_j}{w_i + w_j} \tag{3.1}$$

where w_i and w_j are the weights of x_i and x_j, respectively. The above equation is simplified to the average vector when both weights are equal to one. Merging occurs if and only if the new prototype does not increase the number of misclassified instances. If

merging is successful the distance matrix has to be updated accordingly and the whole process continues for the rest of the instances until no pair of nearest neighbours fulfills the merging criterion. Being the first abstraction method proposed, Chang's algorithm became the foundation for a lot of future work on prototype generation.

In [Bez98b] a new approach of Chang's method, namely Modified Chang Algorithm, was discussed, where the data space is partitioned for more efficient search of prototypes. The merging criterions are the same as in the original method proposed in 1974, but MCA has two main differences compared to its predecessor. The first is the elimination of weights, as the output of the merging process between two prototypes is only their arithmetic mean, hence, (3.1) is simplified to:

$$x' = \frac{x_i + x_j}{2} \tag{3.2}$$

The second aspect introduced, which largely affects the performance of the algorithm, is the partition of the prototype space in homogeneous regions. Using the class labels of prototypes the distance matrix is divided in homogeneous submatrices, in order to optimize search. As a result, search for the minimum distance between two prototypes is only performed within submatrices, so pairs of different class labels are avoided. This process speeds up the algorithm, which, similarly to Chang's algorithm, terminates when no merging between the remaining prototypes is allowed.

Another prototype reduction method that separates the feature space in different class regions, the basic event generation (BEG) algorithm, was introduced for the purpose of classification [Ich79]. BEG generates homogeneous neighbourhoods in the form of hyperrectangles, called events, and when allowed, merging of two events occurs, resulting in the generation of the minimum hyperrectangle that includes both of them. It should be mentioned that a hypperrectangle could be composed by only one instance. So instances of the same class label, x_i and x_j, are merged, if the distance of the minimum hyperrectangle containing both of them, to the nearest enemy is larger than a user defined threshold. The main concept of this algorithm is based on the classifier proposed in [Sto74], and because of a trade-off between the number of generated hyperrectangles and class separability the threshold has to be optimized.

Furthermore, three conditions that manage the merging of samples of the original training set X, so that the resulting set of prototypes is not only prototype consistent, but also cluster consistent, as shown in (Fig. 3.I), were described in [Mol02]. Prototype consistency is achieved when a prototype set can classify all samples in X correctly. On the other hand, a set is cluster consistent when every sample in X is closer to its cluster representative prototype x_i than to any other enemy class prototype. Condition (1) that guarantees cluster consistency states:

$$\left\| x_k - x_j \right\| > 2\max\left(r_k, r_j\right) \forall l \neq l_j \tag{3.3}$$

where x_j is the new prototype of class l_j, x_k represents the closest prototype of x_j and r_k the radius of cluster of prototype x_k in its respective class. The second condition states:

$$\left\| x_k - x_j \right\| > 2\max\left(r_k, r_j\right) \forall x_k \in A : l_k \neq l_j \tag{3.4}$$

where A is the resulting set of prototypes. Finally, condition (3) states:

$$\begin{cases} \left\| x_k - x \right\| > \left\| x_j - x \right\| & \forall x \text{ in the cluster of } x_j \\ \left\| x_k - y \right\| > \left\| x_j - y \right\| & \forall y \text{ in the cluster of } x_k \end{cases} \forall x_k \in A \text{ with } l_k \neq l_i \tag{3.5}$$

So, using these conditions in the order presented above, the algorithm evaluates all resulting prototypes and if at least one of them is true, merging of the two cluster representatives is allowed.

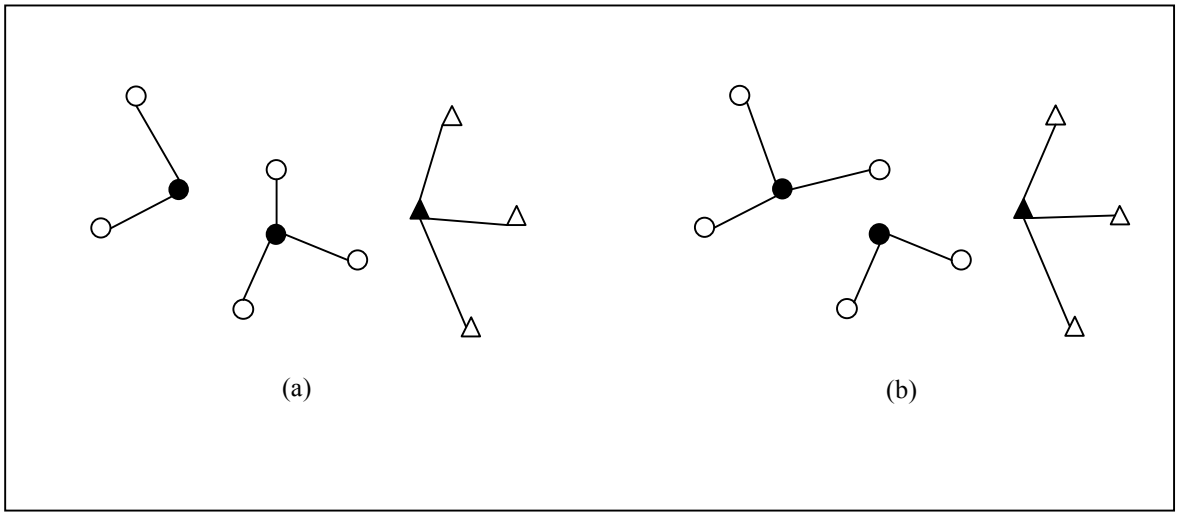


Figure 3.I- A two-class example with three clusters. (a) Example of a cluster consistent set. (b) Example of a prototype consistent set.

Three additional reduction methods that involve partition of the data space, hence, their name Reduction by Space Partition (RSP1), (RSP2) and (RSP3), were proposed in [San04]. The first proposed technique, RSP1, computes the diameter of the training set by calculating the distance between the two farthest points x_i and x_j of the entire training set X. The training set is then divided into two parts, the first one containing all instances that are closer to x_i than to x_j, while the second set includes the rest of the samples that are closer to x_j. Similar divisions of the space continue until the number of partitions reaches a pre-specified value b that is defined by the user. Having a total number of l_c different class labels, during the next step, RSP1 identifies the different classes that exist in each partition and computes the centroids of every class. These leads to a maximum value of $b \times l_c$ centroids, which are then, used as the representative prototypes. It should me noted that after the partition of the space into subsets, the one with the largest diameter will contain the largest number of instances. Therefore, it will

be the one with the highest overlapping degree. In order to tackle this problem, RSP2, which is an expansion of the previously described method, was suggested. A modification in the partition criterion is added so that space is divided according to the overlapping degree. After the first division, which is exactly the same as the first step of RSP1, every step of RSP2 involves the division of only one of the two resulting subsets, more specifically the one with the largest diameter. Therefore, RSP2 achieves better classification results than RSP1. Finally, to further improve the classification performance of this method, the third proposal, RSP3, involves the partition of every subset until each partition contains instances from only one class; hence assuring homogeneity. This is achieved by following the same procedure until each subset becomes a cluster of training instances of the same class. The division criterion can be either the one used in RSP1 or in RSP2, since the final subsets will be homogeneous, and once again the centroids of these subsets are used as the representative prototypes.

Another instance based learning method called prototype generation and filtering (PGF), was proposed in [Lam02$_a$]. This technique consists of two individual components, one that uses filtering and one that generates prototypes by merging. Considering instance abstraction first, PGF incorporates class entropy in the distance metric in order to maintain an acceptable level of homogeneity in the generated prototypes. So the distance metric used is as follows:

$$\|x-y\|_F = \alpha\|x-y\|_E + (1-\alpha)E(x,y) \tag{3.6}$$

where the final distance between two prototypes depends on two measures, the euclidean distance between them and the entropy E. A balancing parameter α is used to weight the importance of these two measures. After the computation of the distance matrix, those instances with the shortest distance are merged. The merging process continues as long as the number of prototypes remains bigger than the number of classes. So class information of the propotypes is indirectly integrated in the entropy, which is defined as:

$$E(x_i) = -\sum_{i=1}^{c} R(x_i,l_i)\log R(x_i,l_i) \tag{3.7}$$

where R is the frequency of occurrence of class l_i in the prototype x_i and c the total number of classes. Small entropy is obtained when most instances of the prototype belong to the same class, while high values of entropy mean class variability. As a result, the distance metric used takes into consideration not only the feature vectors but also their class and the class of the resultant prototype is the class of the majority of instances it consists of. In order to address the problem of noisy instances and outliers, PGF employs a filtering technique called ACC. This method classifies every instance of the training set using the nearest neighbour rule and each time an instance is correctly classified, the classification accuracy of its nearest neighbour is increased. When all of the training set is

scanned through, only prototypes with high accuracy, over a certain threshold, are retained. Two variations of PGF were suggested depending on the order of the two components, as PGF1 filters instances to discard outliers and noise before the generation of the prototype set, while PGF2, filters the generated prototypes since ACC is applied after the merging process.

As already mentioned, border instances hold most of the information needed to accurately describe the underlying distribution. A similarity metric that uses the same and different class similarity between instances to efficiently determine border instances is typicality [Zha92]. Typicality of an instance x_i is defined as the ratio of the average similarity of x_i to all of its friends $F(x_i) = \{y \in X : \psi(y) = \psi(x_i) \wedge y \neq x_i\}$ over its enemies $E(x_i) = X - F(x_i) - \{x_i\}$ from the entire training set *X*. Hence,

$$T(x_i) = \frac{1 - \frac{1}{\alpha_{\max}|F(x_i)|}\sum_{y \in F(x_i)} \|x_i - y\|_2}{1 - \frac{1}{\alpha_{\max}|E(x_i)|}\sum_{y \in E(x_i)} \|x_i - y\|_2} \tag{3.8}$$

where a_{max} is the largest distance in the training set. Instances can be classified in three categories depending on their typicality value. Those with typicality lower than one are noisy instances, while values close to unity indicate border instances. Typical instances normally display values much greater than one. Hence, border and noisy instances can be efficiently determined. Zhang's algorithm is an additive technique that iteratively selects the most typical instance, displaying the largest typicality value. After every addition the resulting set is re-evaluated and the process is repeated until no misclassifications occur. Although this instance based method is a selective algorithm, it is described in this section, because it is the basis of the Integrated Concept Prototype Learning (ICPL) method, which is one of the most efficient abstraction algorithms in the literature.

ICPL is another abstraction method proposed by Lam, which, exactly like the PGF algorithm, consists of two individual components, a filtering and a generation one [Lam02$_b$]. Although no pioneering work has been proposed for the filtering component, since methods used are introduced in [Lam02$_a$], [Wil00] and [Wil72], the abstraction component is novel. In contrast to PGF, ICPL uses the similarity metric typicality in order to distinguish between vectors close to class boundaries and internal samples. All instances of the training set *X* are sorted by their typicality value that is initially computed. Thresholds are then set for each class, depending on their statistical properties, in order to distinguish between noisy, border and non-boundary instances. During the next step of the algorithm, the instance x_i with the highest typicality value is selected and its nearest neighbour x_j is determined. As long as x_j is of the same class as x_i, and is not a border instance or an instance already treated, merging of x_i and x_j takes place. When any

of the three conditions mentioned (different class, border instance, or previously treated) is true, merging ends. The next most typical instance is selected and the merging process is repeated once more. The abstraction procedure terminates when all non-boundary instances have been treated. An additional step is employed by ICPL as the output sets of the filtered and generated prototypes from the respective components are combined and further processed to discard redundant prototypes. Similar to PGF, variations of ICPL have been proposed based on the order the filtering and the abstraction components are evaluated and post-processed.

3.3 Learning Vector Quantization

A widespread competitive prototype-based technique is Learning Vector Quantization (LVQ), which selects a reduced subset of prototype vectors from the original training set. These vectors are then modified according to some rule (LVQ1, LVQ2, etc) in order for the algorithm to define optimal class regions in the data space and achieve high classification accuracy.

The Decision Surface Mapping (DSM) algorithm, a method that belongs to the family of LVQ, was introduced in [Gev91]. Initially, a subset of instances is randomly selected from the original training set, with the number of initial vectors representing every class indicated by their a priori probabilities. All instances of the training set are then passed through the algorithm and in case of correct classification the prototype set remains intact. But, if a misclassification of an instance x_i occurs, prototypes have to be modified, so the nearest neighbour prototype is punished because it belongs to a different class, while the nearest same class prototype is rewarded in order to move towards the misclassified sample. Firstly, the enemy prototype is punished according to the following rule:

$$e(t+1) = e(t) - \alpha(t)\left[x_i(t) - e(t)\right] \tag{3.9}$$

where e is the enemy prototype, a the scalar gain factor and t indicates the iteration. Secondly, the reward of the nearest friend prototype, f, is provided by the following equation.

$$f(t+1) = f(t) + \alpha(t)\left[x_i(t) - f(t)\right] \tag{3.10}$$

The same procedure is repeated for all patterns in the training set until no misclassifications occur. The gain factor is a user-defined scalar that determines the sensitivity of the algorithm and controls the learning process. The gain factor decreases with time and the algorithm terminates when a reaches the value of 0 (or, as mentioned above, when all samples are correctly classified). While DSM is a classification method

that only tries to minimize the error; hence changes only occur when samples are misclassified, LVQ introduced in [Koh86] updates prototypes even after the training sample is correctly classified. The initial steps and the reward and punishment equations are equivalent to the ones suggested in DSM, but for the training sample x_i to be classified, the two nearest neighbour prototypes, e and f, have to form a window of width w given by:

$$\min\left(\frac{d_e}{d_f},\frac{d_f}{d_e}\right) > \left(\frac{1-w}{1+w}\right) \tag{3.11}$$

where d_e and d_f are the distances of x_i from the two prototype vectors, the enemy and friend respectively. Then, the punishment rule for e is given by Eq. (3.9), while Eq. (3.10) defines the reward rule according to which the friend prototype f is updated.

An expansion of the Learning Vector Quantization was introduced, LVQ3 that is used in [Kim03]. In this case, even if the two nearest prototypes belong to the same class as x_i both are updated according to a new function:

$$p(t+1) = p(t) + \varepsilon(t)\alpha(t)\left[x_i(t) - p(t)\right] \tag{3.12}$$

where α and ε are the learning and relative learning rates respectively. Another contribution of the particular algorithm is the fact that it is a hybrid method. LVQ3 uses the LVQ update rule as an "extra step" of the reduction algorithm and it combines it with SVM in order to further improve the obtained prototype set. It is proven that hybridization can enhance the overall performance of such algorithms. The parameters of LVQ3, including the number of prototype vectors, their initial values, and the number of iterations, are optimized in order to determine the best possible prototype set, while the learning rate used is computed as follows:

$$\alpha(t) = \alpha(0)\frac{\mu}{t+\mu} \tag{3.13}$$

where t is the discretized time index, while μ is the number of the iteration. As can be observed the learning rate decreases monotonically with time and in a linear manner which is the trend in the majority of the LVQ algorithms.

An alternative LVQ method that employs the concept of the already defined Nearest Centroid Neighbourhood was introduced in [San06]. Although the basic learning rules remain the same as the ones used in the standard LVQ method, this modification on the neighbourhood has a substantial effect on the final output of the algorithm. As a result, for every input pattern that is processed, the adaptive model updates all prototypes that lie around it. So prototypes are moved in order to surround the input pattern in a small neighbourhood area. This new adaptive algorithm enhances the performance of the LVQ model as it takes into account not only dissimilarities but also the geometrical distributions. In this case, the learning rate is determined using the following equation:

$$a(t) = \frac{a(t-1)}{1 + s(t)a(t-1)} \tag{3.14}$$

where *s(.)*=1 if the two prototypes share the same class label, or *s(.)*= -1 if they belong to enemy classes. Again, as can be observed the learning rate decreases monotonically with time.

Various algorithms make use of the LVQ technique in order to improve the classification accuracy. Another such method was proposed in [Li05]. The LVQ Pruning algorithm (LVQPRU) uses self organising map (SOM) to generate prototypes and employs a weighted distance measure defined as:

$$d(x,p) = \sum_{i=1}^{n} w_i(x_i - p_k) \tag{3.15}$$

where p_k is the generated prototype by SOM and n is the number of instances in the training set. The above distance metric is combined with a functional link network (FLN) classifier that minimizes the following error function,

$$E = \frac{1}{c}\sum_{j=1}^{c}\sum_{i=1}^{n}\left[l_i(j) - l_i'(j)\right]^2 \tag{3.16}$$

where c is the total number of classes, *l(.)* is the class label and *l'(.)* is the desired output of the input instance. For an instance x_i the weight that will be used in the distance metric will be its importance *u*, only normalized:

$$w_i = \frac{u_i}{\sum_{i=1}^{n} u_i} \tag{3.17}$$

The importance of a sample depends on the derivative of the output class label with respect to the relevant instance, and is given by

$$u_i = \frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{c}\left|\frac{\partial l_k(j)}{\partial x_k(i)}\right| \tag{3.18}$$

The initial prototypes are randomly generated and a different SOM is trained for each class. The pruning component of this method also involves the deletion of all empty prototypes along with the application of the LVQ algorithm that is applied in order to fine-tune the locations of the samples and improve the generalization of the method.

A complicated abstraction method for prototype reduction was proposed in Ruta [Rut07]. The entire data space is considered to be an electrostatic field, whereas every instance of the training set *X* acts as a charged particle, attracting and repelling other prototypes. Using the Parzen window technique a density estimation for every instance x_i of the training set, and the total class density estimation are computed. These density estimations act as forces on prototypes, and similar to LVQ, attract (or 'reward') same class vectors and repell (or 'punish') enemy prototypes. All instances of the training set

experience such forces that determine the movement of the data in space. Reduction is achieved if two instances move from their original locations to a distance that is less than a certain user-defined threshold τ. These instances are then merged resulting in a new prototype. Despite the fact that this algorithm, as can be seen in Fig. 3.II, has its foundation in Physics, it displays a clear resemblance to the LVQ technique.

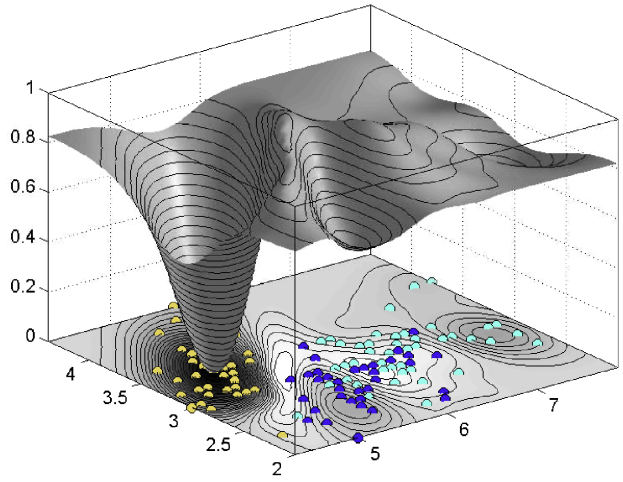


Figure 3.II- Energy between labelled instances in the Iris dataset [Rut07]

As proven in [Cra02], the LVQ model is a special case of a maximal margin principle approach, and despite the family of LVQ algorithms has been around for more than 20 years, new modifications and enhancements are constantly developed.

3.4 Clustering algorithms

In supervised learning, as already explained, training data consists of input instances and their desired output, their class label. On the contrary, in unsupervised learning all input vectors are unlabelled. Therefore, other features of the training data, such as statistical properties or proximity with respect to a certain distance metric, are used to train the algorithm. Data is partitioned into subsets and each one defines a cluster that is represented by a prototype.

Algorithms based on parzen windows, or other density estimation techniques have been previously described, but Astrahan in [Ast70] was one of the first to develop a reduction method that employed density estimation for clustering. A hypersphere of radius r is used in order to obtain an estimate of the density around an instance. During the next step, the sample with the highest density is selected and a different hypersphere centred at the chosen sample and of different radius is defined. Finally, all instances that

lie within it are discarded and the same process is repeated for the rest of the training samples.

In [Ald94] the Dog Rabbit (DR) strategy was introduced which is an unsupervised competitive learning model, which displays relatively large condensation since instances are clustered together. Initially, the DR algorithm generates a number m of randomly distributed prototypes. An instance x_i from the training set is then selected and its distances from all the prototypes are computed in order to determine the nearest one, p_j. The DR algorithm then moves all prototypes towards x_i according to a dynamic update scheme:

$$p_k = p_k + \frac{2\|x_i - p_k\|}{\left(1 + \|x_i - p_k\|\right)^{f_k}}\left(x_i - p_k\right) \qquad k = j \qquad (3.19)$$

$$p_k = p_k + \frac{\|x_i - p_k\|}{\Lambda + \|x_i - p_k\|}\frac{2\|x_i - p_k\|}{\left(1 + \|x_i - p_k\|\right)^{f_k}}\left(x_i - p_k\right) \qquad \begin{matrix} k = 1,...,m \\ k \neq j \end{matrix} \qquad (3.20)$$

where f_k and Λ are the fatigue and the factor determining the inhibition of non winning prototypes, both of which are user-defined parameters. The movement of prototypes terminates when the winning prototype p_j approaches within a certain distance from x_i. The fatigue[4] of the winner is then updated accordingly. The larger the fatigue of a prototype, the lesser it can move, and when a certain threshold of fatigue is exceeded movement is not permitted any more. Another sample of the training set follows and the same process is repeated, until it converges, which means that the total fatigue of all prototypes has exceeded the maximum allowed. Since its introduction in 1994, the DR algorithm has been the focus of analysis and research [Bez98$_b$]. Another modification of the DR technique that is employed for image processing was suggested in [Hil05]. The proposed method uses a pre-processing tool that maps input data onto a toroidal surface. As a result, the negative effect of edges on the identification of clusters is addressed, and the performance of the clustering component is improved.

Recently, a class-based algorithm that performs clustering on the training instances and determines representatives for each cluster was introduced [Che07]. An instance x_i is selected from the original training set X and its nearest enemy instance e that lies at a distance r_{ij} away from x_i is determined. During the next step of the algorithm, all same class instances k that lie within r_{ij} are determined, and are assigned to a cluster C_i that consists of x_i and patterns that lie within the circle $o(x_i,r_{ij})$. A representative of the newly generated cluster is computed using the pair of instances in C_i that display the maximum distance. The first representative is simply their centroid. As a result, the computed centroid becomes the centre of the cluster, while its radius d_i is half their

[4] The rationale behind fatigue is to slow down the movement of prototypes so that they remain close to samples they have already seen.

distance. The same procedure is repeated for the entire training set, but for better cluster representation, additional instances are used as representatives. Therefore, border instances are also taken into consideration and every cluster is denoted by three different cases of prototypes. Firstly, the cluster centre as explained previously. Secondly, samples whose nearest neighbours belong to a different class, hence, are border instances. Finally, the two nearest neighbours of the two first nearest enemies of the cluster centre are cdetermined and are also considered as cluster representatives. Hence, the algorithm succeeds not only in largely condensing the original training set, but also in clearly identifying class borders. In order to improve the time requirements of the proposed method, an optimized search scheme was also presented. The search for the nearest neighbour of a new unseen pattern x_i is not performed over the entire prototype space, but within the hypershpere centered at x_i and with radius r_x, defined as the sum of the radius of the nearest cluster center, d_i, and the distance $\|p_i - x_i\|$, where p_i is the nearest cluster centre prototype.

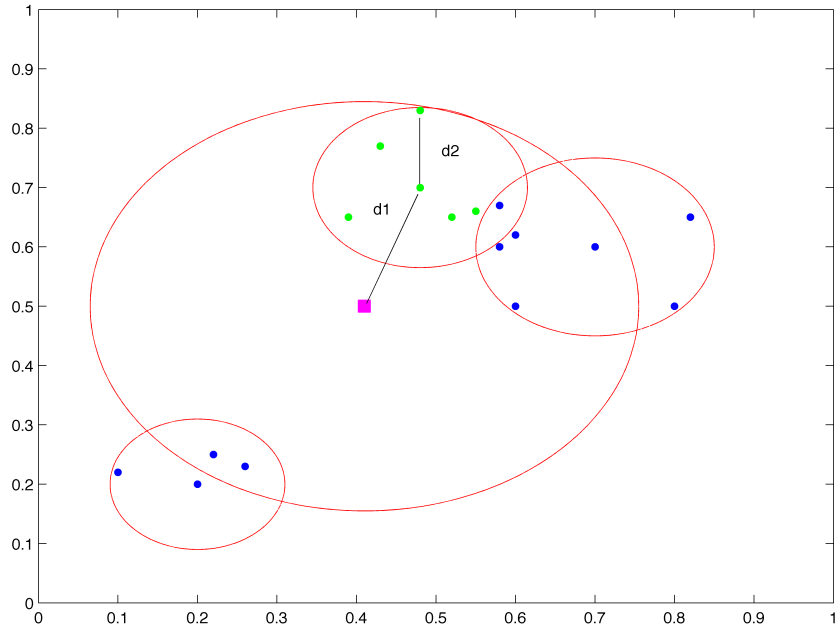


Figure 3.III- Search for the nearest neighbour is performed only within the disc of radius d_1+d_2

As already mentioned, due to the large number of instances the majority of the reduction algorithms, and especially abstraction methods, suffer from large computational complexity and high response times. Therefore, Czarnowski proposed in [Cza10] the application of distributed learning on clustering as he introduced an agent-based method for instance reduction. Similar to the Parallel Condensed Nearest Neighbour [Ang07$_b$], the proposed technique shares a common memory with an initial population of individuals that represent the samples of the training set. Starting from this population a network of agents operates asynchronously and in parallel alternating and concluding to

different population solutions. These agents are in communication with each other as they constantly update the population in order to determine the optimum solution. The improvement of every population is performed locally as every agent aims to independently enhance an individual. The main reduction technique proposed clusters instances according to their similarity coefficient as introduced in [Cza04]. But, the particular agent-based algorithm is also tested with two other reduction techniques that substitute the similarity coefficient clustering and are again locally applied. The first one involves the use of stratification strategy as introduced in [Can06], and has already been described in Chapter 2. This technique aims to map the original training set into disjoint strata of equal class distribution. Compared to the initially suggested technique the stratification strategy is quite inefficient in terms of the computation times required. The second method employed is the K-means clustering technique, which despite its simplicity, can also end up being rather complex in terms of time requirements.

The cluster-based prototype algorithms have been described, that are relevant with the work done in instance reduction, but the literature in clustering is very broad because of the extensive research on exploratory data analysis. Clustering, in general, has been applied in various fields such as machine learning and artificial intelligence, computer sciences and pattern recognition, medical sciences (i.e. genetics biology clinical sciences) or even in finance. Therefore, numerous methods that partition data into different clusters according to their separability or homogeneity have been proposed, which can be subcategorized in more groups based on the techniques used [Xu05]. A small outline of the most important and widely used groups will be presented, along with some of the most successful clustering algorithms, but this thesis will not go into details, as clustering could be a research topic by itself.

- Similarity-based clustering. The algorithms of this group adapt different distance metrics that define the proximity between the individuals. The clusters are a product of this linkage. Such algorithms have been suggested in [Yan04] and [Cha06].
- Hierarchical clustering. It demonstrates similarities to the previous case, but the closeness between clusters and not individuals is examined. Clusters are structured hierarchically, and in most cases the result is presented in a dendrogram. Representatives of such methods can be found in [Mur00].
- PDF estimation. Samples in the clusters are generated according to the various probability distributions. An example of such a method is suggested in [Rub08, Yu05].
- Vector Quantization Clustering (VQC). Based on a certain criterion samples are assigned to clusters of relatively same number of points. No hierarchy is present

and the most common VQC algorithm is the K-means method [Mac67]. Another example of such an approach is [Lee05].

- Combinatorial-based clustering. This group involves adaptive algorithms that use genetic programming to optimize the cluster search. Examples of models with such optimization components include [Cow99] and [Tse01].
- Spectral Clustering. The basic concept of spectral clustering is graph theory, which generates nonlinear hypersurfaces between different clusters and enables their separation, similar to the technique proposed in [Ng02]. More algorithms employing spectral theory are thoroughly examined in [Fil08].
- Kernel-based clustering. These methods perform a nonlinear data transformation to a higher dimensional feature space that increases the linear separability of samples. Such methods include [Gir02] and the well-known Mean-Shift algorithm [Wu07, Che95, Com02] as well as algorithms like [Fuk75] and [Haz07].

For further information in clustering the reader can refer to [Eve01].

3.5 Conclusion

This chapter investigates the field of instance abstraction and presents various algorithms along with their characteristics. As already mentioned, one should decide on what algorithm to use based on the application at hand, and on the behaviour of the method. Instance abstraction algorithms often result in high condensation, due to the freedom of replacing instances, but this may lose track of the contribution of the original instances. Therefore, new prototypes are generated to fill regions in the domain of the problem to improve weak representative samples. In general, abstraction algorithms display higher condensation results compared to selection methods, but as a result, lack in terms of accuracy. Also, time requirements increase due to the generation process. Finally, these observations are illustrated in the results of the following chapters.

CHAPTER 4:

A CLASS BOUNDARY PRESERVING ALGORITHM

4.1 Introduction

In this chapter a novel approach is proposed, the Class Boundary Preserving Algorithm (CBP) [Nik11], which is a hybrid multi-stage method for pruning the training set. CBP aims to preserve samples close to the class boundaries, since these instances can provide most of the required information to effectively describe the underlying distribution. In section 4.2 the four steps of the algorithm are presented. During the first stage, cleaning using Wilson's editing rule [Wil72] is performed, followed by boundary identification that is based on a simple but very effective heuristic. The third stage involves pruning of border instances to remove redundancy using a direct graph approach, namely mutual nearest enemies. The proposed method uses not only selection of instances but also generation of prototypes. Hence, in 4.2.iv the abstraction component of CBP is described. For the latter, the mean shift clustering approach [Wu07, Che95, Com02] is employed on the non-border instances. Finally, the output set involves the combination of the selected border instances and the newly generated prototypes. Section 4.3 presents the results on real datasets and comparisons of the formulated method against other successful prototype condensation algorithms, while section 4.4 concludes the chapter.

4.2 The Proposed Algorithm

Having an initial set $X = \left\{ x \in \Re^d \right\}$ of n d-dimensional instances, where each sample is associated with a unique class label $\psi(x) \in L = \left\{ l_1, ..., l_c \right\}$, the problem in instance reduction is to determine a set of m prototypes (where $m << n$) that best describes the underlying distribution. Internal instances positioned away from class boundaries have little or no effect on classification accuracy. On the contrary, samples that lie close to class boundaries hold enough information to accurately describe the decision surface [Bri02]. Therefore, the proposed framework discards center instances while it retains a suitable number of border patterns. The innovation of this scheme lies in the procedure used to divide the training set X in two subsets; X_B which includes instances close to the decision surface, and X_{NB} which contains internal samples. Because there is a distinctive difference in the importance of the information these two sets hold, two separate reduction processes are applied in each one. Overall, in order to compute the reduced set of prototypes four steps are involved in the proposed algorithm, and are described in detail in the following sub-sections.

i. Smoothing Class Boundaries

In the first step the problem of noise is addressed. Many algorithms suffer from the presence of noisy instances near the class boundaries that degrade classification accuracy. In order to deal with this issue, a filtering component is employed in most instance reduction algorithms, such as [Wil00, Bri02]. Wilson's ENN [Wil72] is the most commonly used noise filter because of its simplicity. CBP also employs ENN as the first stage of the algorithm, in order to discard harmful instances misclassified by their k nearest neighbours (with k=3). A synthetic two-dimensional three-class dataset of 250 samples per class is used to illustrate the operation of the CBP Algorithm in Fig. 4.I. The result of the condensing rule applied is demonstrated in Fig. 4.I(b), where classes do not overlap anymore, thus the decision boundaries have been effectively smoothed by the filtering component.

ii. Distinguishing between Border and Non-border Instances

After boundaries have been smoothed, a novel scheme, that uses the geometric characteristics of the underlying distribution to partition the pre-processed X into the subsets of border X_B and non-border X_{NB} instances, is applied. The reachable set $R(x)$ of a pattern x, as defined in [Smy95], holds instances belonging to $\psi(x)$ that lay closer to x than its nearest enemy; that is, $R(x)$ is a set of all vectors that can provide a correct 1-NN classification for x. In this work, the concept of reachability to multiple levels involving

more than just the nearest enemy is extended. $R_i(x)$ is the reachable set determined by the i-th enemy $\xi_i(x)$ of x defined as:

$$R_i(x) = \left\{ y \in X : \ \psi(x) = \psi(y) \ \wedge \ \|x - y\|_2 \leq \|x - \xi_i(x)\|_2 \right\} \tag{4.1}$$

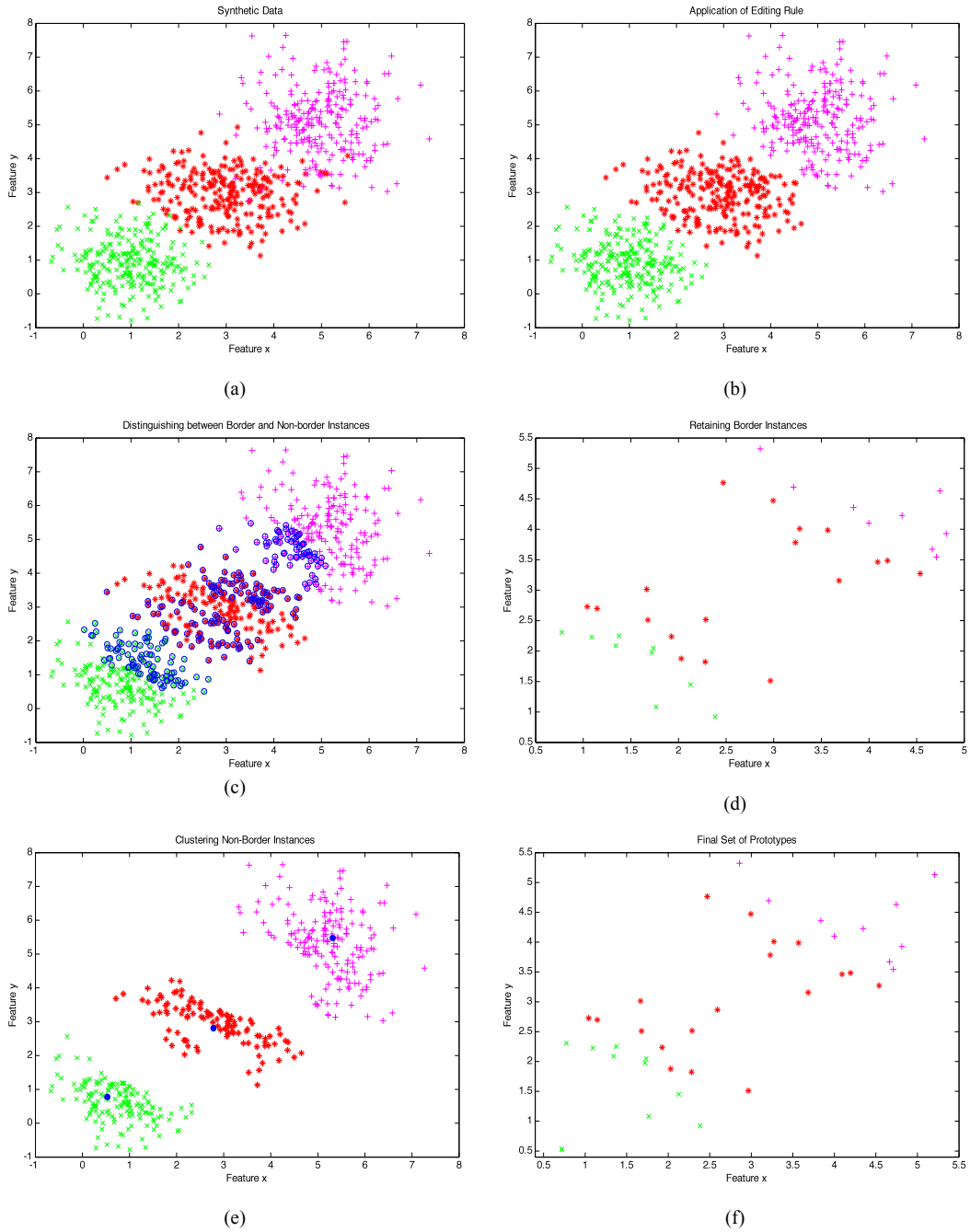


Figure 4.I- (a) A synthetic three class 2-dimensional dataset. **(b)** Wilson's editing rule removes noisy instances and overlap of classes is avoided. **(c)** Class boundaries are determined in step 2. Border instances, which are enclosed by circles, are identified and the decision surfaces are defined. **(d)** Step 3 reduces the number of border instances that are needed to accurately define class borders. **(e)** Clustering of non-border instances takes place in step 4. Cluster centers computed (marked with solid circles) and inserted in the set of prototypes. **(f)** Final reduced set of prototypes consisting of cluster centers and remaining border instances.

where:

$$\xi_i(x) = \underset{\substack{z \in X \\ \psi(z) \neq \psi(x) \\ z \neq \xi_j(x),\ j=1,\ldots,i-1}}{\operatorname{argmin}} \|x - z\|_2 \tag{4.2}$$

and $\xi_1(x)$ is the first nearest enemy of x.

At this stage of the algorithm, the aim is to determine samples close to the decision surface. The above multi-level reachability is used here to make the selection of such samples more resilient to the sparsity of the data and the class boundary irregularities. This is done as follows. Initially, for each training sample x a set $I(x)$ is collected that consists of k_R (k_R is fixed to 3 for all experiments) indices of its nearest enemies $\xi_i(x)$ with $i \in I(x)$. To make these enemies define mostly non-overlapping reachable sets, $I(x)$ is forced to contain those nearest enemies whose line segment from x are at angles larger than a user-defined threshold of φ_R (set to a fixed value of $20°$) from each other. This condition allows for a more universal view of samples around x, because nearest enemy instances can lie very close to each other. This nearest enemy index set is defined as:

$$I(x) = \underset{\substack{J \subseteq \{1,\ldots,n\} \\ |J| = k_R \\ \phi(\xi_i(x)-x, \xi_j(x)-x) \geq \varphi_R,\ \forall i,j \in J,\ i \neq j}}{\operatorname{argmin}} \sum_{i \in J} \|x - \xi_i(x)\|_2 \tag{4.3}$$

where $\phi(\cdot,\cdot)$ returns the angle between two vectors.

Subsequently, in order to decide on whether an instance x is close to class borders towards the enemy classes, there exist the need to determine how samples in $R_i(x)$ are scattered in space. The way these patterns are spread around x with respect to $\xi_i(x)$ can reveal if x is close to the decision boundaries or not. To achieve this, two vectors are defined as y-x and $\xi_i(x)$-x and the cosine similarity $C_{i,x}(y)$ is employed, to judge if the friendly sample y lies near the line connecting x and its enemy $\xi_i(x)$:

$$C_{i,x}(y) = \frac{\langle y - x, \xi_i(x) - x \rangle}{\|y - x\|_2 \cdot \|\xi_i(x) - x\|_2} \tag{4.4}$$

This metric is invariant to rotations and dilations but not to translations. $C_{i,x}(y)$ is computed for every instance y within each $R_i(x)$ and aggregate responses in k_R sets:

$$S_i(x) = \left\{ C_{i,x}(y), \forall\ y \in R_i(x) \right\} \tag{4.5}$$

Finally, all $S_i(x)$ are combined for all enemies in $I(x)$ as:

$$S(x) = \bigcup_{i \in I(x)} S_i(x) \tag{4.6}$$

All the instances $y \in R_i(x)$ are positioned within a sphere passing through $\xi_i(x)$ and centered at x. Therefore, the statistical distribution of the cosine scores in $S(x)$ shows the relative to the enemy scatter of the friendly instances y of x around x. If values in $S(x)$ are mostly positive, most instances y in $R_i(x)$ are restricted to lie within the intersection of the

reachability sphere and a cone with axis x-$\xi_i(x)$, apex at position x and containing $\xi_i(x)$ (the width of this cone is controlled by τ, below in Eq.(4.7)). This means that friendly samples lie in between x and its nearest enemies, so x is positioned away from class boundaries. On the other hand, large negative values in $S(x)$ show a directed scattering of samples y outside the cone containing $\xi_i(x)$. Therefore, values in $S(x)$ give an estimate of how uniformly patterns close to x are distributed with respect to $\xi_i(x)$ for various values of $i \in I(x)$. The above considerations, lead us to the use of a very simple but robust test, to judge whether each training instance lies close to the enemy line or not. The test involves the calculation of the median of $S(x)$ for every instance x, and retains x as a border instance if this value is lower than a threshold $-\tau$. The decision criterion distinguishing between border X_B and internal X_{NB} instances is given by:

$$X_B = \left\{ x \in X : \text{median}\left(S(x)\right) < -\tau \ \vee \ \left|R_1(x)\right| \leq 2 \right\} \tag{4.7}$$

$$X_{NB} = X - X_B \tag{4.8}$$

where samples x in sparse areas with less than three friendly ones are retained unconditionally. Fig.4.I(c) shows the behavior of this step, where instances near the decision surface and internal samples have been successfully determined. A clarifying illustration of the above simple mechanism is presented in Fig. 4.II, for a synthetic two-dimensional example.

Although the above heuristic is reasonably intuitive, it can be shown that it is effective in reducing redundant instances. This can be equivalent to showing that for any query point $q \in \Re^d$, then $\hat{\psi}_{X_B}(q) = \hat{\psi}_X(q)$; here the notation $\hat{\psi}_X(q)$ is used to denote the estimated prediction label l_i of the 1-NN classifier for the point q, using X as the training set. In other words, it is necessary to show that the label of the nearest point to q will not change by removing the set $X_{NB} = X - X_B$. This can be shown as follows. Assume that z_1 is the nearest neighbour of q within the entire dataset X, and z_2 its nearest neighbour within the border set X_B (i.e., before and after removal of X_{NB}, respectively). It is obvious that if $z_1 \in X_B \subset X$ then z_1 will not be removed. In this trivial case:

$$z_2 \equiv z_1 \in X_B \quad \Rightarrow \quad \hat{\psi}_X(q) = \hat{\psi}_{X_B}(q) = \psi(z_1) = \psi(z_2) \tag{4.9}$$

which shows that removal of X_{NB} does not affect q.

The complex case arises when $z_1 \neq z_2$, that is when $z_1 \in X_{NB}$ and $z_2 \in X_B$. Let's make the assumption that $\psi(z_2) \neq \psi(z_1)$. Since z_2 is the new nearest neighbour of q, then $\|z_2 - q\|_2 > \|z_1 - q\|_2$ (ignoring ties), and so z_2 lies outside the sphere centered at q passing through z_1. Furthermore, the fact that z_1 has been removed, means that there exist many points $z \in R_1(z_1)$, such that from Eqs. (4.4-4.7), on average

$\frac{\langle z - z_1, \xi_1(z_1) - z_1 \rangle}{\|z - z_1\|_2 \cdot \|\xi_1(z_1) - z_1\|_2} \geq -\tau$ (for simplicity, we ignore multiple reachable sets for the moment). Then, there exist two cases, depending on whether z_2 is the nearest enemy of z_1 or not.

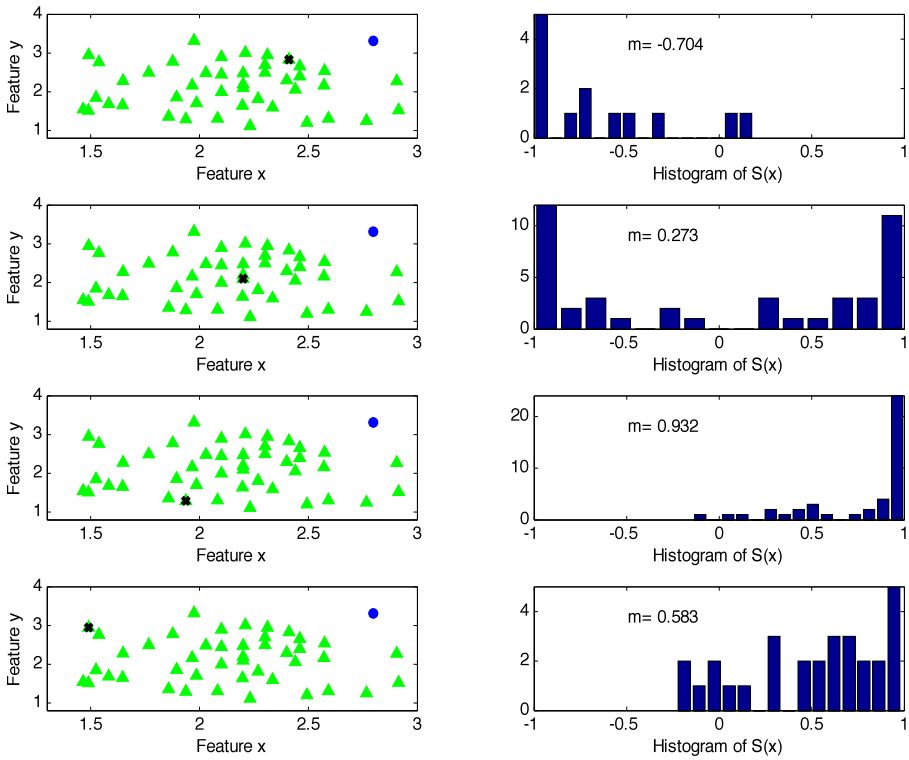


Figure 4.II- This example contains 45 instances of a class (marked as "▲"). Four instances are selected (marked as "✖"); one close to the decision surface, one in the middle of the distribution, and two at borders but away from the enemy class representative (marked by "●"). All four cases are shown in the left hand side. The histograms of the corresponding *S(x)* sets are shown in the right hand side of the figure. Border instances cause large negative values in *S(x)* as can be seen from the first case (top row) where the median value is *m=-0.704*. Internal instances tend to have a uniform scatter, hence, the average value within *S(x)* is close to zero. On the other hand, edge instances away from the enemy class (bottom two cases) exhibit high concentration in positive values and are, therefore, not included in X_B as they do not satisfy Eq. (4.7).

If $z_2 = \xi_1(z_1)$, z_2 would most likely be positioned such that z_1 lies between q and z_2. In this case, all points z which cover for z_1 would lie between z_1 and z_2, and the initial assumption would not hold as z_2 could not be the nearest neighbour of q. To show this, one can assume a value of τ=0 and high density around z_1 and prove that q, z_1 and z_2 must be collinear and arranged in this order, so that all $z \in R_1(z_1)$ are not within the aforementioned sphere around q. Since τ=0, then $(z - z_1)'(z_2 - z_1) \geq 0$, or due to the arrangement of q, z_1 and z_2 $(z - z_1)'(z_1 - q) \geq 0$ which can be equivalently written as:

$$\begin{aligned}\|z-q\|_2^2-\|z_1-q\|_2^2 &\geq (z-z_1)'(z-q)=\|z-z_1\|_2^2+(z-z_1)'(z_1-q)\\ &=\|z-z_1\|_2^2+t(z-z_1)'(z_2-z_1)\end{aligned} \tag{4.10}$$

for some positive t. However, since the last term is nonnegative, $\|z-q\|_2 \geq \|z_1-q\|_2$, which shows that when z_2 is located as far as possible from q on the other side of z_1, all points z are lying after z_1. Furthermore, $\|z-q\|_2$ is upper bounded by the distance $\|z_2-q\|_2$. This is because, since by definition all z lie within $R_1(z_1)$:

$$\begin{aligned}&\|z-z_1\|_2 \leq \|z_2-z_1\|_2 \Leftrightarrow\\ &\|z-z_1+z_1-q\|_2 \leq \|z-z_1\|_2+\|z_1-q\|_2 \leq \|z_2-z_1\|_2+\|z_1-q\|_2=\|z_2-q\|_2\end{aligned} \tag{4.11}$$

using triangular inequality and the arrangement of q, z_1 and z_2. This shows that $\|z-q\|_2 \leq \|z_2-q\|_2$. So far the case for $z_2=\xi_1(z_1)$ has been examined. If this is not the case, then similar arguments to the above can hold, since the heuristic of Eqs. (4.4-4.8) is based on multiple k_R reachable sets and nearest enemies.

iii. Pruning Border Instances

Samples included in X_B are near the class boundaries and, compared to internal instances in X_{NB} that are insignificant for 1-NN classification, they hold most of the information needed to describe the entire structure. Nevertheless, experimentation with different datasets showed that further condensing of X_B is possible, and here a fast heuristic is introduced to do so, using pairs of mutual nearest enemies (the concept of "mutual neighbourhood" was firstly introduced in [Chi79]). A directed edge is defined from every instance x in X_B to its nearest enemy $\xi(x)$ (ignoring subscripts) also within X_B; this results in a directed graph $G=(X_B,E)$ where the edge set is $E=\{(x,\xi(x)) \in X_B \times X_B\}$. Then, all nodes of bi-directional (mutual) enemy preference are unconditionally retained in a temporary set:

$$X'_B=\{x \in X_B : x=\xi(\xi(x))\} \tag{4.12}$$

Subsequently, all remaining edges are sorted in ascending order of their length $\|x-\xi(x)\|_2$. Then, the processing starts from the shortest one and conditionally inserts its two participating nodes to X'_B if neither has been previously added:

$$\text{if } \{x,\xi(x)\} \cap X'_B=\varnothing \;\Rightarrow\; X'_B := X'_B \cup \{x,\xi(x)\} \tag{4.13}$$

Finally, the current X_B is replaced by its reduced version X'_B. Fig. 4.III exemplifies

some cases of mutual and non-mutual enemy pairs between two classes. The above condensing procedure is applied with the purpose of removing redundant border instances, while preserving the actual class boundaries by retaining only nearest enemies with stronger preferences. Although, as discussed in the previous section, samples included in X_B are very close to the boundaries, data sparsity may cause some instances in X_B to be ineffectual to the competence of the classifier.

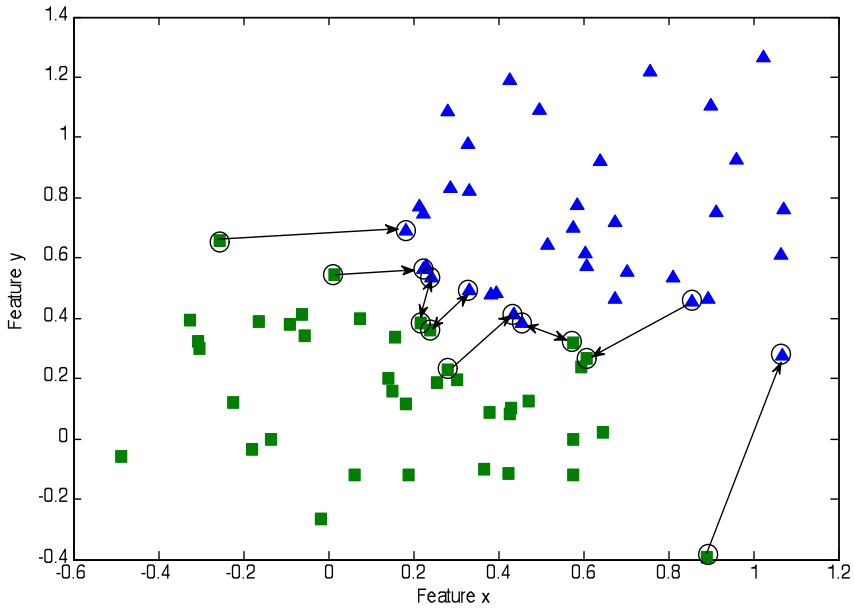


Figure 4.III- A two-class example of the reduction process applied to the border set X_B. Nearest enemy pairs are connected and highlighted in circles. Some have uni-directional preference where only one instance is the nearest enemy of the other, while the mutual nearest enemies, which are given priority, have bi-directional preference between them.

iv. Clustering Non Border Instances

In contrast to border instances, samples that lay close to centers of classes hold hardly any information and most internal instances do not affect the accuracy of the algorithm. But tests on various datasets showed that total exclusion of non-border vectors can, in some cases, reduce performance due to the sparsity of the datasets. Therefore, an unsupervised algorithm has been used that can largely reduce the number of instances held in X_{NB}. X_{NB} is partitioned into clusters which are groups of high local densities that correspond to major subclasses of the underlying distribution. To implement this, the Mean Shift clustering (MSC) [Wu07, Che95, Com02] has been employed, which is a non-parametric technique that uses the gradient of density estimator to determine the stationary points. The MSC algorithm converges to points of maximum density determining the cluster centers of the distributions.

Since X_{NB} consists only of internal instances, clusters obtained by the MSC algorithm will be homogeneous. The advantage of this clustering method over other ones

is that it requires no information on prior knowledge of the cluster number. To set and operate the MSC just one parameter is needed that is the bandwidth h of the kernel $k\left(\frac{x - x_n}{h}\right)$. Many bandwidth selection methods have been proposed (see [Wan06]). In this work a global bandwidth is required, thus the following formula is used that is similar to the one employed in [Gou07]:

$$h\left(X_{NB}\right) = \frac{a}{\left|X_{NB}\right| \times k_h} \sum_{x \in X_{NB}} \sum_{j=1}^{k_h} \left\| x - NN_j(x) \right\| \tag{4.14}$$

where $NN_j(x)$ is the nearest neighbour of x, α is a constant and k_h the number of nearest neighbours used (set to 5 in all experiments). Bandwidth value can largely affect the performance of the CBP algorithm. A very small bandwidth for example can result in an excessive number of clusters, while large values can lead to inhomogeneous clusters. Eq. (4.14) is based on average distances and a user-provided constant α (which is set to 0.1 for all experiments), and it makes MSC insensitive in setting the bandwidth.

In order to reduce the number of clusters obtained to the minimum, a merging process is finally applied. The class labels of cluster centers computed are checked and if nearest neighbour clusters share the same label, merging of the centers occurs. This method condenses the set of cluster centers generated, while it ensures that overlapping of different classes does not take place. An example of this step is in Fig. 4.I(e), where this algorithm has generated three cluster centers for the three classes of the distribution. As can be seen in Fig. 4.I(f), the final set of prototypes X' computed by CBP consists of the border instances in the updated X_B set along with the generated cluster centers. The overall operations of CBP are shown in Fig. 4.IV.

%Initialisation.

- Input user-defined datasets X, $\psi(X)$.
- Set threshold $\tau := 0.5$
- Set $X_B := \varnothing$

Stage 1: %Noise Filtering.
Set X := ENN(X)

Stage 2: %Distinguishing between Border and Non-border instances.
For each pattern x in X
Calculate $\xi_i(x)$, $R_i(x)$, for $i=1,...,k_R$
Calculate $S(x)$
if $median(S(x)) < -\tau$
Set $X_B := X_B \cup \{x\}$
endif
endfor
Set $X_{NB} := X - X_B$

Stage 3: %Pruning Border Instances.
Set X'_B := Mutual_Nearest_Enemies(X_B)
Set $X'_B := X'_B \cup$ Filtered_Nearest_Enemies(X_B, X'_B)
Set $X_B := X'_B$

Stage 4: %Cluster means of Non-border instances.
Set X_{NB} := MSC(X_{NB})
Set X_{NB} := Merge(X_{NB})

Stage 5: %Output final set of prototypes X'.
Set X' := $X_B \cup X_{NB}$

Figure 4.IV- Overall sequencing of operations in the proposed CBP algorithm.

4.3 Experimental Analysis

The problem of the comparative evaluation of instance reduction algorithms is that their overall performance is not characterised only by the classification accuracy they exhibit, but also by the condensation ratio they achieve. Thus, there is an underlying multi-objective optimisation problem in the design and training procedure of such algorithms. These two objectives are conflicting and an improvement in one, often leads to the deterioration of the other. Consequently, there is a trade-off between classification accuracy and condensation ratio, in the sense that excluding noise, the larger the size of the prototype set retained for training, the more information it will hold to describe the underlying structure and as such, there is no solution that can maximise both objectives simultaneously.

The advantage the proposed CBP algorithm provides, is the capability to adjust its performance to the results required. By setting the value of threshold τ, which is used to determine class boundaries, the algorithm can force the output to vary with respect to

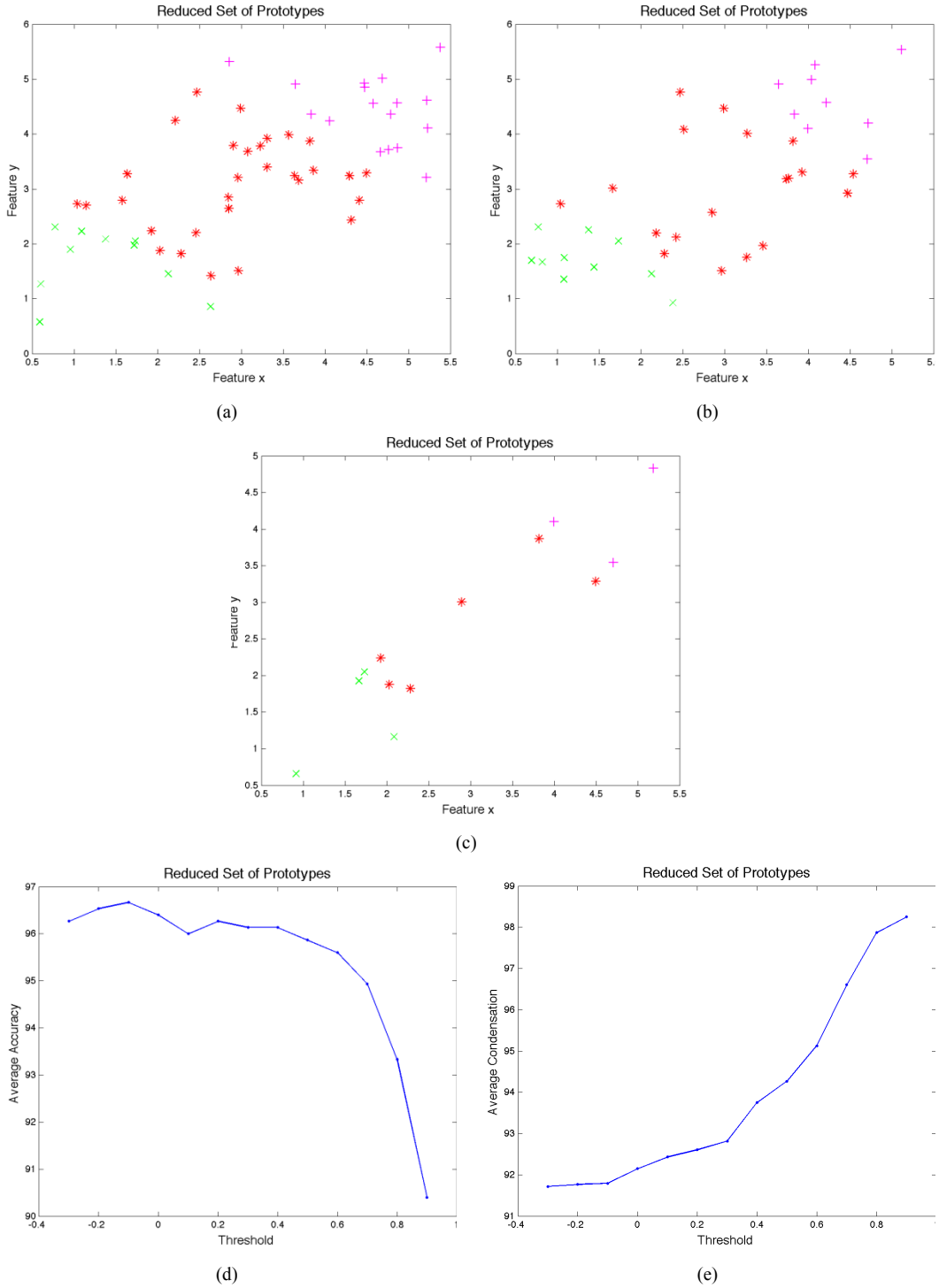


Figure 4.V - The dataset from Fig. 1 is used in this example. The final reduced set of instances obtained by CBP using **(a)** $\tau = 0.1$, **(b)** $\tau = 0.5$, **(c)** $\tau = 0.8$. **(d)** Accuracy decreases as the threshold increases. Fluctuations seen can occur because of noise effects. **(e)** Condensation is a monotonic function of the threshold value τ.

either classification accuracy or condensation ratio. An example of the effect these variations in threshold have on the performance of this method can be seen in Fig. 4.V. The experiment is performed on the two-dimensional three class synthetic dataset previously used in Fig. 4.I. It can be observed in Fig. 4.V(a)-(c) the set of prototypes obtained by CBP for threshold values $\tau = 0.1$, 0.5 and 0.8, respectively. High reduction

can be achieved but with an impact in decreased accuracy as shown in Fig. 4.V(d) and (e). In noise free only cases better classification results are obtained as the prototypes are reduced. According to Eq. (4.7), an increase in τ allows fewer instances to be retained by CBP, which corresponds to a higher condensation ratio. This is also shown in Fig. 4.V(e) where one can notice that the condensation of CBP is a monotonically increasing function of the threshold.

i. Numerical Results

To evaluate the proposed algorithm, this section provides a comparison between CBP and eight previously proposed instance reduction algorithms. These are the ICF [Bri02], HMN [Mar08], LIF [Mar09], CCIS [Mar10], ICPL [Lam02$_b$], TRKNN [Fay09], DROP3 which is the most efficient of the DROP variations introduced in [Wil00], and the editing technique ENN [Wil72]. All algorithms have been assessed on eighteen datasets from the UCI Machine Learning Repository [Bla98] (Table 4-I) and a comparative evaluation in terms of classification accuracy and condensation ratio performances is provided. Wilson's Editing ENN rule was tested using $k=3$, and the same value was also applied in all the algorithms that use it as a pre-processing step for noise filtering. TRKNN was implemented using a threshold value of $a=1.6$ as suggested by the author of [Fay09], while $k=1$ was selected for the CCIS algorithm. In all experiments, the Euclidean distance was used as the distance metric. When the reduced set X' was obtained, the 1-NN rule was used to classify the testing set because of its simplicity, and because this is the trend in all previous works. In each experiment five random permutations of 10-fold cross validation were used to assess the algorithm's performance. The dataset was divided randomly in ten partitions of which nine were used for training the algorithm and one for testing the resulting set of prototypes. Overall, for every single dataset 50 runs were performed and the average percentage of instances retained in the training set, as well as the average classification accuracy over these 50 runs are presented in Table 4-II. CBP receives three inputs, the set of instances X along with their corresponding class labels and a user-defined threshold value τ. As fine tuning τ with additional cross-validation, would require the user to define some balance or weights between the two objectives, to make the comparison direct with other algorithms, this threshold was fixed to a representative value of $\tau=0.5$ for all datasets.

Table 4-I

Details of the 18 datasets used in the experiments, including the total number of instances (the parenthesised summands are the instances per class), the dimensionality *d* and the number of classes *c*.

Dataset	**Total instances (per class)**	***d***	***C***
Diabetes (Pima)	768 (=500+268)	8	2
Ecoli	336 (=143+77+52+35+20+5+2+2)	7	8
Glass	214 (=163+51)	9	3
Haberman	306 (=225+81)	3	2
Heart	270 (=150+120)	13	2
Ionosphere	351 (=225+126)	34	2
Iris	150 (=50+50+50)	4	3
Letter ('A' to 'H')	2400 (=300+300+300+300+300+300+300+300)	16	8
Liver	345 (=200+145)	6	2
Monk	432 (=216+216)	6	2
Musk	476 (=269+207)	167	2
Pendigits	3498 (=363+364+364+336+364+335+336+364+336+336)	16	10
Sonar	208 (=111+97)	61	2
Transfusion	748 (=590+178)	4	2
Vehicle	846 (=220+220+220+186)	18	4
Vowel	990 (=90+90+90+90+90+90+90+90+90+90+90)	10	11
Wine	178 (=71+59+48)	13	3
Yeast	1484 (=244+429+463+44+35+51+163+30+20+5)	8	10

ii. Discussion

As already mentioned, instance reduction is a two-objective optimisation problem and a gain in one objective is accompanied by a worsening of the other. Therefore, Table 4-II presents both accuracy and condensation ratios for all competing algorithms and all experimented datasets. From the table, it is clear that ENN, HMN and LIF, which exhibit slightly higher average accuracies, are not very successful in optimising condensation. Overall, all algorithms manage to have similar accuracies (later on a statistical test for this is provided). In this multi-objective problem, one algorithm is deemed to be better than another if it improves both objectives, or at least one without (significant, in comparison to the objective it improves) deterioration of the other. The condensation ratios of ENN, HMN, LIF and TRKNN are much lower than the ratios of the other algorithms. ENN filters approximately only 20% of the original instances, HMN under 60%, while TRKNN and LIF do not manage to achieve condensation over 35%. For the remaining algorithms, condensation averaged over all datasets is over 69%, while they yield only a slight deterioration in classification accuracy; specifically, these are CCIS (with 69.89%

Table 4-II

Average accuracy (Acc) and condensation (Cond) percentages of the proposed CBP and 8 other compared algorithms over 18 datasets. These results are averaged over 50 runs.

	ENN		TRKNN		HMN		LIF		CCIS		DROP3		ICF		ICPL		CBP	
Datasets	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**
Diabetes	72.63	27.77	66.56	33.99	70.57	62.85	72.22	30.72	68.64	76.89	69.70	94.46	72.55	88.01	68.39	76.53	70.10	92.03
Ecoli	85.83	14.97	72.66	55.74	87.11	47.50	85.05	31.21	82.05	72.11	79.55	92.16	82.53	86.20	81.41	84.11	84.26	92.38
Glass	78.74	24.23	81.42	39.28	79.87	52.20	79.15	31.92	78.22	65.70	70.32	87.46	75.00	86.36	77.90	80.77	73.92	93.45
Haberman	74.69	25.80	64.31	44.38	70.67	61.51	73.53	30.15	64.20	87.48	66.60	80.90	72.36	92.17	67.90	75.33	74.17	91.96
Heart	66.07	33.83	58.67	26.46	65.85	71.52	66.37	38.72	63.93	72.86	63.78	88.11	65.41	82.17	60.89	69.82	65.48	89.16
Iono	83.60	17.01	84.04	67.20	89.40	56.92	86.73	9.73	86.37	79.53	82.68	55.70	81.14	96.07	88.03	86.74	83.09	95.73
Iris	96.67	3.41	93.33	45.21	95.60	43.08	95.20	13.21	95.47	81.67	95.73	71.10	95.47	60.41	93.47	90.52	94.53	94.09
Letter	93.25	6.10	94.68	16.48	91.35	48.99	87.16	48.49	92.04	54.42	84.24	82.38	89.03	75.75	91.13	87.63	85.79	89.54
Liver	66.10	33.79	60.41	23.75	63.68	69.92	66.72	38.49	64.53	72.29	64.98	66.84	61.80	84.32	60.01	70.89	63.65	88.39
Monk	77.09	13.34	77.98	1.21	71.42	82.95	75.86	41.98	71.27	81.11	73.19	80.31	79.30	48.40	71.56	64.68	79.38	57.12
Musk	79.42	16.65	84.38	27.69	83.41	54.65	83.00	23.54	80.00	66.84	71.28	78.49	77.46	78.81	84.66	83.48	79.12	89.38
Pendigit	99.02	1.04	99.30	26.35	98.65	41.88	98.39	31.18	98.94	56.68	96.54	85.18	97.38	89.72	97.93	95.92	94.48	98.38
Sonar	80.24	20.25	81.46	13.98	76.96	62.31	76.94	30.24	75.49	66.65	72.45	72.11	75.55	67.57	79.72	82.03	75.25	89.10
Transfusion	75.64	18.37	62.35	48.26	74.41	66.21	75.10	25.90	63.34	85.98	69.81	82.07	72.33	69.37	73.57	86.11	74.17	88.09
Vehicle	55.35	45.02	57.65	11.95	54.71	68.31	55.75	51.57	52.98	58.54	50.93	90.10	54.87	81.68	54.39	67.96	53.69	89.76
Vowel	92.79	4.44	98.24	9.38	91.82	45.97	70.93	67.41	91.82	46.76	90.69	65.41	90.42	43.38	93.47	82.90	88.65	83.17
Wine	94.92	3.74	94.62	26.39	95.97	48.13	96.44	25.60	95.71	67.94	92.93	74.80	91.47	87.11	93.39	89.43	95.43	95.65
Yeast	56.91	45.86	49.97	22.28	56.76	64.03	58.08	51.81	53.32	69.97	56.18	82.86	54.51	88.10	49.00	65.57	51.98	92.35
Average	**79.34**	**20.03**	**77.38**	**30.24**	**78.47**	**57.69**	**77.37**	**31.31**	**77.13**	**69.89**	**75.09**	**79.47**	**77.14**	**78.09**	**77.05**	**80.02**	**77.09**	**88.97**

Table 4-III

Comparison of CBP against all other algorithms, with accuracy (Acc) and condensation (Cond) shown in the rows. The **PI** columns correspond to the percentage improvement (positive) or percentage deterioration (negative) score, calculated as: (*CBP_score – other_score) / other_score × 100*), where the scores are taken to be the average classification or condensation scores across all datasets (last row of Table II). The columns marked as **p** correspond to the p-values of the Wilcoxon sign-rank test at 0.01 significance, with the null hypothesis that the scores distributions for all individual datasets have equal medians.

	ENN		TRKNN		HMN		LIF		CCIS		DROP3		ICF		ICPL	
	PI(%)	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***
Acc	-2.84	0.00549	-0.37	0.93585	-1.76	0.01001	-0.36	0.04421	-0.05	0.77818	+2.66	0.02688	-0.06	0.87212	+0.05	0.71722
Cond	+344.18	0.00013	+194.21	0.00013	+54.22	0.00029	+184.16	0.00013	+27.31	0.00084	+11.95	0.00430	+13.93	0.00021	+11.18	0.00062

condensation), DROP3 (79.47%), ICF (78.09%) and ICPL (80.02%). Of the latter group, ICPL possesses the best mix of average classification accuracy (77.05%) and condensation (80.02%). Regarding the proposed CBP, it manages the best condensation ratio at 89.46%, while it maintains an accuracy of 77.09% similar to other algorithms. In fact, CBP yields the highest condensation ratio in 13 out of the 18 datasets, and all condensation ratios (apart from the Monk dataset) are over 83%.

The above observations on the comparison of the different methods are corroborated by Table 4-III, which presents the relative percentage improvement (or deterioration) of CBP over each one of the competing methods for the averaged accuracy and condensation ratios. The condensation row of the table shows the PI values are all positive, which means that the CBP is better than all other algorithms in terms of condensation. Specifically, it is better than the second best algorithm (ICPL) by 11.18% and the third best (DROP3) by 11.95% and the fourth best (ICF) by 13.93%. Comparing the accuracy of the CBP with the others, one can observe a 0.05% improvement of accuracy compared to ICPL, along with an improvement of 2.66% percentage over DROP3, while a 0.06% percentage worsening is observed compared to ICF. Nevertheless, as it was stated earlier the algorithms have similar accuracy, so the comparison relies on the condensation objective. In order to numerically quantify this, and further test the statistical significance of these findings, a nonparametric two-sided statistical test is used, the Wilcoxon sign-rank test conducted at a *1%* significance level.

The improvement of CBP over all other algorithms in terms of condensation is seen by the eight very small ($<<0.01$) p-values in the second row of Table 4-III. On the contrary, the first row of the table shows large p-values (>0.01) which means that the null hypotheses of the algorithms yielding similar accuracies are not rejected at the *1*% significance level (apart from ENN, which is better than CBP in accuracy by 2.56%, but 367% worse in condensation). This large values mean that the proposed algorithm is statistically comparable to the other methods, with no significant differences. In this case, the performance is evaluated based on the statistical significance of condensation improvement of CBP.

CBP has the capacity of directing its performance towards either classification accuracy or condensation ratio by choosing the threshold value τ accordingly. Another way to view the threshold-based performance of CBP is to plot the accuracy versus condensation curve for varying values of τ. In such graphs, a condensing algorithm can be characterised better than the rest if it lies closer to the top-right corner, which corresponds to the ideal case of high values of both objectives. Fig. 4.VI shows the behaviour of CBP as the τ parameter that trades accuracy for condensation varies. Four average performing datasets (Diabetes, Heart, Iris and Liver) are examined as τ ranges from -0.7 to 1.0 in steps of 0.1 for 50 runs. As can be seen from most CBP curve

positions in the plots, there are values of τ for which it can perform mostly better than the other algorithms depending on the objective of importance as considered by the user. Most curve portions lie towards the top-right corner, though it is impossible for any algorithm to outperform all others for all parameter values or achieve perfectly monotonic and smooth balancing of the two objectives due to noise and dataset density.

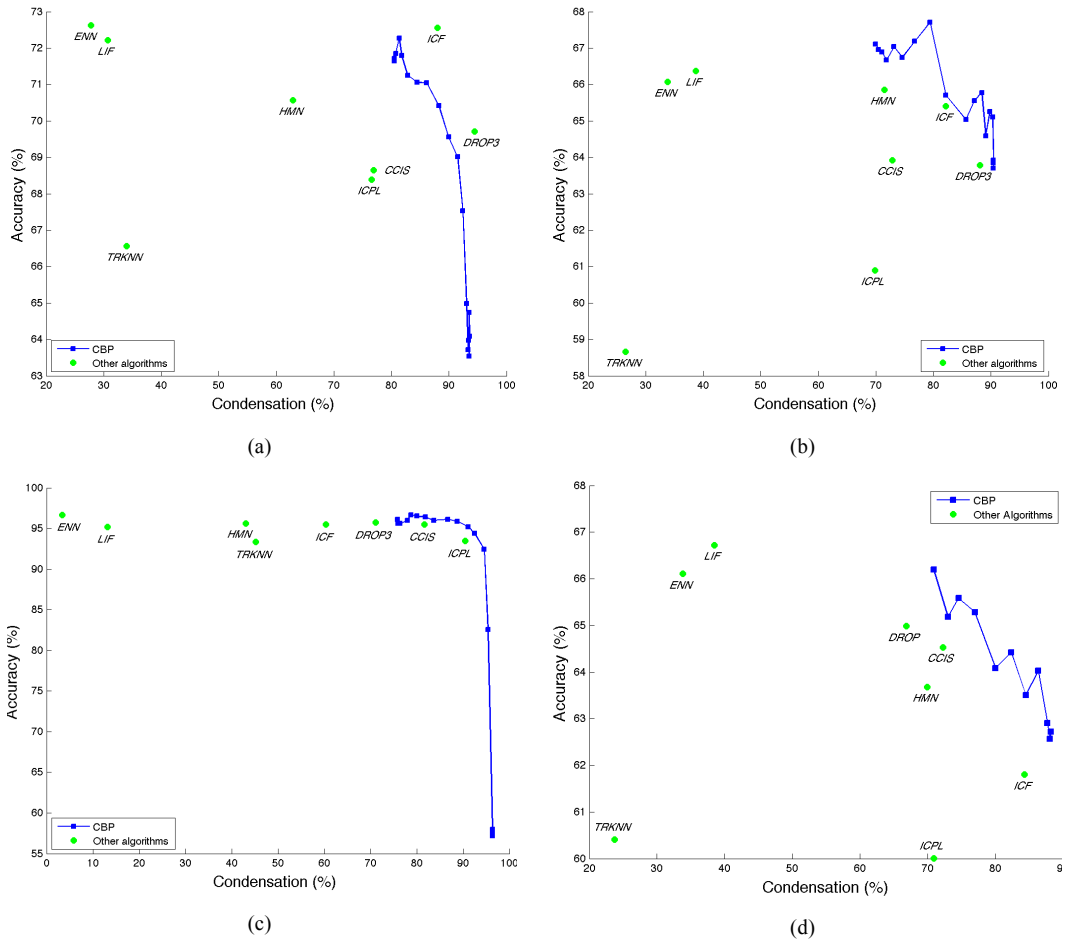


Figure 4.VI- Accuracy vs. condensation graphs (y-axis scale adapted for visibility) of the performance of other algorithms (dots) and CBP (curves) for different values of τ varying from -0.7 to 1.0 in increments of 0.1, for the datasets: **(a)** Diabetes, **(b)** Heart, **(c)** Iris, **(d)** Liver.

Examining the execution speed of CBP, shown in Table 4.IV, CBP does not seem to be as fast as TRKNN, HMN, LIF and CCIS, but these algorithms achieve as discussed earlier very low condensation ratios. However, CBP needs considerably lower computation time than ICPL, the second best algorithm in terms of condensation. It is also faster than the third and fourth best algorithms, these are DROP3 and ICF, respectively. Because the proposed algorithm is an iterative process that involves abstraction of instances, it displays higher time complexity than some of the compared algorithms. CBP, DROP3 and ICF use ENN as a noise filtering stage, which is a very computationally intensive pre-processing stage as seen in the table below.

Table 4-IV

Average computation times of all algorithms (measured in seconds per run) and overall weighted average times (computed using the number of instances of each dataset to weight the execution times of each method and make the comparison more objective).

Dataset	**ENN**	**TRKNN**	**HMN**	**LIF**	**CCIS**	**DROP3**	**ICF**	**ICPL**	**CBP**
Diabetes	81.0	0.3	0.2	0.3	5.2	83.5	92.4	394.4	85.4
Ecoli	7.1	0.1	0.1	0.1	0.7	8.2	7.9	50.6	7.5
Glass	1.3	0.04	0.04	0.1	0.5	1.5	1.5	19.2	1.5
Haberman	3.6	0.1	0.1	0.1	0.8	4.1	4.4	38.9	4.2
Heart	1.9	0.1	0.04	0.1	0.8	2.2	2.1	31.7	2.2
Ionosphere	6.9	0.1	0.1	0.1	1.2	7.5	7.0	55.3	7.5
Iris	0.5	0.03	0.03	0.04	0.4	0.8	0.6	8.3	0.7
Letter	5018.9	2.5	2.5	2.6	91.0	5439.3	5712.4	9426.6	5354.9
Liver	4.4	0.1	0.1	0.1	1.0	5.0	4.9	54.0	4.6
Monk	18.4	0.1	0.1	0.1	1.6	21.2	19.4	127.2	18.8
Musk	25.1	0.4	0.2	0.4	2.5	26.5	26.1	200.0	26.2
Pendigits	11466.3	6.1	6.3	7.6	306.1	11587.3	13212.0	17721.4	11674.7
Sonar	1.2	0.1	0.04	0.1	0.6	1.3	1.3	24.3	1.4
Transfusion	102.9	0.2	0.2	0.3	5.1	108.5	114.8	329.9	111.9
Vehicle	30.1	0.3	0.2	0.2	3.7	31.5	30.4	287.4	34.3
Vowel	301.8	0.5	0.7	0.6	8.7	316.1	305.3	704.0	308.4
Wine	0.7	0.04	0.03	0.05	0.4	0.7	0.8	12.2	1.1
Yeast	426.5	0.9	1.1	1.0	22.7	483.1	469.4	1636.2	433.1
Weighted average	**3805.7**	**2.1**	**2.2**	**2.6**	**96.1**	**4065.6**	**4366.9**	**6340.7**	**3917.6**

The response of CBP to high dimensionalities was also tested and whether the proposed heuristic for removal of non border instances and the other stages of the algorithm are sensitive to large number of features. The chart in Fig. 4.VII depicts dimensionality and accuracy ratios for all datasets. Datasets with large number of features, such as Ionosphere, Musk and Sonar do not show particular bias in terms of accuracy, and the pearson correlation coefficient ρ=*-0.046* numerically verifies this observation. The value is small, very close to zero, indicating no correlation between classification error (accuracy) and dimensionality.

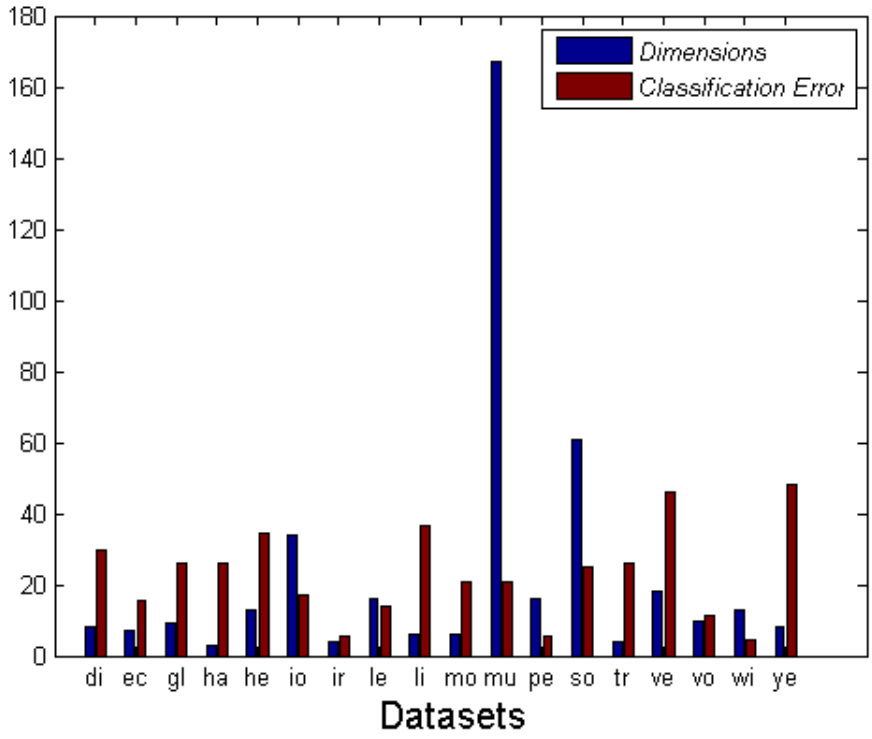


Figure 4.VII- Number of features and average CBP classification errors for all datasets (each denoted by the first two letters of the dataset name).

4.4 Conclusions

This chapter introduced a novel instance reduction method, the CBP algorithm, which employs the technique of instance selection and a simple but powerful heuristic together with the concepts of multi-level reachable sets and nearest enemy pairs, to determine the geometric structure of patterns around every sample in order to proceed with the removal of redundant instances. Its performance has been examined on eighteen datasets and compared the obtained results to eight instance reduction algorithms that were implemented. CBP yields the best condensation ratio for most datasets without compromising significant accuracy, while maintaining competitive execution times.

As already mentioned, instance selection methods display higher accuracy while instance abstraction algorithms exhibit better condensation ratios. The CBP algorithm is a hybrid method that employs both selection and generation of instances. As a result, it achieves very high reduction rates, due to the abstraction component, without compromising much accuracy. Accuracy is comparable not only to other abstraction techniques but also to various selection methods.

CHAPTER 5:

SPECTRAL ORDERING

5.1 Introduction

This chapter presents the effort to devise some sort of ordering in the training set in order to facilitate the reduction process in instance-based learning. This chapter discusses the concepts of Seriation and Spectral Graph Theory, which are the two techniques employed to sort instances in such order that structural information of the underlying distributions are unfold. Firstly, the concept of data seriation is analyzed (section 5.2) and a prototype condensing approach Instance Seriation for Prototype Abstraction (ISPA) [Nik10] that generates a new set of prototypes, is introduced in the following section. This method employs seriation to order the training set, and instances are then merged to create new prototypes based on their class labels. The results of ISPA, which is compared against other pruning algorithms, are discussed in another subsection. Section 5.3 presents the core contribution of this chapter. It introduces a novel framework that employs spectral graph theory to partition the dataset to border and internal instances. The underlying graph is based on a similarity matrix constructed by comparing characterizations of the original instances using a set of representative features. These features are based on concepts such as cosine score [Nik11], reachable and coverage sets [Bri02] and typicality [Zha92]. The proposed method, Spectral Instance Reduction (SIR) [Nik12] is highly accurate and direct, as can be seen by the experiments performed in section 5.3.ii. This subsection includes qualitative and quantitative evaluations that show that the proposed method is capable of effectively locating border instances, as well as a comparison of SIR with other condensing techniques in terms of classification accuracy, data condensation and time. Section 5.3.iii concludes the chapter.

5.2 Sequence Dating

The process of ordering and ranking data according to a dissimilarity function in order to unfold some underlying structural sequence is called sequence dating (or seriation). Seriation was initially used in the field of archaeology in the late nineteenth century by Petrie to date various archaeological findings. It was based on the idea of objects changing with time. Robinson in [Rob51] introduced a method for chronologically ordering archaeological deposits and Kendall in [Ken69] and [Ken71] recognized the mathematical model behind "sequence dating" of patterns. He used matrices of 0's and 1's to solve the problem of re-arrangement.

Since these original works sequence dating has been widely researched and evolved [Hah08, Man08]. Nowadays, sequence dating is not only used in archaeology but has been also applied to various other science fields such as ecology, machine learning and data mining. Bezdek introduced in 2002 a Tool for Visual Assessment of Cluster Tendency (VAT) [Bez02] that performs data seriation to identify the different clusters in the training set. In the VAT algorithm, the dissimilarity matrix is suitably re-ordered according to the pair wise dissimilarity information between instances. Further extensions of this work have been proposed in [Hub05, Bez07].

5.3 Seriation for Instance Abstraction

In this section, given an initial set of samples $X = \left\{ x \in \Re^d \right\}$ of n d-dimensional instances, where each sample is associated with a unique class label $\psi(x) \in L = \left\{ l_i, ..., l_c \right\}$, a simple framework is presented that addresses the problem of instance condensation. ISPA is a prototype abstraction method that orders instances based on the $n \times n$ dissimilarity matrix, and seeks for a highly reduced set of m new vectors (with $m << n$) that can provide the lowest possible error rate.

i. Proposed Framework

In general, algorithms suffer from the presence of noisy instances. Therefore, to make this algorithm noise tolerant Wilson's editing rule is applied, ENN, as a preprocessing step of ISPA. ENN filters the original training set by discarding all instances that are misclassified by their k nearest neighbours. Hence, it affects only instances close to class borders and retains internal samples intact. Removal of harmful

samples effectively smoothes the decision boundaries between classes and ensures that the performance of ISPA is not largely degraded by noise. ENN is widely used as a filter in the latest condensation algorithms including Hit Miss Networks (HMN) [Mar08], DROP [Wil00] and ICF [Bri02].

The next stage of this algorithm involves the extraction of the structural information of the training set. The Euclidean norm is used as the dissimilarity function in order to create an *nxn* dissimilarity matrix *D*. This matrix shows the relation of an instance to every other one in *X*. In order to transform this matrix in an ordered one that reveals a pattern the VAT algorithm is applied, which performs data seriation. A minimum spanning tree links all vertices together by starting at the most isolated instance. So, the pair of vectors with the highest dissimilarity is determined and one of them is chosen as the initial one. An iterative process then takes place as instances are sorted according to their similarity to the later one processed. Hence, no computation takes place, but only reordering of the rows and columns of *D*. The new relational matrix *R* is a transformation of *D*:

$$R = P'DP \tag{5.1}$$

where *P* is the permutation matrix. The set of instances and the class labels are also ordered according to *P* so that:

$$X_R = P'X \tag{5.2}$$

and

$$T = P'L \tag{5.3}$$

It is obvious that the linkage between consecutive instances is very strong. Therefore, one can assume that consecutive instances lie very close to each other on the vector space. This procedure determines the clusters of the distribution as illustrated in Fig. 5.I(a) – (d) for a synthetic example.

The majority of the information needed to accurately describe a set of instances is held by samples close to the class boundaries, while internal samples hold excessive information and can be regarded as redundant instances. Bearing this in mind, ISPA proceeds with a merging technique that favors instances close to the decision surface. The new relational matrix *R* is used to determine the nearest enemy of every sample in the reordered set. Then, an iterative process checks for every sample x_i in X_R its next consecutive sample x_{i+1}, and as long as they share the same class label they are merged. The resulting prototype p_j is the weighted mean of its two 'ancestors' and belongs to the same class. So the new prototype is generated in the following way:

$$p_j = \frac{w_i x_i + w_{i+1} x_{i+1}}{w_i + w_{i+1}} \tag{5.4}$$

where w_i and w_{i+1} are the gaussian weights of x_i and x_{i+1} respectively and are defined as:

$$w_i = \exp\left(\frac{-r_i^2}{2\sigma}\right) \tag{5.5}$$

where r is the distance to the nearest enemy and σ is the smoothing parameter. Defining a $1xn^2$ vector v with the ordered distances, the smoothing parameter is computed as:

$$\sigma = v_l : l = \lfloor 0.5n \rfloor \tag{5.6}$$

From Eq. 5.5 it is obvious that the closer an instance to the opposite class is, the larger the assigned weight will be. Hence, the effect samples near class boundaries have on the resulting prototype is significantly higher than the one internal instances have. As a result, p is moved towards the decision surface, which was the initial goal. The process terminates when no more prototypes can be merged.

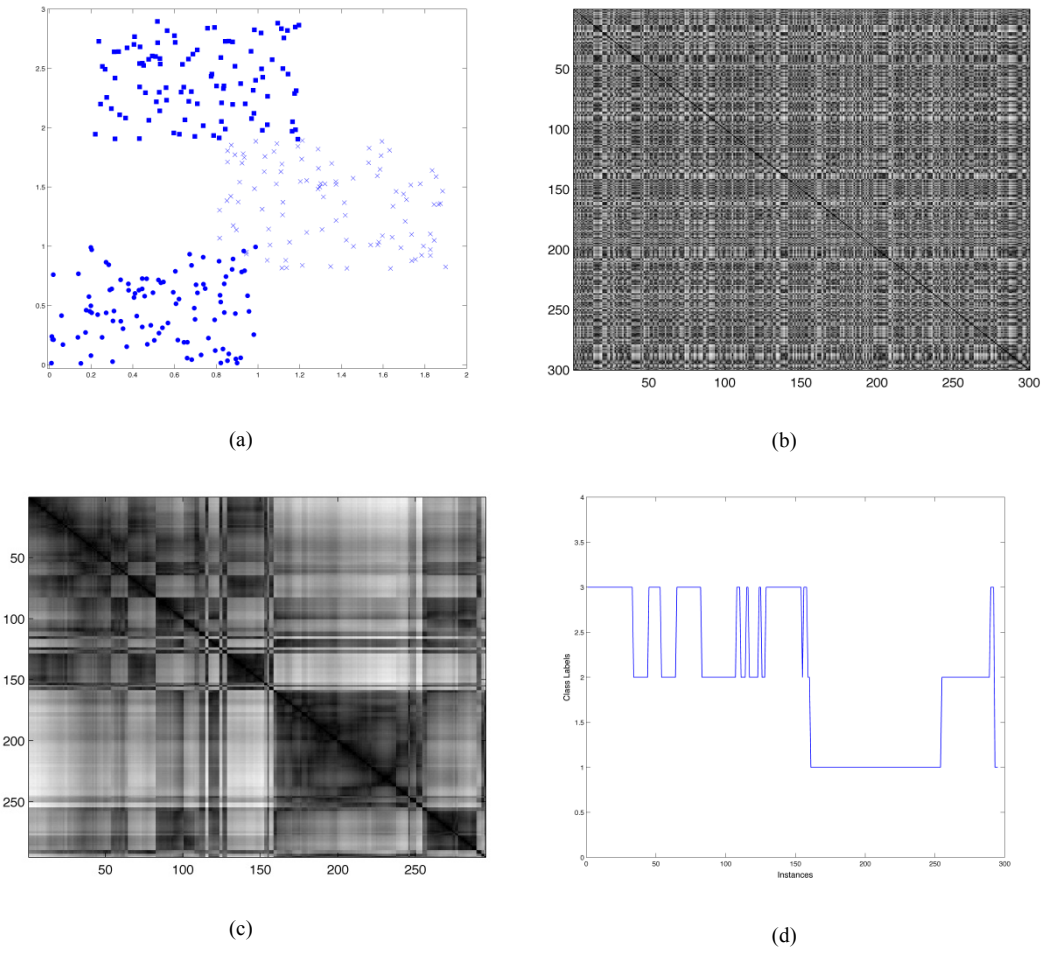


Figure 5.I- (a) A synthetic 2-dimensional dataset of 120 samples. **(b)** Image of the dissimilarity matrix D of the randomly distributed samples. **(c)** Image of the reordered matrix R where the clusters are easily detectable. **(d)** Class labels T of all instances of the reordered set X

ii. Experimental Analysis

The performance of an instance reduction algorithm is a two-objective optimization problem, as it is characterized by two basic outputs, the classification accuracy and the condensation ratio that it exhibits. In order to obtain a wide and

thorough evaluation of the proposed algorithm, ISPA, a comparison with three other pruning methods is made: ICF, which is a selective condensation algorithm, ICPL, which like ISPA, is an abstraction algorithm and HMN that is a filtering method.

a. Numerical Results

In order to perform the comparative evaluation of the four reduction techniques, they have been tested on 10 datasets from the UCI Machine Learning Repository [Bla98]. Wilson's editing rule that is used as a noise filter in ICF and ISPA is applied with a value of k_e=3. In all experiments the Euclidean distance was used as the distance metric and when the reduced vector set was obtained the 1-NN rule was applied for classification. In the experiments 10-fold cross validation with five permutations has been used to assess the algorithms' performance. Hence, 50 runs for every dataset are executed. Each dataset is randomly divided in ten partitions, nine of which are used to train the algorithm and one to test the resulting set of prototypes. In Table 5-I the average percentage of instances retained in the training set and the average classification accuracy over the 50 runs for each tested algorithm are presented. It should be mentioned that the proposed model has no variable inputs apart from the initial set of instances X and their assigned class labels.

b. Discussion

From the results in Table 5-I it is obvious that for the specific datasets the proposed method in terms of quality and precision operates at the same high level as the rest of the algorithms. There is a trade-off between classification accuracy and condensation ratio. In the absence of noise, the larger the size of the final set the better it will describe the underlying distribution. Therefore, it is expected that filtering methods that retain a large number of instances, such as HMN, will exhibit the highest accuracies. In Table I it can be see that although HMN achieved the best classification accuracy with an average of approximately 81.5% it filtered only 60% of the initial training set, which is considerably smaller than the rest of the methods. On the other hand, ICF, one of the best performing selective algorithms, presents a largely improved condensation ratio, as the size of the resulting instance set is nearly 23.5% of the initial one. But abstraction algorithms, ICPL and the proposed method ISPA, present the best mix of classification accuracy and reduction rate. Both methods clearly outperform ICF in terms of condensation with ICPL exhibiting slightly decreased classification accuracy. The small size of the final set is a consequence of the abstraction component of the algorithms. The new prototypes are generated with respect to the underlying distribution, in order to describe it as accurately as possible, so redundancy is tackled. Compared to HMN, abstraction algorithms make a small sacrifice, as HMN performs slightly better in terms of classification accuracy, in order to substantially reduce the size of the prototype set.

Table 5-I

Average Accuracies and Average Condensation (measured in percentages (%)) of ISPA and other competing algorithms over 10 datasets

	ISPA		**ICF**		**ICPL**		**HMN**	
Dataset	Acc	Size	Acc	Size	Acc	Size	Acc	Size
Diabetes	71.26	8.46	72.52	11.7	71.71	15.49	70.09	37.12
Ecoli	83.85	12.25	82.66	13.95	84.29	6.29	87.52	52.9
Glass	74.72	11.12	74.56	13.21	73.97	14.51	79.83	48.67
Heart	64.57	7.59	65.31	17.56	61.85	30.15	66.3	28.7
Ionosphere	82.74	4.92	81.57	4.16	86.51	3.42	90.21	42.96
Iris	94.89	6.54	94.89	39.82	92.67	6.37	96.22	55.98
Liver	64.79	12.99	65.16	15.09	61.32	31.87	65.33	29.95
Monk	79.23	16.38	79.79	52.93	70.68	25.94	71.31	16.77
Wine	96.27	11.29	91.83	12.67	96.67	6.13	96.23	51.31
Zoo	90.42	7.22	88.52	53.4	92.7	15.09	92.03	41.39
Average	**80.27**	**9.88**	**79.68**	**23.45**	**79.24**	**15.53**	**81.51**	**40.58**

In Table 5-I it is obvious that ICPL and ISPA have an equivalent performance in nearly all datasets tested. There is a significant difference, in terms of accuracy, in only three datasets, Ionosphere, where ICPL exhibits better results than ISPA, Heart and Monk, where ISPA is clearly outperforming ICPL. Both methods demonstrate similar classification accuracies, but in terms of average reduction rate ISPA exhibits a significant advantage. The proposed algorithm presents the highest storage reduction in 6 out of the 10 datasets, and demonstrates the best condensation ratio between all the tested methods.

iii. Conclusions

In this work, the Instance Seriation for Prototype Abstraction algorithm has been proposed, which employs the concept of seriation to organize the original training set in a way that facilitates the generation of a new set of prototypes. ISPA is a simple instance-based learning algorithm for prototype reduction and was designed as an attempt to create some sort of ordering between instances to enhance their separability into beneficial and redundant ones. ISPA served as a draft of the final ordering scheme that is thoroughly presented in Section 5.4. But because it demonstrated competitive results only on a number of datasets, more specifically the 10 datasets used in Table 5-I, a less detailed evaluation and analysis has been performed, compared to the other methods proposed in this thesis. Despite its simplicity and its narrow spectrum of applications (datasets), ISPA displayed results worth of mentioning. Therefore, a comparative evaluation of the

proposed method with three recently established reduction algorithms was performed. To obtain a global understanding of the precision of ISPA and of its capabilities it has been compared against a filtering technique, a condensing algorithm using selection of instances and an abstraction method that generates new prototypes. For the reduced set of datasets, as already mentioned, and between the four techniques used in the comparison (HMN, ICF, ICPL and ISPA) the proposed algorithm demonstrated the highest average storage reduction and quite competitive results in terms of classification accuracy.

5.4 Spectral Graph Optimization

In this section, a novel instance selection algorithm, SIR, has been proposed, that is based on spectral graph theory to robustly distinguish between border and internal instances. Firstly, a broad set of existing and new feature transformations of the data samples to border discriminating measures has been presented. This diverse set of border discriminating features (BDFs) that are based on concepts such as cosine score [Nik11], reachable and coverage sets [Bri02] and typicality [Zha92, Lam02$_a$], capture the local friend and enemy profiles of the samples. Secondly, by relying on a graph modeling approach and border feature similarities, the method makes use of graph-cut approximations to efficiently search for the two partitions of border and non-border samples. The underlying optimization is achieved through the eigendecomposition of the corresponding Laplacian matrix. Although the graph Laplacian has been previously used for instance reduction, it had no distinctive relation to spectral theory or graph-cut modeling as [Mar09] used the discrete Laplace operator for the between-class graph, acting on the degree function of the within-class graph, and thresholded the Laplacian score to retain the instances of interest.

i. Border Discriminating Features

Initially, to tackle the effect of noisy instances, a filtering component based on the Edited Nearest Neighbour (ENN) method [Wil72] has been pre-applied. In this way, instances that are misclassified by their 5 Nearest Neighbours (5NNs) are removed. This is a simple but fast modification of ENN, that performs only a single scan over the dataset. The objective of the remaining of the first stage is the design of features that characterize in a representative and compact manner the properties of each prototype x. These properties capture the information needed to build the geometric and statistical profiles of each prototype, in terms of its friend (same class) and enemy (other class) proximal instances, which in turn determine their capacity as border or non-border

samples. To achieve this, ten border discriminating features (BDFs), which are summarized in Table 5-I have been employed. Some of these have been used before in various instance reduction methods, but in a very different context and not as border features within the graph embedding formulation adopted by the proposed SIR. These features are defined and discussed as follows.

Table 5-II

The 10 Border Discriminating Features proposed along with their respective ranges

BDF formulation	Range
$f_1 = \frac{1}{n}\left(\sum_{y\in X-\{x\}}^{n} H(x,y) - \sum_{y\in X}^{n} M(x,y)\right)$	[-1,1)
$f_2(x) = \frac{1 - \frac{1}{\alpha_{\max}\lvert F(x)\rvert}\sum_{y\in F(x)}\lVert x-y\rVert_2}{1 - \frac{1}{\alpha_{\max}\lvert E(x)\rvert}\sum_{y\in E(x)}\lVert x-y\rVert_2}$	[0,∞)
$f_3(x) = \frac{\lvert R(x)\rvert}{\lvert C(x)\rvert}$	[0,n-1]
$f_4(x) = median\left[S(x)\right]$	[-1,1]
$f_{5,\ldots,10}(x) = \frac{\frac{1}{1+\lvert NF_k(x)\rvert}\sum_{y\in NF_k(x)}\lVert x-y\rVert_2}{\left\lvert \frac{1}{k}\sum_{y\in kNN(x)}\lVert x-y\rVert_2 - \frac{1}{1+\lvert NE_k(x)\rvert}\sum_{y\in NE_k(x)}\lVert x-y\rVert_2\right\rvert}$	[0,∞)

The first feature $f_1(x)$ is the scaled difference of hit (number of friends) and miss (number of enemies) degrees, defined similar to [Mar08], [Mar10]. Given a sample x, these degrees are defined using the binary functions:

$$H(x,y) = \begin{cases} 1 & y \in kNF(x) \vee x \in kNF(y) \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

$$M(x,y) = \begin{cases} 1 & y \in kNE(x) \vee x \in kNE(y) \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

where $kNF(x) \subset X - \{x\}$ is the set of k nearest friends of x, and $kNE(x)$ the set of k nearest enemies of x. In [Mar08] the parameter was set as $k=1$, while [Mar10] suggests the use of $k=1, 3$ or 5. For both Eqs. (5.7-5.8) and for all experiments, a fixed value of k=5 was empirically chosen to capture the local neighbourhood of each instance. Lower values of k were found to be sensitive to the sparsity of the data, while higher values did not improve the discriminatory ability of $f_1(x)$. This feature is important because high or low values of $f_1(x)$ correspond to instances lying in a region with high concentration of friend or enemy prototypes, respectively.

Another feature is the value of typicality $f_2(x)$ [Zha92]. It uses the ratio of the average similarity of x to all of its friends $F(x) = \{y \in X : \psi(y) = \psi(x) \wedge y \neq x\}$ over its enemies $E(x) = X - F(x) - \{x\}$ from the entire dataset X (a_{max} is the largest distance in the set). Normally, low values of typicality indicate internal samples, while high values correspond to class boundary prototypes.

The third feature $f_3(x)$ is based on the ratio of the cardinalities of two separate sets introduced in [Bri02]. One is the reachable set given by:

$$R(x) = \{y \in X : \psi(x) = \psi(y) \wedge \|x - y\|_2 \leq \|x - e\|_2\} \tag{5.9}$$

where $e \in 1NE(x)$. That is, $R(x)$ contains all friendly instances lying between x and its nearest enemy e. Intuitively, this means that instances near the class borders, and hence, in closer proximity to their enemies, display values of $|R(x)|$ lower than internal instances. The other set used in $f_3(x)$ is the coverage set defined as:

$$C(x) = \{y \in X : x \in R(y)\} \tag{5.10}$$

$R(x)$ holds all instances that can correctly classify x, while $C(x)$ contains all instances that x can solve. Therefore, redundant samples that lie away from class boundaries exhibit $f_3(x)$ values significantly larger than one, while as the values decrease the importance of the corresponding instances increases.

The cosine score $f_4(x)$ is also employed in a similar manner as introduced in [Nik11]. $S(x)$ is the set of cosine distances between the vector connecting x and other reachable friends, and the vector connecting x and the respective nearest enemy. This is applied to the set of all friend instances formed by the third reachable set; this set is defined as in Eq. (5.9), but with e being the third nearest enemy. Instances with a large number of friends lying in-between their enemies, demonstrate higher $f_4(x)$ values. On the other hand, this feature obtains strongly negative values for boundary prototypes, since friends lie behind them with respect to their enemies.

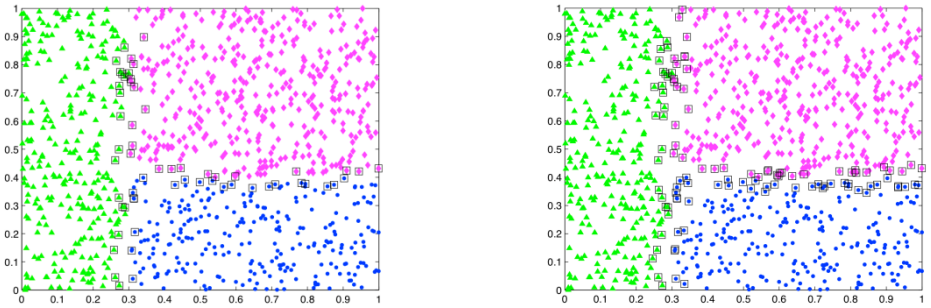
Finally, a new feature based on the average distances of subsets of nearest friends $NF_k(x)$ and enemies $NE_k(x)$ is introduced, and is defined as:

$$NF_k(x) = \{y \in X : \psi(y) = \psi(x), y \in kNN(x)\} \tag{5.11}$$

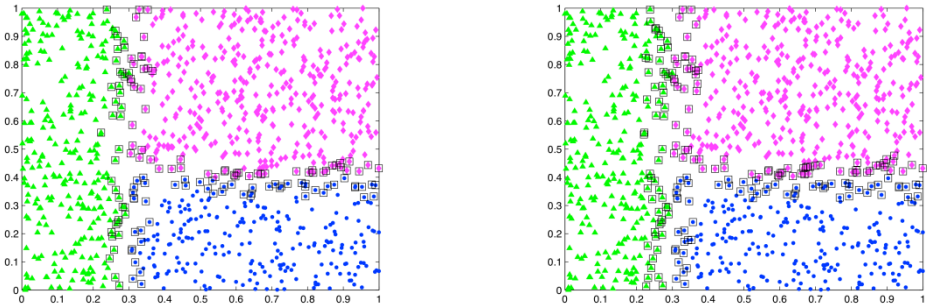
$$NE_k(x) = \{y \in X : \psi(y) \neq \psi(x), y \in kNN(x)\} \tag{5.12}$$

where $kNN(x) \subset X - \{x\}$ is the set of k nearest neighbours of x excluding itself. These subsets allow the examination of the local friend and enemy profiles of each instance x, since $NF_k(x) \cup NE_k(x) = kNN(x)$. Instances close to the class boundaries exhibit higher values for this feature, while lower values indicate internal prototypes. Similar concepts of homogeneous and heterogeneous neighbourhoods have also been used in [Wan07$_a$] for supervised learning. Six versions for this feature are employed, with the

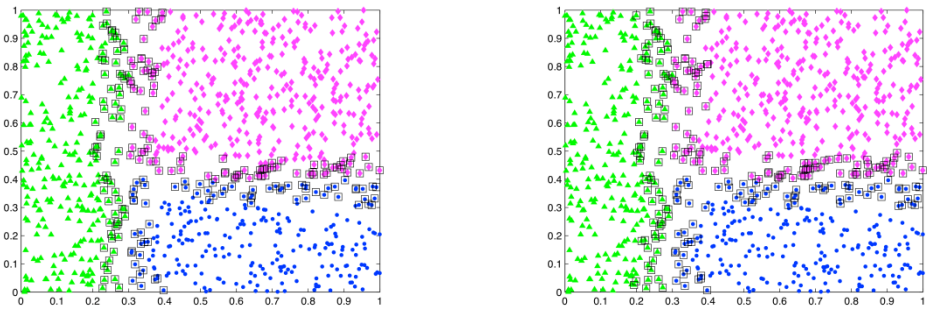
values of k=5, 10, 15, 20, 25, 30, and denote them correspondingly as $f_5(x)$, $f_6(x)$, …, $f_{10}(x)$. A set of multiple values is required in order to capture the friend versus enemy profiles at multiple gradually expanding local neighbourhoods around each instance. Compared to a single value, multiple ones are needed because different datasets exhibit varying data density and inter-class distance characteristics. This feature is quite robust in terms of discriminatory power, but no single value can be perfect for all datasets. As can be observed in Fig. 5-II, as k increases the class boundary becomes denser. For k=5, there is clear distinction in the thin line between border and non-border instances, but there is also some loss of border information. Larger values of k result in more instances being identified as borders, but they do that at the detriment of over-characterizing many instances as border ones.



(a) k=5 (b) k=10



(c) k=15 (d) k=20



(e) k=25 (f) k=30

Figure 5.II- (a)-(f) Experimental analysis of features f_5- f_{10} on a synthetic two-dimensional example. This dataset consists of 3 different classes with a total of 999 instances. Figures show the effect of each feature individually.

The specific values of *k* are used, because for values smaller and larger than 5 and 30 the feature was experimentally determined to be ineffective or insensitive, respectively. Also, the step of five between the different values was found reasonably robust and non-redundant, as smaller steps did not add any discriminatory power (e.g., there was no meaningful effect between *k=10* and *k=11*, or between *k=25* and *k=27*). On the other hand, larger steps work well but not for all datasets (e.g., if only k=*10, k=20,* and *k=30* are used, for some datasets border information at some scales will be lost). A different set of fixed *k* values did not prove beneficial, and as a result the final range and step for *k*, were experimentally selected to support the balance between finer and coarser neighbourhood grid. To conclude, for the wide range of datasets used for experimentation, the proposed method was not found to be sensitive to variations in the range and step values of *k*, because multiple levels for this BDF are used. It should also be mentioned that the above figure is not generated with SIR, but with direct thresholding of each of the proposed features, in order to study its behavior with regard to *k*. This example along with experimentation with each of the 10 features individually proves that each of the features is worth using in the BDF set proposed. It can be noted, that instead of using a fixed set of values for *k*, one could choose to adapt *k* using a direct formula or heuristic, or search for an optimal value using a separate validation set. Although this can potentially increase the performance, it can either be difficult to find a formula that works for all datasets, or very time consuming. Although initially experiments involved BDFs whose parameters can be adapted to a dataset's properties using training-validation or a cross-validation procedure, it was found to be extremely expensive and it made SIR completely impractical.

The final step of the first stage of SIR is to map each original sample $x \in X \subset \Re^d$ to a ten dimensional BDF vector $z \equiv z(x) = \left[f_1(x),...,f_{10}(x) \right]^T \in Z \subset \Re^{10}$. To make the newly generated features comparable, they are standardize using their means and variances.

ii. Border and Non-border Instance Partitioning

The objective of the second stage of SIR is to use the new representation *Z* of the original prototypes *X*, and partition them into two disjoint sets; that is, the border instances Z_B which will be retained, and non-border instances Z_{NB} which can be discarded, such that $Z = Z_B \cup Z_{NB}$. Various algorithms can be used to achieve this partitioning [For10]. In this work, for flexibility and efficiency, the dataset is modeled as a graph whose vertices correspond to samples in *Z*, and employ a balanced graph-cut modeling approach. This is solved using spectral theory [Spi04, Spi07] and graph embedding methodologies, which have been widely used for dimensionality reduction,

supervised learning and clustering [Yan07, Kok09, Ng02, Miu12, He04, Sau03, Row00, Zha09, Zha10$_b$, Xin02, Bel03, Xie11, Ngu11]. Spectral clustering in particular has become very popular in machine learning [Sun08, Bel02, Oze08, Mei01] and a detailed description and analysis of spectral decomposition can be found in [Lux07]. Specifically, it is aimed to minimize the following Min-Max Cut [Din01] problem defined as:

$$Mcut\left(Z_B, Z_{NB}\right) = \frac{cut\left(Z_B, Z_{NB}\right)}{cut\left(Z_B, Z_B\right)} + \frac{cut\left(Z_{NB}, Z_B\right)}{cut\left(Z_{NB}, Z_{NB}\right)} \tag{5.13}$$

where $cut\left(A,B\right) = \frac{1}{2}\sum_{(i,j)\in A\times B} W_{ij}$ is a symmetric graph cut weight score between two vertex subsets A and B. Minimizing Eq. (5.13) locates optimal graph partitions Z_B and Z_{NB}, such that edges within Z_B and edges within Z_{NB} have high similarities. At the same time all edges connecting Z_B and Z_{NB} (cuts) have high dissimilarities. Each weight W_{ij} is set to represent the similarity between the i^{th} and j^{th} graph vertices, or equivalently BDF vectors. Here, each weight is estimated using the Gaussian kernel $W_{ij} = \exp\left(-\left\|z_i - z_j\right\|_2^2 \Big/ \rho\right)$. To automatically adapt the kernel parameter ρ to different datasets, the mean μ is used and standard deviation σ of all the $n(n\text{-}1)/2$ Euclidean distances between elements in Z, and set it to $\rho = \left(\mu - \sigma\right)\big/2$.

A very efficient way of minimizing the balanced graph-cut problem in Eq. (5.13), is to use spectral relaxation procedures that approximate the original problem [Lux07]. Specifically, (5.13) can be solved by finding an optimal vector $q^* \in \Re^{n\times 1}$ from the following optimization:

$$q^* = \underset{\substack{q\in\Re^{n\times 1}\\ \|q\|=1}}{\arg\min}\left\{ q^T L q = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}\left(\frac{q_i}{\sqrt{D_{ii}}} - \frac{q_j}{\sqrt{D_{jj}}}\right)^2 \right\} \tag{5.14}$$

The matrix L is called the normalized graph Laplacian matrix, and is defined in terms of the $n\times n$ weight matrix W as:

$$L = D^{-1/2}\left(D - W\right)D^{-1/2} \tag{5.15}$$

where D is the $n\times n$ diagonal graph degree matrix, given by $D_{ii} = \sum_{j=1}^{n} W_{ij}$. The degree values D_{ii} and D_{jj}, are needed to balance the error contributions of the cut. Minimizing (5.14) produces a vector q^*, whose components q_i^* and q_j^* will be more numerically similar, when the original patterns x_i and x_j are more compatible in terms of their BDFs. According to the Ritz-Rayleigh theorem, the optimization in (5.14) can be efficiently solved through the eigendecomposition of the normalized Laplacian L, and taking q^* to be the eigenvector corresponding to the second smallest eigenvalue [Lux07].

The proposed spectral instance reduction algorithm is based on, firstly, forming the BDF set Z as described in Section 5.4.i. Then, the adjacency matrix W and degree matrix D are estimated, and the Laplacian matrix is formed using (5.15). Finally, q^* is obtained from the eigendecomposition of L. The approximate solution for the partitioning of Z in (5.12), is then found by thresholding the vector $D^{-1/2}q^*$. The optimal threshold value is simply set to the element within q^* that gives the smallest Min-Max Cut value [Din01].

Because from the above two partitions are obtained without any information about which is the border Z_B and which the redundant Z_{NB}, the following simple heuristic is applied. The already calculated BDF values of $f_4(x)$ are re-used and collect the set $T = \{z(x) \in Z : f_4(x) < \tau\}$. Then, a simple test is performed that selects Z_B as the border set if:

$$\frac{vol(Z_B \cap T)}{|Z_B|} > \frac{vol(Z_{NB} \cap T)}{|Z_{NB}|} \tag{5.16}$$

where $vol(A) \equiv \sum_{i \in A} D_{ii}$ is the sum of all weights attached to edges connected to vertices within some vertex subset A. Regarding the threshold value, it was experimentally found that any approximate value for τ that specifies roughly which of the two partitions is border and which not, is acceptable, and cannot affect the accuracy of the proposed method. Since τ is not too sensitive to small variations, the fixed value of τ=-0.5 is used as proposed in [Nik11].

iii. Experimental Analysis

a. *Synthetic Datasets*

Firstly, a qualitative demonstration of the capability of the proposed BDFs succeeded by the spectral partitioning is shown. The aim is to separate a given dataset into internal and border instances from whom the latter ones can be retained. Experiments with three two-dimensional synthetic datasets are shown in Fig. 5.III. All cases are relatively dense with distinct regions of internal and border patterns. The results obtained for these examples are based on thresholding the optimizing eigenvectors, as explained in 5.4.ii. The histograms of these eigenvectors are shown for each individual case in Figs. 5.II(b, d, f). The partitioning results are superimposed in Figs. 5.III(a, c, e), where boxes mark the border patterns. For all cases, it can be seen that the located border instances form concentrations near the actual boundaries of each distribution, neighbouring with the enemy classes. In order to qualitatively compare the performance of this algorithm to the other tested methods Fig. 5-IV is presented, which illustrates the results of the competing

methods on the three synthetic datasets. It can be observed that the function of ENN and LIF is to remove noisy instances, close to the decision surface. This is the reason of their low condensation ratios (especially ENN's which is the reason it has not been included in Table 5-II. It should also be mentioned that although HMN displays relatively very good results on real datasets as already mentioned in Chapter 4 and in the following section of numerical results, it displays rather poor condensation results on these synthetic 2D examples. On the other hand, TRKNN seems to be very efficient for these examples despite its poor performance on real datasets. The DROP3 algorithm although it displays efficient results on real datasets it demonstrates a very poor performance on synthetic data as observed in the figures below, since results are bad for the 3-class example, while borders identified for the cross example are very dense. On the contrary, the rest of the methods, CCIS, ICF, ICPL and CBP, verify their competence with accurately

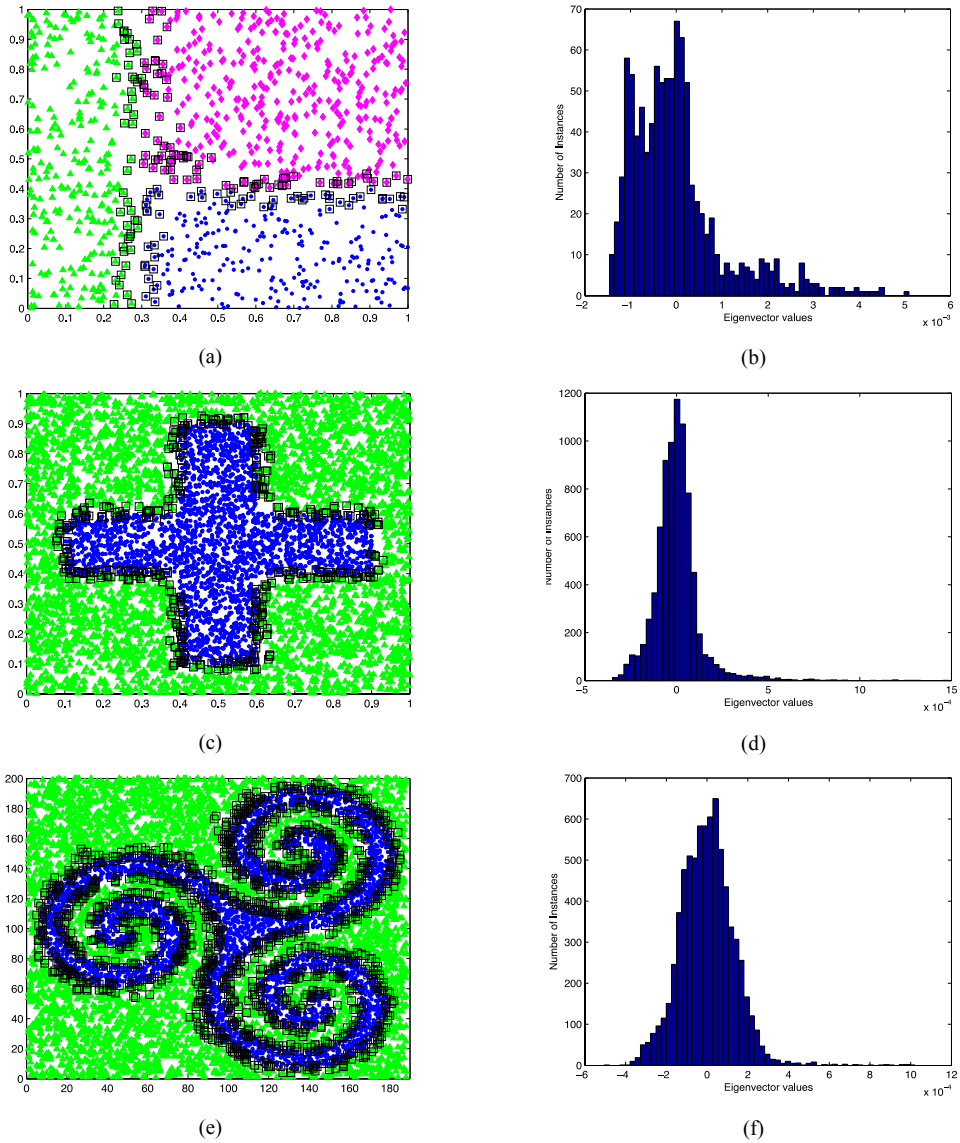
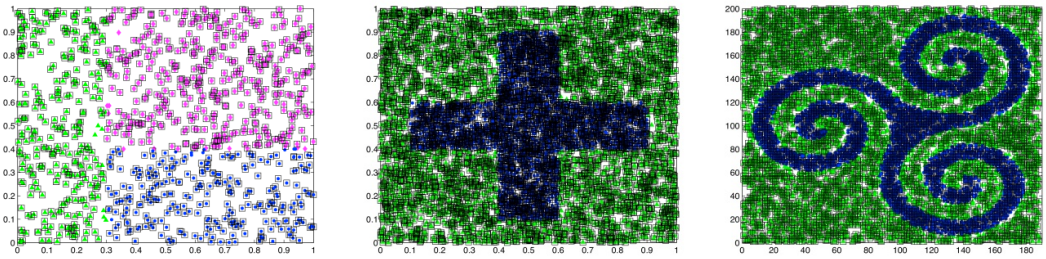


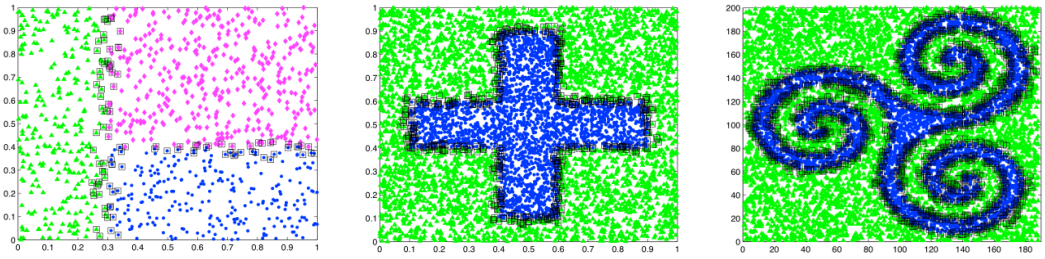
Figure 5.III- Experiments with three synthetic two-dimensional examples: (a) three-class rectangular shapes, (c) two-class cross and (e) two-class tri-spiral. In all cases, the patterns from different classes are marked with symbols "▲", "●", "◆", and the retained border instances are enclosed by "□". The second column shows the histograms of the optimizing eigenvectors, corresponding to the examples of the first column. The border

patterns were located by thresholding the scaled optimal eigenvectors, with the corresponding threshold values of: (b) 0.6×10^{-3}, (d) 1.5×10^{-4}, and (f) 10^{-4}.

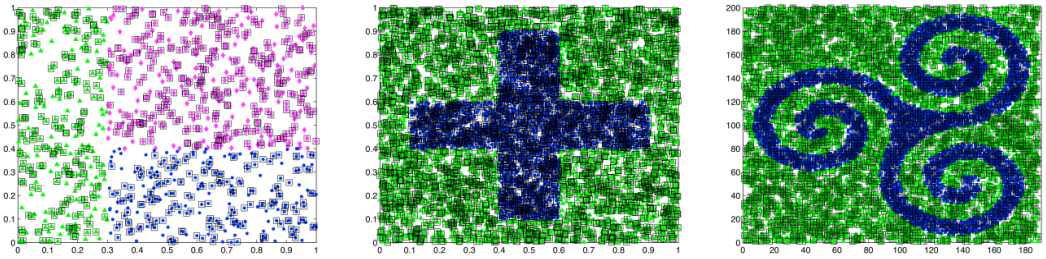
distinguishing the decision surface. It should be mentioned that CCIS seems to have a better condensation ratio on the synthetic examples compared to the real datasets used for the numerical experiments. This different behavior for HMN, TRKNN and DROP3 in terms of 2D and also the real high dimensional datasets arises from the fact that most of these algorithms were designed for multiple dimensions. Also, the increased sample density and the ratio of density over dimensionality are different for the 2D sets above than in the real datasets.



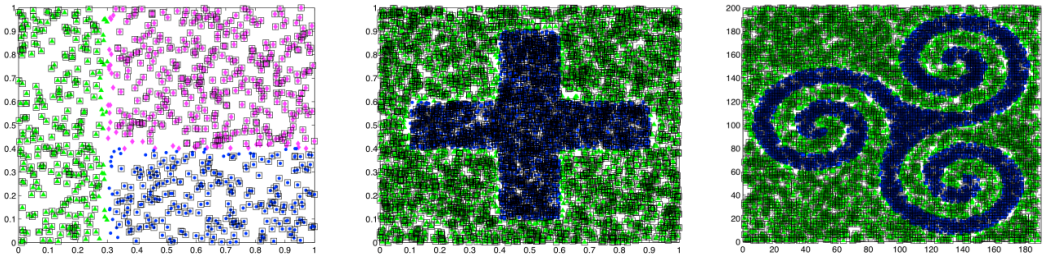
ENN *ENN* *ENN*



TRKNN *TRKNN* *TRKNN*



HMN *HMN* *HMN*



LIF *LIF* *LIF*

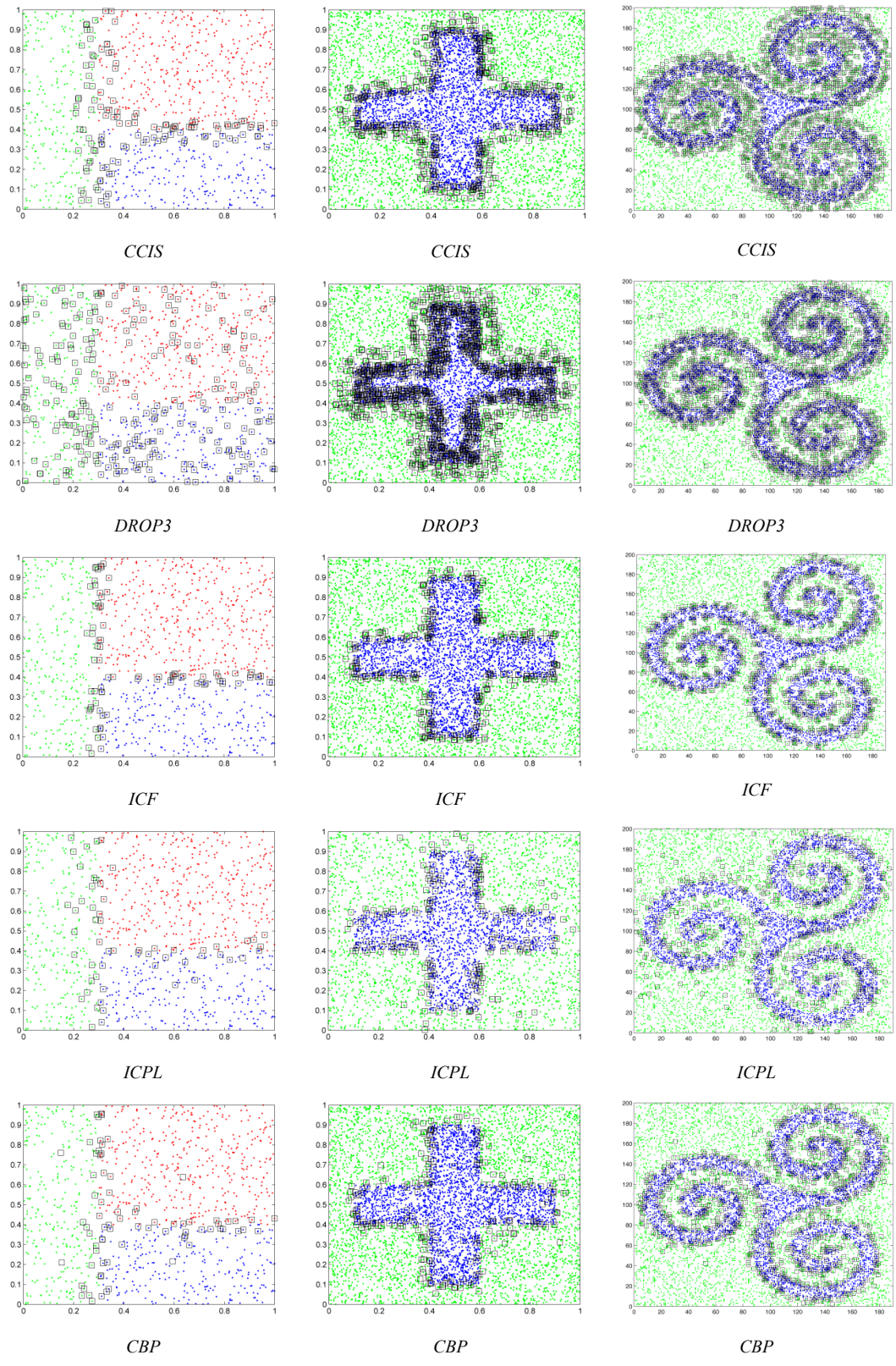


Figure 5.IV- Qualitatetive comparison of all tested methods on the three 2-dimensional synthetic examples that have been already described. Each row of the figure corresponds to one of the tested methods.

Finally, in order to further demonstrate the capability of the SIR algorithm to address rather complicated issues another synthetic dataset example is included. As can be observed in Fig. 5.V, where SIR is tested on a very small class surrounded by a larger

one, the proposed algorithm performs quite robustly, despite how complex the problem is.

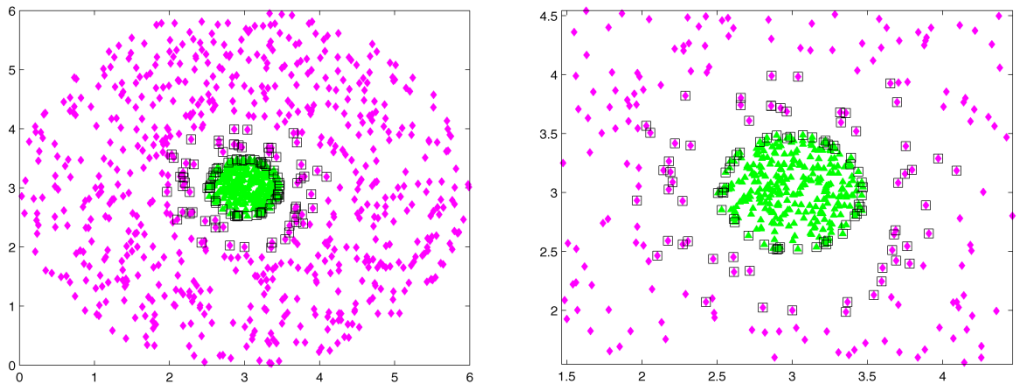


Figure 5.V- A synthetic 2-class dataset of a large class surrounding a much smaller one.**(a)** SIR accurately identifies border points. **(b)** A zoomed version of the same example to illustrate how robust SIR is.

b. Numerical Results

In this section, the performance of the proposed SIR algorithm is quantitatively evaluated, and SIR is compared it in terms of classification accuracy and condensation ratio against eight other well-known instance selection algorithms. These are the Template Reduction for *k*-NN (TRKNN) [Fay09], Hit Miss Networks (HMN) [Mar08], Laplacian Instance Filtering (LIF) [Mar09], Class Conditional Instance Selection (CCIS) [Mar10], DROP3 by Wilson [Wil00], Instance Case Filtering [Bri02], the abstraction method ICPL [Lam02$_b$] and the previously described CBP algorithm [Nik10], that was introduced in chapter 4. All tests are performed over eighteen real datasets from the UCI Machine Learning Repository [Bla98], with characteristics shown in Table 4-I.

Some of the tested algorithms require the setting of user-defined parameters. These were set to their best performing values of α=1.6 for TRKNN, $k=1$ for the CCIS algorithm and $\tau=0.5$ for CBP, and remained fixed in all experiments. For all algorithms, the simple 1NN rule is used to measure the classification accuracy of the retained prototypes, for reasons of simplicity and because this has been the trend in all previous works. For distance measurements in all algorithms the Euclidean distance norm is employed.

To perform the experiments, each dataset was randomly divided in ten folds, nine of which were used for training each algorithm and performing the prototype reduction, and one fold for testing the classification accuracy with the 1NN rule and the reduced prototypes Z_B from the nine folds. In total, five random permutations of a 10-fold cross-validation were used, giving a total of fifty runs per dataset from which the final measures of accuracy and condensation were averaged for each dataset. Numerical results for the

eight competing algorithms and the proposed SIR are aggregated in Table 5-III, together with averages across all datasets.

It should be noted, that the evaluation of performance of an instance reduction algorithm is typically a bi-objective optimization problem, where the user must take into account not only the classification accuracy remaining after prototype removal, but also its condensation capability. These two objectives are conflicting, since when too many instances are removed, the remaining prototypes are unable to sustain class boundary definition and hence, classification accuracy drops. It can be said that one instance reduction algorithm is better than another, when it is better in both objectives, or it is equivalent in one objective and better in the second. But, for real-world applications, the final judgment often relies on what the application prioritizes. If, for example, classification accuracy is critical, then a very bad performance in condensation can be traded off against a slightly better accuracy. [Gar10] explains that the choice for a particular method depends on the application at hand. For example, if accuracy is more important, then one chooses an algorithm, which is better in accuracy even if it exhibits much worse condensation.

As can be seen in Table 5-IV, all algorithms seem to accomplish relatively comparable accuracies, with percentages varying from the highest achieved by HMN with 78.47% (only one surpassing 78%), to the lowest of 75.09% from DROP3. TRKNN demonstrates the second highest accuracy with 77.38%, while LIF follows with 77.37%. Despite the high classification accuracy managed by TRKNN, it can be seen that it is designed to operate as a noise filter rather than a reduction technique, and thus it has a very low condensation ratio of 30.24%. LIF also seems not to achieve high condensation with 31.31%. HMN surpasses the other competitors in accuracy but it displays significantly lower condensation rate reaching an average of 57.69%, when compared to CCIS, DROP3 and ICF that demonstrate reduction rates of 69.89%, 79.47% and 78.09%, respectively. ICPL on the other hand achieves a condensation ratio just above 80%, while the highest one observed is CBP that significantly outperforms all other methods with an average of 88.97%. The latter algorithms, CCIS, ICF, ICPL and CBP, manage nearly identical average classification accuracies as their results lie in a range of 0.09%, with values 77.13%, 77.14%, 77.05% and 77.09% respectively. As already mentioned DROP3, despite having the third best condensation ratio, it displays considerably lower accuracy that the rest of the tested algorithms.

The proposed SIR, with an average of 79.25%, shows to outperform all other competitors in terms of accuracy, as can be seen in Table 5-IV. No other algorithm exceeded the limit of 78.5% accuracy. (If SIR is to be compared with Table 4-IV, only ENN, which is strictly an accuracy enhancing algorithm and not a condensing method, has marginally better accuracy by 0.09%. However, ENN exhibits far lower condensation

than SIR as only 20% of the initial instances are removed compared to nearly 60% achieved by SIR). In terms of condensation, SIR significantly outperforms two competitors TRKNN and LIF, and also outperforms the highly accurate HMN algorithm, by 2.08%. The other competing algorithms, namely CCIS, DROP3, ICF, ICPL and CBP display considerably larger condensation ratios than the 59.77% achieved by the proposed method.

In Table 5-V, the statistical significance of these findings is presented, using a non-parametric two-sided statistical test, the Wilcoxon signed-rank test. It can be seen that SIR is statistically comparable, in terms of accuracy to HMN, LIF, DROP3, ICF and ICPL, as indicated by the large *p*-values, and is better than CBP and CCIS at 2% significance level. Also, in terms of condensation, at 1% significance level, SIR outperforms TRKNN and LIF, since the null hypothesis that the distributions difference has zero median is rejected, while it yields comparable results to HMN and ICF. The rest of the tested methods (CBP, DROP3, ICPL and CCIS) are shown to be better at 1% significance level.

The above observations on the comparison of the different methods are corroborated by Table 5-V, which presents the relative percentage improvement (or deterioration) of SIR over each one of the competing methods for the averaged accuracy and condensation ratios. Comparing the condensation of SIR with the others, a significant improvement of the reduction rate can be seen compared to TRKNN and LIF, with 97.65% and 90.9% respectively, along with an improvement of 3.61% percentage over HMN. On the other hand, a percentage worsening is observed compared to the condensation ratios of the rest of the algorithms. Nevertheless, the accuracy row of the table shows the PI values are all positive, which means that SIR is better than all other algorithms in terms of classification accuracy. More specifically, it improves the second best algorithm (HMN) by nearly 1%, while a percentage improvement of 2.42% and 2.43% can be observed when compared to TRKNN and LIF respectively. For the rest of the methods (namely, CCIS, DROP3, ICF, ICPL and CBP) demonstrates and even larger improvement with values ranging from 2.74% up to a maximum of 5.54% observed for DROP3.

Another way of comparing the previous algorithms is time complexity. Regarding SIR, it consists of two phases. The construction of border discriminative features with a complexity of $O(n^2)$ and the graph partition algorithm that identifies between border and non-border instances. In the latter phase, the eigen-decomposition of the similarity matrix, is the one that dominates the complexity of SIR; this has a measure of $O(n^r)$, where $r<2.376$ is the matrix multiplication exponent as explained in [Pan99]. For the competing methods, TRKNN, LIF, HMN and CCIS have at worst case scenario $O(n^2)$. On the other hand, ENN has complexity $O(n^3)$, and since CBP, DROP3, ICF and

ICPL use ENN as a preprocessing noise filter, which is a very computationally intensive method, their complexity is also $O(n^3)$. Additionaly, in terms of time requirements ICPL, as experiments showed (Table 4-IV) is even more expensive than the rest of the methods. As one can observe, SIR is a somewhat slower than four other methods, but as it is stated in the reviewing article of [Gar10], time requirements are not too important (as for most real applications the reduction stage is executed initially once), unless the method takes excessive times to complete and thus becomes impractical for real applications. But it should be mentioned that all methods but CCIS, that surpass SIR in terms of condensation, namely, DROP3, ICF, ICPL and CBP, are all significantly slower.

In order to evaluate whether SIR is sensitive to large number of features its response to high dimensionalities is evaluated. The chart in Fig. 5.VI depicts dimensionality and classification error for all tested datasets. Datasets with large number of features, such as Ionosphere, Musk and Sonar do not show particular bias in terms of accuracy, and the pearson correlation coefficient ρ=*-0.049* numerically verifies this observation, as the small value indicates no correlation between classification error (accuracy) and dimensionality.

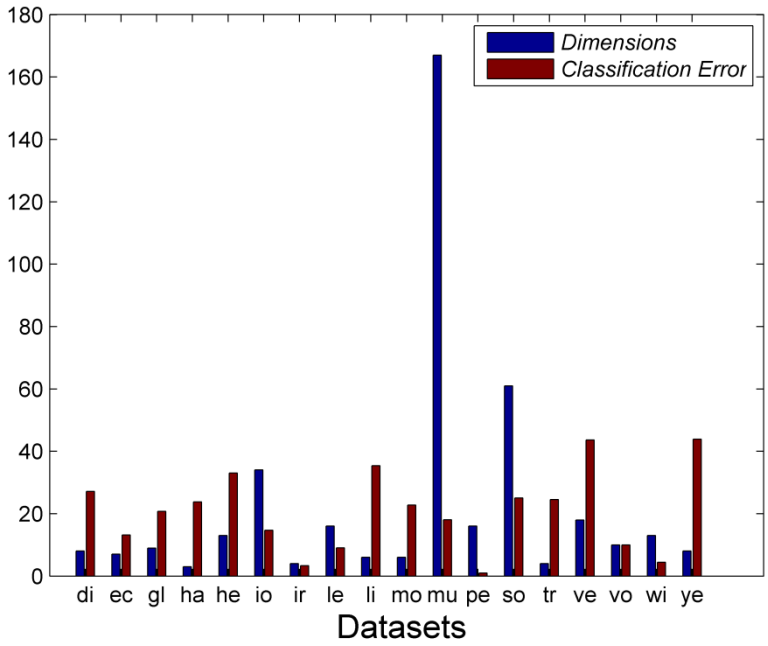
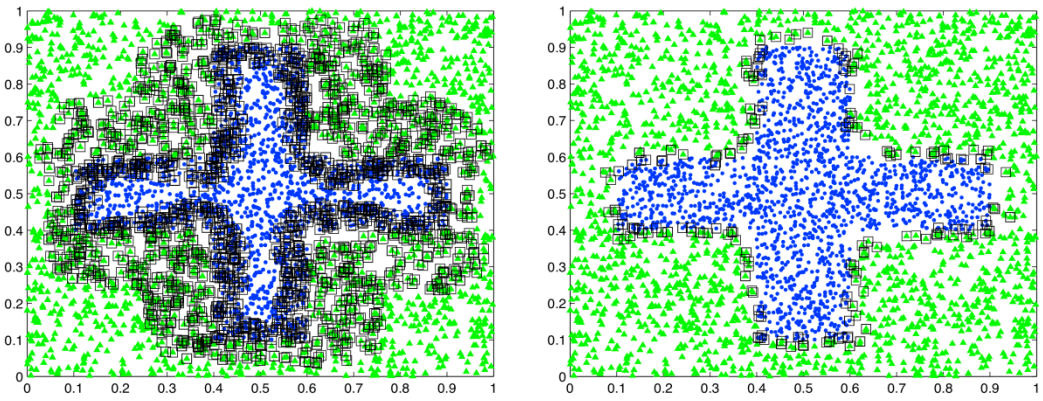


Figure 5.VI- Number of features and average classification errors for all datasets (each denoted by the first two letters of the dataset name) for the proposed algorithm SIR.

A final issue is the use of graph-cut modeling to partition the instances to Z_B and Z_{NB}. In this work, a graph-cut modeling approach has been selected, which is solved with a very fast relaxation method based on matrix decomposition. However, in the reviewing article of [For10] there are many more methodologies for graph partitioning, such as traditional methods (hierarchical [Dun74, Bez81], and spectral clustering [Don73, Spi96]), divisive algorithms(such as CONGA [Gre07] and the Newman-Girvan algorithm

[Gir02, New04]), modularity-based methods (fast modularity [Cla04] and MSG [Sch08]), dynamic algorithms (like Walktrap [Lat05] and MCL [Don00]) and statistical inference methods (variational Bayesian inference [Jor99, Bea03, Hof08]). Using the proposed BDFs here, those methods could be potentially useful to replace the second stage (Section 5.4.ii) of SIR. Experimentation with a few different partitioning algorithms from [For10] has taken place, namely modularity methods (fast modularity, MSG), random walk (walktrap), variational Bayesian inference, divisive algorithms (CONGA), and clustering (k-means). The setup (the BDF part) for these experimentations was identical to the proposed SIR, but replaced the spectral optimisation module of SIR using eigen-decomposition, with one of the above methods. However, it was found that most of these methods, either did not scale well and even with moderately large datasets took excessive times to complete, or produced unbalanced prototype reductions. More specifically, for the CONGA algorithm a single run on a simple dataset, such as ecoli, took more than 33 hours for completion. On the other hand, k-means clustering on the BDF's, although it was very fast, its results were inconsistent, since every run gave totally different output, as can be observed in Fig. 5.VII. The borders identified by the algorithm are either too dense (a) or too sparse (b).



(a) (b)

Figure 5.VII- The 2-dimensional 2-class cross dataset. Both figures were obtained using the same model and the same dataset as input and despite the simplicity of the synthetic dataset, k-means clustering gave not only inconsistent results but also bad ones.

Therefore, CONGA and k-means are excluded from Table 5-IV. For the model assessment of the rest of the graph-cut methods (Fast-Modularity, MSG, Walktrap and Variational Bayesian Inference) 18 real datasets and multiple cross-validation runs were used. So, Table 5-IV presents results on accuracy, condensation and timing (average seconds). It can be seen that all other methods do not scale well for big datasets. For at least two datasets, each algorithm (marked by "-") either did not terminate (after a few hours),or crashed, or the obtained partitions were unbalanced (i.e., rejected almost all samples from a single class, thus making classification and evaluation impossible).

Comparing the table with the accuracy (79.25%), condensation (59.77%) and average time (1.63 sec) of SIR, it can be seen that SIR with eigen-decomposition is strongly outperforming all other methods. Firstly, its timing requirements are much better that all the other methods. It also scales far better and is much more robust than all other methods as it produces results (as shown in Table 5-III) for all datasets. Its accuracy is also better than all other methods, while regarding the last evaluation criterion, three of the other methods show to produce better condensation results. To conclude, for the completed datasets, spectral partitioning yielded better accuracies at much faster speeds (between 4 and 33 times faster), but some of those methods produced better condensation. In addition to these methods, other ones could be used for partitioning, such as the network-based stochastic method of [Sil12$_a$], [Sil12$_b$]. Also, the multilevel method of [Wan07$_b$] could be adapted to enhance the proposed partitioning scheme and enable SIR to cope with large datasets.

Table 5-III

Accuracy, condensation and timing results of competing graph-cut methods. This methods are tested over 18 datasets with multiple cross-validation runs.

Datasets	F-Modularity			MSG			Walktrap			V.B. Inference		
	Acc	**Con**	**Time**	**Acc**	**Con**	**Time**	**Acc**	**Con**	**Time**	**Acc**	**Con**	**Time**
Diabetes	68.13	88.63	68.1	73.7	63.9	18.5	48.11	86.01	11.3	73.62	62.66	32.2
Ecoli	81.78	87.74	81.8	86.2	57.54	6.3	73.73	81.7	3	82.29	55.79	10.7
Glass	54.67	95.38	54.7	77.17	61.4	1.7	74.14	75.7	0.9	66.98	63.82	5.5
Haberman	-	-	-	74.66	63.62	2.5	64.17	89.62	1.5	-	-	-
Heart	58.74	93.52	58.7	66.15	66.54	1.9	57.11	76.62	1.2	67.04	66.19	5.7
Ionosphere	71.12	90.04	71.1	83.59	57.58	5.2	87.19	77.08	3.1	-	-	-
Iris	-	-	-	-	-	-	-	-	-	-	-	-
Letter	-	-	-	-	-	-	-	-	-	-	-	-
Liver	59.95	90.54	60	66.07	67.05	3.1	55.39	81.55	1.7	-	-	-
Monk	80	34.12	80	77.96	52.38	10.4	62.07	65.78	5.2	73.46	54.88	18.8
Musk	64.64	85.23	64.6	79.39	56.15	10.2	71.31	73.08	5.8	76.46	56.13	18.3
Pendigit	-	-	-	-	-	-	-	-	-	-	-	-
Sonar	-	-	-	74.98	59.74	1.6	67.05	69.76	0.9	68.72	59.25	4.9
Transfusion	71.24	88.17	8.5	74.47	61.51	9.3	65.11	83.3	5.3	71.84	62.61	16.7
Vehicle	48.16	90.16	6.3	54.46	74.58	6.4	45.26	86.38	3.5	51.28	74.16	14.1
Vowel	63.39	85.68	51.9	87.8	52.41	54.6	76.85	71.23	35.8	84.93	39.88	89.2
Wine	-	-	-	95.97	52.83	1.7	92.43	75.31	1	94.68	52.73	5.2
Yeast	52.59	89.22	34.3	56.65	75.74	33.6	45.98	87.54	19.4	56.72	72.4	51.7
Average	**64.53**	**84.87**	**53.33**	**75.28**	**61.53**	**11.13**	**65.73**	**78.71**	**6.64**	**72.34**	**60.04**	**22.75**

Table 5-IV

Average accuracy (Acc) and condensation (Cond) percentages of the proposed SIR and 8 other compared algorithms over 18 datasets. These results are averaged over 50 runs.

	TRKNN		**HMN**		**LIF**		**CCIS**		**DROP3**		**ICF**		**ICPL**		**CBP**		**SIR**	
Datasets	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**
Diabetes	66.56	33.99	70.57	62.85	72.22	30.72	68.64	76.89	66.56	94.46	72.55	88.01	68.39	76.53	70.10	92.03	72.84	64.10
Ecoli	72.66	55.74	87.11	47.50	85.05	31.21	82.05	72.11	72.66	92.16	82.53	86.20	81.41	84.11	84.26	92.38	86.85	57.52
Glass	81.42	39.28	79.87	52.20	79.15	31.92	78.22	65.70	81.42	87.46	75.00	86.36	77.90	80.77	73.92	93.45	79.24	60.92
Haberman	64.31	44.38	70.67	61.51	73.53	30.15	64.20	87.48	64.31	80.90	72.36	92.17	67.90	75.33	74.17	91.96	76.19	63.15
Heart	58.67	26.46	65.85	71.52	66.37	38.72	63.93	72.86	58.67	88.11	65.41	82.17	60.89	69.82	65.48	89.16	66.96	66.30
Iono	84.04	67.20	89.40	56.92	86.73	9.73	86.37	79.53	84.04	55.70	81.14	96.07	88.03	86.74	83.09	95.73	85.31	57.53
Iris	93.33	45.21	95.60	43.08	95.20	13.21	95.47	81.67	93.33	71.10	95.47	60.41	93.47	90.52	94.53	94.09	96.67	51.04
Letter	94.68	16.48	91.35	48.99	87.16	48.49	92.04	54.42	94.68	82.38	89.03	75.75	91.13	87.63	85.79	89.54	90.95	53.43
Liver	60.41	23.75	63.68	69.92	66.72	38.49	64.53	72.29	60.41	66.84	61.80	84.32	60.01	70.89	63.65	88.39	64.65	66.41
Monk	77.98	1.21	71.42	82.95	75.86	41.98	71.27	81.11	77.98	80.31	79.30	48.40	71.56	64.68	79.38	57.12	77.27	52.13
Musk	84.38	27.69	83.41	54.65	83.00	23.54	80.00	66.84	84.38	78.49	77.46	78.81	84.66	83.48	79.12	89.38	81.97	56.34
Pendigit	99.30	26.35	98.65	41.88	98.39	31.18	98.94	56.68	99.30	85.18	97.38	89.72	97.93	95.92	94.48	98.38	99.06	50.58
Sonar	81.46	13.98	76.96	62.31	76.94	30.24	75.49	66.65	81.46	72.11	75.55	67.57	79.72	82.03	75.25	89.10	74.96	59.13
Transfusion	62.35	48.26	74.41	66.21	75.10	25.90	63.34	85.98	62.35	82.07	72.33	69.37	73.57	86.11	74.17	88.09	75.46	61.97
Vehicle	57.65	11.95	54.71	68.31	55.75	51.57	52.98	58.54	57.65	90.10	54.87	81.68	54.39	67.96	53.69	89.76	56.34	74.82
Vowel	98.24	9.38	91.82	45.97	70.93	67.41	91.82	46.76	98.24	65.41	90.42	43.38	93.47	82.90	88.65	83.17	90.06	53.33
Wine	94.62	26.39	95.97	48.13	96.44	25.60	95.71	67.94	94.62	74.80	91.47	87.11	93.39	89.43	95.43	95.65	95.62	51.50
Yeast	49.97	22.28	56.76	64.03	58.08	51.81	53.32	69.97	49.97	82.86	54.51	88.10	49.00	65.57	51.98	92.35	56.15	75.66
Average	**77.38**	**30.24**	**78.47**	**57.69**	**77.37**	**31.31**	**77.13**	**69.89**	**75.09**	**79.47**	**77.14**	**78.09**	**77.05**	**80.02**	**77.09**	**88.97**	**79.25**	**59.77**

Table 5-V

Comparison of SIR against all other algorithms, with accuracy (Acc) and condensation (Cond) shown in the rows. The **PI** columns correspond to the percentage improvement (positive) or percentage deterioration (negative) score, calculated as: (*SIR_score – other_score) / other_score × 100*), where the scores are taken to be the average classification or condensation scores across all datasets (last row of Table II). The columns marked as **p** correspond to the p-values of the Wilcoxon sign-rank test at 0.01 significance, with the null hypothesis that the scores distributions for all individual datasets have equal medians.

	TRKNN		**HMN**		**LIF**		**CCIS**		**DROP3**		**ICF**		**ICPL**		**CBP**	
	PI(%)	*P*	**PI(%)**	***P***	**PI(%)**	***p***	**PI(%)**	**PI(%)**	***p***	**PI(%)**	***P***	**PI(%)**	***p***	**PI(%)**	**PI(%)**	***P***
Acc	+2.42	0.21	+0.99	0.53	+2.43	0.42	+2.75	0.01	+5.54	0.21	+2.74	0.53	+2.86	0.42	+2.8	0.01
Cond	+97.65	0.0003	+3.61	0.1080	+90.90	0.0003	-14.48	0.0057	-24.79	0.0003	-23.46	0.1024	-25.31	0.0003	-32.82	0.0057

iv. Conclusion

The novelty of the proposed framework is noticeable on the two contributions of this work that involve, firstly, the border discriminating features that enable for the proper setup (ordering) of instances, and, secondly, the spectral modeling used for the graph partitioning. So, this work has focused on the creation of a set of border discriminating features capable of capturing different types of local geometric characteristics of the data samples, in terms of their friend and enemy profiles. These features were employed by a graph-cut modeling approach and processed using standard spectral graph theory, in order to generate a partition vector that divides the instances into border and internal ones.

The main advantage of the proposed algorithm is that it manages relatively high condensation without compromising the classification accuracy, as it demonstrated the highest accuracy among all the tested condensation algorithms. It should also be mentioned that it is robust to spatial arrangements of the class distributions because of the use of a diverse set of BDFs, and it does not require any user-defined parameters. On the other side, limitations include that, like other methods that use distances between samples, it may fail to remove the right instances in highly dimensional spaces due to the distance concentration effect. Despite the fact that SIR displays competent condensation results the average reduction is not as good as some of the existing methods such as CBP that was proposed in Chapter 4 of this work. Also, if there is high degree of noise, a more effective filter than the employed ENN-based one may be needed to remove misclassified samples. Finally, as SIR is designed with the 1NN rule in mind, if a different classifier is required, removal of some non-border instances may have impact on the formation of the decision boundaries, especially in datasets with a large number of classes.

In relation to other works in the literature, SIR displays the typical behaviour of selection algorithms. Competent results in terms of instance removal that account for the very high accuracy achieved by the method. Although SIR is one of the most accurate algorithms, as experiments showed, this work did not tackle the problem of instance selection algorithms that is the low condensation ratio compared to abstraction techniques.

Future work could include the improvement of SIR in terms of condensation by combining it with instance abstraction. Speeding up the method and using incremental updates of the partitioning eigenvector for applications where new samples are frequently generated can be useful. The design of additional BDFs better tailored to the employed classifier can also be beneficial, together with adapting the cardinality of friends and enemies for the different BDFs according to the characteristics of the individual datasets.

CHAPTER 6:

PROTOTYPE REDUCTION BASED ON DIRECT WEIGHT OPTIMIZATION

6.1 Introduction

This chapter presents an instance reduction technique that uses a set of binary weights to directly control, which samples will be discarded and which retained. In order to guarantee that every sample is correctly classified, the proposed Direct Weight Optimisation algorithm (DWO), aims to retain the ratio of distances of its nearest friend over its nearest enemy to a minimum. In section 6.2 the four components of the algorithm are presented. During the first stage, analyzed in 6.2.i., the model involving instance weight learning is presented. In section 6.2.ii., the optimisation component of DWO is described that uses a genetic algorithm to obtain solutions in terms of the two essential objectives, accuracy and condensation ratio. The final stage of the proposed algorithm involves a set of heuristics that are employed in order to improve the performance of DWO and accelerate the entire process. Similar to SIR, the proposed method uses explicitly selection of instances as it involves no prototype generation. Section 6.3 presents the results on real datasets along with simulations on synthetic data and comparisons of the formulated method against other successful prototype condensation algorithms (as well as CBP and SIR described in chapters 4 and 5 respectively). Experiments show that DWO is competent and efficient as it displays the highest classification accuracy along with competitive condensation results. Finally, section 6.4 concludes the chapter.

6.2 The Proposed Algorithm

Assuming a dataset X of n d-dimensional patterns $\mathbf{x}$, each associated with a discrete label $\psi(\mathbf{x})$. The principal objective is to reduce X to a much smaller number of $m<<n$ patterns, that match the classification performance of the original X as close as possible. As it is the norm in all previous works [Wil00, Bri02, Nik12, Mar08] for reasons of simplicity, the 1-nearest neighbour (1NN) rule is used to measure the classification accuracy supported by the dataset.

i. Instance Weight Modelling

The proposed model depends on a set of design parameters, which directly control which instances are removed and which retained. These parameters are a set of n binary weights $w(\mathbf{x})\in\{0,1\}$ each corresponding to an original pattern $\mathbf{x}\in X$. A sample $\mathbf{x}$ is discarded or preserved when its associated $w(\mathbf{x})$ has the value of zero or one, respectively. This simple and explicit modelling enables us to optimize the entire weight vector $\mathbf{w}\in\{0,1\}^n$, such that the classification accuracy is compromised minimally. The overall optimisation is bi-objective. Firstly, the condensation ratio defined as

$$\frac{n-m}{n}=1-\frac{1}{n}\sum_{\mathrm{x}\in X} w(\mathrm{x}) \tag{6.1}$$

needs to be maximized, in order to remove as many instances as possible. However, removing too many instances will deteriorate the system's performance. Thus, the second objective is to simultaneously maximize the overall accuracy.

A straightforward way for measuring this accuracy using the 1NN rule is to use the ratio of the distance of $\mathbf{x}$ from its nearest friend (where friends are other instances in X from the same class) to its nearest enemy (where enemies are instances in X from other classes than $\mathbf{x}$). In the absence of noise, if this ratio is less than the unity, then the sample $\mathbf{x}$ is supported by X, otherwise it is misclassified. Nevertheless, in the proposed model nearest friends and enemies are not static, as their existence depends on the current state of $\mathbf{w}$. This is because all samples are involved in the model optimization procedure. To incorporate the state of $\mathbf{w}$ into the above modelling, there exist the need to firstly express the nearest friend and enemy of each instance $\mathbf{x}$, also as a function of $\mathbf{w}$, as

$$\begin{aligned} F(\mathbf{x},\mathbf{w}) &= \arg\min_{\substack{\mathbf{z}\in X-\{\mathbf{x}\}\\ \psi(\mathbf{z})=\psi(\mathbf{x})\\ w(\mathbf{z})\neq 0}} \|\mathbf{z}-\mathbf{x}\|_2 \\ E(\mathbf{x},\mathbf{w}) &= \arg\min_{\substack{\mathbf{z}\in X\\ \psi(\mathbf{z})\neq\psi(\mathbf{x})\\ w(\mathbf{z})\neq 0}} \|\mathbf{z}-\mathbf{x}\|_2 \end{aligned} \tag{6.2}$$

where F(**x**,**w**) is the nearest surviving neighbour of **x**, and E(**x**,**w**) its nearest surviving enemy. Then, under **w**, the ratio

$$\gamma(\mathbf{x},\mathbf{w}) = \frac{\left\|\mathbf{x} - F(\mathbf{x},\mathbf{w})\right\|_2}{\left\|\mathbf{x} - E(\mathbf{x},\mathbf{w})\right\|_2} \tag{6.3}$$

is used to test whether **x** is classified correctly or not.

To keep the accuracy objective high for all patterns, it is not possible to force the ratios for all samples $\mathbf{x} \in X$ to have $\gamma(\mathbf{x},\mathbf{w})<1$ as a set of hard constraints within the optimization, due to noise, sparse sampling or the nature of the dataset. Instead, accuracy is optimized in a soft way using penalty functions $H[\cdot]$ that penalise cases with ratios exceeding the unity in an aggregate way. To facilitate the optimisation a smooth penalty function is used, which is defined as

$$H[k] = \frac{1}{1+\exp(\alpha(1-k))} \tag{6.4}$$

where α is a fixed parameter that controls the shape of the sigmoid penalty curve, as shown in Fig. 6.I. If α is set to a high value, the curve becomes a step function and gives zero or one penalty values to ratios below or above the unity, respectively. However, a smoother penalty curve allows for a better balancing of the two competitive objectives and copes with cases of noise and sparse datasets better.

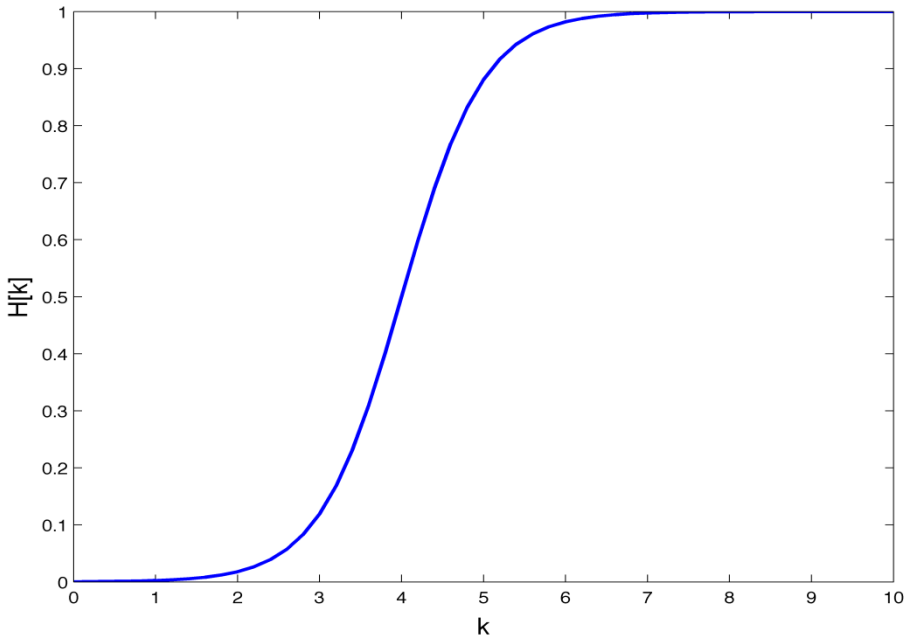


Figure 6.I- Shape of the smooth sigmoid penalty function H[k], where values of classification ratios k<1 receive less penalty.

Finally, the two maximising objectives of condensation and accuracy can be combined to a single objective function *J(w)*, using a weighted sum according to

$$\max_{\mathbf{w}\in\{0,1\}^n} J(\mathbf{w}) \equiv \sum_{\mathbf{x}\in X} w(\mathbf{x}) + \lambda \sum_{\mathbf{x}\in X} H\left[\gamma(\mathbf{x},\mathbf{w})\right] \tag{6.5}$$

with λ being a user defined parameter that balances the competition between the two objectives during the optimisation process.

ii. Optimisation Procedure

The index *J(**w**)* of Eq.(6.5) corresponds to a binary optimisation problem, which can be difficult to solve for large datasets in terms of efficiency and also in terms of finding an acceptable, globally optimal solution. In this work, a genetic algorithm (GA) is used, because GAs can in general find good quality solutions for large scale combinatorial problems [Gou11]. Especially for problems with binary design parameters, their standard bit-string genetic encoding naturally follows the problem formulation. Additionally, as will be described in Section 6.2.iii, it is easy to add performance accelerating heuristics tailored to the problem at hand. GAs have been previously used for prototype reduction in [Kun95], [Gar08], and in machine learning works such as [Tse01, Cow99, Mau00, Gal05].

For the current work, a standard GA with bit-string chromosome modelling is employed, where each gene of every *n*-length member in each population, corresponds to the weight $w(\mathbf{x})$ of the specific sample $\mathbf{x}$ the gene corresponds to. The exploitation and exploration aspects of the genetic search are supported with a standard uniform crossover, and a bit-flip mutation. A two member elitism operator copies the best two members to the subsequent generation. Also, to guard against invalid chromosomes, a repair procedure has been included, which when crossover or mutation create an offspring with all samples of some class having zero weight (i.e., no class representatives remain), one of those discarded samples is randomly selected and reinstated in the chromosome.

iii. Performance Acceleration Heuristics

Although GAs constitute competent search strategies for large scale combinatorial problems, performance may be slow for large dimensionalities. As metaheuristic techniques, GAs lend themselves easily to enhancements by taking advantage of the knowledge and structure of the problem at hand.

The first enhancement is population hybridisation. Assuming to have p_{size} members in the population, before genetic search is initiated a random population has to be created. Although the standard procedure is to generate uniformly random zero/one values for the entries of the $p_{size}\times n$ population matrix, this is not efficient. Here two tests are used to approximately estimate whether a sample $\mathbf{x}$ is a border instance of its class distribution and can be retained. Although border instances are not the only ones that can

be retained, an early flagging of such instances can speed up the recovery of useful patterns in the dataset. The first test is based on the concept of typicality as introduced in [Zha92], and is given by

$$T_1(\mathbf{x}) = \frac{1 - \frac{1}{d_{\max}|FR(\mathbf{x})|}\sum_{\mathbf{y}\in FR(\mathbf{x})}\|\mathbf{x}-\mathbf{y}\|_2}{1 - \frac{1}{d_{\max}|EN(\mathbf{x})|}\sum_{\mathbf{y}\in EN(\mathbf{x})}\|\mathbf{x}-\mathbf{y}\|_2} \tag{6.6}$$

This test evaluates the ratio of the average similarity of **x** to all of its friends FR(**x**) over all its enemies EN(**x**) (d_{max} is the largest distance in the dataset). High values of $T_1(\mathbf{x})$ indicate that **x** is a border instance. The second test employed is defined as

$$T_2(\mathbf{x}) = -\frac{|R(\mathbf{x})|}{|C(\mathbf{x})|} \tag{6.7}$$

and is based on the ratio of the cardinalities of two sets introduced in [Bri02]. One is the reachable set given by

$$R(\mathbf{x}) = \left\{\mathbf{y}\in X : \psi(\mathbf{x}) = \psi(\mathbf{y}) \wedge \|\mathbf{x}-\mathbf{y}\|_2 \le \|\mathbf{x}-E(\mathbf{x})\|_2\right\} \tag{6.8}$$

where E(**x**) is the nearest enemy of **x**. That is, R(**x**) contains all friendly instances lying between **x** and its nearest enemy. The other set is the coverage one, defined as

$$C(\mathbf{x}) = \left\{\mathbf{y}\in X : \mathbf{x}\in R(\mathbf{y})\right\} \tag{6.9}$$

Their negative ratios $T_2(\mathbf{x})$ obtains higher values when **x** is most likely a border instance. Using these tests, each initial gene (i.e., weight) of the i^{th} population chromosome is set as a uniformly random zero/one value with probability 1-p_{hyb} (that is, as in a standard GA), or it is hybridised with probability p_{hyb} as follows. A random test index $k\in\{1,2\}$ is chosen and the weight is set according to the rule

$$w_i^{(initial)}(\mathrm{x}) = \begin{cases} 0 & \text{if } T_k(x) \le mean\left(\left\{T_k(x)\right\}_{x\in X}\right) \\ 1 & \text{otherwise} \end{cases} \tag{6.10}$$

In this way, any of the existing tests $T_k(\mathbf{x})$ can be used to approximate whether each **x** is a likely border sample and set its corresponding initial weight to one (i.e., flag it in the population as more likely to remain). On average, $p_{hyb}\times p_{size}\times n$ of the genes are created in this informed manner, and the remaining randomly. Of course, any inaccuracies introduced in the population from these tests, do not constitute a burden to the evolutionary search, as the population can gradually find the better optimum. This procedure was found to speed up the convergence significantly.

The second heuristic, employed to accelerate the optimisation process, is a memetic component used to enhance the condensation of the proposed algorithm, via incorporating some local fine-tuning of existing members in the population. A fraction of p_{mem} members from the offspring created in each generation are checked to establish

whether any of their retained instances **x** have reachable sets larger than the coverage ones. If this is found to be true, the corresponding weights are set to zero. This procedure is helpful, since for an instance **x**, $|R(\mathbf{x})| > |C(\mathbf{x})|$ means that there are more friends of **x** that can be used to classify **x** correctly, than **x** can [Bri02]. To avoid positional bias, the check is performed for all weights set to one, in random order. Also, the rule is applied as long as it does not lead to the creation of an invalid chromosome.

The last heuristic of the proposed method is based on accelerating the search by directly reducing the dimensionality of the problem from n to $n'=[p_{share} \times n]$, given a user defined parameter p_{share} (Section 6.3 summarises all parameters and their values). This is achieved with weight sharing, where samples are collected in small groups that can share the same fate of being retained or discarded. This is implemented as follows. For each **x**, the values of the previous tests are used and the vectors $[T_1(\mathbf{x}), T_2(\mathbf{x})]^T$ are formed. Then a simple clustering algorithm is applied to these two-dimensional features, such as *k*-means, and n' clusters L_i, for i=1,…,n' are obtained. Finally, by using the cluster memberships, all n patterns **x** are arranged into n' groups. Therefore, all patterns within the same group share a single weight value according to

$$\forall i \in \{1,...,n'\}, \forall \mathrm{x},\mathrm{y} \in L_i \subset X, w(\mathrm{x}) \equiv w(\mathrm{y}) \tag{6.11}$$

The rationale behind this heuristic, is that when different samples share similar border test profiles, they have more chances in having the same fate. This is because they can be located in similar key or unimportant positions in terms of the class distributions and whether they can facilitate the classification of other friends and support the separability from the enemy classes.

6.3 Experimental Analysis

As already explained, the problem of instance reduction is a multi-objective optimisation problem, as the overall performance of algorithms is not characterised only by the classification accuracy they exhibit, but also by the condensation ratio they achieve. The conflicting nature of these two objectives, since an improvement in one, often leads to the deterioration of the other, along with other important aspects such as complexity and robustness require an in depth analysis of the experimental results. In order to obtain the results presented in this section, the parameters of the optimisation algorithm were set to the values specified in Table 6-I. These values are the result of thorough analysis and intense experimentation. But, the genetic algorithm used as the optimisation component offers a significant advantage to DWO, since its performance can be adjusted to the results required. So, by differentiating the values of the table below, the performance of

the algorithm can vary to favour classification accuracy or condensation ratio respectively.

Table 6-I

Parameters settings for the proposed optimisation algorithm.

Parameter	Value	Description
α	20.0	Penalty curve control
λ	3.3	Bi-objective balancing parameter
p_{size}	100	Population size
p_c	0.8	Crossover rate
p_m	0.01	Gene mutation probability
p_{hyb}	0.15	Population hybridisation ratio
p_{mem}	0.1	Memetic component frequency
p_{share}	0.9	Weight sharing dimensionality fraction
t_{max}	200	Maximum number of generations allowed

An example of the effect these variations can have on the output of DWO can be seen in Fig. 6.II. These experiments are performed on twelve (out of the total eighteen) real dataset used for all experiments. Because of the multiple runs necessary and the time requirements for all these experiments, the choice of the datasets was done based on their size. As illustrated in Fig. 6.II, the parameters of the genetic algorithm can largely affect the performance of the proposed algorithm. In the first row of the figure, one can observe that classification accuracy rises as the balancing parameter λ increases until it reaches a maximum value from where decline starts. Condensation on the other hand is monotonically decreasing (Fig. 6.II (1c)). The second row depicts the effect increasing α has on the performance of DWO. Again, accuracy and condensation are conflicting, since the first one is increasing, while the latter one declines. The next parameter investigated is the size of the population. As p_{size} rises the condensation of DWO boosts, while accuracy shows an initial incline before it starts deteriorating. Finally, it can be seen that a bigger number of generations leads to larger values of condensation as well as higher classification accuracy. To further demonstrate the capabilities of the proposed DWO algorithm Fig. 6.III is included, that illustrates the results of this algorithm on the same three two-dimensional synthetic datasets (3-class example, cross and spiral) that are used in previous experiments. The selection of instances from DWO is illustrated in the figures below, and as can be observed, it demonstrates quite high reduction rate as only a small percentage of the initial training set is retained. Although the selected instances do not lie exactly on decision surface, they are optimized in such a way to guarantee that the nearest neighbour of each sample is of the same class label.

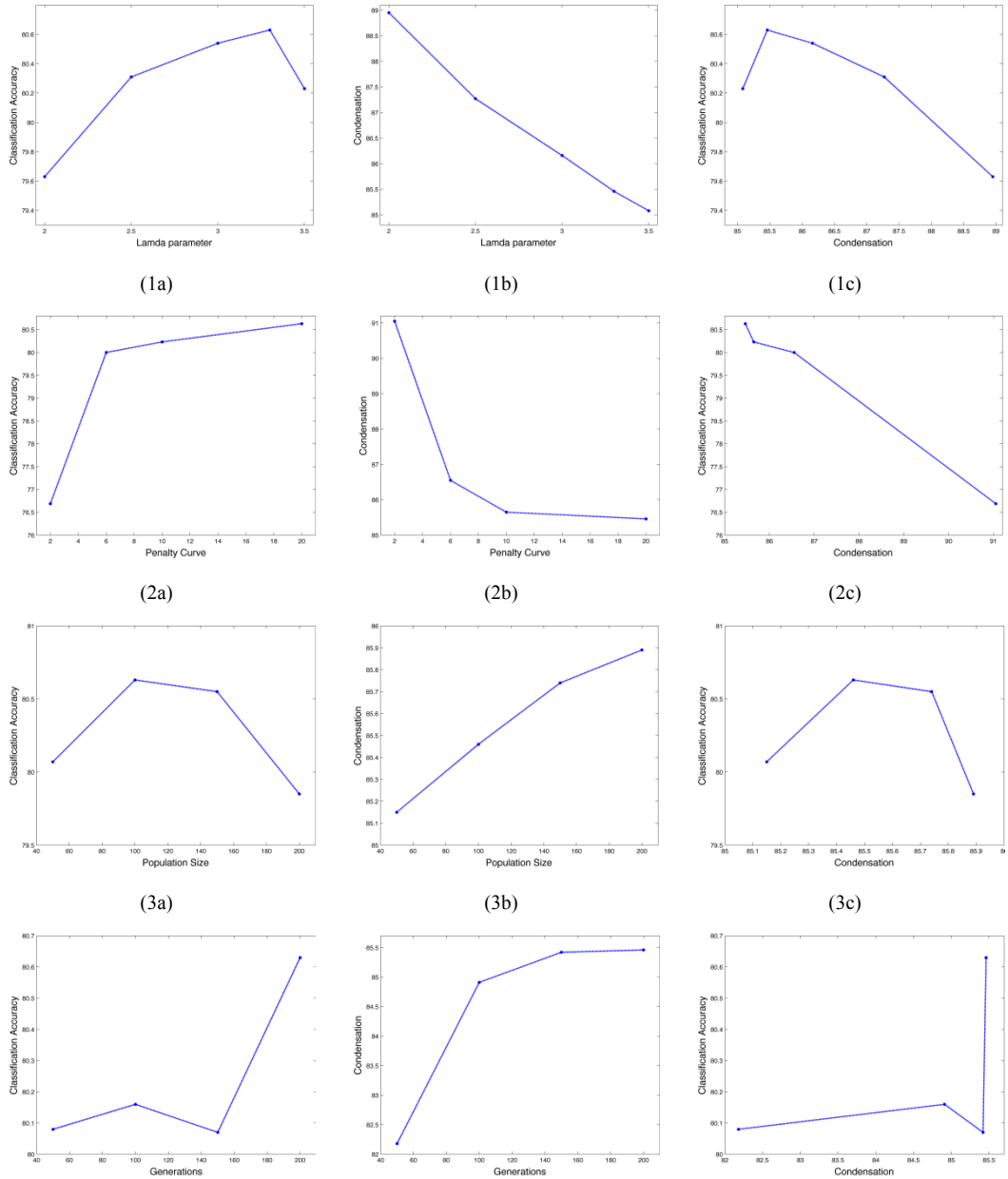
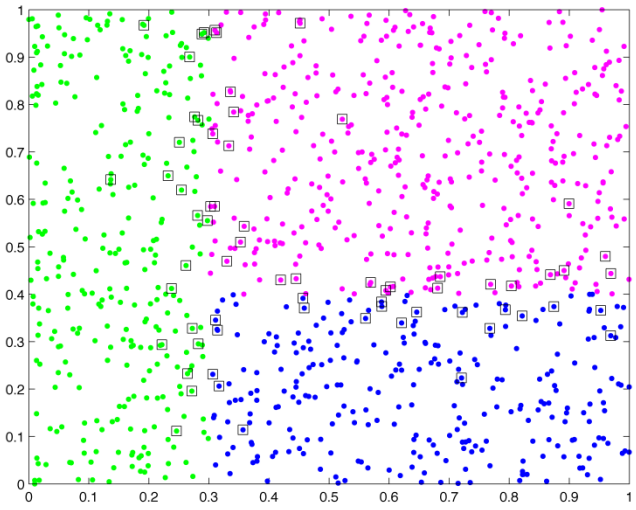
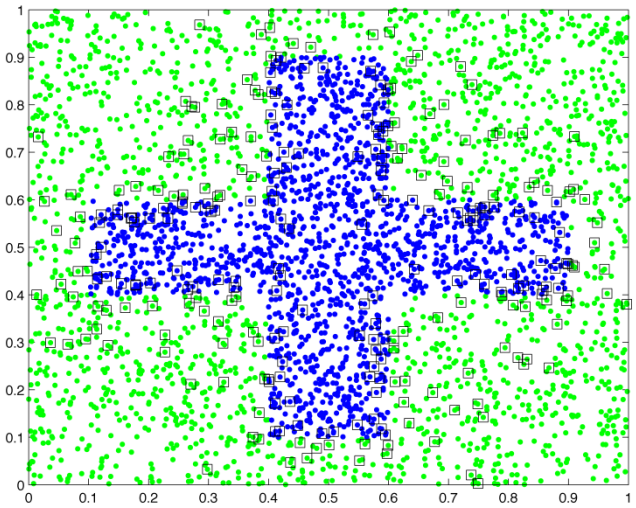


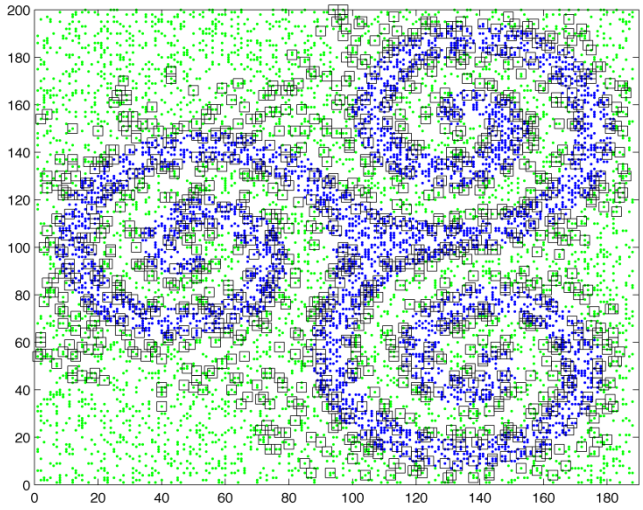
Figure 6.II- Effect of GA parameters on classification accuracy and condensation ratio. **1(a)** Classification accuracy of DWO as the balancing parameter λ increases. **1(b)** Effect of λ on condensation ratio. **1(c)** Accuracy over condensation as the balancing parameter increases. **2(a)-(c)** Changes on classification accuracy and condensation ratio as the control parameter α of the penalty curve increases. **3(a)-(c)** Effect of the population size on the accuracy and condensation ratio of DWO. **4(a)-(c)** Classification accuracy and condensation of DWO as the number of generations is increased.



(a)



(c)



(c)

Figure 6.III- Experiments with three synthetic two-dimensional examples: **(a)** three-class rectangular shapes, **(b)** two-class cross and **(c)** two-class tri-spiral. In all cases, the patterns from different classes are marked with symbols "▲", "●", "◆", and the retained border instances are enclosed by "□".

i. Numerical Results

In this section, in order to evaluate the proposed DWO algorithm, a comparison against eight previously proposed instance reduction algorithms is provided. These methods include Hit Miss Networks, LIF and CCIS proposed by Marchiori [Mar08, Mar09, Mar10], DROP3, the most efficient of the DROP algorithms introduced in [Wil00], ICF [Bri02], ICPL [Lam02$_b$] and two previously described algorithms, CBP and SIR [Nik10, Nik12]. All tests are again performed over the same eighteen datasets from the UCI Machine Learning Repository [Bla98] that are described in Table 4-I.

As explained in previous chapters, the user defined parameters required were set to their best performing values of $k=1$ for the CCIS algorithm and $\tau=0.5$ for the CBP, and remained fixed in all experiments, while DWO was used without the weight sharing heuristic. For reasons of simplicity and because this has been the trend in all previous works, the simple 1NN rule was used to measure the classification accuracy of the output prototypes, while in all algorithms the Euclidean distance norm was used as the distance metric. Finally, each dataset was randomly divided in ten folds, nine of which were used for training each algorithm and performing the prototype reduction, and one fold for testing the classification accuracy with the 1NN rule. In total, five random permutations of a 10-fold cross-validation were used giving a total of fifty runs per dataset from which the final measures of accuracy and condensation were averaged for each dataset. Numerical results for the eight competing algorithms and the proposed DWO are aggregated in Table 6-II, together with averages across all datasets.

ii. Discussion

As already mentioned, Table 6-II presents both accuracy and condensation ratios for all competing algorithms. Since, instance reduction is a bi-objective optimisation problem one instance reduction algorithm can be considered better than another, when it is better in both objectives, or if it is equivalent in one objective and better in the second. But, in reality the final decision depends on the application at hand and what the user prioritizes. For example, if condensation is critical, an algorithm that exhibits high reduction rate can be chosen even if it does not demonstrate the highest accuracy.

From the table below, it is clear that all competing algorithms manage to have similar accuracies as all averages lie within a range of 4.16% from the highest one achieved by SIR at 79.25%, which is the only one that surpasses 79%, to the lowest one observed for DROP3 at 75.09%. The two algorithms, HMN and LIF, proposed by Marchiori demonstrate the second and third highest accuracy at 78.47% and 77.37%, while ICF, CCIS, CBP and ICPL follow with nearly identical classification accuracies at 77.14%, 77.13%, 77.09% and 77.05% respectively. Despite the high classification accuracy managed by LIF, it can be seen that it is designed to operate as a noise filter

rather than a reduction technique, and thus it has a very low condensation ratio of no more than 31.31%. SIR and HMN that display the best accuracy results exhibit competent condensation ratios, 59.77% and 57.69% respectively, but significantly lower that CBP and ICPL that achieve the highest reduction rates at 88.97% and 80.02% respectively. It should be noted that these two are the only ones that overcome the 80% limit in terms of condensation, as the rest of the algorithms demonstrate considerably lower values. DROP3 with 79.47% and ICF with 78.09% follow, while CCIS achieves a condensation of nearly 70%. As already mentioned DROP3, despite having the third best condensation ratio, it displays considerably lower accuracy that the rest of the tested algorithms.

As can be observed in Table 6-II, the proposed algorithm, DWO, with average classification accuracy of 79.75% outperforms all other algorithms. No other algorithm exceeds the limit of 79.5% accuracy, not even ENN that is strictly an accuracy-enhancing algorithm and according to Table 4-IV exhibits an average accuracy of no more than 79.34%. SIR and HMN, which are the two methods with the highest accuracies, are clearly outperformed by 0.5% and 1.28% respectively. So DWO is the most accurate algorithm from all the tested methods. On the other hand, in terms of condensation, HMN, LIF, CCIS, DROP3, ICF, ICPL and SIR display considerably lower reduction rates than the 83.72% achieved by DWO. Hence, DWO clearly surpasses all other algorithms but CBP. However CBP has noticeably lower accuracy than DWO.

In Table 6-III, statistical confidence values are also included that were obtained using the Wilcoxon signed-rank test, which is a non-parametric two-sided statistical test. It can be seen that DWO is significantly better, in terms of accuracy, than CCIS, DROP3, ICF, ICPL, and CBP as indicated by the small *p*-values, and is better than HMN at 7% significance level. The large p-values for LIF and SIR show that it is statistically comparable to these methods. Also, in terms of condensation, at 1% significance level, DWO clearly outperforms four out of the eight methods (namely, HMN, LIF, CCIS and SIR) since the null hypothesis that the distributions difference has zero median is rejected, while it yields comparable results to DROP, ICF and ICPL (the null hypothesis is rejected at 8% significance level). Only CBP is shown to be better than DWO at 1% significance level.

The further demonstrate the improvement achieved by the proposed method; Table 6-III is included with the relative percentage improvement (or deterioration) of DWO over each one of the competing methods for the averaged accuracy and condensation ratios. Comparing the accuracy row of the table it shows that DWO performs better than all other algorithms in terms of classification accuracy, since all signs are positive. The percentage improvement varies from a maximum +6.21% observed for DROP3 to a minimum of +0.63% for the second best algorithm (SIR). An improvement of over 1.5% is observed for all other methods. On the other hand, in terms

of condensation ratio, improvement continues when DWO is compared against all other algorithms but CBP. More specifically, it displays a worsening of 5.9% against CBP, but it improves the third best algorithm (ICPL) by 4.62%, and the fourth best (DROP3) by nearly 5.5%. For the rest of the methods (namely, HMN, LIF, CCIS, ICF, and SIR) DWO demonstrates even larger improvement with values ranging from 7.21% (ICF) up to a maximum of 167.39% observed for LIF.

Instance reduction algorithms can also be compared based on their complexity and time requirements. The complexity of the proposed algorithm is dominated by the optimizing component used to solve this model. Hence, the complexity of DWO is the complexity of the genetic algorithm that in the worst case scenario, and if no early termination occurs, has a measure of $O(t_{max}np_{size})$. Three of the methods LIF, HMN and CCIS, which are the fastest ones, have a complexity of $O(n^2)$, while SIR, as already mentioned, displays a complexity of $O(n^r)$, where $r<2.376$ is the matrix multiplication exponent. On the other hand, CBP, DROP3, ICF and ICPL have an even higher complexity because they use Wilson's editing rule (ENN) of $O(n^3)$ complexity as a preprocessing step. Therefore, their complexity is also $O(n^3)$. But, ICPL, because of the merging component, exhibits even larger time requirements than the rest of the methods. As one can observe, the speed of DWO depends on the parameters of the GA used. It should be noted that the reduction of instances in real applications is performed only once, in the beginning of the process. Therefore, time requirements are not of high importance as long as the method does not become impractical for real applications.

A final issue of DWO is its sensitivity to datasets with a large number of features. In order to show how the proposed method responds to high dimensionalities, Fig. 6.IV is added that illustrates dimensionality and classification error for all tested datasets. As can be observed for large datasets such as musk and sonar, there is no particular bias with respect to accuracy. This case is also proven from the small value of the pearson correlation coefficient that was measured to be $\rho=-0.159$. This indicates that DWO is robust to high dimensionalities, since accuracy is not affected.

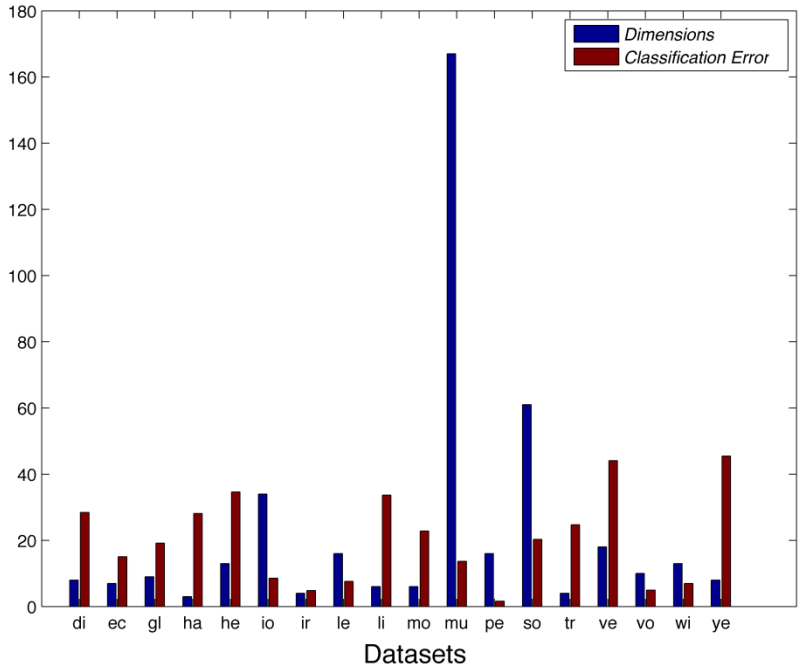


Figure 6.IV- Number of features and average classification errors for all datasets (each denoted by the first two letters of the dataset name) for the proposed algorithm DWO.

Table 6-II

Average accuracy (Acc) and condensation (Cond) percentages of the proposed CBP and 8 other compared algorithms over 18 datasets. These results are averaged over 50 runs.

	HMN		LIF		CCIS		DROP3		ICF		ICPL		CBP		SIR		DWO	
Datasets	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**	**Acc**	**Cond**
Diabetes	70.57	62.85	72.22	30.72	68.64	76.89	69.70	94.46	72.55	88.01	68.39	76.53	70.10	92.03	72.84	64.10	71.55	86.38
Ecoli	87.11	47.50	85.05	31.21	82.05	72.11	79.55	92.16	82.53	86.20	81.41	84.11	84.26	92.38	86.85	57.52	84.96	90.33
Glass	79.87	52.20	79.15	31.92	78.22	65.70	70.32	87.46	75.00	86.36	77.90	80.77	73.92	93.45	79.24	60.92	80.84	82.18
Haberman	70.67	61.51	73.53	30.15	64.20	87.48	66.60	80.90	72.36	92.17	67.90	75.33	74.17	91.96	76.19	63.15	71.88	87.52
Heart	65.85	71.52	66.37	38.72	63.93	72.86	63.78	88.11	65.41	82.17	60.89	69.82	65.48	89.16	66.96	66.30	65.41	83.19
Iono	89.40	56.92	86.73	9.73	86.37	79.53	82.68	55.70	81.14	96.07	88.03	86.74	83.09	95.73	85.31	57.53	91.46	92.36
Iris	95.60	43.08	95.20	13.21	95.47	81.67	95.73	71.10	95.47	60.41	93.47	90.52	94.53	94.09	96.67	51.04	95.20	94.07
Letter	91.35	48.99	87.16	48.49	92.04	54.42	84.24	82.38	89.03	75.75	91.13	87.63	85.79	89.54	90.95	53.43	92.38	81.23
Liver	63.68	69.92	66.72	38.49	64.53	72.29	64.98	66.84	61.80	84.32	60.01	70.89	63.65	88.39	64.65	66.41	66.30	80.85
Monk	71.42	82.95	75.86	41.98	71.27	81.11	73.19	80.31	79.30	48.40	71.56	64.68	79.38	57.12	77.27	52.13	77.16	83.20
Musk	83.41	54.65	83.00	23.54	80.00	66.84	71.28	78.49	77.46	78.81	84.66	83.48	79.12	89.38	81.97	56.34	86.35	79.95
Pendigit	98.65	41.88	98.39	31.18	98.94	56.68	96.54	85.18	97.38	89.72	97.93	95.92	94.48	98.38	99.06	50.58	98.36	93.80
Sonar	76.96	62.31	76.94	30.24	75.49	66.65	72.45	72.11	75.55	67.57	79.72	82.03	75.25	89.10	74.96	59.13	79.69	75.88
Transfusion	74.41	66.21	75.10	25.90	63.34	85.98	69.81	82.07	72.33	69.37	73.57	86.11	74.17	88.09	75.46	61.97	75.30	84.15
Vehicle	54.71	68.31	55.75	51.57	52.98	58.54	50.93	90.10	54.87	81.68	54.39	67.96	53.69	89.76	56.34	74.82	55.93	76.96
Vowel	91.82	45.97	70.93	67.41	91.82	46.76	90.69	65.41	90.42	43.38	93.47	82.90	88.65	83.17	90.06	53.33	95.07	68.96
Wine	95.97	48.13	96.44	25.60	95.71	67.94	92.93	74.80	91.47	87.11	93.39	89.43	95.43	95.65	95.62	51.50	93.04	91.88
Yeast	56.76	64.03	58.08	51.81	53.32	69.97	56.18	82.86	54.51	88.10	49.00	65.57	51.98	92.35	56.15	75.66	54.55	74.05
Average	**78.47**	**57.69**	**77.37**	**31.31**	**77.13**	**69.89**	**75.09**	**79.47**	**77.14**	**78.09**	**77.05**	**80.02**	**77.09**	**88.97**	**79.25**	**59.77**	**79.75**	**83.72**

Table 6-III

Comparison of CBP against all other algorithms, with accuracy (Acc) and condensation (Cond) shown in the rows. The **PI** columns correspond to the percentage improvement (positive) or percentage deterioration (negative) score, calculated as: (*DWO_score – other_score) / other_score × 100*), where the scores are taken to be the average classification or condensation scores across all datasets (last row of Table II). The columns marked as **p** correspond to the p-values of the Wilcoxon sign-rank test at 0.01 significance, with the null hypothesis that the scores distributions for all individual datasets have equal medians.

	HMN		LIF		CCIS		DROP3		ICF		ICPL		CBP		SIR	
	PI(%)	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***	**PI(%)**	***p***
Acc	+1.63	0.07	+3.08	0.36	+3.40	0.0014	+6.21	0.0006	+3.34	0.0056	+3.50	0.0003	+3.45	0.0074	+0.63	0.7439
Cond	+45.12	0.0020	+167.39	0.0020	+19.79	0.0003	+5.35	0.25	+7.21	0.1570	+4.62	0.0778	-5.90	0.0033	+40.07	0.0002

6.4 Conclusions

Instance weight learning is a widely used technique in the field of machine learning [Zuo08, Mod03, Lan09, Qu08]. In order to deal with the problems arising in instance-based learning, a framework is proposed that employs instance-weight modelling to determine a small subset of the original prototypes that can accurately represent the entire dataset. The novelty of the Direct Weight Optimization algorithm lies on the binary weights that control the preservation or removal of instances. These weights are computed by optimizing an objective function (Eq. 6.5) that involves the ratio between the nearest friend and the nearest enemy of every instance. An instance of the training set is correctly classified as long as this ratio is less than one. Hence, the rationale behind the design of DWO is to keep this ratio to a minimum, in order to guarantee that the majority of sample patterns are not misclassified by the nearest neighbour rule. A major advantage of the proposed method is the fact that the optimization procedure is performed by a genetic algorithm. Not only because GAs can find good quality solutions, but also because the various parameters of the GA offer DWO the ability to adjust its performance according to the needs of the user and the application at hand.

As experiments showed the proposed algorithm manages very competitive results as it yields very high reduction rate, the second best between the tested algorithms, without compromising classification accuracy. It is the best performing method in terms of accuracy since it surpasses even accuracy enhancing methods like ENN. It should also be mentioned that the proposed heuristics can efficiently accelerate DWO for large datasets, compensating for the increased complexity of the optimization component. On the downside, although there does not seem to be correlation between classification error and dimensionality, like other methods that use distances between samples, it may fail to remove the right instances in highly dimensional spaces due to the distance concentration effect. Finally, for small datasets that their size is significantly lower than t_{max}xp_{size} (number generations and population size), DWO is rather slow despite the accelerating enhancements.

Unlike other methods, DWO models both objectives of classification accuracy and data condensation explicitly in a simple but very effective manner. So this work displays not only high accuracy results, as expected by selection techniques, but also very good condensation results. Experiments showed that DWO deals extremely well with the issue of condensation in selection algorithms, as the reduction rate achieved is equivalent if not better than the ones displayed by abstraction algorithms like CBP and ICPL.

Since DWO displayed the best accuracy, future work should aim on improving

its performance in terms of condensation and time requirements. The design of further heuristics, similar to the ones already introduced, can be beneficial in terms of speeding up the optimization process.

CHAPTER 7:

EPILOGUE

7.1 Conclusions

This thesis presented a comprehensive survey on instance reduction algorithms, including instance selection techniques as well as prototype abstraction algorithms. It also proposed four original algorithms, namely Class Boundary Preserving algorithm, Spectral Instance Reduction, Instance Seriation for Prototype Abstraction and Direct Weight Optimization algorithm, which can efficiently deal with the problems instance-based learning faces. Apart from the obvious advantages every algorithm displayed as analyzed in the respective chapters (i.e. the accuracy of DWO, the condensation ratio of the CBP algorithm or the overall performance of SIR that exhibited high accuracy, competent condensation as well as very good speed) the major contribution of this thesis is the novelty of the frameworks proposed. All methods designed offer new possibilities and provide new insights in the field of prototype reduction.

First of all, CBP is a powerful heuristic that determines the geometric structure of patterns around every sample, which is represented as the cosine score, in order to determine which instances should be discarded. It aims at identifying border samples, since the major information required to effectively describe the underlying distribution is held by samples close to class boundaries. Qualitatively, CBP exhibits the highest condensation ratio between all the tested algorithms, in the region of 89%, without compromising classification accuracy that is competent at an average of 77.09%.

Compared to other algorithms, DWO outperforms all others in terms of classification accuracy with an average of 79.75%, while it yields the second best reduction rate (83.72%), behind only CBP. But, the most important contribution of DWO is the instance-weight modelling proposed that employs a number of parameters (binary weights) focusing on maintaining misclassifications to a minimum. This is achieved with the appropriate objective function that is based on the ratio of the distances of a sample to

its nearest friend over its nearest enemy instance. Another major advantage of this algorithm is the objective function of the model, along with the parameters of the optimizing component that can be modified to fit the needs of the application at hand.

The SIR algorithm displays relatively high accuracies, third best as only DWO and ENN surpass its average of 79.25%, along with competent condensation results, as nearly 60% of the original samples are removed, and very good speeds (lower complexity than DWO and CBP). The novelty, though, of the specific framework is the totally new insight it offers to instance reduction with two major contributions. Firstly, the border discriminating features that capture different types of local geometric characteristics of the data samples, in terms of their friend and enemy profiles. Secondly, the spectral graph theory that is employed in order to partition between instances close to the decision surface, hence, worthwhile maintaining, and internal ones that should be removed. Essentially, samples are projected to a new space, which favors the separation between them.

Even ISPA, which can be consider as an early draft of SIR since it was designed in an effort to create some sort of ordering of sample patterns, has its own slight contribution to the field of prototype reduction. Although seriation has been known for over 50 years, it has only recently been used in machine learning and more specifically, in pattern recognition. So, it should be mentioned that despite the lack of competent results, as ISPA cannot be compared to the other very successful reduction algorithms designed or implemented in this thesis, it is the first method that tried to employ seriation on data reduction. It focused on organizing the training set in a way that favors direct merging of prototypes.

Table 7-I

Comparison of the three proposed algorithms in terms of classification accuracy, condensation ratio and computational complexity. Algorithms are numbered from the best performing (1) to the least (3). The complexity of DWO varies with the parameters of the GA and has lower complexity than CBP for larger datasets and higher for small datasets.

	Accuracy	**Condensation**	**Complexity**
CBP	3	1	3 (2)
SIR	2	3	1
DWO	1	2	2 (3)

Finally, many statistics have been introduced [Dem06, Alp99] to decide which algorithm is better, and various methods have been proposed to determine which feature is most important [Wit08], but every method will conclude on a different result. A simple comparison between the three main schemes in terms of classification accuracy, condensation and computational complexity can be seen in Table 7-I. But, as explained in [Tri11] and [Gar10] for real-world applications, the final judgment often relies on what the application prioritizes and on the needs of the user. For example, if high condensation were required, setting accuracy and speed aside, the best solution would be CBP. On the other hand, if a robust and fast method is needed, then a worse performance in condensation can be traded off against a slightly better accuracy obtained in the lower possible time. So, SIR would be ideal. If accuracy is critical, DWO is the obvious choice. So, the final choice comes down to the application at hand.

To conclude, this thesis provided a thorough analysis of instance selection and prototype abstraction techniques and identified the advantages and disadvantages of each group respectively. Algorithms that generate new prototypes display high condensation ratios along with large time requirements, similar to CBP. On the other hand, selection methods are generally faster and highly accurate, as proven by SIR and other similar algorithms. So this work proposed novel techniques that not only improved the strong points of each group respectively, but also introduced DWO that enhances accuracy of already existing selection algorithms while simultaneously strengthens their weakness in terms of reduction rate.

7.2 Future Work

In this thesis four different algorithms have been presented and as experiments showed, all of them, apart from their advantages exhibit some limitations. In this section, possible extensions are presented that can improve the proposed work.

First of all, CBP, as already explained, is a reduction method that consists of multiple heuristics. Therefore, extensions on this method can be done in the form of adding extra stages or replacing already existing ones. One of the major drawbacks of the CBP algorithm is its high complexity that leads to large response times. One improvement would be to substitute the initial noise filter (ENN) that is very computationally expensive. A simpler and faster filtering process would largely benefit CBP. Another improvement that could have a significant effect on the classification accuracy and the condensation ratio of CBP is the use of an adaptive threshold τ instead of the fixed one that is currently used. Although having a threshold that adjusts to the

processed dataset could prove to be valuable for the overall performance, it could largely increase the computational requirements of the method.

Regarding the ISPA algorithm proposed in chapter 5, the whole abstraction component would have to be restructured in order to obtain competitive results. Since the solely contribution of this algorithm is the ordering of the distance matrix, this is the only stage of the method that is worth further investigation. On the other hand, for the SIR algorithm that is described in the same chapter, future work could include its improvement in terms of condensation by combining it with instance abstraction. It has been proven that the majority of abstraction methods [$Lam02_a$, $Lam02_a$, Tri11] display large reduction rates. Another enhancement would be to use incremental updates of the partitioning eigenvector for applications where new samples are frequently generated. This could prove to be very beneficial for the performance of SIR. Despite the successful operation of the border discriminating features, they present an aspect of the algorithm that can efficiently and easily be improved. One such case would be to better tailor them to the employed classifier, or to adapt the cardinality of friends and enemies for the different BDFs according to the characteristics of the individual datasets.

Finally, the DWO technique introduced in chapter 6 can be considered to be the best performing algorithm from the tested ones as it displayed the best overall performance with the highest possible accuracy while it outperformed all other methods but CBP in terms of condensation ratio. Therefore, speeding up the process would be the most advantageous enhancement. This could be done by improving the already proposed accelerating heuristics or by designing new better suited ones.

All these suggestions involve the proposed algorithms and can simply enhance their operation. There exist, though, two areas in data mining that it would be useful to work on in the future. The first topic is the specific case of imbalanced datasets, which has been addressed [Lau01, Cha02, Cha03, Cha05, Mal03, and Bat04] but not investigated in depth. The second one, involves simultaneous instance as well as dimensionality reduction. Similar to the idea analyzed in [Vil08], research should be focused on combining the two data mining techniques in order to achieve condensation of the original training set in both directions at the same time.

REFERENCES

[Aha91] Aha, D.W., Kibler, D. and Albert, M.K., 1991. "Instance-based learning algorithms." *Machine Learning*, 6, pp. 37 – 66.

[Ald94] Alder, M., and McKenzie, P., 1994. "Unsupervised learning: The dog rabbit strategy." In *IEEE Int. Conf. on Neural Networks.*

[Alp99] Alpaydin, E., 1999. "Combined 5x2 cv F test for comparing supervised classification learning algorithms." *Journal Neural Computation*, vol. 11, pp. 1885-1892.

[And02] Andrew, R.W., 2002. *Statistical Pattern Recognition*. 2nd Edition. John Wiley and Sons Ltd.

[Ang07$_a$] Angiulli, F., 2007. "Fast nearest neighbour condensation for large data sets classification." *IEEE Trans. Knowledge and Data Eng.,* vol. 19, pp.1450-64.

[Ang07$_b$] Angiulli, F., 2007. "Distributed nearest neighbour-based condensation of very large data sets." *IEEE Trans. Knowledge and Data Eng.,* vol. 19, pp. 1593-1606.

[Ast70] Astrahan, M.M., 1970. "Speech analysis by clustering, or the hyperphoneme method." *Stanford A. I. Project Memo*, Stanford University, California.

[Atk97] Atkeson, C.G., Moore, A.W., and Schaal, S., 1997. "Locally weighted learning." *Artificial Intelligence Review*, vol. 11, pp. 11-73.

[Bab96] Babich, G.A., and Camps, O.I., 1996. "Weighted parzen windows for pattern classification." *IEEE Trans. Pat. Anal. Mach. Intel,* vol.18, pp. 567-70.

[Bar05] Barandela, R., Ferri, F.J. and Sanchez, J.S., 2005. "Decision boundary preserving prototype selection for nearest neighbour classification." *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, pp. 787-806.

[Bar05] Barandela, R., Ferri, F.J., and Sanchez, S.J., 2005. "Decision boundary preserving prototype selection for nearest neighbour classification." *Int. Journal of Pattern Recognition and Artificial Intelligence,* vol. 19, pp. 787-806.

[Bat04] Batista, G.E.A.P.A., Prati, R.C., and Monard, M.C., 2004. "A study of the behavior of several methods for balancing machine learning training data." *ACM SIGKDD Explorations Newsletter*, vol. 6, pp. 20-29.

[Bea03] Beal, M.J., 2003. "Variational algorithms for approximate bayesian inference." *PhD Thesis*, Gatsby Computational Neuroscience Unit, University College London, UK.

[Bel02] Belkin, M., and Niyogi, P., 2002. "Laplacian eigenmaps and spectral techniques for embedding and Clustering." In *Advances in neural information processing systems*, vol. 14, T.K. Leen, T.G. Dietterich, & V. Tresp Editions, Cambridge, MA: MIT Press.

[Bel03] Belkin, M., and Niyogi, P., 2003. "Laplacian eigenmaps for dimensionality reduction and data representation." *Neural Computation*, vol. 15, pp.1373-1396.

[Ber00] Bermejo, S. and Cabestany, J., 2000. "Adaptive soft *k*-nearest-neighbour classifiers." *Pattern Recognition*, vol. 33, pp. 1999-2005.

[Bez02] Bezdek, J.C., and Hathaway, R.J., 2002. "VAT: A tool for visual assessment of (cluster) tendency." In *Proc. of 2002 Int. Joint Conference on Neural Networks*, vol. 1-3, pp. 2225-2230.

[Bez07] Bezdek, J.C., Hathaway, R.J., and Huband, J.M., 2007. "Visual assessment of clustering tendency for rectangular dissimilarity matrices." *IEEE Trans. on Fuzzy Systems*, vol. 15, pp. 890-903.

[Bez81] Bezdek, J.C., 1981. *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, Norwell, USA.

[Bez98$_a$] Bezdek, J.C., and Kuncheva, L.I., 1998. "Nearest prototype classification: clustering, genetic algorithms, or random search?" *IEEE Trans. Syst., Man., Cybern.,* vol.28, pp. 160-164.

[Bez98$_b$] Bezdek, J.C., et al. 1998. "Multiple-prototype classifier design." *IEEE Trans. Syst., Man., Cybern.,* vol.28, pp. 67-79.

[Bha05] Bhattacharya, B., Mukherjee, K., and Toussaint, G., 2005. "Geometric decision rules for instance-based learning problems." In *Proc. of 1st Int. Conf. on Pattern Recognition and Machine Intelligence,* Kolkata, India.

[Bis06] Bishop, C.M., 2006. *Pattern Recognition and Machine Learning*. New York: Springer Science and Business Media, LLC.

[Bla98] Blake, M., 1998. "UCI Repository of machine learning databases." www.ics.uci.edu/~mlearn/, Irvine, CA, University of California, Department of Information and Computer Science.

[Bri02] Brighton, H., and Mellish, C., 2002. "Advances in instance selection for instance-based learning algorithms." *Data Mining and Knowledge Discovery,* vol.6, pp. 153-72.

[Bri99] Brighton, H., and Mellish, C., 1999. "On the consistency of information filters for lazy learning algorithms." *In Principles of 3rd Conference on Data Mining and Knowledge Discovery.* Prague, Czech Republic.

[Can05] Cano, J.R., Herrera, F., and Lozano, M., 2005. "Stratification for scaling up evolutionary prototype selection." *Pattern Recognition Letters*, vol. 26, pp. 953-963.

[Can06] Cano, J.R., Herrera, F., and Lozano, M., 2006. "A study on the combination of evolutionary algorithms and stratified strategies for training set selection in data mining." *Applied Soft Computing*, vol. 6, pp. 323-332.

[Cao08] Cao, H., et al. 2008. "Enhancing effectiveness of density-based outlier mining." In *Int. Symp. On Information Processing.*

[Cha02] Chawla, N.V., Bowyer, K.W., Hall, L.O., and Kegelmeyer, W.P., 2002. "SMOTE: Synthetic minority over-sampling technique." *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357.

[Cha03] Chawla, N.V., 2003. "C4.5 and imbalnced data sets: Investigating the effect of sampling method, probabilistic estimate, and descision tree structure." In *Proc. of ICML Workshop on Class Imbalances*.

[Cha05] Chawla, N.V., 2005. "Data mining for imbalanced datasets: An overview." *Data Mining and Knowledge Discovery Handbook*, vol. 6, pp. 853-867.

[Cha06] Chang, H., Yeung, D.Y., and Cheung, W.K., 2006. “Relaxation metric adaptation and its application to semi-supervised clustering and content-based image retrieval.” *Pattern Recognition*, vol. 39, pp. 1905-1917.

[Cha74] Chang, C., 1974. “Finding prototypes for nearest neighbour classifiers.” *IEE Trans. On Computers,* vol. C-23, pp. 1179-84.

[Cha96] Chaudhuri, B.B., 1996. “A new definition of neighbourhood of a point in multi-dimensional space.” *Pattern Recognition Letters,* vol. 17, pp. 11-17.

[Che07] Chen, T.S., Chiu, Y.H., and Lin, C.C., 2007. “Fast nearest neighbour classification using class-based clustering.” In *Proceedings of 6th Int. Conf. on Machine Learning and Cybernetics.*

[Che95] Cheng, Y., 1995. “Mean shift, mode seeking, and clustering.” *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 17, pp. 790-799.

[Chi79] Chidananda, G.K., and Krishna, G., 1979. “The condensed nearest neighbour rule using the concept of mutual nearest neighbourhood.” *IEEE Trans. On Information Theory*, vol.25, no. 4, pp. 488–490.

[Cho06] Chou, C.H., Kuo, B. H., and Chang, F., 2006. “The generalized condensed nearest neighbour rule as a data reduction method.” In *Proceedings of 18th International Conference on Pattern Recognition.*

[Cla04] Clauset, A., Newman, M.E.J., and Moore, C., 2004. “Finding community structure in very large networks.” *Physics Rev.*, E70, 066111.

[Com02]Comaniciu, D., and Meer, P., 2002. “Mean shift: A robust approach toward feature space analysis.” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5.

[Cow99]Cowgill, M., Harvey, R. and Watson, L., 1999. “A genetic algorithm approach to cluster analysis.” *Comput. Math. Appl.*, vol. 37, pp. 99-108.

[Cra02] Crammer, K., Bachrach, R.G., and Navot, A., 2002. “Margin analysis of the LVQ algorithm.” *Advances in Neural Information Processing Systems*, pp. 462-469.

[Cza04] Czarnowski, I., Jedrzejowicz, P., 2004. “An approach to instance reduction in supervised learning.” *Research and Development in Intelligent Systems*. Coenen, F., Preece, A., and Macintosh, A., Springer Editions, vol. XX, pp. 267-282.

[Cza10] Czarnowski, I., 2010. “Prototype selection algorithms for distributed learning.” *Pattern Recognition*, vol. 43, pp. 2292-2300.

[Das94] Dasarathy, B.V., 1994. “Minimal consistent set (MCS) identification for optimal nearest neighbour decision systems design.” *IEEE Trans. Syst., Man., Cybern.,*vol.24, No. 3, pp.511-517.

[Deh07] Dehzangi, O. et al., 2007. “An efficient nearest neighbour classifier using an adaptive distance measure.” In *Proceedings of* 12th International Conference on Computer Analysis of Images and Pattern, pp. 970-978.

[Dem06] Demsar, J., 2006. “Statistical comparisons of classifiers over multiple data sets.” *Journal of Machine Learning Research*, vol. 7, pp. 1-30.

[Der10] Derrac, J., Garcia, S., and Herrera, F., 2010. “IFS-CoCo: Instance and feature selection based on cooperative coevolution with nearest neighbour rule.” *Pattern Recognition*, vol. 43, pp. 2082-2105.

[Dev80] Devijver, P.A., and Kittler, J., 1980. “On the edited nearest neighbour rule.” In *Proc. of 5th Int. Conf. on Pattern Recognition*, pp. 72-80.

[Din01] Ding, C.H.Q., He, X.F., Zha, H.Y., Gu, M. and Simon, H. D., 2001. “A min-max cut algorithm for graph partitioning and data clustering.” In *Proc. IEEE International Conference on Data Mining*, pp. 107-114.

[Dom02] Domeniconi, C., Peng, J., and Gunopulos, D., 2002. “Locally adaptive metric nearest neighbour classification.” *IEEE Trans. on Patt. Anal. Mach. Intell.*, vol. 24, pp. 1281-1285.

[Dom05] Domeniconi, C., Gunopulos, D., and Peng, J., 2005. “Large margin nearest neighbour classifiers.” *IEEE Trans. on Neural Networks*, vol. 16, pp. 899-909.

[Don00] van Dongen, S., 2000. “Graph clustering by flow simulation.” *PhD Thesis*, Dutch National Research Institute for Mathematics and Computer Science, University of Utrecht, Netherlands.

[Don73] Donath, W., and Hoffman, A., 1973. “Lower bounds for the partitioning of graphs.” *IBM Journal Res. Dev.*, vol. 17, pp. 420-425.

[Dud01] Duda, R.D., Hart, P.E. and Stork, D. G., 2001. *Pattern Classification.* 2nd Edition. New York: John Wiley & Sons, Inc.

[Dui08] Duin, R.P.W., and Pekalska, E., 2008. “On refining dissimilarity matrices for an improved NN learning.” In *19th Int. Conf. on Pattern Recognition (ICPR)*, Tampa, USA.

[Dun74] Dunn, J.C., 1974. “A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters.” *Journal of Cybernetics*, vol. 3, pp. 32-57.

[Eve01] Everitt, B., Landau, S., and Leese, M., 2001. *Cluster Analysis*. London: Arnold.

[Fay09] Fayed, H.A. and Atiya, A.F., 2009. “A novel template reduction approach for the k-nearest neighbour method.” *IEEE Trans. on Neural Networks*, vol. 20, pp. 890-896.

[Fer07] Fernandez, F., and Isasi, P., 2007. “Local feature weighting in nearest prototype classification.” *IEEE Trans. Neural Networks*, pp. 40-53.

[Fil08] Filippone, M., Camastra, F., Masulli, F., and Rovetta S., 2008. “A survey of kernel and spectral methods for clustering.” *Pattern Recognition*, vol. 41, pp. 176-190.

[Fis36] Fisher, R.A., 1936. “The use of multiple measurements in taxonomic problems.” *Annals of Eugenics*, pp. 179-188.

[For10] Fortunato, S., 2010. “Community detection in graphs.” *Physics Reports*, 486, pp. 75-174.

[Fuk75] Fukunaga, K., 1975. “The estimation of the gradient of a density function, with applications in pattern recognition.” *IEEE Trans. Information Theory*, vol. 21, pp. 32-40.

[Gal05] Gallagher, M., and Frean, M., 2005. “Population-based continuous optimization, probabilistic modelling and mean shift.” *Journal Evolutionary Computation*, vol. 13, pp. 29-42.

[Gar08] Garain, U., 2008. “Prototype reduction using an artificial immune model”. *Pattern Analysis Applications*, vol. 11, pp. 353-363.

[Gar09] Garcia, A.H., and Pedrajas, N.G., 2009. “A divide-and-conquer recursive approach for scaling up instance selection algorithms.” *Data Mining and Knowledge Discovery*, vol. 18, pp. 392-418.

[Gar10] Garcia, S., Cano, J.R., and Herrera, F., 2008. “A memetic algorithm for evolutionary prototype selection: A scaling up approach.” *Pattern Recognition*, vol. 41, 2693-2709.

[Gar10] Garcia, S., Derrac, J., Cano, J.R., and Herrera, F., 2010. “Prototype selection for nearest neighbour classification: Taxonomy and empirical study.” *IEEE Trans. Pat. Anal. Mach. Intel.* (in press).

[Gat72] Gates, G.W., 1972. “The reduced nearest neighbour rule.” *IEEE Trans. On Information Theory*, pp. 431 – 433.

[Gev91] Geva, S., and Sitte, J., 1991. “Adaptive nearest neighbour pattern classification.” *IEEE Trans. On Neural Networks,* vol. 2, pp. 318-22.

[Gir02] Girolami, M., 2002. “Mercer kernel-based clustering in feature space.” *IEEE Trans. on Neural Networks*, vol. 13, pp. 780-784.

[Gir02] Girvan, M., and Newman, M.E.J., 2002. “Community structure in social and biological networks.” In *Proc. Natl. Acad. Sci.*, USA, pp. 7821-7826.

[Gir03] Girolami, M., and He, C., 2003. “Probability density estimation from optimally condensed data samples.” *IEEE Transactions on Pattern Analysis and Machine Inteligence,* vol.25, pp. 1253-1264.

[Gou07] Goulermas, J.Y. et al. 2007. “Density-driven generalized regression neural networks (DD-GRNN) for function approximation.” *IEEE Trans. on Neural Networks*, vol.18, pp.1683-1696.

[Gou11] Goulermas, J.Y., 2011. *Evolutionary Optimisation*, lecture notes distributed in Computational Intelligence II ELEC475/675 at University of Liverpool, UK.

[Gre07] Gregory, S., 2007. “An algorithm to find overlapping community structure in networks.” In *Proc. 11th European Conf. on Princ. Pract. Knowl. Disc. Dat., PKDD 2007*, Springer-Verlag, Berlin, Germany.

[Guo05] Guo, G., et al., 2005. “Similarity-based data reduction and classification.” In International Workshop on Monitoring, Security, and Rescue Techniques in Multiagent Systems, pp. 227-238.

[Hah08] Hahsler, M., Hornik, K., and Buchta, C., 2008. “Getting things in order: An introduction to the R package seriation.” *Journal of Statistical Software*, vol. 25, pp. 1-34.

[Har68] Hart, P.E., 1968. “The condensed nearest neighbour rule.” *IEEE Trans. On Information Theory,* pp. 515-516.

[Has96] Hastie, T., and Tibshirani, R., 1996. “Discriminant adaptive nearest neighbour classification.” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 18, pp. 607-615.

[Hay99] Haykin, S.S., 1999. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, London, UK.

[Haz07] Hazelton, M.L., and Turlach, B.A., 2007. “Reweighted kernel density estimation.” *Computational statistics and Data Analysis*, vol. 51, pp. 3057-3069.

[He02] He, X., and Niyogi, P., 2004. “Locality preserving projections.” *Advances in Neural Information Processing System*, vol. 16, Cambridge, MA: MITPress.

[Hil05] Hill, E.J., Alder, M., and deSilva, C.J.S., 2005. “An improvement to the DR clustering algorithm.” *Pattern Recognition Letters*, vol. 26, pp. 101-107.

[Hof08] Hofman, J.M., and Wiggins, C.H., 2008. “Bayesian approach to network modularity.” *Phy. Rev. Lett.*, vol. 100 (25), 258701.

[Hu08] Hu, Q., Yu, D., and Xie, Z., 2008. “Neighbourhood classifiers.” *Expert Systems with Application*, vol. 34, pp. 866-876.

[Hua06] Huang, D. and Chow, T.W.S., 2006. “Enhancing density-based data reduction using entropy.” *Journal of Neural Computation*, vol. 18, pp.470-495.

[Hua07] Huang, X., et al. 2007. “Weighted kNN model-based data reduction and classification.” In 4th International Conference on Fuzzy Systems and Knowledge Discovery.

[Hub05] Huband, J.M., Bezdek, J.C., and Hathaway, R.J., 2005. “bigVAT: Visual assessment of cluster tendency for large data sets.” *Pattern Recognition*, 38, pp. 1875-1886.

[Ich79] Ichino, M., 1979. “A nonparametric multiclass pattern classifier.” *IEEE Trans. on Systems, Man, and Cybernetics*, vol. smc-9, pp.345-352.

[Jan04$_a$] Jankowski, N., and Grochowski, M., 2004. “Comparison of instances selection algorithms I. Algortihms survey.” In Proc. of 7th Int. Conference on Artif. Intel. and Soft Comp., Zakopane, Poland.

[Jan04$_b$] Jankowski, N., and Grochowski, M., 2004. “Comparison of instances selection algorithms II. Results and comments.” In Proc. of 7th Int. Conference on Artif. Intel. and Soft Comp., Zakopane, Poland.

[Jar92] Jaromczyk, J.W., and Toussaint, T., 1992. “Relative neighbourhood graphs and their relatives.” In *IEEE Proceedings,* vol. 80, pp. 470-479.

[Jor99] Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., and Saul, L.K., 1999. “An introduction to variational methods for graphical models.” *Machine Learning*, vol. 37, pp. 183-233.

[Kan08] Kang, P. and Cho, S., 2008. “Locally linear reconstruction for instance-based learning.” *Pattern Recognition*, vol. 41, pp. 3507-3518.

[Kay93] Kay, S.M., 1993. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall. Chapter 7.

[Ken69] Kendall, D.G., 1969. “Incidence matrices, interval graphs and seriation in archaeology.” *Pacific Journal of Mathematics*, vol. 28, pp. 565-570.

[Ken71] Kendall, D.G., 1971. “Abundance matrices and seriation in archaeology.” Wahrscheinlichkeitstheorie, pp. 104-112.

[Kim03] Kim, S.W., and Oommen, B.J., 2003. “Enhancing prototype reduction schemes with LVQ3-type algorithms.” *Pattern Recognition,* vol. 36, pp. 1083-93.

[Koh86] Kohonen, T. 1986. “Learning vector quantization for pattern recognition.” *Technical Report*, Helsinki University of Technology.

[Kok09] Kokiopoulou, E., and Saad, Y. , 2009. “Enhanced graph-based dimensionality reduction with repulsion Laplaceans.” *Pattern Recognition*, vol. 42, pp. 2392–2402.

[Kun95] Kuncheva, L.I., 1995. “Editing for the *k*-nearest neighbours rule by a genetic algorithm.” *Pattern Recognition Letters*, vol. 16, pp. 809-814.

[Lam02$_a$]Lam, W., Keung, C.K., and Ling, C.X., 2002. “Learning good prototypes for classification using filtering and abstraction of instances.” *Pattern Recognition,* vol. 35, pp. 1491-1506.

[Lam02$_b$]Lam, W., Keung, C.K., and Liu, D., 2002. "Discovering useful concept prototypes for classification based on filtering and abstraction." *IEEE Trans. on Pattern Analysis and Machine Inteligence,* vol.24, No 8, pp. 1075-1090.

[Lan09] Lan, M., Tan, C.L., Su, J., and Lu, Y., 2009. "Supervised and traditional term weighting methods for automatic test categorization." *IEEE Trans. Pattern Analysis Machine Learning*, vol. 31, pp. 721-735.

[Lat05] Latapy, M., and Pons, 2005. Lecture notes on computer science, 3733, pp. 284-293.

[Lau01] Laurikkala, J., 2001. "Improving identification of difficult small classes by balancing class distribution." In *Proc. 8th Conf. Artificial Intelligence Medicine*, pp. 63-66.

[Law06] Law, M.H.C., and Jain, A.K., 2006. "Incremental nonlinear dimensionality reduction by manifold learning." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, pp. 377-391.

[Lee05] Lee, J., and Lee, D., 2005. "An improved cluster labelling method for support vector clustering." *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 27, pp. 461-464.

[Li05] Li, J., Manry, M.T., Yu, C., and Wilson, R.D., 2005. "Prototype classifier design with pruning." *Int. Journal on Artificial Intelligence Tools,* vol. 14, pp. 261-280.

[Lin91] Lin, J., 1991. "Divergence measures based on the Shannon entropy." *IEEE Trans. On Information Theory*, vol. 37, no 1, pp. 145-151.

[Loz03] Lozano, M.T., Sanchez, J.S., and Pla, F. 2004. "Using the geometrical distribution of prototypes for training set condensing." In 10th Conference of the Spanish-Association-for-Artificial-Intelligence, pp. 618-627.

[Lux07] von Luxburg, U., 2007. "A tutorial on spectral clustering." *Statistics and Computing*, vol. 17, pp. 395–416.

[Mac67] MacQueen, J., 1967. "Some methods for classification and analysis of multivariate observations." in *Proc. 5th Berkeley Symp.*, vol. 1, pp. 281-297.

[Mal03] Maloof, M.A., 2003. "Learning when data sets are imbalanced and when costs are unequal and unknown." In *Proc. of ICML Workshop on Learning from Imbalanced datasets*.

[Man08] Mannila, H., 2008. "Finding total and partial orders from data for seriation." *Lecture Notes in Computer Science*, Springer.

[Mar08] Marchiori, E., 2008. "Hit miss networks with applications to instance selection." *Journal of Machine Learning Research,* vol.9, pp. 997-1017.

[Mar09] Marchiori, E., 2009. "Graph-based discrete differential geometry for critical instance filtering." In *Joint European Conf. on Machine Learning (ECML)/ European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD*), Bled, Slovenia.

[Mar10] Marchiori, E., 2010. "Class conditional nearest neighbour for large margin instance selection." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 364-370.

[Mau00] Maulik, U., and Bandyopadhyay, S., 2000. "Genetic algorithm-based clustering technique." *Pattern Recognition*, vol. 33, pp. 1455-1465.

[Mei01] Meila, M., and Shi, J., 2001. "A random walks view of sectral segmentation." In *8th International Workshop on Artificial Intelligence and Statistics*.

[Mit02] Mitra, P., Murthy, C.A., and Pal, S.K., 2002. "Density-based multiscale data condensation." *IEEE Trans. Pat. Anal. Mach. Intel.,* vol. 24, pp. 734-47.

[Miu08] Miu, T., 2008. "Machine learning for pattern classification." *PhD Thesis*, University of Liverpool, Liverpool, UK.

[Miu12] Miu, T., Goulermas, J.Y., Tsujii, J., and Ananiadou, S., 2012. "Proximity-based frameworks for generating embeddings from multi-output data." *IEEE Trans. Pattern Anal. Mach. Intell*, (in press).

[Mod03] Modha, D.S., and Spangler, W.S., 2003. "Feature weighting in *k*-means clustering." *Journal of Machine Learning*, vol. 52, pp. 217-237.

[Mol02] Mollineda, R.A., Ferri, F.j., and Vidal, E., 2002. "An efficient prototype merging strategy for the condensed 1-NN rule through class-conditional hierarchical clustering." *Pattern Recognition,*vol.35, pp. 2771-2782.

[Muh03] Muhlenbach, F., et al. 2004. "Identifying and handling mislabelled instances." *Journal of Intelligent Information Systems,* vol.22, pp. 89-109.

[Mur83] Murtagh, F., 1983. "A survey of recent advances in hierarchical clustering algorithms." *Computational Journal*, vol. 26, pp. 354-359.

[New04] Newman, M.E.J., and Girvan, M., 2004. "Finding and evaluating community structure in networks." *PhysicsRev.*, vol. E69, 026113.

[Ng02] Ng, A.Y., Jordan, M.I., and Weiss, Y., 2002. "On spectral clustering: Analysis and an algorithm." *Advances in neural information processing systems*, vol. 14, pp. 849–856.

[Ngu11] Nguyen, C.H., and Mamitsuka, 2011. "Discriminative graph embedding for label propagation." *IEEE Trans. Neural Networks*, vol. 22 (9), pp. 1395-1405.

[Nik10] Nikolaidis, K., Rodriguez, M.E., Goulermas, J.Y., and Wu, Q.H., 2010. "Instance seriation for prototype abstraction." In *Proc. of IEEE 5th BICTA International Conference*, Liverpool, pp. 1351-1355.

[Nik11] Nikolaidis, K., Goulermas, J.Y., and Wu, Q.H., 2011. "A class boundary preserving algorithm for data condensation." *Pattern Recognition*, vol.44, pp. 704-715.

[Nik12] Nikolaidis, K., Rodriguez, M.E., Goulermas, J.Y., and Wu, Q.H., 2012. "Spectral graph optimization for instance reduction." *IEEE Trans. Neural Networks*, vol. 23, pp. 1169-1175.

[Owe84] Owen, A., 1984. "A neighbourhood-based classifier for LANDSAT data." *The Canadian Journal of Statistics*, vol. 12, pp. 191-200.

[Oze08] Ozertem, U., Erdogmus, D., and Jenssen, R., 2008. "Mean shift spectral clustering." *Pattern Recognition*, vol. 41, pp. 1924-1938.

[Pan99] Pan, V., and Chen, Z., 1999. "The complexity of the matrix eigenproblem." In *Proc. 31st ACM Symposium on Theory of Computing*, New York.

[Par00] Paredes, R., and Vidal, E., 2000. "Weighting prototypes. A new editing approach." *In Proc. 15th International Conference on Pattern Recognition (ICPR)*.

[Par06$_a$] Paredes, R., and Vidal, E., 2006. "Learning weighted metrics to minimize nearest neighbour classification error." *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 28, pp.1100-10.

[Par06$_b$] Paredes, R., and Vidal, E., 2006. “Learning prototypes and distance: A prototype reduction technique based on nearest neighbour error minimization.” *Pattern Recognition*, vol. 39, pp. 180-188.

[Ped08] Pedrajas, N.G., Castillo, J.A.R., and Boyer, D.O., 2008. “A cooperative coevolutionary algorithm for instance selection for instance-based learning.” *Machine Learning*, vol. 78, pp. 381-420.

[Pek06] Pekalska, E., Duin, P.W. and Paclik, P., 2006. “Prototype selection for dissimilarity-based classifiers.” *Pattern Recognition*, vol. 39, pp. 189-208.

[Pek08] Pekalska, E., and Duin, P.W., 2008. “Beyond traditional kernels: Classification in two dissimilarity-based representation spaces.” *IEEE Trans on Systems, Man, and Cybernetics*, vol. 38, pp.729-744.

[Qu08] Qu, C., Li, Y., Zhu, J., Huang, P., Yuan, R., and Hu, T., 2008. “Term weighting evaluation in bipartite partitioning for text clustering.” In *Proc. 4th Information Retrieval Technology Conf.*, pp. 393-400, Berlin, Germany.

[Ric99] Ricci, F., and Avesani, P., 1999. “Data compression and local metrics for nearest neighbour classification.” *IEEE Trans. Pat. Anal. Mach. Intel,* vol.21, pp. 380-4.

[Rie09] Riesen, K., and Bunke, H., 2009. “Dissimilarity based vector space embedding of graphs using prototype reduction schemes.” *Lecture Notes in Computer Science*, vol. 5632, pp. 617-631.

[Rit75] Ritter, G.L. et al., 1975. “An algorithm for a selective nearest neighbour decision rule.” *IEEE Trans. On Information Theory,* pp. 665 – 669.

[Rob51] Robinson, W.S., 1951. “A method for chronologically ordering archaeological deposits.” *American Antiquity*, vol. 16, pp. 293-301.

[Rod10] Rodriguez, M.E., Nikolaidis, K., Goulermas, J.Y., Ralph, J.F., and Miu, T., 2010. “Collaborative projection pursuit for face recognition.” In *Proc. of IEEE 5th BICTA International Conference*, Liverpool, UK.

[Row00] Roweis, S.T., and Saul, L.K., 2000. “Nonlinear dimensionality reduction by locally linear embedding.” *Science*, vol. 290, pp. 2323-2326.

[Rub08] Rubio, E.L., and Ortiz-de-Lazcano-Lobato, J.M., 2008. “Soft clustering for nonparametric probability density function estimation.” *Pattern Recognition Letters*, vol. 29, pp. 2085-2091.

[Rut07] Ruta, D., and Gabrys, B. 2007. “Reducing spatial data complexity for classification models.” *Comp. Meth. In Science and Eng., Theory and Computation: Old Problems and New Challenges.* vol.1, pp. 603-613.

[Sal91] Salzberg, S., 1991. “A nearest hyperrectangle learning method.” *Machine Learning*, vol. 6, pp. 277-309.

[San04] Sanchez, J.S., 2004. “High training set size reduction by space partitioning and prototype abstraction.” *Pattern Recognition,* vol. 37, pp. 1561-64.

[San06] Sanchez, J.S., and Marques, A. I., 2006. “An LVQ-based adaptive algorithm for learning from very small codebooks.” *Neurocomputing*, vol. 69, pp. 922-927.

[Sau03] Saul, L.K., and Roweiss, S.T., 2003. “Think globally, fit locally: Usupervised learning of low dimensional manifolds.” *Journal Machine Learning Research*, vol. 4, pp. 119-155.

[Sch08] Schuetz, P., and Caflisch, A., 2008. "Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement." *Physics Rev.*, vol. E77, 046112.

[Sil12$_a$] Silva, T.C., and Zhao, L., 2012. "Stochastic competitive learning in complex networks." *IEEE Trans. Neural Networks and Learning Systems*, vol. 23 (3), pp. 385-398.

[Sil12$_b$] Silva, T.C., and Zhao, L., 2012. "Network-based stochastic semisupervised learning." *IEEE Trans. Neural Networks and Learning Systems*, vol. 23 (3), pp. 451-466.

[Smo10] Smola, A., and Vishwanathan, S.V.N., 2010. *Introduction to Machine Learning*, University of Cambringe, Cambringe, UK.

[Smy95] Smyth, B., and Keane, M.T., 1995. "Remembering to forget." *In Proceeding of the 14th International Conference on Artificial Inteligence,* pp. 377 – 382.

[Son06] Son, S.H., and Kim, J.Y., 2006. "Data reduction for instance-based learning using entropy-based partitioning." In *Proc. Inter. Conf. Computational Science and its Applications*, pp.590-599.

[Spi04] Spielman, D.A., 2007. *Spectral Graph Theory and its Applications* lecture notes distributed in Applied Mathematics 500A at Yale University, USA.

[Spi07] Spielman, D.A., 2007. "Spectral Graph Theory and its Applications." In *48th IEEE Annual Symposium Foundations of Computer Science*, pp. 29-38.

[Spi96] Spielman, D.A., and Teng, S.H., 1996. "Spectral partitioning works: Planar graphs and finite element meshes." In *IEEE Symposium on Foundation of Computer Science*, pp. 96-105.

[Sta86] Stanfill, C. and Waltz, D., 1986. *Toward Memory-Based Reasoning Communications*. New York: ACM. vol 29, pp. 1213 – 1228.

[Sto79] Stoffel, J.C., 1974. "A classifier design technique for discrete variable pattern recognition problems." *IEEE Trans. Comput.*, vol. C-23, pp. 428-441.

[Sug06] Sugumaran, V., Muralidharana, V., and Ramachandrana, K.I. "Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing." *Mechanical Systems and Signal Processing*, 21, pp. 930-942.

[Sun08] Sun, L., Ji, S., and Ye, J., 2008. "Hypergraph spectral learning for multi-label classification." In *Proc. Of the 14th ACM SIGKDD International Conf. On knowledge Discovery and Data Mining*, Las Vegas, Nevada, USA, pp. 668-676.

[Sus02] Susheela Devi, V., and Narasimha, M., 2002. "An incremental prototype set building technique." *Pattern Recognition*, vol. 35, pp. 505-513.

[The99] Theodoridis, S., and Koutroumbas, K., 2006. *Pattern Recognition*. 3rd Edition. Academic Press, London, UK.

[Tom76] Tomek, I., 1976. "Two modifications of CNN." *IEEE Trans. Systems, Man., and Cybernetics*, pp. 769 – 772.

[Tou79] Toussaint, G.T., and Poulsen, R.S, 1979. "Some new algorithms and software implementation methods for pattern recognition research." In *Proc. IEEE Int. Computer Software Applications Conf.,* Chicago.

[Tou80] Toussaint, G.T., 1980. "The relative neighbourhood graph of finite planar set." *Pattern Recognition*, pp 261 – 268.

[Tri11] Triguero, I., Derrac, J., Garcia, S., and Herrera, F., 2011. "A taxonomy and experimental study on prototype generation for nearest neighbour classification." *IEEE Trans. Sys., Man. And Cyb.,* part C. (Accepted for publication)

[Tse01] Tseng, L., and Yang, S., 2001. "A genetic approach to the automatic clustering problem." *Pattern Recognition*, vol. 34, pp. 415-424.

[Ull74] Ullmann, J.R., 1974. "Automatic selection of reference data for use in a nearest-neighbour method of pattern classification." *IEEE Trans. On Information Theory*, pp. 541 – 543.

[Vil08] Villegas, M., and Paredes, R., 2008. "Simultaneous learning of a discriminative projection and prototypes for nearest-neighbour classification." In *IEEE Proc.on Conf. Computer Visionand Pattern Recognition (CVPR)*, Alaska, USA.

[Wan06] Wang, H., Cao, C., and Leung, H., 2006. "An improved locally weighted regression for a converter re-vanadium prediction modelling." In *Proceedings of 6th World Congress on Intelligent Control and Automation*, Dalian, China.

[Wan07$_a$] Wang, F., and Zhang, C., 2007. "Feature extraction by maximizing the average neighbourhood margin." *Comp. Vis. Patt. Rec.*, pp. 1-8, 17-22 June.

[Wan07$_b$] Wang, F., and Zhang, C., 2007. "Fast multilevel transduction on graphs." In *Proc. 7th SIAM Conf. on Data Mining(SDM)*, Minneapolis, USA.

[Wei09] Weinberger, J. and Saul, L.K., 2009. "Distance metric learning for large margin nearest neighbour classification." *Journal of Machine Learning Research*, vol. 10, pp. 207-244.

[Wil00] Wilson, D.R., and Martinez, T.R., 2000. "Reduction techniques for instance-based learning algorithms." *Machine Learning,* vol.38, pp. 257-286.

[Wil72] Wilson, D.L., 1972. "Asymptotic properties of nearest neighbour rules using edited data." *IEEE Trans. Systems, Man., and Cybernetics,* SMC-2(3), pp. 408-421.

[Wil97] Wilson, D., R., and Martinez, T., R., 1997. "Improved heterogeneous distance functions." *Journal of Artificial Intelligence Research,* 6, pp. 1-34.

[Wit08] Witten, D.M., and Tibshirani, R., 2008. "Testing significance of features by lassoed principal components." *The Annals of Applied Statistics*, vol. 2, pp. 986-1012.

[Wu02] Wu, Y., Ianakiev, K. and Govindaraju, V., 2002. "Improved *k*-nearest neighbour classification." *Pattern Recognition*, vol. 35, pp. 2312-2318.

[Wu07] Wu, K.L., and Yang, M.S., 2007. "Mean shift-based clustering." *Pattern Recognition*, vol. 40, 3035-3052.

[Xie11] Xie, B., Wang, M., and Tao, D., 2011. "Toward the optimization of normalized graph laplacian." *IEEE Trans. Neural Networks*, vol. 22 (4), pp. 660-666.

[Xin02] Xing, E.P., Ng, A.Y., Jordan, M.I., and Russell, S., 2002. "Distance metric learning, with application to clustering with side-information." *Advances in Neural Information Processing Systems 16 (NIPS2002)*, MIT Press, pp. 521-528.

[Xu05] Xu, R., Wunsch II, D, 2005. "Survey of clustering algorithms." *IEEE Trans. on Neural Networks*, vol. 16, pp. 645-678.

[Yan04] Yang, M.S., and Wu, K.L., 2004. "A similarity-based robust clustering method." *IEEE Trans. on Pattern AnalysisMachine Intelligence*, vol. 26, pp. 434-448.

[Yan07] Yan, S., Xu, D., Zhang, B., Zhang, H. J., Yang, Q., and Lin, S., 2007. "Graph embedding and extensions: A general framework for dimensionality reduction." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, pp. 40-51.

[Yu 08] Yu Su, Shiguang, S., Xilin, C., and Wen, G., 2008. "Classifiability-based optimal discriminatory projection pursuit." In *IEEE Conf. on Computer Vision and Pattern Recognition.*

[Yu05] Yu, J., 2005. "General C-Means Clustering Model." *IEEE Trans. Pattern Analysis ana Machine Learning Intelligence*, vol. 27, pp. 1197-1211.

[Zen09] Zeng, Z., Tung, A.K.H., Wang, J., Feng, J., and Zhou, L., 2009. "Comparing stars: on approximating grah edit distance." In *Proc. 35th International Conference on Very Large Data Bases (VLDB)*, Lyon, France.

[Zha08] Zhang, N., Wang, X.Z., and Xiao, T. 2008. "An instance selection algorithm based on contribution." In *Proceedings of 7th Int. Conf. on Machine Learning and Cybernetics.*

[Zha09] Zhang, T., Tao, D., Li, X., and Yang, J., 2009. "Patch alignment for dimensionality reduction." *IEEE Trans. On Knowledge Data Engineering*, vol. 21, pp. 1299-1313.

[Zha10$_a$] Zhang, T., Fang, B., Tang, Y.Y., Shang, Z., and Xu, B., 2010. "Generalized discriminant analysis: A matrix exponential approach." *IEEE Trans. Sys., Man. And Cyb.,* vol. 40, pp. 186-197.

[Zha10$_b$] Zhang, T., Huang, K., Li, X., Yang, J., and Tao, D., 2010. "Discriminative orthogonal neighbourhood-preserving projections for classification." *IEEE Trans. Sys., Man. And Cyb.,* vol. 40, pp. 253-263.

[Zha92] Zhang, J., 1992. "Selecting typical instances in instance-based learning." In *Proc. Int. Conf. Machine Learning,* pp. 470-479.

[Zuo08] Zuo, W., Zhang, D., and Wang, K., 2008. "On kernel difference-weighted *k*-nearest neighbour classification." *Pattern Analysis Applications*, vol. 11, pp. 247-257.