LEABHARLANN CHOLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH | TRINITY COLLEGE LIBRARY DUBLIN
Ollscoil Átha Cliath | The University of Dublin

Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Aggregating Case-Based Reasoners in Ensembles: an Approach in support of Explanation

A Thesis submitted to the

University of Dublin, Trinity College

for the degree of

Doctor of Philosophy

Gabriele Zenobi

Department of Computer Science

Trinity College,

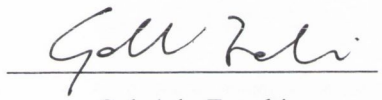
Dublin

April 2003

TRINITY COLLEGE
28 JAN 2004
LIBRARY DUBLIN
THESIS
7863

Declaration

I, the undersigned, declare that this work has not been previously submitted as an exercise for a degree to this or any other University, and that, unless otherwise stated, it is entirely my own work.

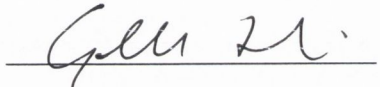


Gabriele Zenobi

April 2003

Permission to lend or copy

I, the undersigned, agree that the Trinity College Library may lend or copy this thesis upon request.



Gabriele Zenobi

April 2003

Acknowledgements

My first and most sincere gratitude goes to Prof. Pàdraig Cunningham, who during these years has been a constant source of help, suggestions, motivation and friendship. It would be hard to express in a few words how much I've learned from him, but it's not too hard to express that without him this thesis would certainly not exist.

My personal gratitude goes to my parents and my sister who have always given to me great support and encouragement, especially when we happened to be not so geographically close.

A sincere acknowledgement goes to all the people of the Medilink Project I had the pleasure to work with and learn from. A particular mention goes to Prof. Jane Grimson, Prof. Bill Grimson and Dr. Lucy Hederman that gave me the great chance to join such a highly skilled group of people, and to Sebastien Pardon and Mary Sharp, whose help to my work was truly appreciated.

I also wish to thank sincerely Dr. Paul Walsh and Dr. Stephen Byrne for the time they spent to supply me with their expertise for some of the evaluations in this research.

A special thank goes to all my colleagues in the A.I. Group of the Computer Science Department, for all they've done to make a lost Italian in Dublin look less lost: from technical suggestions to personal support, from English language assistance to pubs recommendations. In order of "appearance" in my Trinity-life film: Conor, Shane, Michelle, Joe, John, Francisco, Ulisses, Patrick, Brian, Kevin, Mick, Marco, Lorcan, Brian and Deirdre. A very special thank goes to Robert, whose outstanding technical skills have given more than a simple help to my whole research.

I also owe an acknowledgement to all the technical and administrative staff of the Computer Science Department, who has always been very kind and helpful in any circumstance.

A very special personal thank goes to all the people I have met during these years and with whom I have shared great moments. To Marco, for those hundreds "aglio e olio" in those hundreds rainy nights. To Paul, who has been such a great flatmate and such a great friend. To Susana, Francesca, Giuliana, Carole, George, I-Wei, Antonella, Micheal, Annarita, Massimo, Mariella and all the others that have contributed in making this time of my life unforgettable. To Rosalie. And to John and Viola O'Connor, for their infinite kindness and hospitality.

In any research project there are turning points. I owe Derek Bridge an acknowledgement for that refreshing conversation we had and that contributed so much in making me realize what was the right direction to develop this research.

Abstract

Among the reasons for the success Case-Based Reasoning (CBR) has achieved in tackling supervised learning problems, is certainly the capability to give a ranking to any case stored in the database depending on its similarity to the query and the subsequent possibility to retrieve a small set of cases to explain the predicted output. Many areas, like medical domains, electronic commerce applications, diagnosis tasks, recommender systems, greatly benefit from this characteristic of CBR.

Recent developments in Machine Learning research have shown that a number of predictors aggregated to form an Ensemble can be more accurate than a single predictor. This is true both for Eager Learning systems, like neural networks, and for lazy learning systems like Case-Based Reasoning and k-Nearest-Neighbours (k-NN). The aggregation step is normally performed by either simple voting or with a linear weighted average of the single classifiers' output.

However in the case of Case-Based Reasoning systems this increased accuracy comes at the cost of interpretability. The number of cases retrieved by an Ensemble to return a prediction in reply to a submitted query is dramatically larger (and so less interpretable) if compared to the number of cases retrieved by a single predictor. Furthermore, if cases are retrieved by ensemble members that differ on the feature representation, they are also intrinsically represented by a different semantic and cannot be compared. Thus there seems to be an apparent trade-off between performance improvement and interpretability deterioration when using an ensemble of CBR systems, deterioration that is a considerable loss if we consider explanation to be one of the most powerful CBR characteristics.

This thesis presents *Meta k-NN*, a new technique to aggregate ensembles of *k*-Nearest Neighbours that produces very accurate predictions and allows the identification of a small number of similar cases in support of explanation through a strategy that assigns coherent ranking to the cases. The technique is based on a two layered "nearest neighbour" approach. In the first layer a set of *k*-Nearest Neighbours classifiers are trained giving to each classifier a different feature subset; the second layer (aggregation) might be viewed as a transformation process whereby cases are transformed from their feature based representation to a representation based on the predictions of ensemble members.

The evaluation shows that *Meta k*-NN obtains accuracies either better or comparable to more traditional weighted average-based aggregation strategies, both in equally distributed and skewed distributed class domains; in this last case evaluation is based on the Harmonic Means of class accuracies and uses a modification of the standard *k*-NN algorithm. Also, the quality (usefulness) of the retrieved cases proved to be very good in an analysis performed by domain experts.

Table of Contents

Acknowledgements iv

Abstract vi

List of Figures xi

List of Tables xii

Chapter 1. Introduction 1

1.1. Problem Context and Motivations..... 1

1.2. Thesis Contributions..... 3

1.3. Related Publications..... 4

1.4. Structure of the Thesis..... 4

Chapter 2. Lazy Learning and Case-Based Reasoning 6

2.1. Lazy Learning and Eager Learning algorithms..... 6

2.2. Case-Based Reasoning and *k*-Nearest Neighbours..... 8

2.2.1. A formal description of k-Nearest Neighbours..... 9

2.2.2. Loss Functions..... 11

2.2.3. Distinctive Features of k-NN, CBR and Lazy Learning..... 12

2.3. Case-Based Reasoning in support of Explanation..... 14

2.3.1. CBR for Explanation in the Literature..... 15

2.4. Limitations of Case-Based Reasoning..... 17

2.5. Summary of the Chapter..... 18

Chapter 3. The Feature Selection problem 20

3.1. Introduction to Feature Selection and Weighting..... 20

3.2. The selection of the Best Feature Subset..... 24

3.2.1. Types of Evaluation Functions..... 25

3.2.2. Types of Search Algorithms..... 27

3.3. Knowledge Discovery and the Instability Problem..... 32

3.4. Summary of the Chapter..... 35

Chapter 4. Ensembles of Case-Based Reasoners 37

4.1. Introduction to Ensembles..... 37

4.1.1. Basic concepts and definitions..... 37

4.1.2. Different sources of diversity in the literature..... 41

4.2. Theoretical foundations of Ensemble techniques..... 43

4.2.1. Bias/Variance decomposition and the Overfitting problem 45

4.2.2. Ambiguity decomposition: the importance of Diversity..... 49

4.2.3. Ambiguity and Bias/Variance for ensembles of k-Nearest Neighbours classifiers..... 51

4.2.4. Measures of diversity for classification problems..... 53

4.3. Use of Ambiguity to select Ensembles of k-Nearest Neighbours 56

4.3.1. HC: The Default Search Hill Climbing strategy..... 57

4.3.2. AmbHC: a Hill Climbing Strategy that uses Ambiguity... 58

4.3.3. Evaluation and Discussion..... 60

4.4. Case Representation Issues from Ensemble Research..... 65

4.4.1. Single Best Feature Subset Versus Ensembles..... 65

4.4.2. Implications for CBR..... 66

4.5. Summary of the Chapter..... 68

Chapter 5 The *Meta k*-NN aggregation strategy 69

5.1. Limits of the use of Ensembles for Lazy Learning..... 69

5.2 The search for the "best"cases..... 71

5.2.1. Attempts to select representative cases through Ensembles 72

5.3. The *Meta k*-NN aggregation strategy..... 75

5.3.1. An example to introduce a different data representation... 76

5.3.2. The Meta k-NN algorithm 77
5.4. Considerations on the nature of *Meta k*-NN 79
5.4.1. Considerations on the problem representation of Meta k-NN 80
5.4.2. Considerations on possible extensions of Meta k-NN 83
5.4.3. Considerations on Computability Issues 83
5.5. Summary of the Chapter 84

Chapter 6. Evaluation and Discussion **86**

6.1. Evaluation on Datasets with equally distributed classes 87
6.1.1. Results of the Evaluation 88
6.1.2. Discussion 91
6.2. The Skewed Class Distribution Problem 93
6.2.1. A Modification of k-NN to deal with skewed class 94
6.2.2. Use of Harmonic Mean to Evaluate Performance 95
6.3. Evaluation on Datasets with Skewed Class Distribution 99
6.3.1. Results of the Evaluation 99
6.3.2. Discussion 100
6.4. Evaluation of the "quality" of the retrieved cases 107
6.5. Summary of the Chapter 112

Chapter 7. Conclusions and Future Work **114**

7.1. Thesis Summary and Conclusions 114
7.2. Suggestions for Future Work 117
7.2.1. Research Suggestions to develop the Meta k-NN approach 117
7.2.2. Research Suggestions to develop Ensemble analysis 118
7.2.3. Research Suggestions for the Feature Relevance Analysis. 118

Bibliography **120**

Appendix A. Datasets used in the Evaluations **133**

Appendix B. *Meta k*-NN: an example of Implementation **134**

List of Figures

Fig. 2.I. Voronoi diagram for a 1-Nearest Neighbour classifier 13

Fig. 3.I. The Filter Strategy 25

Fig. 3.II. The Wrapper Strategy 26

Fig. 4.I. An example of aggregation of 3-NN in Ensemble 39

Fig. 4.II. An example of low complexity (a) and high complexity (b) classifiers 47

Fig. 4.III. The principle of Averaging to reduce the error. 48

Fig. 4.IV. An example showing how diversity in the ensemble leads to better classification 53

Fig. 4.V. *HC:* the default selection strategy for generating ensembles using error only. 59

Fig. 4.VI. *AmbHC:* The algorithm for generating ensembles while emphasising diversity in ensemble members 58

Fig. 4.VII. Generalization error of different ensemble sizes on the Pima data.. 62

Fig.4.VIII. Generalization error of different ensemble sizes on the Warfarin data. 62

Fig. 4.IX. Generalization error of different ensemble sizes on the Heart data. 63

Fig. 5.I. An example of Pool of retrieved cases by an Ensemble 73

Fig. 5.II. The two Phases of Classification through the Meta *k*-NN aggregation strategy 79

Fig. 5.III. The Meta *k*-NN algorithm 80

Fig. 6.I Results for Heart data 89

Fig. 6.II Results for Pima data 89

Fig. 6.III Results for Warfarin data 90

Fig. 6.IV Results for Cylinder data 90

Fig. 6.V. Fidelity error of the *Meta k*-NN classifiers to the ensemble on which they are based. The 10% and 5% error lines are shown for comparison. 91

Fig. 6.VI. The *Duplicate k*-NN principle 95

Fig. 6.VII Harmonic Mean of two Accuracies 98

Fig. 6.VIII. Study on the Spect dataset 101

Fig. 6.IX. Study on the Ionosphere dataset 102

Fig. 6.X. Study on the Horse Colic dataset 103
Fig. 6. XI. Study on the Pima Indians dataset 104
Fig. 6.XII. Study on the Warfarin (3classes) dataset 105
Fig. 6.XIII. Evaluation of for the Bronchiolites dataset 111
Fig. 6.XIV. Evaluation of for the Warfarin dataset 111

List of Tables

Table 3.a. Average Hamming Distance of the chosen feature subset for classifiers trained in different subsets of the Case-Base 34

Table 4.a. An example with three 3-nearest neighbours classifiers and 5 data points. The top half of the table shows ensembled predictions allowing continuous values and the bottom half shows 0/1 predictions 54

Table 4.b. Error and ambiguity measures for the scenario shown in Table 4.a. 54

Table 4.c. A summary of all the evaluations showing the best ensemble generated for each data set and the corresponding average member error and ambiguity 63

Table 4.d. The ensembles built using *AmbHC* have significantly less features on average that those built using the default search algorithm. 63

Table 4.e. An error comparison of ensembles of *k*-NN classifiers compared with best single *k*-NN classifiers found using hill-climbing search 66

Table 4.f. An error comparison of ensembles of *k*-NN classifiers compared with best single *k*-NN classifiers. 66

Table 5.a. A sample data set of *n* cases each described by *f* features. 76

Table 5.b. A transformation of the data shown in Table 5.a based on the outputs of the *m* classifiers in the ensemble. 77

Chapter 1

Introduction

1.1 Problem Context and Motivations

Case-Based Reasoning (CBR) is one of the most efficient and widely used *Machine Learning (ML)* methodologies for supervised learning problems. Its origins arise from the idea that a number of 'similar' cases in the space of the instances of a certain domain can be retrieved to make predictions about a new instance (query). Some similarity criteria must be specified to perform the retrieval of the closest cases. One of the main characteristics that make Case-Based Reasoning so strongly appealing is its similarity to the way human beings reason: much of human expertise is experienced based and CBR is considered to capture this idea.

Thus, the output of a CBR system can be thought as composed by two separate pieces of information: one is the associated query prediction in reply to the supervised learning problem, the other is the set of most similar cases to the query. While the first piece of information is a standard type of reply given by any Machine Learning classifier, the second one is specific to Case Based Reasoning. In particular, the ability to give a ranking of similarity to all the stored cases and to retrieve the ones that look most similar to the new case gives remarkable support to either *weak theory* domains or to any domains where case retrieval for explanation plays a central role (Cunningham et al., 2003). Diagnosis tasks, medical domains, e-commerce applications, recommender systems are only examples of a considerable number of areas that greatly benefits from this characteristic of CBR. We say colloquially that this is a way to use CBR for *Explanation* in the sense that it is shown why the query is

assigned to a certain prediction through the characteristics of the cases that seem more similar to it.

In the research, a great number of ways to improve the performance of Case Based Reasoning system in terms of scored generalization error have been proposed. Feature Selection is a simple and effective technique that tries to remove irrelevant and redundant features to obtain better performance and to contribute to knowledge discovery on the domain; however recent research has shown several doubts arise for this last issue, due to instability problems (Dunne et al., 2002).

One of the most significant developments in Machine Learning research in recent years is the discovery that ensembles of models can offer remarkable improvements in accuracy and stability over single models (Nitzan & Paroush, 1985; Boland, 1989). An ensemble is defined by two elements: a set of properly trained classifiers and an aggregation mechanism that composes the single predictions into an overall outcome. Several ways to train ensembles have been explored, from training each classifier in a subpart of the training set, to giving each classifier a subset of the features available. Usually the aggregation step is performed through a weighted average or a simple majority vote over the output of the ensembles, e.g. (Breiman, 1996); however, more complex linear or non-linear combination of the component predictions may be used, e.g. (Jacobs et al., 1991, Heskes, 1998a).

If on the one hand the aggregation in ensembles improves the performance of single models, on the other it poses serious issues for the CBR paradigm. For example it naturally raises questions about the reliance of CBR on feature-vector representation of cases, seeing that a combination of models outperforms any single model.

However the main problem posed by the use of ensembles in Case-Based Reasoning is that the potential for explanation through case ranking and retrieval, that is a very typical and desirable characteristic of a single CBR system, is somehow lost when moving to an aggregation of CBR systems. The reason for this is that the aggregation process introduces an extra layer of complexity, in comparison to a single CBR system, between the cases and the proposed solution and there is no longer a small number of cases to use for explanation.

So there appears to be a fundamental incompatibility between the ensemble idea when applied to Case-Based Reasoning and the interpretability of CBR itself; if we

consider explanation to be one of the most appealing characteristics of CBR systems this is a considerable loss.

1.2 Thesis Contributions

This thesis makes contributions to both Case-Based Reasoning and Ensemble Research in several directions.

- It presents *Meta k-NN*, a new approach to the ensemble aggregation process that obtains excellent results in terms of performance and simultaneously maintains the benefit of case retrieval for explanation (a solution to the interpretability loss mentioned in 1.1), through a process that gives coherent ranking to cases and identifies a small number of cases for use in explanation. This new approach might be viewed as a representation transformation process for each case, whereby the feature-based representation is transformed to a representation based on the predictions of ensemble members (see Chapter 5). In the case of ensembles of *k*-Nearest Neighbours (*k*-NN) classifiers trained on different feature subsets this aggregation approach has shown very good generalization accuracies, either better or comparable to that of a standard weighted average approach, with the considerable advantage to keep the interpretability properties of single CBR systems.

- It presents *AmbHC*, an approach to train ensemble of classifiers with a modification of a classic Hill Climbing technique that emphasizes the diversity in the ensemble. It shows a comparison between ensembles of *k*-Nearest Neighbours classifiers that are trained with the two Hill Climbing approaches, where the technique emphasizing ambiguity proves to outperform the strategy considering error only, provided the ensemble size is sufficiently large.

- It presents *Duplicate k-NN*, a modification of the standard k-Nearest Neighbour approach designed to deal with databases that present an unbalanced class distribution. While the Duplicate *k*-NN idea is somehow an obvious and simple mechanism for addressing problems caused by unbalanced class distributions, it has not so far been reported in the research literature to our knowledge. It is also argued that to give more significant performance measures in

the case of skewed class databases, Harmonic means indexes are better suited to the task. Results are shown applying the algorithm in the context of *Meta k*-NN aggregation strategy.

- It raises some issues about case representation emerging from analyses of the results obtained with ensemble techniques.

1.3 Related Publications

- Zenobi, G., Cunningham, P., (2002) An Approach to Aggregating Ensembles of Lazy Learners that Supports Explanation, *5th European Conference on Case-Based Reasoning (ECCBR 2002)*, eds S. Craw & A. Preece, LNAI 2416, pp436-447, Springer Verlag.
- Zenobi, G., Cunningham, P., (2001) Using Diversity in Preparing Ensembles of Classifiers Based on Different Feature Subsets to Minimize Generalization Error, *12th European Conference on Machine Learning (ECML 2001)*, eds L. De Raedt & P. Flach, LNAI 2167, pp576-587, Springer Verlag.
- Cunningham, P., Zenobi, G., (2001) Case Representation Issues for Case-Based Reasoning from Ensemble Research, in *Proceedings of 4th International Conference on Case-Based Reasoning* eds D. W. Aha, I. Watson, LNAI 2080, pp146-157, Springer Verlag.

1.4 Structure of the Thesis

The thesis begins in Chapter 2 giving an overview of the main concepts and definitions of Lazy Learning and Case-Based Reasoning for supervised learning problems. Then it specifically concentrates on the *k*-Nearest Neighbours paradigm and raises some issues about *k*-NN in support of explanation.

Chapter 3 is dedicated to the Feature Selection issue. After the problem definition, it gives an overview of the techniques proposed in literature for the search of the best feature subset. The last section of the Chapter shows an evaluation in reference to the instability problem.

Chapter 4 introduces ensembles theory. Specific attention is given to the Bias/Variance and Ambiguity decomposition tools and to the several ways proposed in literature to introduce Diversity in ensembles. Then it focuses on *AmbHC,* a hill climbing technique proposed to train ensembles with a high degree of diversity; both the theoretical approach and the evaluation are shown. The final section outlines some considerations about case-representation emerging from results obtained with ensemble research.

Chapter 5 presents the *Meta k*-NN paradigm, an aggregation strategy designed to maintain the characteristic of case ranking and retrieval in support to explanation that are typical of single CBR systems.

Chapter 6 is entirely dedicated to the evaluation of the *Meta k*-NN aggregation strategy. Evaluation is performed both in domains with equally distributed and skewed distributed classes; in the last case *Duplicate k*-NN, a modification of the standard *k*-NN paradigm is introduced and applied to *Meta k*-NN. To perform evaluation in the case of skewed class distribution, Harmonic Mean indexes are also introduced. The final section of the Chapter presents an evaluation of the quality of the retrieved cases, performed with the contributions of some domain experts.

Finally Chapter 7 draws the conclusions and outlines some possible developments for future research.

Chapter 2

Lazy Learning and Case-Based Reasoning

2.1 Lazy Learning and Eager Learning algorithms

During the past decades of *Machine Learning (ML)* research a considerable number of learning algorithms has been proposed and developed to approach a solution for the *supervised learning problem.*

To give a definition of the supervised learning problem, suppose our task is to learn an unknown target function $t(\boldsymbol{x})$ where $\boldsymbol{x}=(x_1, x_2, \ldots, x_n)$ is a vector of variables (features) and $t(\boldsymbol{x})$ the *outcome* of the function. Typically the feature values $x_1, x_2, \ldots x_n$ are either real-valued or discrete or symbolic (for example weight, age in years, blood type) and the outcome belongs either to a finite set of classes or to a real-value interval. Suppose a set of measured couples $(\boldsymbol{x}_i , y_i)$ is drawn from the original distribution with some noise (y_i represents the outcome of vector $\boldsymbol{x}_i$); we will call each couple a *case,* and the whole set of couples $(\boldsymbol{x}_i , y_i)$ *the training database T or Case-Base T.*

A supervised learning problem is defined when, given a vector of new feature values $\boldsymbol{q}$ called *query*, a prediction of the query outcome $t(\boldsymbol{q})$ has to be given on the basis of the already stored Case-Base *T*. A predictor is a model that takes as input the Case-Base and the query $\boldsymbol{q}$ and outputs a prediction $f(\boldsymbol{q})$ of the outcome $t(\boldsymbol{q})$;

The set which y_i, $f(\boldsymbol{q})$ and $t(\boldsymbol{q})$ belong to will be indicated as *outcome set Y*. When the outcome belongs to a finite set of classes we call it a *classification problem* while

when it belongs to a real-value interval we call it a *regression problem*. Usually when dealing with classification problems the predictor is called *classifier*. It is frequently the case that a classification problem has a very small number of possible classes; when the number of classes is 2 it's then called a *binary classification problem*, because the two classes can be easily mapped into the values 1/0. (for example it's quite straightforward to map "Yes" as 1 and "No" as 0).

When talking about algorithms that address the supervised learning problem, a preliminary distinction can be made between *lazy* learning and *eager* learning algorithms (Mitchell, 1997), (Aha et al., 1991), (Aha, 1997).

Eager learning algorithms greedily compile an intensional concept description using the input data assuming its learning bias is appropriate to generalize for future queries. Any input used to build such a model is then discarded. For example learning algorithms like neural networks or decision trees belong to this category.

A *lazy learner,* on the contrary, delays the processing of the input data until a request for a new query is received; only at this time the information stored in the database are combined and used to reply. The query constructed and intermediate results are discarded after the query is processed. The algorithms that belong to this category mostly refer to the learning paradigms named Instance-Based Reasoning and Case-Based Reasoning: their philosophy is based on the idea that an answer to the supervised learning problem can be given through the retrieval of a number of "similar" cases in the space of the instances, that are combined with some strategies to obtain a prediction about the query outcome. Some similarity criteria must be specified to perform the retrieval of the closest cases. The *k*-Nearest Neighbours algorithm is among the most popular examples of Case-Based Reasoning and will be investigated in detail in the following section.

Lazy learners and eager learners have considerable differences; we will outline here some intuitive explanations, while in section 2.2 we will show in more detail distinctive features of some of these algorithms. Some of the most evident differences are in terms of computational costs both during the training and the generalization process and also in terms of storage costs. Typically lazy algorithms have a smaller computational cost than eager algorithms during the training, since that the eager ones have to build a model based on the training data while the lazy ones just have to store the data and wait for the query; as a consequence, in terms of storage cost obviously

lazy algorithms need a greater capacity. On the other hand the eager algorithms, once they have built a global model after the "training", are much faster when the time comes to classify for a new query, while the lazy ones need a greater time, as for any new query a different local model is built from the stored data. Another advantage of lazy learning systems is that there is no need for a new training when a considerable number of new cases is added to the database, while the eager learning must rebuild the model if they want to remain up to date.

In fact there is not such a neat distinction between lazy and eager learners. Machine learning literature is full of examples of algorithms that combine the two philosophies in order to achieve better performance results. In the present work we will focus our attention on *k*-Nearest Neighbours classifiers and we will show in the following chapters how some typically eager learning "injections" may contribute in improving substantially the performance of the tool.

The following sections are organised as follows: section 2.2 will introduce principles and relative notation of Case-Based Reasoning and of the *k*-Nearest Neighbours algorithm in particular, showing in detail some of their distinctive features, while section 2.3 will focus on the use of Case-Based Reasoning in support of explanation. Section 2.4 will instead point out some limitations of Case-Based Reasoning.

2.2 Case-Based Reasoning and *k*-Nearest Neighbours

Among the several lazy learning algorithms that have been proposed we will focus our attention on those ones that use a *distance function* (or a *similarity function*) to generalize from the stored data. These algorithms are generally known as *Case-Based Reasoning (CBR)*, with obvious reference to the fact they perform their "reasoning" based on a number of samples stored in a Case-Base.

We will start illustrating the *k-Nearest Neighbours (k-NN)* algorithm (Cover & Hart, 1967, Cover, 1968, Duda & Hart, 1973), which turns out to be one of the simplest and most effective ones, and that will also be the one used throughout this research. It assigns to the query the class inferred from the classes of the *k* nearest cases, where the distance between two cases is usually computed through the

Euclidean distance metric. Here below we report a brief summary of the concepts and the notation involved in the theory of the *k*-Nearest Neighbours classifiers; at the end of this section we will outline a bit more in detail how *k*-NN relates to Case-Based Reasoning. A good survey on the role of CBR in science and applications can be found in Aha (1998).

2.2.1 A formal description of *k*-Nearest Neighbours

When dealing with a classification problem, the *k*-NN algorithm assigns to the query q the most probable class among all the classes c_j, as shown in (2.2.i). To simplify notation, in the rest of this dissertation we will use regular font instead of bold font (i.e. q instead of $\boldsymbol{q}$) to indicate vectors, unless specifically demanded by the mathematical context.

$$\text{Class}_{k\text{-NN}}(q) = \arg\max_{c_j} \text{p}(c_j|q) \qquad \textbf{(2.2.i)}$$

with the posterior class probabilities are computed as:

$$p(c_j|q) = \frac{\sum_{x \in Kq} 1(\text{Class}(x) = c_j) \cdot K(d(x,q))}{\sum_{x \in Kq} K(d(x,q))} \qquad \textbf{(2.2.ii)}$$

Class(x) is the measured outcome of case x, (the notation simply remarks it is a classification problem) while the function 1(Class(x) =c_j) is defined as

$$1(\text{Class}(x) = c_j) = \begin{cases} 1 & \text{Class}(x) = c_j \\ 0 & \textit{otherwise} \end{cases} \qquad \textbf{(2.2.iii)}$$

K_q is the set of the k nearest neighbours, i.e. the stored examples that have the shortest distance to the query q; *K(.)* is a kernel function of the distance measure. There are several definitions for the Kernel functions, however the most widely used ways to define it are either as a constant:

$$K(d(x,q)) = 1 \qquad \textbf{(2.2.iv)}$$

if we want each of the k nearest neighbours to have the same effect on the distance function (i.e. they are equally important so that the average becomes a "simple count" on the times each class was retrieved), or as the inverse of the distance:

$$K(d(x,q)) = \frac{1}{d(x,q)} \qquad \textbf{(2.2.v)}$$

if we want each of the k nearest neighbour to have a decreasing weight with the distance value, assigning to a closer element a higher importance for the prediction than a further element. With the refinement proposed in (2.2.v) it may happen that the query is assigned to a different class from the majority class of the retrieved cases, if these last ones happen to be at a considerable distance from the query.

As mentioned above the distance is usually defined as the Euclidean one:

$$d(x,q) = \sqrt{\sum_{f \in F} \delta(x_f, q_f)^2} \qquad \textbf{(2.2.vi)}$$

Since any feature f may happen to be either symbolic or numerical (discrete and continuous), the distance function needs to take into account this potential diversity. It is very frequently the case that all the numerical features are normalised so that the highest distance between two cases in a single feature can at most be 1; as for the symbolic features the usual strategy is to count a distance of 1 if they don't match and a distance of 0 in the case they match. Below we report a formalization of the delta function used in (2.2.vi):

$$\delta(x_f, q_f) = \begin{cases} 0 & f \text{ symbolic and } x_f = q_f \\ 1 & f \text{ symbolic and } x_f \neq q_f \\ |x_f - q_f| & f \text{ numerical normalized} \end{cases} \qquad \textbf{(2.2.vii)}$$

To generalise the k-NN algorithm to the case of a regression problem is relatively easy; it is sufficient to use the same philosophy that lead us to equation (2.1.ii). This time instead of the function 1(Class(x) =c_j) that compares the neighbours' classes to the query class we will use the neighbours' measured outcome (continuous value) as it is stored in the Case-Base.

$$\text{Outcome}_{k\text{-NN}}(q) = \frac{\sum_{x \in Kq} \text{Outcome}(x) \cdot K(d(x,q))}{\sum_{x \in Kq} K(d(x,q))} \qquad \textbf{(2.2.viii)}$$

2.2.2 Loss Functions

To measure the performance of a Case-Based Reasoning system it is very common to use some standard statistical indexes. In the case of a classification problem the most natural way to compute the *Generalization Error* (the error on unseen data) is through the *0/1 Loss Function*: (both the outputs *t(q)* and *f(q)* are a class among a finite set of classes):

$$\mathsf{E}_{0/1}(t(q), f(q)) = \begin{cases} 0 \text{ if } t(q) = f(q) \\ 1 \text{ if } t(q) \neq f(q) \end{cases} \qquad \textbf{(2.2.ix)}$$

In the case of a regression problem the most widely used Loss Function is the *Mean Squared Error (MSE)*:

$$\mathsf{E}_{MSE}(t(q), f(q)) = \frac{1}{2K} \sum_{i=1}^{K} (t^i(q) - f^i(q))^2 \qquad \textbf{(2.2.x)}$$

where t^i and f^i are the i-th components of the K-dimensional vectors.

Whatever the chosen Loss Function is, it's frequently the case that the error is estimated through its mean value over a certain set of queries (test set), indicated as:

$$\overline{\mathsf{E}}(t, f) = \langle \mathsf{E}(t, f) \rangle_{\underline{X}}$$

A more detailed analysis of the error will be outlined in Chapter 4 when dealing with Ensembles of classifiers/predictors.

Usually, if there are no reasons to assign different weights to the queries, the mean value is computed as a simple arithmetic average over the query set; thus, when dealing with classification problems, the error simply becomes the following ratio:

$$E = \frac{\text{number of queries incorrectly classified}}{\text{Total number of queries}} \qquad \textbf{(2.2.xi)}$$

It is also very common to use the *Accuracy A*, a complementary measure to the Error; it is simply computed as the complement to one of the Error:

$$\mathsf{A} = 1 - \mathsf{E} \qquad \textbf{(2.2.xii)}$$

In the case of simple arithmetic average for classification problems accuracy becomes:

$$A = \frac{\text{number of queries correctly classified}}{\text{Total number of queries}} \qquad \textbf{(2.2.xiIi)}$$

2.2.3 Distinctive Features of *k*-NN, CBR and Lazy Learning

A first observation emerging from the definitions shown in 2.2.2 is that the concept of "similarity" strongly depends on the choice of the distance function. If we choose the usual Euclidean metric we make the (strong) assumption that the space of the instances can be represented as a Euclidean space. Another downside of equation (2.2.vii) is the fact symbolic and numerical features affect the final distance measure with a very different amplitude; for example there will be a very small probability that the distance between two numerical features is 1 (this happens only in the case of distance between the maximum and the minimum value) with a net result that in practice the symbolic features may have a significantly greater influence on the similarity computation. In research several alternative approaches to the Euclidean one have been proposed; most of them handle the difference between all the features through a set of different weights assigned dynamically through some strategies. However, they will not be investigated any further by this research; we would just like to mention here an interesting approach proposed by Wilson and Martinez (1997), that defined a number of Heterogenous Distance Functions based on the concept of Value-Distance Metric, to address classification problems with Datasets that present non-homogeneous features.

The *k*-NN is perhaps one of the most widely studied and used among the lazy learning algorithms, due to its simplicity and effectiveness. Many are the generalizations proposed in literature, and this could well be the subject of a wide survey; in the context of our research we will only mention here the works by Gates (1972), who introduced the idea of Reduced Nearest Neighbour rule, and the fundamental work by Hart (1968) who introduced the Condensed Nearest neighbours Rule, a methodology that reduces the number of cases of the Case-base to give more efficiency in the performance, an idea which gave birth to the branches of Case-Base Editing and Case Selection. A couple of interesting refinements of the Condensed Nearest Neighbours Rule were also proposed by Tomek (1976).

The *k*-NN, like any other lazy learning algorithm, doesn't form an explicit general hypothesis about the target function *t*. So it is not possible to have a clear and straightforward mathematical model to describe the relationship between the input (query) and the output of a *k*-NN predictor or classifier. Nevertheless is still possible to ask ourselves what the implicit learning bias is. The following figure illustrates the

Voronoi diagrams for a 1-NN classifier: the convex polygon around each training case indicates the area of the space that would be assigned to that case.

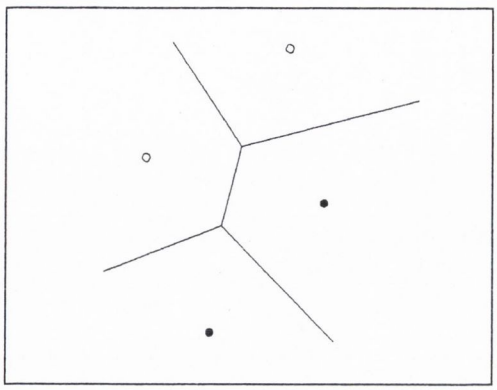


Fig. 2.I. Voronoi diagram for a 1-Nearest Neighbour classifier

As we have shown in section 2.1, there are some key differences between eager learning algorithms in general and lazy learning ones, for example lazy learning requires higher storage capacity and higher computational time in the prediction phase than eager learning algorithms. However, there are some other very important differences between the two approaches. A first one is concerned with the inductive bias in learning. Eager learning chooses a *global* approximation to the target function *before* any query is presented. Thus it must commit to one single hypothesis to cover the entire space of the instances. On the other side, lazy learning can generalise from the stored case-base each time in a different way, depending on the query (through its closest neighbours in the case of *k*-NN). Thus it chooses each time a *local* representation that better approximates the query. In other words a fundamental distinction between lazy and eager learning algorithms can be thought about the "scope" of the approximation to the target function: for lazy learners it is a local one while for eager learners it is a global one. Another difference, extremely important for our purpose, regards interpretability of the results; we will dedicate to this issue the whole of section 2.3.

Before finishing this section we would like to clarify some aspects of the jargon used in this thesis. To give a clear and straightforward definition of the differences between *k*-Nearest Neighbours and Case-Based Reasoning is a tricky job. The use of a new jargon can sometimes lead to subtle and small differences between different research communities, and this is certainly the case of Case-Based Reasoning. In his comprehensive survey, Aha (1998) defines Case-Based Reasoning as "*a multi-*

disciplinary subject that focuses on the reuse of experiences" but immediately shows the difference on the interpretation given by Cognitive Scientists, Artificial Intelligence Researchers and Expert System Practitioners. Following this approach it appears that the *k*-Nearest Neighbours algorithm is simply an example of the more general Case-Based Reasoning category. In his fundamental book on Machine Learning, Mitchell (1997) points out instead that *k*-Nearest Neighbours and Case-Based reasoning are two different examples of "instance-based" reasoning, *k*-NN being defined only for numerical continuous vectors in Euclidean spaces, while Case-Based reasoning deals with a more complex and rich representation of the data (for example function graphs) and a subsequent non-Euclidean similarity measures, like in the case of CADET (Sycara et al., 1992) a tool that assists the design of simple mechanical devices.

To show a detailed classification of the Machine Learning jargon goes far beyond the purpose of the present research. For this reason, as the *k*-Nearest Neighbours algorithm will be the algorithm used through this entire research, with a little abuse of notation from now onward we will use "*k*-Nearest Neighbours" and "Case-Based Reasoning" as if they were synonymous, unless specifically mentioned. For the same reason we will often use "Lazy Learning" to indicate the same categories of algorithms.

2.3 Case-Based Reasoning in support of Explanation

Case-Based Reasoning is based on a very intuitive paradigm that is frequently used in real life: given a set of past experiences we will make a prediction on a new situation based on a subset of the past experiences that look most similar to the new case. Consider for example the case of a medical practitioner; first, he either consciously or unconsciously compares a new patient with the patients he had the chance to visit and treat in the past. Then he compiles a list of "most matching" cases (for example patients that presented same symptoms, or same results to tests), has a look at the successful treatment he prescribed for the similar cases in the past and finally decides to prescribe it for the new patient. One of the reasons that has made a simple tool like *k*-NN so popular is certainly the parallel that can be drawn with the human mind. Ramberg (1996) has shown in a study about physicians' expectations

from computer support for consultation, that the most important requirement was the capability to explain the reasoning. In the course of this research we had the chance to speak with some specialist practitioners, and they all seemed to agree that a tool that retrieves the most "similar" patients to the one currently examined, especially in very large databases, can certainly be a big support for diagnosis tasks and the relevant decision for an adequate treatment.

Thus, a very distinctive feature of Case-Based Reasoning is in particular the capability to retrieve cases that looks mostly similar to the query case. This is certainly a very important piece of information provided to the user in addiction to the prediction about the query outcome. We can call this additional piece of information *explanation*. The use of the term "explanation" comes from the idea that with a set of closest retrieved cases we could imagine the system suggesting something like: *"the query belongs to class X because the closest case to it also belongs to class X and so does the third closest case"*. This can be considered as an "explanation" to the query.

A recent study by Cunningham et al. (2003) has evaluated the usefulness of examples-based explanation, as provided by CBR, in comparison to more traditional principles-based explanations, as provided by Rule-Based Systems for a blood-alcohol level prediction task. The results showed that the average rating given by the users indicated CBR as more convincing than Rule-Based Systems in explaining the predictions.

2.3.1 CBR for Explanation in the Literature

In the literature the importance of Case Based Reasoning for explanation has been stressed in a number of works. A good source of information for the case indexing, ranking and retrieval and its relationship with interpretation and evaluation can be found in the work by Kolodner (1993) and Leake (1996). Another way to look for the "best" case retrieval was proposed by Salzberg et al. (1995) through a theoretical model that was giving upper and lower bounds to the sample complexity using geometric properties.

Other less straightforward approaches to explanation, different from the case ranking, have been proposed. Leake (1995) for example, introduced some similarity assessment methods to guide retrieval of candidate explanations together with evaluation of retrieved candidates. Bergmann et al. (1994) introduced the idea of

explanation-based similarity, an approach to integrate Domain Knowledge into Case Based Reasoning, successfully applied both to a diagnosis task and to a planning task. In this approach domain knowledge is modelled with several levels of abstraction, and two explanations matching on a lower level of abstraction are assigned a higher similarity than two matching on a higher level of abstraction. Aamolt (1990) proposed a system called CREEK to integrate case-based with explanation-based methods; also in (1994) he described a generic reasoning method that makes use of rich knowledge model in producing context-dependent explanations. Ram (1993) introduced the idea of incremental learning of explanatory cases when dealing with novel problems.

The explanation feature of Case-Based Reasoning makes it particularly desirable in any *weak theory* domains. Here below we give some examples of areas where CBR proved to be successful, particularly with regards to its retrieval and explanation capabilities; a more complete discussion of the areas where CBR has been successfully applied can be found in Bartsch-Sporl et al. (1999) and Aha (1998).

An area where CBR has obtained a considerable number of successes is certainly Diagnosis Tasks (Lenz et al., 1998). A successful example of CBR in support to Diagnosis can be found in Medicine. Medicine is certainly a subject where complete theoretical models are very rarely available, while on the other side there is the availability of a good number of databases of monitored patients. Thus the CBR approach is particularly indicated (Gierl et al., 1998), since it also works in a "cognitive" way which is very similar to the way doctors work; we have already mentioned how practitioners feel that the possibility to retrieve cases for explanation gives a very important piece of information for the domain knowledge. Good examples can be found in Gierl & Stengel-Rutkowsky (1994) who proposed GS.52, a system for the diagnosis of dysmorphic syndromes and in Ong et al. (1997) who described CARES, a system for predicting recurrence of colorectal cancer. Another successful application of CBR to Diagnosis Tasks is Help-Desk for Interactive Troubleshooting. The cases are usually described by a set of questions and answers to the observed "symptoms" while the solution is usually returned as a short piece of text indicating how to manage the problem. Many are the world-renown companies that have used this approach (e.g. Compaq, IBM, HP); an interesting reference for Conversational CBR on this issue can be found in Breslow & Aha (1997). Also, troubleshooting of machines other than PCs proved to be very efficient if supported by

CBR; see for example Heider (1996) who implemented a troubleshooting system for engines used in Boeings 737. Other interesting examples can be found in McSherry (2001), who described Strategist, a toy fault diagnosis system, in Pews & Wess (1993), who described MoCAS, a system designed to identify faulty components in technical systems, and Richter (1991), who described PATDEX a system in support of technical diagnosis.

A second area that greatly benefits of the CBR characteristics is certainly Electronic Commerce; a comprehensive survey can be found in Wilke (1997) and Wilke et al. (1998). There are many applications of this type available from the Web, for example product selection, both in pre-sales and after-sales, can greatly benefit from CBR tool assistance. An interesting example is also described in Lentz (1994) where a CBR tool is outlined to deal with the huge number of holiday offers in a travel agency.

Another type of application where CBR has played a major role is the Recommender Systems one. By nature, a recommender retrieves a set of cases that it considers "suitable" to the user, once he has given a set of preferences. Thus CBR looks the most "natural" paradigm for this kind of task. In this field as well there are many Internet applications available, sometimes coupled with the possibility of a purchase (e-commerce) where a number of suggestions for further purchases is given depending to the user profile. There are many others applications where CBR has been used with great success; we would only like to mention here the use of CBR for Configuration, Planning, Design, and Document Retrieval.

2.4 Limitations of Case-Based Reasoning

Together with all the benefits described above, k-NN and CBR show some downsides as well. If on one side they proved to be very effective and robust to noisy training cases, provided there is a sufficiently accurate choice of k, a major problem is posed instead by the *feature relevance*. In equation (2.2.vi) we can see that all the features are included to compute the distance between two cases. This can become a serious problem if some of the features are either irrelevant or redundant. The problem of selecting the subset of most predictive features is known in Machine Learning as

Feature Selection problem. In the course of this research we will concentrate with particular attention on *k*-NN based on a subset of features, so we will dedicate the whole of Chapter 3 to a detailed description of the Feature Selection problem.

We have already introduced in section 2.1 the idea that to improve the performance of a lazy learning system it is very common to use some eager learning "injections". Feature Selection is an example of such injections. As a matter of fact every time we introduce a "training" process (as is selection of the most predictive feature subset) we move a step further from the pure lazy learning paradigm. A pure lazy learning paradigm doesn't spend any time at all for the training: in a pure lazy learning system the training is a mere storage step. Furthermore, after the training step we always reach a certain "configuration" that looks the best for that particular Case-Base it was trained on, but if some new cases are added to the Case-Base the chosen configuration may need an update. A purely lazy system doesn't need any re-configuration, it just generalises every new query on the basis of the whole set of cases stored until that moment. Once again, the reason that justifies such eager learning "injections" is the fact they can improve the performance of a lazy learning system, while the nature and the principles behind the lazy learning system remain the same (for example in the case of CBR, case retrieval for explanation remains a distinctive characteristic). In Chapter 4 we will also see another layer of "training": the aggregation of an ensemble of nearest neighbours.

2.5 Summary of the Chapter

In this Chapter we have outlined an introduction on Lazy Learning and Case-Based Reasoning (CBR). In Section 2.1 we have presented some characteristics of Lazy and Eager learning paradigms to address supervised learning problems. In section 2.2 we have focussed on CBR and in particular on the *k*-Nearest Neighbours algorithm, that will be the basic "tile" on which we will build more complex algorithms in the following chapters. In section 2.3 we have focussed on the use of CBR in support of explanation, as interpretability will play very important roles in the course of this thesis. Finally in section 2.4 we have outlined some limitations of CBR algorithms.

In the next Chapter 3 we will focus on the Feature Selection issue. Special attention will be paid to Feature Selection techniques applied to *k*-NN algorithms, and to the limits of Feature Selection use for Knowledge Discovery in relation to instability problems.

Chapter 3

The Feature Selection Problem

3.1. Introduction to Feature Selection and Weighting

One of the main problems that affect a lazy learner is the fact that all the features are considered in the computation of the distance to retrieve the nearest cases, even though some of them might be irrelevant or redundant. This is also known as *feature relevance* problem.

A feature is said to be *irrelevant* when it is not predictive for that particular classification or regression task. To give a simple example, if the task is to predict the hair colour of a newly born child and together with the features "*father's hair colour*" and "*mother's hair colour*" we consider the feature "*father's job*", the new classifier will obtain a worse result because it will compute distances to the other cases also considering a feature that is completely irrelevant to this task.

A feature is said to be *redundant* when it is predictive for the task, but it could be inferred from the other features. It will affect the prediction in a more subtle way, because if it somehow repeats already present information, this information will be counted two times as if it were twice as important. For example, if in a certain classification problem we have the features "distance in Km" and "distance in Miles" one of them is obviously redundant as it can be derived from the other one.

It is important to remark that it might happen that a feature is globally relevant for a certain task, but irrelevant or redundant in some sub-areas of the space of the instances (*local relevance*), as shown in Domingos (1994).

In the literature a large number of works that tried to address and solve the feature relevance problem have been proposed. One of the most intuitive and effective technique is named *Feature Selection (FS)*.

From a mathematical point of view the Feature Selection approach simply introduces binary weighting factors in the distance computation so that a feature is counted only if it is considered predictive. Equation (2.2.vi) becomes then:

$$d(x,q) = \sqrt{\sum_{f \in F} w_f \cdot \delta(x_f, q_f)^2} \qquad \textbf{(3.1.i)}$$

As mentioned above, weights are binary:

$$\forall f, w_f = \begin{cases} 1 & \text{if feature } f \text{ is considered predictive} \\ 0 & \text{if feature } f \text{ is not considered predictive} \end{cases} \qquad \textbf{(3.1.ii)}$$

From the geometrical point of view this simply means that in the space of the instances we drop the axes corresponding to the irrelevant features; the dimension of the space decreases by the number of dropped features.

A straightforward generalization of the Feature Selection approach is the *Feature Weighting* approach. Feature Weighting follows the same philosophy of Feature Selection and uses the weighted Euclidean distance shown in (3.1.i) with the difference each single weight can assume any value in the real interval [0,1]. This technique allows us to refine the importance given to the features: each one can be considered not only as "completely" relevant or irrelevant, but also as "partially" relevant; features with lower weights give a lower contribution to the distance function than features with higher weights. From the geometrical point of view this means in the space of the instances we stretch or compress the axes by a factor corresponding to each feature weight. Furthermore feature weighting seems to appear a good technique to deal also with the problem of symbolic features, as we mentioned in section 2.2.3.

Seen from a purely mathematical point of view Feature Selection may be considered as a sub-case of Feature Weighting. Thus it may appear obvious that, first, there is no need to separate the two theoretical approaches and, second, the general continuous-value Feature Weighting scores in general better results compared to the sub-case of Feature Selection. The reality is a lot less straightforward and there are good reasons both to keep the two approaches separate and to prefer Feature Selection

to Feature Weighting in some domains. What follows tries to explain these reasons in more detail.

First, there is a computability issue. Feature Selection can be seen as a "training" process, in the sense that we need to explore a whole set of feature combinations to decide which one best represents the domain. As mentioned in section 2.4, this can also be seen as an eager learning "injection", as when we apply a feature selection technique we lose some of the characteristics of a pure lazy learning system (but we obviously gain in performance). Suppose we are dealing with a domain described by n features. For a Feature Selection problem, the total number of feature subsets we can draw from the entire set of features is 2^n (a feature can be either "present" or "absent", so that it can have 2 possible states). The generalization to a Feature Weighting problem leads to an infinite number of sets of weights, precisely corresponding to the n-dimensional space generated by the interval [0,1], indicated as $[0,1]^n$. As we will better see in next section, most of the time in a feature selection problem it is computationally impossible to examine all the possible configurations and some strategies must be defined to guide the search in the space of feature subsets. In the case of Feature Weighting the definition of a strategy for the search in the space of the feature weights, due to its infinite dimension, may in some complex domains become problematic.

Second, there is a representational issue. When dealing with Feature Selection, each feature subset can be represented by an n-dimensional binary vector (0 for feature absent, 1 for feature present), while in the case of Feature Weighting each set of weights has to be represented by n-dimensional real-value vectors. Thus, most of the techniques that perform a search through the feature subsets space are defined keeping into consideration the nature of this representation and almost impossible to extend to the case of the search through the weights space in the case of Feature Weighting. Think for example of a sequential technique that, based on a certain configuration reached in the current step, moves toward next step flipping a bit from its current value to the neglected one; this would certainly be hard to define if we deal with non-binary values as in the case of Feature Weighting. On the other hand, it can happen that a technique defined for Feature Weighting cannot be extended to Feature Selection, because of the continuous value representation.

Both the computational and the representational issues could be partially addressed by the use of a finite set of weights. The idea is to introduce a binary coding so that each feature has associated with it k bits that encode 2^k possible weights. As a result the feature weights vector becomes in this case a binary vector whose length is $k*n$. However, even though the number of possible states is in this case not infinite anymore, it still remains considerably higher than a feature selection situation (2^{kn} against 2^n) so that the computability difference would be only partially reduced. Furthermore, some of the most powerful feature weighting techniques, like EACH (Salzberg et al., 1991), IB4 (Aha, 1992) or RELIEF (Kira & Rendell, 1992) are defined by iterative methods that update continuous weights (or cumulative weights) and forcing them to fall into a discrete set of weights would greatly reduce their effectiveness.

Third, there is an *overfitting* issue. Overfitting is a serious problem that affects machine learning algorithms. Every time we perform a training step we choose a configuration based on the sampled cases stored in the Case-Base. These cases are obviously perturbed by some random noise. We have an overfitting problem when a learning model trained on the sampled cases has such a high complexity that it also models the random noise of the training data. When computing the generalization error on unseen data this tool may score poor results, as the unseen data may be affected by a completely different random noise from the training data. A more detailed digression on the Overfitting problem will be made in Chapter 4 when dealing with ensembles of classifiers; what instead is interesting to us in the context of Feature Selection/Weighting is the fact that using continuous weights (a "refinement" of simple binary weights) can in some domains lead to an overfitting problem. A recent study by Kohavi et al. (1997) has clearly shown there are several domains where a simple binary weighting (i.e. Feature Selection) outperforms a more complex continuous weighting, due to the overfitting problem. Overfitting can generally affect any learning algorithm, including Feature Selection (Kohavi & Sommerfield, 1995), but the higher complexity of Feature Weighting makes it potentially more susceptible to this problem. However at present this issue remains still open for further research. Also, when deciding to use Feature Weighting instead of Feature Selection, a balance between the effective improvement against the increase in complexity must be

evaluated; in some cases it may happen there is a too little improvement to "justify" the complexity increase.

Starting from these considerations we have chosen to focus this research on *k*-Nearest Neighbours algorithms refined through a Feature Selection process. One of the main reasons for this choice is the fact that in next chapters we will deal with ensembles of *k*-Nearest Neighbours classifiers. Ensembles have recently emerged as very effective technique to improve classification/regression performance. The downside is that the aggregation in ensembles introduces an extra layer of complexity. Feature Selection seemed to us more indicated as it keeps complexity smaller than Feature Weighting and is better suited for ensembles. As for the overfitting issue, we will see in Chapter 4 that the use of an ensemble can greatly contribute in reducing this problem.

The following sections will be organised as follows. In section 3.2 we will give an overview of the most widely used techniques to perform the search for the best feature subset. Of course, this will not be a comprehensive survey as the focus of this research is not the Feature Selection issue itself; more detailed references can be found in Doak (1992), Wettsheterek et al. (1995), Atkeson et al. (1996) and Aha (1998). In section 3.3 we will instead show some considerations and an experimental evaluation about the use of Feature Selection for knowledge discovery in relation to instability problems.

3.2 The selection of the Best Feature Subset

In the previous section we have shown that if a domain is described by n features the total number of possible feature subsets is 2^n. When the number of features n is not small it becomes prohibitive to compute the performances scored by all the possible subsets. So it is necessary to define some criteria to "guide" the search for the best element in the feature subsets space, where with "best element" here we mean the most predictive feature subset. To perform this search we need two elements: a *Search Algorithm* and an *Evaluation Function*. This section will explain briefly the role of both elements and give a very quick overview of solutions proposed in the literature.

The Search Algorithm is a process that, at each step of the search, selects one or more new elements of the feature space from the area that seems to have the highest potential in terms of feature relevance. This selection is made taking into account the performance measure scored by the current feature subset compared to the ones scored by the feature subsets selected in the previous step. The Evaluation Function has the task to give a performance measure to the selected feature subsets, in other words to return a measure of "how good" the current element is compared to the previously examined ones. Through the result of this evaluation the Search Algorithm is able to understand in which "direction" performing the selection of a new element (feature subset); this in turn is given to the Evaluation Function for the performance measure and the cycle goes on this way. The cycle usually starts with an "ad Hoc" element, chosen either using the information available on the particular domain or with a simple random selection if no information is available, and it continues until a certain condition is reached, for example a predetermined level of accuracy or simply an assigned number of iterations.

3.2.1 Types of Evaluation Functions

In literature two different approaches to define the Evaluation Function and the way it relates to the Search Algorithm have been proposed: the *Filter* and the *Wrapper* model. They are illustrated in the following figures:

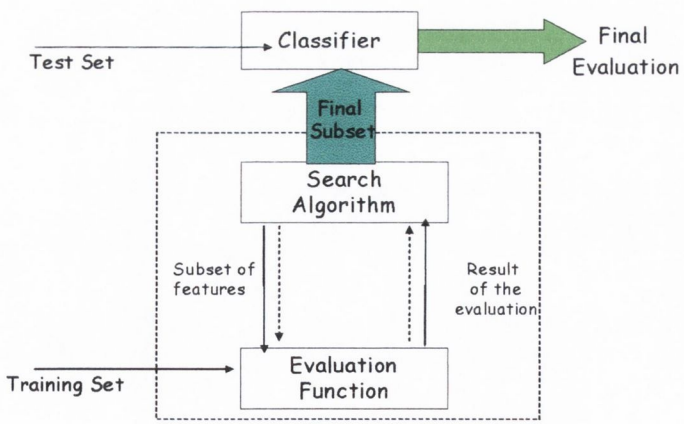


Fig. 3.I. The Filter Strategy

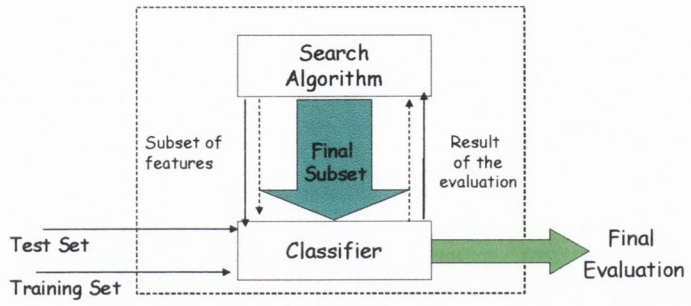


Fig. 3.II. The Wrapper Strategy

In the Filter model the Evaluation Function is based on information that can be directly extracted from the data through some of their intrinsic properties. In the case of Feature Selection the most used Evaluation Functions are indexes like probabilistic distance and dependence measures, interclass distance measures, or entropy measures; in the case of Feature Weighting other indexes have been defined based on conditional probabilities, class projection, or mutual information. This schema is also known as "*non feedback*" because the classifier/predictor is only used to perform the prediction on the final element selected and it doesn't perform on any of the intermediate steps.

In the Wrapper model the evaluation is instead performed at each step by the classifier/predictor itself, this way leading to a closed-loop or "*feedback*" schema. The most widely used evaluation index in this case is the Generalization Error of the classifier/predictor (or the complementary Accuracy) as defined in equation (2.2.ix) in the case of a Classification problem, and the Average Squared Error as defined in equation (2.2.x) in the case of a Regression problem. Other indexes for the Wrapper approach have been proposed in literature; we will only mention here the work by Domingos (1997), who replaced the simple Accuracy for a Feature Selection problem with the Laplace-Corrected Accuracy, and the work by Yang & Honavar (1997), who defined a more complete Fitness Function, weighting both the Accuracy and the Cost that comes from measuring the selected feature subset. A detailed analysis of the Wrapper approach can be found in Kohavi & John (1998).

Several researches in literature have analysed a comparison between the Filter and Wrapper approaches. In the greatest majority of the cases this comparison seems to indicate the Wrapper model outperforms the Filter model; some references can be found in Aha & Bankert (1994) and Aha & Bankert (1995). The reason why this happens may rely on the fact that the Filter model uses an Evaluation Function with an intrinsic different bias from the Classifier's one, so that it negatively affects its performance on the selected feature subset. On the contrary the Wrapper model performs a feedback on the Classifier and the search is guided by the same bias that gives the final result. Next sub-section will show some of the most successful examples of Search Algorithm for Feature Selection proposed in literature.

3.2.2 Types of Search Algorithms

The Search Algorithm guides the selection of the most predictive feature subset based on the results given by the Evaluation Function. Since it is frequently the case the search space is large, a good search algorithm should examine only a small portion of the space, possibly the area where the best feature subsets reside. As shown in the comprehensive survey by Doak (1992), Search Algorithms can be classified in three big categories: Exponential Algorithms, Sequential Algorithms and Randomized Algorithms. We will give a quick overview of each category here below.

a) Exponential Algorithms

These algorithms either perform an exhaustive search, i.e. the whole space of feature subsets is examined, or they use a "branch and bound" technique to visit only a part of the space when the evaluation function is monotone, i.e. a feature subset will perform no better that any other subset of which it is a proper subset. They obviously are guaranteed to find the optimal feature subset, but their main problem is that they can be used only with a very small search space (usually not more than 15 features).

b) Sequential Algorithms

These algorithms take their name from the fact they sequentially add or remove features at the end of each cycle, on the basis of the evaluation function response, until an optimal or a sub-optimal feature subset is reached. It is a partial search, in the sense that only a small fraction of the search space is visited. The purpose is to guide the

search toward the absolute minimum of the evaluation function in the search space, anyway it may be the case that these algorithms reach only a local minimum.

The *Forward Sequential Selection (FSS)* is among the oldest and simplest (complexity $O(n^2)$) sequential algorithm. It starts with an initial empty set of features then repeatedly adds, at the end of each cycle, the feature that scored the best result when taken together with the current best subset. The selection stops when any further addition will damage the accuracy obtained with the current subset. The main disadvantage comes from its assumption: features are first tested separately (we start with an empty set) and not for their effect when taken together with the others, so that there is a high risk for the algorithm to be trapped in a local minimum.

A partial solution to this problem can be reached with the *Backward Sequential Selection (BSS)* algorithm: it follows the same idea of the Forward technique but in the opposite direction. It starts with the full set of features and repeatedly removes one feature at the end of each cycle, choosing the one which produces the best result if removed. The selection stops when a further removing will not improve the accuracy of the current subset. This algorithm has the same complexity of the FSS one, but it's much heavier computationally, as it has to calculate any distance with a considerably higher number of features (it starts to calculate n different accuracy for subset of n-1 features, while the FSS starts with n subsets of 1 single feature). Furthermore it doesn't solve the problem that a discarded feature cannot be reconsidered anymore.

Some refinements of Forward and Backward Sequential selections can also be considered. For example the *Bi-directional Search* performs together FSS and BSS; at each step of FSS care must be taken to avoid adding a feature that has already been discarded by BSS and, as well, BSS must never discard a feature that has already been added by FSS. The two algorithms are forced this way to converge to the same feature subset. A generalization of both FSS and BSS is given by the *(p,q)-Sequential Selection.* Given the two non-negative integers p and q, the algorithm starts either a forward sequential selection by adding the best p features and than removing the worst q in the case $p<q$, or a backward sequential selection by removing the worst q before adding the best p in the case $p>q$. Of course, if $p=1$ and $q=0$ this algorithm becomes the classical FSS, while if $p=0$ and $q=1$ it becomes the BSS.

A more sophisticated variant, called *BEAM Search*, was proposed by Doak (1992). It involves the use of a bounded queue of the Best-Visited states, which is updated

regularly at each step. It might be based on FSS or on the BSS. In this last case the technique is a "reduction" of the BSS, in the sense that at each step not all the possible states are visited, but only those ones that can be derived from the states appearing in the queue (i.e only the sub-branches of the BSS tree that have their route included in the queue).

Another interesting category of Sequential Search Algorithms is *Hill-Climbing (HC)*. These are gradient descendent techniques that start from a random point (a random feature subset) then perform a search through the direction where the gradient indicates a minimum for the evaluation function. Even though they present a little randomness (the starting point) these can be considered as sequential techniques: once the starting point is defined there is only one possible path to the solution. A practical implementation in the case of feature selection is shown in (Cunningham & Carney, 2000): the algorithm starts computing the accuracy of the initial feature subset, then it flips the first "bit" (first feature) of the feature vector (if it was "1" turns into "0" and vice-versa). If the Accuracy after the flip is greater than the one before the flip, the flip itself is accepted otherwise it is rejected. The cycle is then carried on sequentially for all the other bits until the last one is reached. Then a new cycle is repeated, starting from the first up to the last bit; the whole process stops when no more flips are accepted. This is a relatively simple and effective algorithm; its main downside is the high risk of getting trapped into a local minimum if the starting point is not chosen sufficiently far from it. However we will see in Chapter 4 that when used in an Ensemble the possibility of getting trapped in local minima has a much smaller impact on the final prediction. Some refinements based on the Hill-Climbing paradigm can be found in Caruana & Freitag (1994) and Skalak (1994).

c) Randomized Algorithms

Randomized algorithms perform the search with a completely different philosophy from the sequential ones. At each step the new feature subset is not selected following the Evaluation Function's response, but using instead a random operator. In these algorithms the Evaluation Function performs its task in a slightly different way; it gives to all the "population" of new elements a score and only those with the highest ones will be retained (the others are discarded).

Usually the random operator is formulated following some of the examples that exist in nature, *Genetic Algorithms (GA)* and *Simulated Annealing (SA)* are among the most popular ones.

The philosophy behind Genetic Algorithms is to simulate the natural selection of the best individuals in evolutionary processes in the hope to find efficient solutions. The first step for a Genetic Algorithm is to construct an initial population of candidate individuals (solutions); then the selection process begins and at the end of each cycle a new generation of individuals is created, using the genetic operators *Crossover* and *Mutation*; at this stage the Evaluation Function selects the best individuals that will be the "parents" in the next generation, and the loop goes on until a predetermined number of generations is reached. The Crossover operator simulates what happens when two individuals generate two other completely new individuals: their genetic information is mixed together and the new individuals have some information from one of the parents and some others from the other one. In the case of Feature Selection every individual's chromosome is the feature vector. Here below is an example of One-Point crossover:

1100 ¦ 0110	1st parent
0101 ¦ **0101**	2nd parent
1100 ¦ 0101	1st child
0101 ¦ 0110	2nd child

The Mutation operator acts immediately after the Crossover; it allows the search algorithm to "jump" into an unexplored region of the search space, by randomly selecting a digit of the chromosome-vector and mutating it with a certain probability P_m. In the case of Feature Selection the mutation operates by substituting the element with its binary complement.

The main advantage of Genetic Algorithms is the fact they may overcome the problem of getting stuck in local minima; on the other hand Genetic Algorithms are extremely sensitive to the choice of the parameters that characterize the selection, like the population size, the maximum number of generations, the Crossover rate and the Mutation rate. A wrong selection of one or more of these parameters may lead to very poor performances. Good examples of Genetic Algorithms applied to feature selection can be found in Jarmulak & Craw (1999) and Yang & Honavar (1997).

Simulated Annealing is inspired by the physical thermal phenomenon of cooling a substance to its low-temperature equilibrium. In this model, given an initial state and

an initial value of a variable T referred as "temperature", changes of states are always accepted if they lead to states with a lower energy E (in the analogy corresponding to a lower cost) and accepted with a certain probability $p = \exp(-dE/T)$ if they lead to a higher energy state. The temperature T (which is an artificial variable that in the analogy regulates the cooling process) is slowly decreased until an equilibrium-state is reached. The algorithm performs two nested loops: the outer one is on the temperature decreasing, the inner one performs a search on the "neighbours" elements and it is decreased only when a state is not accepted, so that it is possible - at each step of the outer loop – to visit several elements before decreasing the temperature. A crucial element to define a good Simulated Annealing algorithm is the "transform" operator (i.e. "how" to select a neighbour state): usually in the case of binary vectors bit flipping is performed with a certain probability P. Like Genetic Algorithms, Simulated Annealing are also algorithms with high sensitivity to the parameter selection. Examples of Simulated Annealing applied to feature selection can be found in Boese & Kahng (1994) and in Sutter et al. (1995).

To say which one, among all these techniques, would be the best is almost an impossible task. The best type of search can be strongly dependent to the particular kind of task examined. As a general guideline we might say that when the search space is not highly complex a sequential search may be the best to reach the optimal state; on the contrary, for a highly complex search space we might be considering a randomized algorithm, that rarely reaches the optimal state (absolute minimum), but is more robust about the possibility of being trapped in a local minimum. We already mentioned the fact that, as we will use classifiers in the context of ensembles, the impact that the possibility of getting trapped in local minima has in the final accuracy will be dramatically reduced. This is one of the reasons why in next Chapter 4 we will make use of ensembles of classifiers based on different feature subsets that are trained with Hill-Climbing techniques; a more detailed explanation of such a choice will be given in the same chapter.

3.3 Knowledge Discovery and the Instability Problem

So far we have considered the Feature Selection approach exclusively as a mean to improve Classification accuracy (or Squared Error in the case of regression); basically as a strategy that tries to find the subset of features that maximizes performance. In reality this is only the first of three main motivations that lead to the Feature Selection theory. The second one is the idea Feature Selection may help in achieving economy of representation and allowing problems/phenomena to be represented as succinctly as possible. The third one is its use for knowledge discovery: Feature Selection may help to discover what features are and are not influential in weak theory domains.

We have shown in the previous sections with numerous examples taken from research how the first issue is certainly fulfilled by the Feature Selection process. The second one is quite straightforward: the use of Feature Selection will certainly reduce the dimension of the problem space, this follows directly from the definition. The third one is a lot more contradictory and we will spend the rest of this section to discuss it.

At first sight it may seem obvious that Feature Selection can have a key role in knowledge discovery. After all if we can find which, among the entire set of features, are the predictive ones we can also draw some conclusions about which are the variables describing the domain. This characteristic of Feature Selection certainly looks very appealing and can be though as a further step toward explanation performed by CBR systems (after the capability to retrieve the most similar cases to the query). The problem is, we cannot deduce that the feature subset scoring the best performance on a given Case-Base is the set of features that really represents the phenomenon. First we give some theoretical reasons then we proceed with an experimental evaluation.

The *k*-Nearest Neighbours paradigm is based on the assumption that the domain we are dealing with can be represented with a Euclidean model. This is a strong assumption that may not correspond to what the domain really embodies. In this case the search for the best feature subset is a search only in the sense of the "performance": we can only say that a feature subset is the one that best approximates the real domain function in a Euclidean scenario. But there is more. When we train the best feature subset for a CBR tool, this is always referred to a given Case-Base. The

Feature Selection training process is particularly sensitive to the variations that may occur in the sampled data, especially when it happens the case-base is an under-sampled version of the real phenomenon. This is also known as the *instability problem* for Feature Selection. An interesting work on the subject can be found in Dunne et al. (2002); that research shows how the performance of a wrapper-based feature selection process can be highly sensitive to the training dataset variations (in this context, a low sensitivity process would be the one capable to retrieve the same feature subset irrespective of the variations in the partitioning of the training data). It is worth remarking here that, while the process of finding the best feature subset can be highly sensitive to the variations in the training set, the performance score (accuracy) of a *k*-NN is instead not very sensitive to the same variations.

Here below we show an experimental comparison to measure how sensitive Feature Selection is to the variations in the Case-Base. For each of six different databases, we have run 15 times a Backward Sequential Selection (see section 3.2.2) to select the best feature subset, each time randomly removing 20% of the cases taken in order to have 3 different configurations of 5 disjointed subsets each. We have performed the BSS algorithm each time on the remaining 80% of the training dataset with a leave one out technique as if it were the entire dataset; we have then measured the average Hamming distance between the best feature subsets found in all the 15 different cases. We recall here that the Hamming distance between two binary vectors can be computed as:

$$\sum_{i=1}^{n} 1[v_1(i) \neq v_2(i)] \qquad \textbf{(3.3.i)}$$

where the function 1[.] is computed as:

$$1(v_1(i) \neq v_2(i)) = \begin{cases} 1 & v_1(i) \neq v_2(i) \\ 0 & otherwise \end{cases}$$

In other words the Hamming distance counts the number of different digits between the two binary vectors. The results are shown in the following table:

Name of Database	Total Number of Features.	Average Hamming Distance	% Hamming Distance on the n. of feat.
Heart Disease	13	3.74	28.75%
Horse Colic	27	8.28	30.67%
Ionosphere	34	10.88	32.0%
Pima Indians	8	2.58	32.25%
Spect	22	6.74	30.64%
Warfarin	22	7.80	35.46%

Table 3.a. Average Hamming Distance of the chosen feature subset for classifiers trained in different parts of the Case-Base

The data in table (3.a) show a clear instability problem. In any of the six databases, when training on two different random subsets (corresponding to 80% of the entire database), the average Hamming distance between the two best feature subsets found is around 30% of the total number of features; we can roughly say 1 feature over 3 differs.

This certainly looks like a remarkable difference in the chosen feature subset, so that it doesn't appear fair to draw any conclusion about how representative of that phenomenon a certain feature subset is. In other words, if the feature subset described well the domain model we would certainly expect it to appear in a more remarkable way when varying the initial condition. Some techniques to overcome this problem have been proposed. For example, in the research mentioned above Dunne et al. (2002) have proposed to repeat the whole training process a certain number of times with different training set sub-samples, and then to compute statistics about how frequently any feature appears in the chosen feature subset. The features with the highest number of "votes" are finally included in the subset of the most predictive ones. It is interesting to notice how this idea uses a sort of "averaging" approach to remove instability; this will be also the leading philosophy behind Ensemble techniques that we will show in next Chapter 4.

An additional issue that raises some doubts about the use of global feature selection for knowledge discovery comes from research on local feature selection.

Some recent studies (examples can be found in Domingos, 1997; Aha, 1998b; Wettschereck et al., 1997 and Howe & Cardie, 1997) have stressed that a single feature might be predictive only in sub-regions (especially in very complex spaces) and irrelevant in the rest of the search space. Under this perspective the best *global* feature subset may be only considered as a sort of rough "average measure" of the features relevance throughout the entire space: any other attempt to give a meaning to the feature subset that globally describes the domain would hardly be acceptable. A good example of local feature selection algorithm is given in Domingos (1997), who proposed a Context Sensitive feature selector called *RC (Relevance in Context)* that performs the search in parallel for all the instances in a clustering-like fashion: each instance looks for the nearest example in the same class and hypothesizes that the features along which they differ are irrelevant. This is tested by checking whether deleting those features from the instance has a positive or negative effect on the accuracy of the classifier: at each cycle the algorithms keeps on deleting features until a negative effect is verified; at this time the selection stops and all the remaining features are retained. This is processed for each instance, coming to the point of separate different subsets for several different zones in the search space.

Research about the "global VS local" issue in Feature Selection is still open and some new developments may be found in the future. However, for the purposes of our study we won't consider any further the search of the best feature subset as a mean to support knowledge discovery; we will simply use Feature Selection as a mean to improve the accuracy in classification/regression performance.

3.4 Summary of the Chapter

In this Chapter we have presented the Feature Selection problem. In section 3.1 we have introduced the notation for Feature Selection and Feature Weighting and stressed some of the most important differences between these two approaches. In section 3.2 we have revised some of the most successful strategies to select the best feature subsets proposed in the literature. In section 3.3 we have outlined some considerations and shown an experimental evaluation about the use of Feature Selection in support to Knowledge Discovery, particularly in relation to Instability Problems.

In Chapter 4 we will introduce the idea of ensembles of classifiers/predictors; we will see how combining a number of different classifier leads us to improvements both in terms of performance and stability. We will also draw some considerations about local feature relevance in relation to ensembles.

Chapter 4

Ensembles of Case-Based Reasoners

4.1 Introduction to Ensembles

The main idea that leads to the definition of an ensemble of predictors comes from a simple observation: it is frequently the case that, in any field, a group of experts can take better decisions than a single one. Expressing this colloquially: "two heads are better than one". The intuitive motivation comes from the observation that there rarely is an expert that can have such a wide knowledge to cover all the possible aspects of a domain. Each expert can instead specialize on a particular aspect of the domain so that when the time comes to take a decision for a pool of such experts more information is available. The ensemble theory is actually a formalization of this very intuitive approach. It is interesting to notice that with just an intuitive approach we have already a feeling of how the "diversity of opinion" is important for this ensemble to succeed; we will see later on in this chapter a formalization of the concept of diversity.

4.1.1 Basic concepts and definitions

Suppose we have a set of m different predictors (the meaning of "different" will be stated in more detail in the next section), a training set T and a query case q. Each predictor will output its prediction on the query case depending on the training set. Let's denote the prediction of the predictor i with $f_i(q, T_i)$. In general we may have each predictor making use of a different subset T_i of the case-base T; in the case where every predictor uses the whole case-base then we will omit for simplicity the dependency on the training set T.

An ensemble E of predictors is defined by providing a set of predictors together with a *combination rule*. A combination rule is a function $F: Y^m \rightarrow Y$ that states how all the single-predictor outcomes will be combined to obtain an outcome from the whole "pool"; this is also called the *aggregation function*. An ensemble is itself a predictor on the training set T whose prediction f_E is defined by the combination rule applied to the set of single predictors:

$$f_E(q,T) = F\{ f_1(q,T_1), f_2(q,T_2), ..., f_m(q,T_m) \} \quad \textbf{(4.1.i)}$$

in the case that every T_i is the whole case-base T we will write:

$$f_E(q) = F\{f_1(q), f_2(q), ..., f_m(q)\} \quad \textbf{(4.1.ii)}$$

In literature there is a large variety of proposed ways to combine the outcomes of an ensemble of predictors. They mainly depend both on the type of predictor in use (k-NN, decision trees, neural nets, etc.) and the type of supervised learning problem (whether it is classification or regression). Here we will briefly review the most common ones.

Perhaps the simplest approach to aggregate an ensemble when dealing with a classification problem is by simple voting (Boland, 1989). As we have seen in chapter 2, a simple strategy for a k-Nearest Neighbours to assign a class to the query is to count the number of times each class occurs and return the most frequent one; this corresponds to using equation (2.2.ii) with a constant Kernel function, like in (2.2.iv).

It is quite straightforward then to extend such a simple principle to an ensemble of classifiers, returning the class that has obtained the majority of the votes among all the single classifiers; obviously the single classifier probabilities (real values) are mapped into classes (discrete values). The following figure (4.I) shows how this would operate in the case of a 3-nearest neighbours and a binary classification problem (the two classes are marked as a white and black dots).

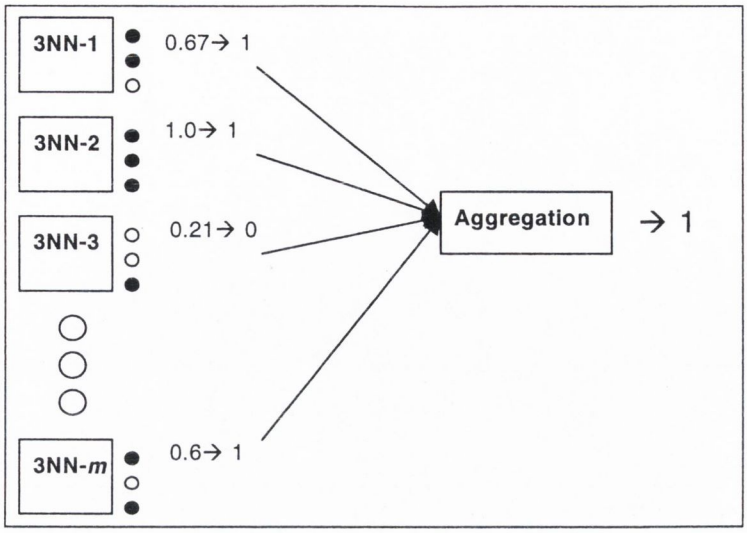


Fig. 4.I. An example of aggregation of 3-NN in Ensemble

This simple method presents some problems. First, it is clear how to compute the majority class in the case of a binary classification problem and an odd number of classifiers, but this is less straightforward when we have an even number of classifiers or, even worse, when we have a higher number of classes. Second, it doesn't take into account how confident of a prediction each single classifier is; for example a classifier whose probability outcome for a certain class is 51% will count as much as a classifier whose probability outcome for the same class is 99%.

These problems lead to the definition of the *weighted average* combination function, undoubtedly the most popular in the field of ensembles. The weighted average can be used both for classification and regression problems.

Given a query q and a set of m classifiers, the weighted average combination rule ensemble outcome is defined as:

$$f_E(q,T) = \sum_{i=1}^{m} w_i \cdot f_i(q,T_i) \qquad \textbf{(4.1.iii)}$$

where $f_E(q,T)$ is the ensemble outcome, $f_i(q,T_i)$ is the outcome of the predictor i while w_i is the weight associated to the same predictor. Usually the weights have to obey to a normalization constraint:

$$\sum_{i=1}^{m} w_i = 1 \qquad \textbf{(4.1.iv)}$$

There are many ways to define weights. The simplest one is to assign every predictor the same weight in the following manner $w_i=1/m$. Another popular way is to give each predictor a weight inversely proportional to its error, so that the most accurate predictors count more than the less accurate ones. Some others more sophisticated ways to determine weights have been proposed in the literature; anyway this particular topic will not be investigated any further in the course of this research. All the experiments we will carry out in the evaluation sections are based on the equal weights strategy, which has proved to be fast enough and satisfactory for our goals.

Let's examine more in detail the weighted average aggregation function. In the case of a regression problem equation (4.1.iii) is quite straightforward: the ensemble outcome (a real value) can be obtained through simple algebraic steps.

In the case of a classification problem we need to apply the same weighted average principle to each of the class probabilities; this time the outcome f_i of each classifier and the outcome f_E of the ensemble in equation (4.1.iii) become the class probabilities p_i and p_E. Here below we show the version of equation (4.1.iii) for classification problems.

$$p_E(c_j|q,T) = \sum_{i=1}^{m} w_i \cdot p_i(c_j|q,T_i) \qquad \textbf{(4.1.v)}$$

Once the probabilities p_E are computed for each class we can then associate a class to the ensemble prediction by simply taking the *arg max* (p_E), the same way we have already seen in Chapter 2, equation (2.2.i), for the case of a *k*-Nearest Neighbours classifier.

It is interesting to notice that in the case of a binary classification problem it is always possible to avoid the computation of both class probabilities by simply mapping the two classes into 0 and 1 and taking as output the probability associated to the class 1. In this case the output has the information for both classes 0 and 1 (for example an output of 0.32 says that class 1 has 32% probability and class 0 has 68% probability, while an output of 0.99 says that class 0 has 1% probability and class 1 has 99% probability), so that we can apply directly equation (4.1.iii) we would in the case of regression.

Several other aggregation functions have been proposed in the literature, like for example the logarithmic pool (Heskes, 1998). However this topic will not be

investigated by this research. From now onwards we will only concentrate on the simple voting and weighted average aggregation functions.

4.1.2 Different sources of diversity in the literature

In the previous sections we have shown that the two elements needed to define an ensemble are: first, a group of "diverse" predictors, second an aggregation function (combination function). In the last section we have shown some of the most widely used combination functions, in this section we will concentrate on how to select a group of diverse predictors.

Several ways to differentiate members of an ensemble of classifiers have been proposed in literature. Here we will only briefly outline some of the main ways to produce "diverse" members of an ensemble, a more complete survey on the subject can be found in Dietterich, (1997, 2000a).

a) training set manipulation

Perhaps the most common source of diversity is by training members on different subsets of the training data. In ML research it is well known that ensembling will improve the performance of unstable learners. Unstable learners are learners where small changes in the training data can produce quite different models and thus different predictions. Thus, a ready source of diversity is to train models on different subsets of the training data. This can be done systematically by bootstrapping (sampling with replacement) different training sets from the training data. Such an approach has been applied with great success in eager learning systems such as Neural Networks (Hansen & Salamon, 1990) or Decision Trees (Breiman, 1996). Another training sampling method that gives promising results is performed by leaving out disjoint subsets of the training data, similarly to the Leave One Out testing methodology (Parmanto, Munro & Doyle, 1996). A third popular way for manipulating training sets was proposed by Freund & Shapire (1995, 1996) with the *AdaBoost* algorithm; Adaboost manipulates the training set in order to draw multiple hypothesis; it associates a set of weights over the training cases and, in iteration j, the algorithm tries to minimize a weighted error on the training set, and returns the current hypothesis H_j. Then weights are updated, based on the current hypothesis error, so that misclassified examples are weighted more in the next iterations. At any iteration the

learning problems become gradually more difficult, until a sufficient number of iterations is performed; then the final classifier is built by a weighted average of the individual classifiers, each weight depending on the single classifier accuracies.

All the researches mentioned above shows that, for difficult classification and regression tasks, ensembling will improve the performance of unstable learning techniques such as Neural Networks and Decision Trees.

b) feature subsets

Ensembling will also improve the accuracy of lazy learners such as *k*-Nearest Neighbours classifiers, however *k*-NNs are relatively stable in the face of changes in training data so other sources of diversity must be employed, unless the training dataset is very large and for computational reason it is wise to have a reduced training set for each classifier (Alpaydyn, 1995).

A few studies have been done on the use of feature selection to create an ensemble of classifiers; among them those ones made by Cherkauer (1995), who trained an ensemble of neural networks to identify volcanoes on Venus, Ho (1998a, 1998b), Guerra-Salcedo & Whitley (1999a, 1999b), Tumer & Ghosh (1996) and Cunningham & Carney (2000) give the most promising results.

However, if the use of ensembles improves the performance on the one hand, on the other it reduces the other benefits of feature selection. It is clear that an ensemble of feature subsets doesn't fulfill the goal of economy of representation (see section 3.3), seen that the complexity dramatically increases when using a pool of classifiers instead of a single one. As for the knowledge discovery issue, when we move to an ensemble we cannot say anymore that the outcome of a phenomenon depends on a particular subset of features. Anyway this doesn't seem to be the biggest problem: we have shown in 3.3 there are serious reasons to doubt about knowledge discovery even in the case of a single classifier, as the feature selection process proved to be quite unstable with regards to the chosen feature subset. In the last section of this chapter we will outline the proposal that the lack of interpretability associated with ensembles may be recoverable if the ensemble members prove to be *local* learners.

c) output target manipulation

Another approach to build an ensemble of classifiers, called *error-correcting output coding*, was proposed by Dietterich & Bakiri (1995) in the case of a large

number of classes. The philosophy behind this idea is to partition repeatedly the training set into two subclasses and give a simple binary classification problem to a new classifier in any iteration; following this principle input data are re-labeled with a "0" or a "1" depending on which partition they belong to, then the learning algorithm constructs a classifier on this newly labeled training set. The process is then repeated a number of times, generating each time a different partition of the training set and a subsequent different associated classifier. The query is then given to each classifier; if it replies "0" than all the classes corresponding to the partition associated to "0" receive a vote; the opposite if "1"receives a vote. The class with the highest vote is returned as the prediction for the query. Recently Schapire (1997) has also shown that Adaboost can successfully be combined with this technique, while Ricci & Aha (1997, 1998) have obtained good results combining error-correcting with feature selection.

Several other techniques for the construction of a good ensemble of "diverse" classifiers have also been proposed in literature, following other philosophical approaches. They are not particularly related to the work we are concerned with here, so we will only mention the works by Neal (1993), who approximated Bayesian voting on a number of hypotheses with Markov chain Monte Carlo methods, the works by Kolen & Pollack (1991), Kwok & Carter (1990), Dietterich (2000b), who tried to differentiate neural networks and decision trees by injecting randomness into their parameters, and the work by Raviv & Intrator (1996), who combined the bootstrap sampling idea with the injection of noise into the input features.

4.2 Theoretical foundations of Ensemble techniques

After an introduction on the basic ideas behind ensemble techniques and the corresponding definitions, a question arises very naturally: why should ensembles work? So far nothing can guarantee us that the process of aggregating several different predictors or classifiers will really produce the hoped results without introducing any other related problem. This section answers the question by providing some theoretical background. From now onwards we will refer always to a classification problem unless it is not otherwise specified.

Informally, we can say that the main hope behind ensemble research is: if a classifier or predictor is either inaccurate or unstable then an ensemble of such classifiers voting on the outcome should produce better results – better in terms of accuracy and stability. A classifier is "accurate" when it predicts well the outcome of the queries, in other words when its generalization error is low. A classifier is "unstable" when its performance varies substantially depending on the initial conditions; what is a good classifier with a certain set of queries referred to a given case-base T_1 may become a poor classifier with another set of queries referred to a different case-base T_2. We have already met the instability problem, under a slightly different guise, in Chapter 3 when we introduced the feature selection paradigm. We will see how these two problems may be faced with the use of an ensemble of classifiers; let's start first with some "history" on the topic.

While the use of ensembles in Machine Learning research is fairly new, the idea that aggregating the opinions of a committee of experts will increase accuracy is not new. The Condorcet Jury Theorem states that:

> *"If each voter has a probability p of being correct and the probability of a majority of voters being correct is M, then $p > 0.5$ implies $M > p$. In the limit, M approaches 1, for all $p > 0.5$, as the number of voters approaches infinity."*

This theorem was proposed by the Marquis of Condorcet in 1784 (Condorcet, 1784) – a more accessible reference is (Nitzan & Paroush, 1985). We know now that M will be greater than p only if there is diversity in the pool of voters. And we know that the probability of the ensemble being correct will only increase as the ensemble grows if the diversity in the ensemble continues to grow as well. (Typically the diversity of the ensemble will plateau as will the accuracy of the ensemble at some size between 10 and 50 members.).

Dietterich (2000a) gives a very simple and intuitive survey on the reasons why ensemble techniques can improve the performance of a single predictor. He focuses on three main reasons: statistical, computational and representational. The first occurs when the amount of training data is too small compared to the size of the hypotheses space: without sufficient data many different hypotheses will score the same accuracy so that averaging will reduce the risk of choosing the wrong one. The second occurs when there is enough training data but the hypotheses selection may be affected by getting stuck in local minima (like Hill Climbing or gradient descending techniques):

in this case averaging will provide a better approximation to the target function. The third occurs when the target function doesn't belong to the hypotheses space (so it cannot be represented by any of the models of our space): in this case asymptotic properties of many machine learning tools may be combined with averaging to obtain an acceptable approximation of the target function.

Apart from intuitive reasons, there is a whole branch of statistics research that has obtained excellent results in regards of explaining ensemble techniques. The next sub-section 4.2.1 will introduce a very powerful tool, the Bias/Variance decomposition, and show how it relates to the problem of Overfitting, while in 4.2.2 we will see how the concept of diversity may be formalized for an ensemble of predictors. Subsection 4.2.3 will relate these concepts to CBR tools and in particular to ensembles of *k*-Nearest Neighbours, while subsection 4.2.4 will focus on ambiguity measures for ensembles of *k*-NN classifiers.

4.2.1 Bias/Variance decomposition and the Overfitting problem

In the literature, some powerful statistical tools have been proposed to analyze the error scored by predictors (regression and classification problems), both in the case they perform singly and in the case they are aggregated to form an ensemble. Many are the references available on this topic; an excellent work that collects the formalization and the analyses of all the most meaningful decompositions available from nearly two decades of research is the PhD dissertation by Hansen (2000). In this section we will briefly introduce one of the most important decompositions for a single classifier, the Bias/Error decomposition; the notation will mainly refer to the work by Hansen mentioned above.

Given a predictor f, the error E gives a measure of how wrong the predictor is, in respect to the target function t. In Chapter 2 we have outlined some of the most widely used loss functions to quantify error both in classification and regression problems. Generally, it is common to use a 0/1 Loss Function (equation 2.2.ix) in the case of classification problems and a Mean Squared Error (MSE) in the case of regression problems (equation 2.2.x); a final measure of the Error is usually given as the mean value over a certain validation set (test set) as follows:

$$\overline{\mathsf{E}}(t,f) = \langle \mathsf{E}(t,f) \rangle_{\underline{X}} \qquad \textbf{(4.2.i)}$$

It is common to assign equal weights to the queries so that the mean value is computed as a simple arithmetic average over the query set (see 2.2.2).

When dealing with a single classifier a general request is to have a robust measure of its performance; it could be the case, for example, that a classifier performs well on a certain subset of queries and poorly on another one so that having an error estimation with only one "trial" may be misleading. In this case it's common to test our classifier with a number of query subsets and then average the results over all the subsets. We will indicate this type of averaging in the following way:

$$\left\langle \overline{\mathsf{E}}\left(t, f(T_i)\right)\right\rangle_{\underline{T}} = \left\langle \mathsf{E}\left(t, f(T_i)\right)\right\rangle_{\underline{X,T}} \qquad \textbf{(4.2.ii)}$$

With $f(T_i)$ we have indicated the classifier tested on a certain subset T_i (we have omitted for simplicity the dependence on q); for example, in the case of Leave One Out it corresponds to one of the N folders used for the cross-validation.

A classic decomposition of the generalization error for a single classifier is the one where the error is decomposed in two parts with different meaning: *Bias and Variance*. In the following equation we can see the expression of the two terms in the case of the Mean Squared Error loss function:

$$\left\langle \overline{\mathsf{E}}_{MSE}\left(t, f(T_i)\right)\right\rangle_{\underline{T}} = \left\langle \left[t - f(T_i)\right]^2 \right\rangle_{\underline{X,T}} =$$
$$= \left\langle \left[t - \overline{f}\right]^2 \right\rangle_{\underline{X}} + \left\langle \left[\overline{f} - f(T_i)\right]^2 \right\rangle_{\underline{X,T}} = \overline{\mathsf{E}}_{MSE}\left(t, \overline{f}\right) + \left\langle \overline{\mathsf{E}}_{MSE}\left(\overline{f}, f(T_i)\right)\right\rangle_{\underline{T}}$$

(4.2.iii)

The average on the input space $\underline{X}$ is indicated with $\overline{\mathsf{E}}$ while with $\overline{f} = \left\langle f(T_i)\right\rangle_{\underline{T}}$ we have indicated the average of the predictor trained over all the different training sets; we will call $\overline{f}$ *Average Predictor*.

The left member of the equation is the generalization error, while the right member is a sum of two parts, the first one is called *Bias* and the second one is called *Variance*. The bias is defined as the generalization error of the Average Predictor while the variance is defined as the average (over both the input space and all the different fold) of the error between the Average Predictor and the predictors trained on $T \backslash T_i$ and tested on T_i (if T is the whole dataset, and T_i the chosen test subset, then $T \backslash T_i$ becomes the training subset), for each i.

This decomposition allows us to analyse the relationship between the complexity of a single classifier and the error scored and give an intuitive explanation of bias and variance. In this context we assume as *complexity* of a classifier the number of parameters characterizing the classifier itself (for example in a neural network it would be the number of units in the hidden layer or in a k-NN it would be k).

In the figure shown below we can see the difference between a "simple" and a "complex" classifier (the axis units are simply indicative and not important in this context).

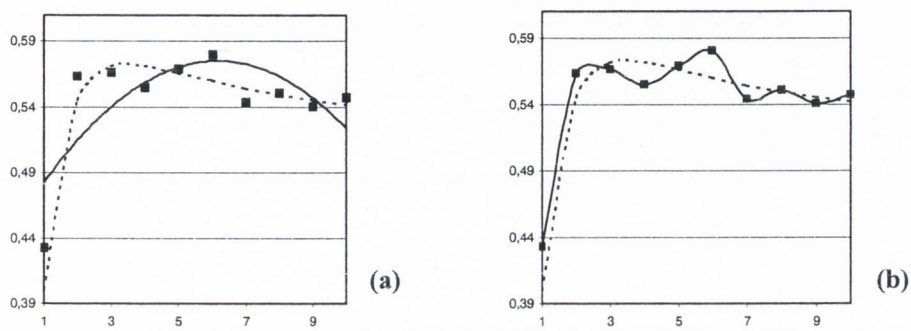


Fig. 4.II. An example of low complexity (a) and high complexity (b) classifiers.

The dotted line represents the target unknown function, the small squares are the sampled data drawn randomly from the distribution (affected by some noise) while the continuous line represents the model derived from the sampled data. In simple words we may say that the *Bias* is concerned with the way the low-complexity classifier in figure 4.II(a) errs: the complexity is too low to interpolate the curve in a sufficiently good way. On the other side the *Variance* is more concerned with the way the high-complexity classifier in figure 4.II(b) errs: the classifier's complexity is so high that it tries to interpolate the noise as well. We have seen this last problem is also known in statistics as the problem of *overfitting*.

In both cases the scored generalization error will be poor, even though in each case a different component of the Error (Bias or Variance) is dominant. When choosing the complexity of a classifier it is common practice to choose the point where the *sum* of Bias and Variance reaches the minimum.

Of course, the theory behind Bias and Variance decomposition can be extended far beyond the simple example presented and it is possible to show that some Machine

Learning techniques may have an intrinsic higher bias or variance than others. However, even though there are two different components they are not equally bad in the way they affect the generalization error; as a general trend we can affirm the bias can be more difficult to address compared to the variance.

On the complexity side we can observe that a "too complex" classifier embeds somehow the information we need while the "too simple" classifier doesn't have enough information, so that they are not equivalent: we prefer to have a classifier that knows "too much" instead of having one that knows "too little". The point is: how to get the information we need from a classifier that knows too much? In other words, how do we get information from a classifier which highest component of the error is variance, so to face and solve the problem of overfitting?

The answer to these questions is given by a simple principle: given that the variance is a stochastic component of the error, averaging will reduce it. The following figure illustrates the principle:

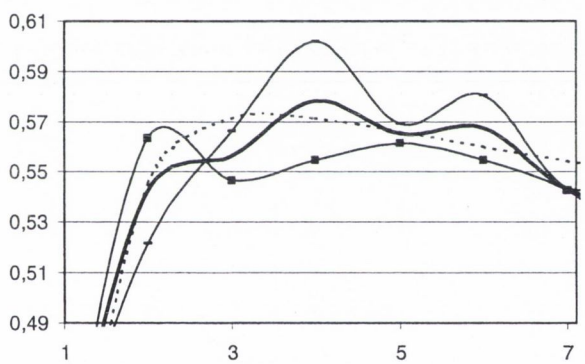


Fig. 4.III. The principle of Averaging to reduce the Error.

The thin dotted line is the unknown target function, the two thin continuous lines are two different complex classifiers obtained through a random sampling of the function space (both of them show a serious problem of overfitting), while the thick continuous line is the linear average of the two classifiers. As we can see the average scores a better result than the two single classifiers taken singularly. The whole theory of ensembles is based on this principle of averaging. In the next section we will see how the idea of averaging is combined together with the idea of the diversity of classifiers to give an answer to the overfitting problem.

We will end this section with some other useful references. In statistics and Machine Learning literature there is a whole branch of research that has successfully tried to extend the powerful Bias/Variance decomposition to many other Loss Functions. Some very interesting works have been made by Hansen&Krogh (2000a) who studied a generalization with target independent variance of error functions derived from the exponential family of distributions, Domingos (2000) who studied a unified bias-variance decomposition for 0/1 and Squared Loss, Kohavi&Wolpert (1996) and James&Hastie (1996) who discussed some other extensions to the 0/1 Loss, Heskes (1998a, 1998b) who proposed an extension to the Kullback-Leibler Loss function, and Tibshirani (1996) who derived bootstrap estimates of the components of prediction error from its decomposition.

4.2.2 Ambiguity decomposition: the importance of Diversity

A fundamental work on ensemble of neural networks was made by Krogh & Vedelsby (1995). In that paper a theoretical framework for ensembles is presented. It is shown how the error in an ensemble is directly proportionate to the average error of the single classifiers minus a term which counts the diversity in the predictions of the components of the ensemble, as measured by variance.

The whole theory is drawn for the Mean Squared Error (MSE) loss function and leads to a concise formula that describes how the ensemble generalization error depends on both the average error of a single classifier and the Ambiguity, a term which counts "how differently" the classifiers of the ensemble predict.

$$E = \overline{E} - \overline{A} \qquad \textbf{(4.2.iv)}$$

The definitions are the following; for consistency we have used the same symbolic notation used in the previous sections and not the one used by Krogh & Vedelsby (1995):

E is the *Generalization Error* of the ensemble, obtained as

$$E = \overline{\mathsf{E}}_{MSE}(t, f_E) = \left\langle [t - f_E]^2 \right\rangle_{\underline{X}} \qquad \textbf{(4.2.v)}$$

$\overline{E}$ is the *Average Error* of all the single classifiers:

$$\overline{E} = \left\langle \overline{\mathsf{E}}_{MSE}(t, f_i) \right\rangle_{\underline{F}} = \left\langle [t - f_i]^2 \right\rangle_{\underline{X,F}} \qquad \textbf{(4.2.vi)}$$

$\overline{A}$ is the *Ambiguity* of the ensemble, defined as:

$$\overline{A} = \left\langle \overline{\mathrm{E}}_{MSE}(f_E, f_i) \right\rangle_{\underline{F}} = \left\langle [f_E - f_i]^2 \right\rangle_{\underline{X},\underline{F}} \qquad \textbf{(4.2.vii)}$$

The average $\langle . \rangle_{\underline{F}}$ must be computed over all the classifiers f_i of the ensemble (they belong to the set of all the possible different classifiers $\underline{F}$) using the same set of weights w_i chosen to compute the ensemble output f_E as shown in equation (4.1.iii). These classifiers may differ using any of the strategies mentioned in section 4.1.2.

The equation (4.2.iv) introduces a fundamental decomposition for ensembles: in practical terms it formalizes the intuitive approach that for an ensemble to succeed it must be guaranteed a certain diversity in the classifiers' output.

If we examine carefully the Bias/Error and the Ambiguity decompositions in the case of MSE we notice that the two equations are formally very similar (we have brought the Ambiguity term on the other side of the equality):

$$\left\langle \overline{\mathrm{E}}_{MSE}(t, f(T_i)) \right\rangle_{\underline{T}} = \overline{\mathrm{E}}_{MSE}(t, \overline{f}) + \left\langle \overline{\mathrm{E}}_{MSE}(\overline{f}, f(T_i)) \right\rangle_{\underline{T}} \qquad \textbf{(Bias/Error)}$$

$$\left\langle \overline{\mathrm{E}}_{MSE}(t, f_i) \right\rangle_{\underline{F}} = \overline{\mathrm{E}}_{MSE}(t, f_E) + \left\langle \overline{\mathrm{E}}_{MSE}(f_E, f_i) \right\rangle_{\underline{F}} \qquad \textbf{(Ambiguity)}$$

In the Bias/Error decomposition $\overline{f} = \langle f(T_i) \rangle_{\underline{T}}$ is the Average Predictor, $f(T_i)$ is the "basic" single predictor trained and tested each time using different subsets T_i, while in the Ambiguity decomposition f_i is the i-th of a number of different predictors aggregated together in the ensemble function f_E. It is quite straightforward to make an observation: if in the ensemble we choose to select different classifiers using the same classifier structure trained on a number of different training subsets the two equations become equivalent. The average over the different ensemble classifiers $\langle . \rangle_{\underline{F}}$ becomes in this case the average over the different training subsets $\langle . \rangle_{\underline{T}}$ and the ensemble output f_E becomes equivalent to the Average Predictor $\overline{f}$. Some of the classic ensemble techniques, like Bagging (Breiman, 1996) rely exactly on this principle of equivalence (Friedman, 1996). From the Bias/Variance point of view, in this context an ensemble has a low bias only if the predictors forming the ensemble are complex enough to fit reasonably well the sampled data; this, in turn, will keep the variance large, but the process of averaging will greatly reduce it (see previous section). Seen from the

Ambiguity point of view, to have a small ensemble error we want the ensemble members to show high ambiguity, provided the average error of the single classifiers is not too high.

Other interesting statistical analyses on ensembling can be found in (Friedman, Hastie & Tibshirani, 1998), where the Boosting algorithm is analysed with additive logistic regression, and in (Freund, Mansour & Schapire, 2001) where a detailed analysis of the overfitting avoidance through the averaging process is presented.

Some recent research (Hansen, 2000), has extended the Ambiguity decomposition presented in Equation (4.2.iv) to a number of other Loss Functions, including those deriving from the Exponential Family of distributions and the Kullback-Leibler Loss Function (Hansen & Krogh, 2000a, 2000b), using as a driving principle the equivalence with the Bias/Error decomposition. Despite the excellent results obtained, at present there is no equivalent ambiguity decomposition available for the 0/1 loss function, that remains the simplest, both in terms of computability and interpretability, and most widely used Loss Function for classification.

4.2.3 Ambiguity and Bias/Variance for ensembles of *k*-Nearest Neighbours classifiers

From the previous section we have seen that we have a complete equivalence of Bias/Variance and Ambiguity decompositions in the case that:

i) both the decomposition are available for a chosen loss function;

ii) the ensemble members are chosen using different training subsets (as if they were different sampling of the same distribution).

In the case of an ensemble of *k*-Nearest Neighbours, though, condition ii) seems not to hold. It is well known that *k*-Nearest Neighbours classifiers are very stable in the face of small changes in training data, so that differentiating ensemble members by training on different data subsets is almost never used; instead a widely used technique is training over different feature subsets. In this case the equivalence between Bias/Variance and Ambiguity decomposition doesn't hold anymore.

Furthermore, while in the case of a regression problem (where MSE is somehow a "natural" loss function) it's still possible to make use of the only Ambiguity decomposition even for an ensemble of *k*-NN trained over different feature subsets, in the case of a classification problem condition i) appears not to hold as well. It is quite

straightforward that the most interpretable loss function for classification problems is the 0/1 loss function and it is no surprise that any domain expert is much happier with a result that shows the percentage of scoring of any classifier (say, for example "76% accuracy") rather than with any alternative index that has to be somehow interpreted (see example in 4.2.4). Unfortunately for the 0/1 loss function it has not been possible to find such a direct Ambiguity decomposition, so that even the Ambiguity tool on its own doesn't hold anymore in classification.

So two main open questions that arise are:

a) does it still make sense to see what happens to the Bias and Variance component of the ensemble error in the case of ensembles of *k*-Nearest Neighbours?
b) is it still possible to talk about Ambiguity in the case of a classification problem?

A first step to reply to the first question comes clearly out through the study by Bay (1996) where a set of *k*-NN classifiers with random feature subsets are aggregated together to form an ensemble. In this research it is reported, among several other results, an analysis of the bias and variance components of the error scored by the ensemble; it is shown that the ensemble improves the accuracy of a single classifier with two different modes of operations: on one side it decreases variance through the aggregation, on the other one it "trades" bias for variance through the feature selection process. It is quite intuitive that, even though the *k*-NN paradigm has its own "natural bias", two classifiers with different feature subsets may have substantially different biases, in contrast to the case of the same classifier (same feature subset) trained over different subsets of the Case-Base. Another important result shown by the same paper is that an ensemble of *k*-NN trained over different feature subsets loses the asymptotic optimality properties of a single *k*-NN classifier.

To reply to the second question we need to make an observation. Even though we don't have a formalized Ambiguity decomposition for 0/1 loss function, it's quite clear that the *uplift* due to the ensemble depends on the diversity in the ensemble members.

Informally, we can say that; if the ensemble members are more likely on average to be right, and when they are wrong they are wrong at different points, then their decisions by majority voting are more likely to be right than that of individual

members. But they must be more likely on average to be right and when they are wrong they must be wrong in different ways.

The example in figure (4.IV), if a little contrived, illustrates these principles in action. The test set is shown at the top of the figure and the five component classifiers have an average accuracy of 65% on this. The ensemble decision is by simple voting and the ensemble has an accuracy of 100% on the test data. If the four voting scenarios are examined it is clear that the tendency to be right coupled with the tendency to be wrong in different ways accounts for the improvement due to the ensemble. We will show in next section 4.3 how results on ensemble analysis validate this hypothesis.

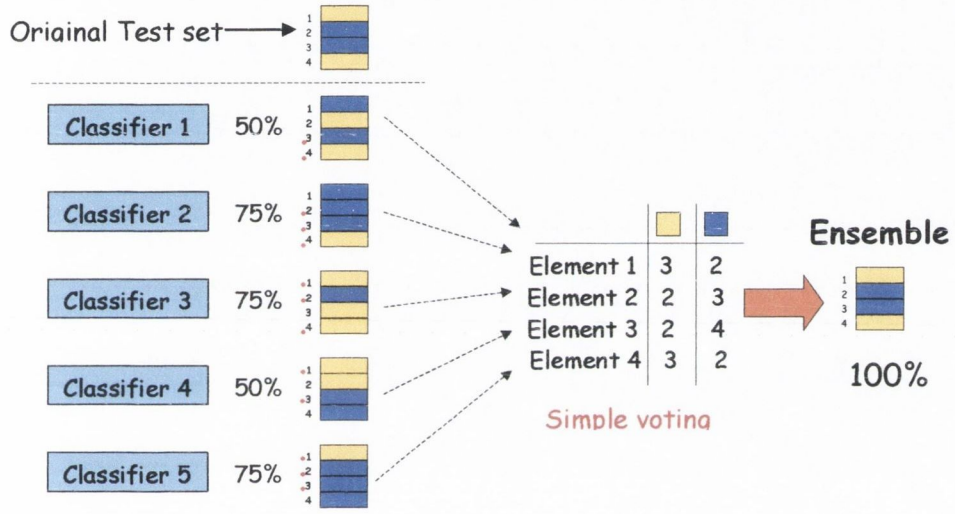


Fig. 4.IV. An example showing how diversity in the ensemble leads to better classification.

4.2.4 Measures of diversity for classification problems

There are a variety of potential ways to quantify accuracy, a measure of the ensemble diversity, in the case of a classification problem.

The simplest way to define Ambiguity is to refer to the associated error measure. For example in a regression problem (continuous output problem) it's normal to measure accuracy by the squared error so we have seen, as suggested by Krogh & Vedelsby (1995), a diversity measure can be variance, defined as:

$$a_i(x_k) = [f_i(x_k) - f_E(x_k)]^2 \qquad \textbf{(4.2.viii)}$$

where a_i is the ambiguity of the i^{th} classifier on a given example x_k, and as usual f_i and f_E are, respectively the i^{th} classifier and the ensemble predictions on x_k.; in this

scenario equation (4.2.iv) holds. The equation (4.2.iv) also holds for classification, provided that the loss function used is the squared error function and that the ensemble prediction is still given as the weighted average of the single classifier predictions. Provided also, that we are happy to deal with real-valued class membership figures, like in the example shown in the following Table (4.a).

However, for classification the most commonly used error measure is a simple 0/1 loss function, so a measure of ambiguity in this case is:

$$a_i(x_k) = \begin{cases} 0 & \text{if} \quad f_i(x_k) = f_E(x_k) \\ 1 & \text{otherwise} \end{cases} \qquad \textbf{(4.2.ix)}$$

where this time the outputs of classifier f_i and ensemble f_E to the case labelled as x_k are classes instead of real numbers.

Another proposed measure, associated with a conditional-entropy error measure, is based on the concept of entropy (Cunningham & Carney, 2000). This entropy measure of diversity can be defined as:

$$\tilde{A} = \frac{1}{M} \sum_{x=1}^{M} \sum_{k=1}^{K} -P_k^x \log(P_k^x) \qquad \textbf{(4.2.x)}$$

This quantifies the overall entropy of an ensemble on a test set of M cases where there are K possible classes.

Our objective here is to identify an ambiguity measure that will help us determine the contribution of an individual ensemble member to diversity. The entropy-based measure shown in (4.2.x) is not useful because it does not allow us to do that. Both equations (4.2.viii) and (4.2.ix) can quantify the contribution of an individual member to ensemble diversity and the variance based measure has the advantage that it directly quantifies the improvement due to the ensemble. To see how these would be applied in practice a simple example is shown in Table (4.a) and Table (4.b). In order to use squared error and variance it is necessary that the outputs of the ensemble members are real valued. This is achievable in a variety of ways with nearest neighbour classifiers where a degree of class membership can be aggregated from the similarity to nearest neighbours (see Chapter 2 for details).

Table 4.a. An example with three 3-nearest neighbours classifiers and 5 data points. The top half of the table shows ensembled predictions allowing continuous values and the bottom half shows 0/1 predictions

Value	**1**	**0**	**1**	**1**	**0**	E_i
Cl 1: pred	1	0.33	1	0.33	0.67	1
Cl 2: pred	0	0	0.33	0.67	0.33	1.67
Cl 3: pred	0.33	0.67	0.67	1	0	1
Ensemb.	**0.44**	**0.33**	**0.67**	**0.67**	**0.33**	**0.75**
Cl 1: 0/1	1	0	1	0	1	0.4
Cl 2: 0/1	0	0	0	1	0	0.4
Cl 3: 0/1	0	1	1	1	0	0.4
Ensemb.	**0**	**0**	**1**	**1**	**0**	**0.2**

Table 4.b. Error and ambiguity measures for the scenario shown in Table 4.a.

	E	$\overline{E}$	$\overline{A}$	$\overline{E}-\overline{A}$
Squared Err.	0.75	1.22	0.47	0.75
0/1 Loss	0.2	0.4	0.33	0.07

In the first scenario the outputs from the individual classifiers are real valued and we can see in Table (4.b) that the Ambiguity measure directly determines the improvement due to the ensemble as Krogh and Vedelsby predict. Using 0/1 loss and the ambiguity measure proposed above the ensemble still produces an improvement but it is not directly related to the ambiguity figure.

While the squared error and variance figures have this very elegant relationship these real valued class membership figures are not particularly meaningful; as this research focuses especially on interpretability, we will use throughout the rest of the present work the 0/1 loss error measure and the ambiguity metric proposed in equation (4.2.ix).

4.3 Use of Ambiguity to select Ensembles of *k*-Nearest Neighbours

In literature different approaches to the ensemble training issue have been presented; generally speaking they can anyway be classified into two main categories.

A first approach, quite simple and straightforward, to selecting an ensemble of classifiers consists of two separate steps: first a group of independently "good" classifiers is selected, then they are aggregated to form an ensemble. It can be considered a sort of "sequential" training. Such an approach has the advantage of simplicity, both conceptually and computationally, but the main disadvantage is that the classifiers are selected for the results they obtain singly and not for their contribution in the context of the ensemble. Most of the ensemble techniques described so far in the previous sections have followed this approach.

Following the work of Krogh and Vedlsby (1995), which demonstrated the crucial role played by the disagreement (Ambiguity) in the final prediction of an ensemble, other less straightforward approaches have been proposed to build an ensemble of good classifiers that have a high degree of disagreement. Among them the most relevant results were obtained by Liu (1999), who introduced a negative correlation penalty term to train ensembles of neural networks, and by Optiz and Shavlik (1996), who used the notion of ambiguity to find a diverse ensemble of neural networks using a genetic algorithm.

In this section we provide the definition of an algorithm based on a Hill Climbing strategy that makes use of the concept of ambiguity; here we will collect and summarize most of the results we have showed in two papers: (Zenobi & Cunningham, 2001) and (Cunningham & Zenobi, 2001). The results show that this technique outperforms the classic two-steps strategy for selecting ensembles via hill-climbing search.

The aim of the study presented here is to show how using ambiguity to select ensembles of classifiers will improve performance. As explained in section 4.2, one thing that appears to be clear is that to obtain good results an ensemble must include classifiers with a high degree of disagreement. It is this disagreement that gives the potential to correct the errors made by a single classifier. In the extreme case that all

the classifiers are good but make mistakes over the same subset of data, the ensemble will not give a better performance than any single classifier.

The choice of using algorithms based on Hill Climbing search to compare the default selection strategy, that doesn't take into account diversity, and ours, which makes use of ambiguity, was made for a couple of reasons. First, we have a way to compare two ensembles performances that is not affected by any random event, once we state the same starting point (i.e. an initial set of feature masks). If we used for example a genetic algorithm it would have been more difficult to make a direct comparison, due to its randomized nature. Second, a hill-climbing strategy is computationally less expensive then alternative stochastic search techniques. After all, we are interested in evaluating the heuristic to guide the search rather than the comprehensiveness of the search strategy.

4.3.1 *HC*: The Default Search Hill Climbing strategy.

In a classic *Hill Climbing* strategy *(HC)* that performs feature selection (Cunningham & Carney, 2000) a "good" classifier is selected by flipping each bit of the feature mask and accepting this flip if the classifier error E_i decreases. (A feature subset is a mask on the full feature set.) This process is repeated until no further improvements are possible – i.e. a local minimum in the feature set space is reached.

The error is measured using leave-one-out testing. To produce an ensemble this process is repeated for each classifier and at the end all the classifiers are aggregated to form the ensemble. This approach is illustrated in Figure (4.V).

From what we have already shown in this Chapter about the importance of diversity, clearly this approach has the disadvantage that the improvement due to the ensemble may not be so evident because there is no means of promoting diversity in the ensemble.

The dominant loss function used in classification is 0/1 loss and we have seen in 4.2.3 it is difficult if not impossible to derive a simple and linear equation that relates E to $\overline{E}$ and $\overline{A}$. However it is still clear that the *uplift* due to the ensemble depends on the diversity in the ensemble members (Cunningham & Carney, 2000). In the evaluation that follows we will use 0/1 loss and the associated ambiguity introduced with equation (4.2.ix).

```
generate a random ensemble of feature subsets;

for every classifier i in the ensemble {
    calculate initial error E_i;
    do {
        for every bit j of the mask {
            flip j^th bit of i^th mask;
            calculate new E_i';
            if E_i<= E_i'
              flip back j^th bit of i^th mask; //flip rejected
            else E_i= E_i'; //flip accepted
        }
    } while there are changes in the mask AND not maximum number of iterations;
}
aggregate classifiers to obtain ensemble prediction;
```

Fig. 4.V. *HC:* the default selection strategy for generating ensembles using error only.

4.3.2. *AmbHC*: a Hill Climbing Strategy that uses Ambiguity

Assuming a homogeneous distribution of the instances (so that the average is simply obtained by an arithmetic average, i.e. dividing by n, the number of training samples) and equal weights in the ensemble, equation (4.2.vii) that defines Ambiguity becomes:

$$\overline{A} = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n}\sum_{k=1}^{n}a_i(x_k) = \frac{1}{m}(A_1 + A_2 + ... + A_m) = \frac{1}{m}\sum_{i=1}^{m}A_i$$

where $a_i(x_k)$ is given by equation (4.2.ix) while the ambiguity A_i of the i^{th} classifier is defined as

$$A_i = \frac{1}{n}\sum_{k=1}^{n}a_i(x_k) \qquad \textbf{(4.3.i)}$$

On the basis of these definitions we may think of a new algorithm, *AmbHC*, that taking the Hill Climbing strategy as a starting point, tries to build an ensemble of classifiers with a high degree of disagreement. This approach considers every classifier in the context of the ensemble, and at each step accepts or rejects the flip depending on two parameters: the classifier error E_i and the classifier ambiguity A_i, as defined in the equation (4.3.i). If the improvement of one of the two parameters leads

to a "substantial" deterioration of the other, then the flip is rejected. With "substantial" here we mean that a threshold value *(Thresh)* is given for the highest acceptable deterioration (if we consider acceptable a deterioration of 5% then *Thresh* takes the real value 0.05). The condition to accept or reject the flip of a bit is the following: if the improvement of one of the two parameters is less than the threshold value, then the highest acceptable deterioration of the other parameter is given by the improvement of the first one; if the improvement of one of the two parameters is instead greater than the threshold value, then the highest acceptable deterioration is the threshold itself. This technique allows us to avoid the selection of a set of good classifiers that make mistakes over the same subspace of the instances; it is illustrated in Figure (4.VI).

In the next subsections we present a complete comparison of the results obtained by the two different Hill Climbing algorithms described above, *HC* and *AmbHC*, while in section 4.4 we present with some considerations about implications for CBR in terms of case representation.

```
generate a random ensemble of feature subsets;
do {
 for every classifier i in the ensemble {
     calculate initial error E_i and contribution to ambiguity A_i;
     for every bit j of the mask {
       flip j^th bit of i^th mask;
       calculate new E_i' and new A_i';
       if { {          [ E_i' < E_i ] AND
                       [     [( E_i' ≤ (1-Thresh) × E_i ) AND (A_i' ≥ (1-Thresh) × A_i )] OR
                             [( E_i' > (1-Thresh) × E_i ) AND (A_i' ≥ E_i'/E_i × A_i ) ]    ] } OR
                       {     [A_i' > A_i] AND
                       [     [( A_i' ≥ (1+Thresh) × A_i ) AND (E_i' ≤ (1+Thresh) × E_i )] OR
                             [( A_i' < (1+Thresh) × A_i ) AND (E_i' ≤ A_i'/ A_i × E_i ) ]    ] }    }
       E_i = E_i' ; A_i = A_i';  //flip accepted
       else flip back j^th bit of i^th mask;  //flip rejected
     }
 }
} while there are changes in the masks AND not maximum number of iterations;
calculate final ensemble prediction;
```

Fig. 4.VI. *AmbHC:* The algorithm for generating ensembles while emphasising diversity in ensemble members.

Before showing all the results we would like to make an observation about computability. The *AmbHC* strategy is computationally comparable to the *HC* one. The only operation additional to *HC*, is the re-calculation at each step (i.e. every time we flip a bit of a classifier mask) of the classifier ambiguity A_i. This, in turn, needs a

re-calculation of the whole ensemble prediction, but as the ensemble prediction is a linear combination of all the classifiers' predictions, at each step we only need to re-calculate a single classifier prediction (any classifier prediction is not affected by a change in another classifier), just like it happens with a simple *HC*. Practically this means we need only to include a matrix that contains all the predictions, update it for any new value (in the row corresponding to that classifier) and recalculate two averages (ensemble prediction and ambiguity A_i). This adds a computational effort that is not problematic.

4.3.3 Evaluation and Discussion.

We present in this subsection a complete comparison of the *HC* and *AmbHC* selection strategies. We show that if the ensemble members are forced to be diverse then a better ensemble accuracy can be achieved with ensemble members that have poor overall accuracy, provided we include a sufficiently high number of classifiers in the ensemble. Also, these diverse ensemble members prove to have fewer features than ensemble members selected without consideration for diversity; we will extend this consideration later on in section 4.4.

We have run the two algorithms on three datasets; two available from the UCI repository (Pima Indians, Heart Disease) and the Warfarin data-set described in (Byrne et al., 2000). These were chosen on the basis of the following criteria:

- we have restricted our experimental comparison to 2-class datasets, turning a problem into a 2-class classification task if necessary;
- we have considered datasets which do not have a skewed class distribution, as simple 0/1 error measures are questionable for datasets with very unbalanced class distributions. The Pima Indians dataset shows a moderate unbalance (the two classes distribution is roughly 2 to 1) and it is a "borderline" case that could be considered either as a non-skewed or as a skewed distribution dataset without any remarkable contradiction. Since there is a limited availability of datasets with nearly perfect class distribution, we will use it in this context as a non-skewed class dataset. However in Chapter 6 we will also show a complete evaluation on the dataset with skewed-class techniques.

In the main evaluation section of this thesis (Chapter 6), after introducing the new Meta *k*-NN ensemble aggregation strategy, we will remove these limitations, outlining all the necessary considerations each case needs.

For each dataset we have run the two algorithms described in the previous section *(HC & AmbHC)*, varying their initial ensemble size. For each ensemble size we have also repeated the process with 4 different starting points (initial sets of feature masks), averaging the results obtained, as the hill-climbing strategy is quite sensitive to the initial condition. The scoring of any ensemble is determined using a 5-fold cross validation; in the 5-fold cross validation the data is divided into 5 parts and the ensemble is tested on each part in turn having been trained on the other 4 parts. The training involves the search processes described in figures (4.V) and (4.VI) and the fitness is determined using leave-one-out testing. The results are then averaged over the 5 validation sets. The threshold used for the *AmbHC* algorithm was set for all the datasets at 2.5%, after some preliminary tests.

The evaluation on the three datasets shows that the ensembles trained with the *AmbHC* algorithm (higher diversity) have lower generalization errors than those trained with the simple *HC*, provided the size of the ensemble is sufficiently large (see Figures (4.VII) to (4.IX)). Because of the nature of the ensemble training process *HC* ensembles have corresponding *AmbHC* ensembles allowing us to use a paired *t*-test to test the hypothesis that the *AmbHC* ensembles have lower error. We have randomly selected 9 different ensembles in each of the three datasets and performed a paired *t*-test; the results gave a confidence of >80% for Warfarin, and >95% and >99% for Heart and Pima respectively. These figures are very satisfactory - the weaker figure for Warfarin is probably accounted for by the small impact of the ensemble given the already low error of the individual classifiers. This is the first main result of our study: the algorithm that takes into account diversity while selecting ensembles outperforms the simply error-only strategy.

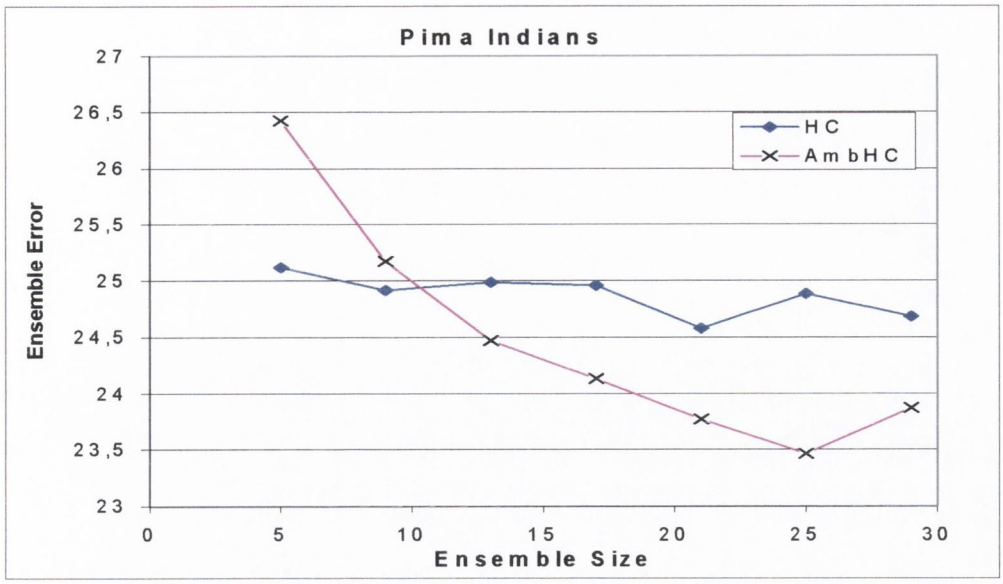


Fig. 4.VII. Generalization error of different ensemble sizes on the Pima data..

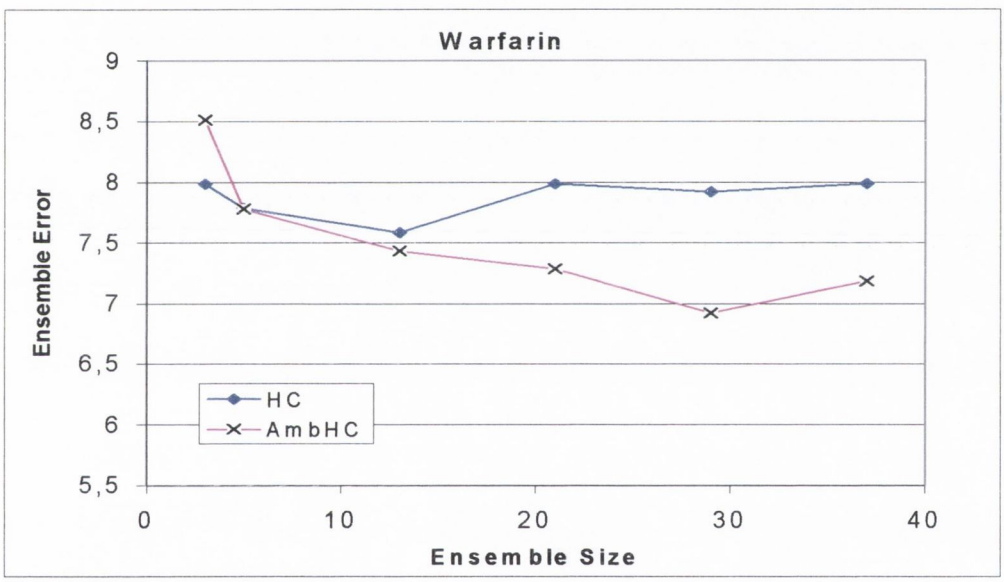


Fig.4.VIII. Generalization error of different ensemble sizes on the Warfarin data.

In the following two tables we focus on some other aspects of the comparison between classifiers trained with the two different algorithms. In Table (4.c) we show, for each dataset and for both the algorithms, respectively the error obtained by the best ensemble, the average error and the average ambiguity of the single classifiers in the ensembles. In Table (4.d) we show instead, for each dataset, respectively the total number of features and the average number of features of the masks trained with *HC* and with *AmbHC*.

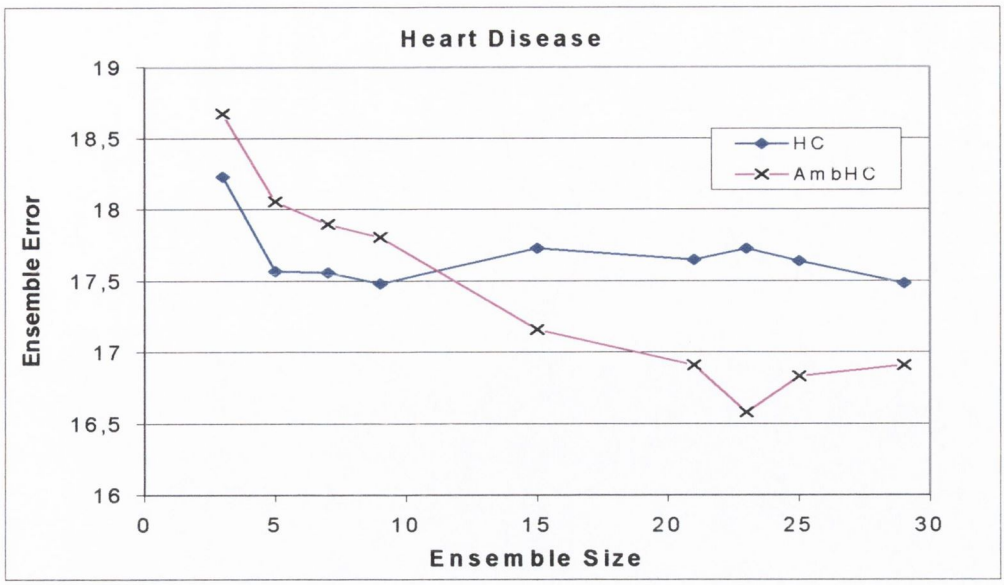


Fig. 4.IX. Generalization error of different ensemble sizes on the Heart data.

Table 4.c. A summary of all the evaluations showing the best ensemble generated for each data set and showing the corresponding average member error and ambiguity.

Data	Algorithm	Best Ensemble	Average Error	AVERAGE Ambiguity
Pima	*HC*	24.6	27.0	13.7
	AmbHC	23.4	31.5	22.3
Heart	*HC*	17.4	21.8	12.2
	AmbHC	16.6	24.8	18.8
Warfarin	*HC*	7.6	8.6	3.3
	AmbHC	6.9	14.1	10.6

Table 4.d. The ensembles built using *AmbHC* have significantly less features on average that those built using the default search algorithm.

Data	Total	Average: *HC*	Average: *AmbHC*
Pima	8	4.8	3.5
Heart	13	7.4	6.0
Warfarin	22	12.9	10.7

As we can see from Table (4.c), the classifiers in the ensemble selected with the *AmbHC* algorithm have a higher average error than those selected by the simple *HC* algorithm: the increase in ambiguity (diversity) comes at the cost of significantly higher errors in the ensemble members. It seems to us that the only way to account for the improvement in overall performance in the face of deterioration of the ensemble members is that the members are local specialists; here we will outline some considerations about this hypothesis. Further observations in its favour will be made in section 4.4 when we talk about the case representation issues emerging from ensemble research.

The use of ambiguity in the *AmbHC* algorithm means that the ensemble selection is made by choosing classifiers that disagree on a higher number of elements compared to those ones selected without ambiguity. For the first ones, the higher average ambiguity seems to compensate their higher error by "distributing" the prediction of the different individuals over more diverse regions of the space of the instances; as a result we get a lower ensemble error.

This view is reinforced by another interesting result: the ensemble members produced using ambiguity have fewer features on average than the others (Table (4.d)). It seems reasonable that fewer features are required to discriminate in these local regions. This observation also suggests a new perspective on the knowledge discovery issue connected to the feature selection problem (see section 3.3). It may be useful to reconsider this as a process of finding the best ensemble of local feature subsets rather than a global feature subset. However, the problem remains that if the ensemble is performing a problem space decomposition then it is doing so implicitly and the decomposition is not accessible. An interesting avenue for future research will be to use clustering to see if meaningful regions of the problem space can be identified where ensemble members specialize.

The local learners hypothesis helps us also in explaining the behaviour shown in the Figures (4.VII) to (4.IX), where, in each case, a minimum number of classifiers are needed for *AmbHC* to surpass *HC*. Since classifiers trained with the *AmbHC* algorithm have a higher average error than the ones trained with *HC* it is clear that each classifier will cover (i.e. predict correctly) a "smaller" region of the problem space. So, to have the majority of the ensemble voting correctly we need a larger number of classifiers. Then, when the ensemble size is small (e.g. 5) even though we

do not use diversity (in the *HC* algorithm) it is very probable that a set of classifiers randomly chosen has its own 'natural' diversity. As we increase the ensemble size, it becomes more probable that new members of *HC* ensembles will be similar to existing ones. While with the *AmbHC* algorithm diversity is still emphasized and variety is maintained. Thus, a diverse bunch of "mediocre" classifiers outperforms a good bunch of classifiers with less diversity.

However, as mentioned in section 4.2 when discussing the Condorcet Jury Theorem, this addition of new diverse members does not continue to deliver benefit indefinitely. At best, it will not continue to be possible to find more diverse members and the reduction in error will bottom out. What is happening in the three examples here is actually slightly worse than that in that an overfitting effect is evident. Since the selection of the ensemble members is a training process there is the potential for the ensemble as a whole to overfit the training data and that is clearly evident in the three graphs shown here. So with this *AmbHC* approach there is an optimal ensemble size which appears to be between 25 and 30 for these data sets.

4.4 Case Representation Issues from Ensemble Research

In parallel to the evaluation shown in the previous section, ensemble research has allowed us to obtain some other results that have interesting implication for CBR. The objective of the evaluation outlined in this section is to show that ensembles of *k*-NN classifiers can out-perform the best single *k*-NN classifier available.

4.4.1 Single Best Feature Subset Versus Ensembles

We start with two data-sets, an InVitro Fertilisation (IVF) data-set presented in (Cunningham & Carney, 2000) and the Hepatitis data from the UCI repository. The IVF data has 53 features and Hepatitis has 19 features so it is not practicable to search through the full space of feature subsets to find the best mask. In each case 100 good masks are found using hill-climbing search as described above; see also (Cunningham & Carney, 2000). Six ensembles of size 20 are produced using these masks and the error figures are shown in Table (4.e). The ensembles outperform the best masks found.

Table 4.e. An error comparison of ensembles of *k*-NN classifiers compared with best single *k*-NN classifiers found using hill-climbing search.

Dataset	Average Mask	Best Mask Found	Average Ensemble	Best Ensemble
IVF	41.1%	38.3%	35.7%	35.5%
Hepatitis	20.7%	17.4%	16.8%	15.5%

In the next evaluation two data-sets are chosen with few enough features to perform an exhaustive search of the space of feature masks. These are the Pima and Abalone data-sets from the UCI repository and each has 8 input features. The Abalone data is converted to a classification format by partitioning the outcome variable into two classes (age 7&8 → Class 1; age 11,12&13 → Class 2). In this case ensembles are produced using the algorithm presented in Fig. (4.IV) that emphasizes diversity. These ensembles are compared using 5-fold cross validation with the best masks that have been found by exhaustive search. Using a one tailed paired *t*-test the ensemble is better than the best mask with 80% confidence for the Pima data and 90% confidence for the Abalone. In the 5-fold cross validation the data is divided into 5 parts and the ensemble is tested on each part in turn having been trained on the other 4 parts. The scoring of the masks in determining the best mask is done in the same fashion.

Table 4.f. An error comparison of ensembles of *k*-NN classifiers compared with best single *k*-NN classifiers.

DATASET	ALL FEATURES	BEST MASK	BEST ENSEMBLE	AVERAGE ENSEMBLE MASK
Pima	23.9%	23.8%	22.5%	30.7%
Abalone	18.1%	16.5%	15.9%	22.5%

4.4.2 Implications for CBR

From the results shown above we can make some observations. When we train an ensemble of classifiers we are able to obtain a better performance than any possible single classifier. This brings into question the practice of feature subset selection and illustrates some shortcomings of the feature vector representation of cases. Searching

for *the* predictive feature subsets is questionable because a combination of feature subsets, maybe poorly predictive if taken singularly, can give a better prediction when aggregated in an ensemble. In other words, in a case-based reasoning system a single classifier (in this case, a single feature subset) might have an upper limit to its performance that can be raised by the use of an ensemble.

This result, together with all the consideration we have made in Chapter 2 about the instability of the feature selection process, suggests the hypothesis that the search space is "adapted" by a single classifier (dropping some of the features) in order to fit better to its intrinsic metric (usually the Euclidean one). A single classifier uses a "global adaptation" of the space, while in the case of an ensemble we have a number of different "adaptations" that can be seen as making errors in different regions of the search space.

So in terms of the interpretability of the knowledge representation in the ensemble, at the extreme, there are two possibilities:

1. The ensemble embodies a distributed knowledge representation, similar to a neural network, and is not readily interpretable.
2. The ensemble is a set of local specialists and these specialists are locally interpretable.

Clearly, from the results obtained and the subsequent considerations outlined throughout this Chapter, we are inclined to this second view. As we have already mentioned, in our *AmbHC* algorithm for generating diverse ensembles, by forcing the ensemble to be diverse the accuracy of the ensemble is increased but the average overall accuracy of the ensemble members is decreased. Coupled with this is the curious effect that the average number of features in the ensemble members is reduced compared to ensemble members that are optimized for accuracy only. We would suggest that this poor overall accuracy and small "specialized" feature set indicates that the ensemble members are local specialists. As previously suggested, an interesting line of research for future work would be to use an ensemble of lazy learners to discover a problem space decomposition that would allow for a hierarchical CBR solution that would have a performance comparable to that of the ensemble and better than a single CBR system.

4.5 Summary of the Chapter

In this Chapter we have introduced Ensemble techniques. In section 4.1 we have discussed the basic ideas and related definitions for an Ensemble of classifiers/predictors; we have also outlined some of the most successful applications proposed in literature. In section 4.2 we have inspected in more detail the theoretical foundations of Ensembles; in particular we have shown two of the most important error decompositions, the Bias/Variance and the Ambiguity ones, especially in relation to the Overfitting problem. Then we have outlined some strategies to apply these concepts to ensembles of nearest neighbours. In section 4.3 we have proposed *AmbHC*, a Hill-Climbing based algorithm designed to train ensembles of nearest neighbours with a high degree of diversity; results show that this strategy outperforms classic Hill-Climbing techniques based on error only. Finally in section 4.4.we have drawn some considerations about case representation issues emerging from the better results obtained with Ensembles if compared to the best single learner available.

The advantage of ensembling derives from aggregating diverse models. This advantage seems incompatible with CBR because if case-based learners are aggregated then the interpretability of CBR is lost. The case retrieval capabilities that make CBR such a powerful Machine Learning paradigm are seriously worsened by the use of an ensemble; on the other hand, the use of an ensemble can produce better results in terms of accuracy and stability if compared with a single classifier based on a feature subset. In the next Chapter 5 we will face the problem of CBR interpretability, with particular reference to the explanation characteristic through case retrieval of CBR. We will propose a new aggregation function for ensemble, named *Meta k*-NN, that will allow us to keep the explanation characteristic of CBR when using an ensemble, obtaining comparable results to classic aggregation functions in terms of accuracy.

Chapter 5

The *Meta k*-NN aggregation strategy

5.1. Limits of the use of Ensembles for Lazy Learning

As we have shown in the previous chapter, the use of ensemble techniques can provide great benefits to the classification/regression task, particularly in terms of accuracy and stability. However, to some extents the strength of the ensemble idea poses on the other side some serious issues for lazy learning in general and to Case-Based Reasoning in particular. For example, with the ensemble approach several learners have to be consulted at run-time and lazy learners do all their work at run-time. With lazy learners there has been little or no offline processing of the training data to build a hypothesis or model – this work is deferred to run-time. Furthermore, in terms of model complexity, the ensemble adds another dimension (number of classifiers) to the problem, and this is certainly not an advantage.

If the ensemble approach poses problems for lazy learning in general then it proposes fundamental problems for Case-Based Reasoning in particular. As we have seen in Chapter 2, the key simple idea in CBR that makes it such a powerful tool is that a single case or a small number of cases are retrieved during problem solving and these cases are reused to produce a solution for the new problem. Thus, as a problem solving technique, CBR has the special advantage of interpretability. As previously explained, this CBR feature makes it particularly desirable in any field where explanation plays a vital role; we have already mentioned how areas like medical and financial domains, diagnosis and recommendation systems, among the others, can greatly benefit from the explanation and interpretation support provided by CBR tools.

Also, recent research has shown (Cunningham et al., 2003) that the use of CBR for explanation can in some cases be even more meaningful than other Machine Learning tools that are traditionally used for knowledge discovery, like rule-based systems.

The motivation in ensemble research is quite different; instead of there being a single line of reasoning in the problem solving process there are several parallel lines of enquiry and the results of these parallel efforts are aggregated to form a single solution. Thus it is difficult to *explain* the output of an ensemble. For this reason it appears difficult to bring the advantages of the ensemble idea to CBR without losing the interpretability advantage of CBR. In other words there seems to be an apparent trade between accuracy and stability against interpretability when using an ensemble of Case-Based Reasoners.

Furthermore, the improvement obtained by the ensemble depends on the members of the ensemble being diverse; a characteristic that arises naturally with decision trees or neural networks trained using different data sets. With CBR tools like nearest neighbour, that are instead more stable in the face of changing data, the most common way to differentiate ensemble members is to base them on different feature subsets (Ho, 1998a; Guerra-Salcedo & Whitley, 1999a, 1999b); again, in this case diversity plays a vital role (Ricci & Aha, 1998;Cunningham & Carney, 2000; Zenobi & Cunningham, 2001).

As shown in Chapter 3, feature selection may present an instability problem. We have seen that for CBR the use of an ensemble will help solving this problem (as much as in the case of neural networks and decision trees it helps solving the instability on data samples); the important point here is that this improved performance is not based on a single problem representation but on a variety of different representations. So a further threat to the CBR idea arises from ensembles where the diversity in the ensemble is achieved by having different problem representations in the ensemble components. This is somehow a confirmation of what we found in section 3.3: feature selection on its own may not be very helpful for the knowledge discovery issue, since an ensemble of different feature subsets can outperform any single subset.

The ensemble might be considered to embody a distributed problem representation. Alternatively it might better be viewed as a committee of local problem solvers using a variety of local representations. This contrasts with the fixed

representations that are common in case-based reasoning. In Richter's knowledge container terms (Richter, 1998), the vocabulary knowledge container is difficult to characterise in these ensembles based on different feature subsets.

5.2 The search for the "best"cases.

In Chapter 2 we have seen that the output of a CBR system to any query is given by two separate pieces of information: the main part is a query prediction (a class in the case of a classification problem) and secondary we have a set of retrieved "best" cases that can be used to "explain" the query. This last one is an important piece of information because the user has the chance to see what are the cases that the system suggests to be similar to the query; we simply recall here we could imagine the system giving explanation in the form of suggestions like: *"the query belongs to class X because the closest case to it also belongs to class X and so does the third closest case"*. In section 2.3 we have described a number of successful CBR applications in support of explanation proposed in literature, in the context of a great variety of problem domains.

However, all the approaches mentioned above were dealing with a single Case Based Reasoner. When moving to the higher complexity of the ensembles, the neighbours retrieval paradigm together with the search for the "best" cases for explanation become a lot less straightforward. Nevertheless this piece of information remains a crucial distinctive feature of Case-Based Reasoning that, if lost, would considerably reduce the high potential of CBR applicability.

The main goal of this thesis will thus be the search of a paradigm that allows us a consistent case ranking and the subsequent capability to retrieve a (possibly small) set of cases to explain the query when using ensembles.

The key point in presenting the following example is to show that a large number of cases contribute to the prediction of the ensemble. For example, in an ensemble of 25 classifiers each retrieving 5 neighbours, 125 neighbours will be retrieved. Clearly there will be duplicates in this set but with the prediction from each classifier being made on the basis of a different feature subset, it is very likely that the nearest neighbours for one particular classifier will not be the same as for another. Clearly, the

"naïve" attempt to solve the problem of case-retrieval by considering the entire set of retrieved cases of the ensemble is not an acceptable solution.

A refinement of such an attempt may be made by putting all the retrieved cases in a "pool" and defining a rule to select the most representative ones. There are two main difficulties with this approach.

The first one is theoretical. All the retrieved cases come from a set of nearest neighbours that operates on a variety of representations, in particular on representations based on different feature subsets. In turn the sets of retrieved cases have an intrinsic different semantic for each classifier and it appears very difficult to define a consistent rule to compare cases retrieved by classifiers based on different feature subsets. For example any ranking process of each classifier would be made on the basis of its particular feature subset and it doesn't look very reasonable to rank all the cases based on the rank given by each different classifier, as they may have scores with remarkably different scales.

The second difficulty is practical. Even if we didn't consider the theoretical issue explained above, it is very difficult to define a rule to extract a small set of cases in a consistent way with the final ensemble prediction. If we consider, for example, a single 3-NN classifier with a binary task (say black/white) that for a given query predicts "black", it's quite straightforward that at least two of the retrieved cases will be "black". (it may at most happen that only the first case is "black" but it's so much "closer" to the query – i.e. it has a much higher raking - that the *k*-NN still assigns a higher probability to the "black"). In any case, the set of retrieved cases is consistent with the final prediction for a single *k*-NN, which is something that appears obvious if we think that the final prediction is made on the basis of the retrieved cases themselves. In the case of an ensemble this relationship cannot be so clearly expressed anymore. The following subsection outlines a simple example showing how difficult it gets to choose a reasonable and consistent rule to select the best cases in a "pool" of cases retrieved by an ensemble of classifiers.

5.2.1 Attempts to select representative cases through Ensembles

Consider an ensemble of five 3-Nearest Neighbours classifiers for a simple binary task (black/white); we have used for simplicity a constant Kernel function (see equation (2.2.iv)) to compute class probabilities. In the following figure is reported an

example of how the predictions and retrieved cases may look like (to each classifier we have associated a purely indicative different feature subset).

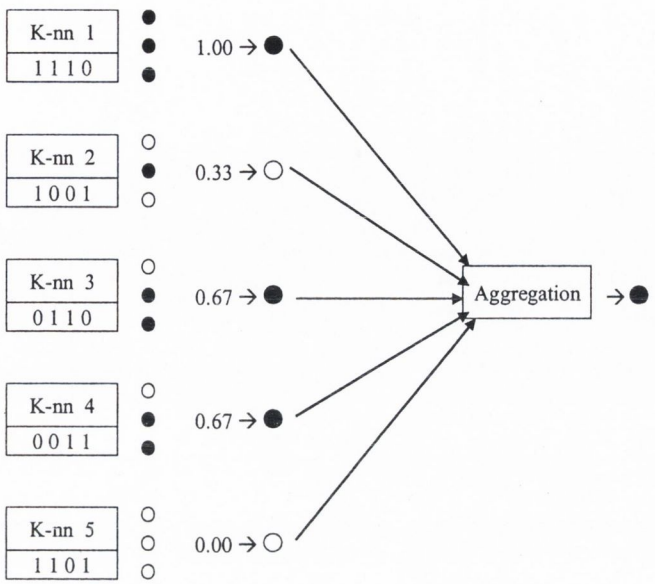


Fig. 5.I. An example of Pool of retrieved cases by an Ensemble

Giving as a starting point figure (5.I), let's try and define a rule to extract the most representative cases from the pool of all the cases (in this case the pool is given by 15 cases). As we can see, the aggregation process returns, for a given query, the reply "black", both in the case we use a simple voting aggregation strategy (3 blacks against 2 whites) and in the case we use a weighted average (it would be a prediction of 0.56 toward black using equal weights). Below we report some attempts to define a fair retrieval rule from this example.

First attempt: we put in the pool all the cases labelled with their own ranking score (each one coming from its own *k*-NN classifier) and as retrieval rule we get the first *k* cases, where *k* is the same as each classifier (in this case *k*=3). Apart from the theoretical inconsistency explained above, the problem that arise is that the "white" cases that are at the top of classifiers 2,3,4,5 may have a higher ranking than the black one at the top of classifier 1 and we may end up with 3 white retrieved cases and a prediction returning "black", that is clearly inconsistent.

Second attempt: we do not define a pool and simply ask each classifier to retrieve its "best" case; all these cases are given back as an "explanation" to the query. In this case we return a number of cases equal to the number of classifiers. The problem in

this case is, classifiers 2,3,4,5 would retrieve a "white" case against the only black retrieved by classifier 1 and we would end up with 4 white cases retrieved over 5 and a prediction of "black", clearly inconsistent.

***Third attempt*:** we ask only the classifier whose outputs match with the final prediction to retrieve their best case. In this case they would be classifiers n. 1, 3, 4 but, as we can clearly see, it might happen that their best case would be anyhow different from the ensemble output. Here it happens that classifiers 3 and 4 retrieve a "white" so that we retrieve 2 whites over 3 with inconsistency to the final ensemble output.

Fourth attempt: each classifier retrieves only the best case with the same label as its own final prediction (so that for example classifier n.3 would retrieve the second case, because it is the best "black" case). With this solution we will have for sure 3 blacks and 2 whites retrieved. Apart from the fact this starts to appear a rather "artificial" strategy, there still is a problem in the case of duplicates. It may happen that, for example, some of the blacks retrieved by the different classifiers are in reality the same case (say the best "black" case for classifier 1,3 and 4 is always the same so that as a matter of fact we retrieve only 1 case instead of 3) and still it may occur that the retrieved cases are inconsistent with the final prediction.

***Fifth attempt*:** we retrieve the cases that have the highest frequency of occurrence among all the classifiers. Even without considering the theoretical inconsistency arising from the fact each classifier has a different semantics, nothing can guarantee us that a "white" case may be the one that appears with the highest frequency even though the prediction is "black" (for example a noisy case); this solution looks quite problematic as well. Furthermore, one of the characteristics of ensembles is, it may happen that a rarely retrieved case gives in some circumstances a contribution to the ensemble that creates the uplift in respect to a simple classifier (see 4.2.3); That case will never be retrieved by this strategy.

We have limited the description of these attempts to the five most "reasonable" ones; we have shown that the idea of using the pool of all the retrieved cases with some retrieval rules to select a small fraction of them will break down in fairly simple situations. There obviously could be more sophisticated strategies but, apart from the fact the more complex a strategy is the furthest from a clear explanation goal it looks,

in any case the theoretical inconsistency due to the different semantics of all the classifiers still remains.

Thus, some other ideas will have to be explored if we want to address the problem of case ranking and retrieval when using an ensemble. In the following section we present our *Meta k*-NN solution to this problem.

5.3 The *Meta k*-NN aggregation strategy.

To move some steps toward a solution for the interpretability problem introduced by the use of ensembles we need to make a crucial observation. As explained in Chapter 4, there are mainly two different steps in the definition of an Ensemble: the first one is the definition of a set of classifiers with a certain degree of diversity (for lazy learners it's common the case to differentiate classifiers on the feature subsets), while the second one is the definition of an aggregation function, usually chosen as weighted average strategy (4.1.iii) or as simple voting strategy (only for classification problems).

Of these two steps the one that is somehow "losing" the nature of CBR is certainly the aggregation process. In fact, no matter what the retrieved set of cases in a CBR ensemble is, the weighted average (or a simple voting) aggregation strategy will "forget" the way each classifier makes its own prediction and use all the classifiers' predictions only as pure numbers. If we think that the weighted average strategy is used in ensembles of nearest neighbours, neural networks and decision trees, that are certainly classifiers with completely different approaches and natures, we have a better understanding of what we mean with the previous sentence. With no doubt the use of a weighted average aggregation strategy will make us lose some of the features that makes CBR so powerful, the possibility of retrieving cases.

Taking this observation as a starting point it is then possible to elaborate a new way to aggregate classifiers that preserves the nature of CBR, in particular of case retrieval. Whatever this new aggregation technique may be it will certainly have to take the set of single CBR problems (single classifiers) as first step and then "generate" a new CBR problem that will allow us to aggregate all the single classifiers' predictions, maintaining the CBR nature. So it will have to be a new CBR

model (aggregation step) based on a number of different CBR models (single classifiers), in other words a *Meta CBR model*. In the rest of this chapter we define such an aggregation strategy, called *Meta k*-NN, for the case of ensembles of nearest neighbours classifiers; in 5.3.1 we show a simple example introducing a different data representation that leads to this new aggregation strategy, in 5.3.2 we give a detailed description of the algorithm, while in section 5.4 we outline some considerations on the nature of *Meta k*-NN.

5.3.1 An example to introduce a different data representation.

Consider a database consisting of a set of n cases, each one described by a set of f features (an example is shown in Table (5.a)). For simplicity suppose all the features are normalized numerical and that the outcome is a simple binary classification mapped to the classes 0 and 1. We will call this dataset the *Original Database*.

Table 5.a. A sample data set of n cases each described by f features.

	Feature 1	**Feature 2**	**...**	**Feature *f***	**CLASS**
Case 1	0.23	0.16	...	0.98	0
Case 2	0.14	0.56	...	0.32	1
Case 3	0.45	0.16	...	0.42	0
...	...	...	...	...	...
Case *n*	0.56	0.18	...	0.0	1

Suppose we train an ensemble of m k-NN classifiers differing on the feature subset chosen as described in 4.1.2(b). For each single case in the training data, every classifier of the ensemble can use Leave One Out and associate a prediction about its class (in the form of a probability between 0 and 1).

It is then possible to build with the ensemble a new $n \times m$ matrix associated with the original dataset described in Table (5.a), where in the position (i, j) is stored the prediction given by the classifier j for the case i. In other words each case is described by a new set of features representing how the ensemble (through the predictions of each one of its classifiers) *sees* the case. An example of such a matrix is shown in Table (5.b). The arrows indicate what would have been the final prediction if the classifier were used on its own. This new matrix can be seen as a transformation of the

data that in some sense reflects how the ensemble has modeled the problem domain. For reasons that will be explained in next section we will indicate this problem representation as the *Meta Database*. If classifier *i* gives similar class probabilities to two different cases (i.e. it considers the two cases as similar with respect to the class outcome) this will be translated in the Meta Database representation by the fact that the two cases will have feature *i* with a similar value.

To obtain a "richer" Meta Database representation it is a good practice to use *k*-NN classifiers with a non-constant Kernel function, for example a Kernel function defined as the inverse of the distance (equation 2.2.v). This will give a higher refinement to the class probabilities assigned to each case and subsequently a higher variety to the features of the Meta Database representation; otherwise for example any query that has two cases labeled as "1" and one case labeled as "0" as nearest neighbours, will have a class probability of 0.67.

Table 5.b. A transformation of the data shown in Table 5.a based on the outputs of the *m* classifiers in the ensemble.

	Classifier 1	**Classifier 2**	**...**	**Classifier *m***	**CLASS**
Case 1	0.95 →1	0.08 →0	...	0.21 →0	0
Case 2	0.67 →1	1.0 →1	...	0.0 →0	1
Case 3	0.21 →0	0.19 →0	...	0.69 →1	0
...	...	...	...	...	...
Case *n*	0.61 →1	0.32 →0	...	0.15 →0	1

5.3.2 The *Meta k*-NN algorithm

The important thing about the way to represent data introduced in 5.3.1 is that it suggests what the new two-stage process of classification may be. When a target query is given to the ensemble, in the first stage it is presented to the ensemble members as before; each classifier returns an output for the target based on the original dataset. In the second stage the outputs of the ensemble members is used as a representation of the query case, in analogy to what happened to each single training case when building the Meta Database. At this stage it becomes quite straightforward how to replace the weighted average aggregation process used so far with a new case-based reasoning process. As the Meta Database and the query case are now

represented through the same semantics, it is possible to classify the query with another nearest neighbour tool using as Case-Base the Meta Database. We will call this new nearest neighbour classification process the *Meta k-NN aggregation strategy*; the name simply reminds us the fact we have a nearest neighbour tool that works on a number of nearest neighbour tools.

The thing that really makes this new technique so appealing is that in the aggregation process the "nature" of CBR is preserved (being the aggregation step a CBR problem itself) and it will be possible to retrieve a small number of cases to explain the query without the inconsistencies we illustrated in section 5.2.1. For example, using as *Meta* k-NN a 3 nearest neighbours classifier we will have for each new query a set of 3 retrieved cases to explain it. The benefits of having a small set of cases to represent the query do not need to be explained any further.

To avoid any confusion emerging from these two "layers" of nearest neighbour tools, we will keep using k to indicate the number of neighbours retrieved by each single classifier in the ensemble, while we will use k_M to indicate the number of neighbours retrieved by the nearest neighbour tool performing the aggregation on the Meta Database. The notation "*Meta* k-NN" refers mainly to the fact the aggregation step is performed on a number of k-NNs but it should be kept in mind that the final number of retrieved neighbours by a *Meta* k-NN is k_M. Note that k and k_M are completely unrelated and in general may be set to different values.

The whole training/classification process is illustrated in figure (5.II); for simplicity we have kept the same numerical values shown in Table (5.a) and (5.b). On the left side of the figure we have the Original training database and the query case with the original set of features, while on the right side we have the transformation into the Meta Database with the associated *Meta* k-NN problem. As we can see, the original training database feeds each k-NN of the ensemble to produce the Meta Database (Phase 1: Training). After this, the query case is given to each ensemble member (to avoid a too confusing picture we have omitted the arrows that would represent this) and is translated into the new problem representation; it is then ready to be classified with the *Meta* k-NN technique (Phase 2: Classification).

The following figure (5.III) illustrates instead the *Meta* k-NN algorithm in a more formalized way.

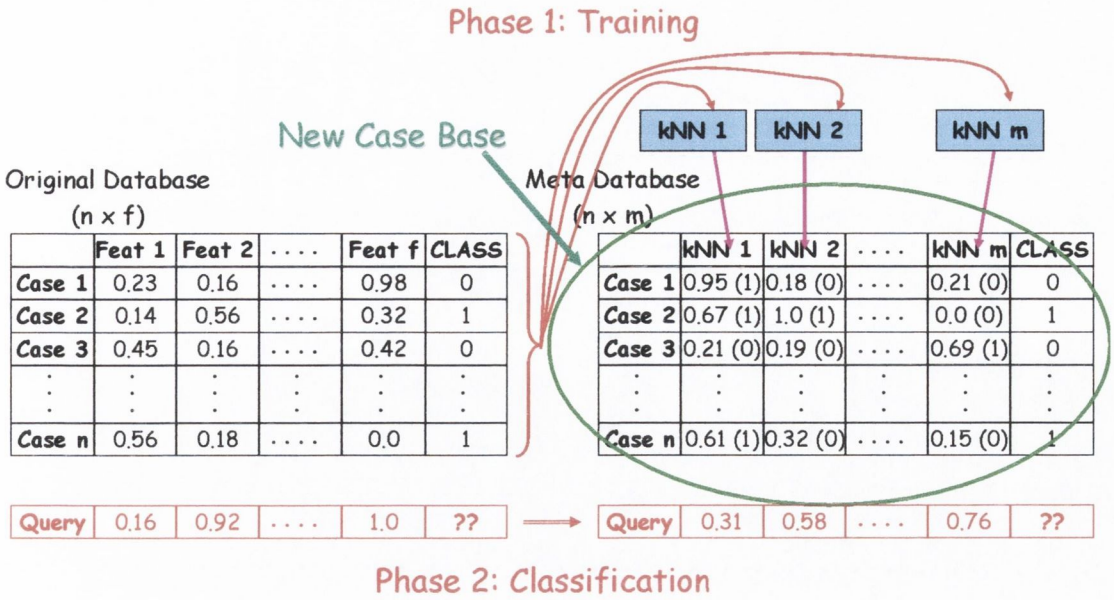


Fig. 5.II. The two Phases of Classification through the Meta *k*-NN aggregation strategy

5.4 Considerations on the nature of *Meta k*-NN.

Before we proceed with the evaluation of the *Meta k*-NN aggregation technique we need to consider a few things. A first straightforward observation is the fact that, similarly to what happened with ensembles aggregated with weighted average, also in the case of aggregation through *Meta k*-NN there are two separate phases for the Training and the Classification (see Figure (5.III)). In other words, the *Meta k*-NN aggregation strategy introduces another layer of Eager Learning "injections" into Lazy Learning systems (see Chapter 2), after the one brought by the Feature Selection process.

A second observation can be outlined about the theoretical assumptions behind the *Meta k*-NN aggregation strategy. As a matter of fact it is not possible to describe with a consistent formal language a statistical analysis of an ensemble of *k*-nearest neighbours based on different feature subsets, as we have shown in 4.2.3. Nor it is possible, when using the *Meta k*-NN aggregation strategy, to relate with a simple mathematical model the query prediction of the ensemble to the set of the single members predictions, as it happened instead in the case of the weighted average. This is due to the fact the aggregation step is performed through a nearest neighbour process and we know (see Chapter 2) that nearest neighbour classifiers do not build a

unique model of the problem domain but rather construct a local answer to the query in a Euclidean distance-based space.

Even though there is a lack of a theoretical background for developing a more rigorous analysis of the *Meta k*-NN strategy, it is still possible to outline a few considerations about the problem representation embedded by the *Meta k*-NN and the implicit assumption it relies on. The next sub-section (5.4.1) will be dedicated to this issue, while in 5.4.2 we will address possible extensions of *Meta k*-NN to regression and non-binary classification problems and in 5.4.3 we will consider the computability issue.

input: a set of m k-NN classifiers based on different feature subsets (trained with diversity);
a *query case* q;
a case-base O, the Original Database, a matrix $n \times (f+1)$; column $(f+1)$ contains the class labels.
output: a prediction about the class of q;
a set of k_M cases from case-base O that better approximate the query q (nearest neighbours).

```
// Phase 1: Training - Construction of the Meta Database
initialise a matrix M of dimension n x (m+1) to null values;
for every case i in the case-base O {
      for every classifier j of the ensemble {
        O(i,j) = compute leave-one-out prediction of classifier j on case i;
      }
      M(i,m+1) = O(i, f+1);    //the class is simply copied
}

// Phase 2: Classification on the new query
initialise a vector q' of dimension m to null values;
for every classifier j of the ensemble {
      q'(j) = compute prediction of classifier j on query q;
}
assign a new k_M - nearest neighbours classifier (Meta k-NN) to Meta case-base M;
p_M = compute prediction of Meta k-NN classifier on q' using Meta case-base M;
compute set of indexes for closest cases to q in case base M {n_1, n_2, ..., n_Km}

return prediction p_M;
return set of closest cases with indexes {n_1, n_2, ..., n_Km} from case-base O
//the cases are returned in the Original database representation;
```

Fig. 5.III. The Meta k-NN algorithm

5.4.1 Considerations on the problem representation of *Meta k*-NN

As explained in Chapter 4, one of the main ideas that leads to the definition of ensemble of classifiers based on different feature subsets is the fact that in many cases a single feature subset doesn't seem to be suitable to represent the whole problem

domain. A set of feature subsets working as "local experts" looks much closer to what the underlying unknown distribution may be. This is especially true if we think that when we classify a query with a single *k*-NN based on a certain feature subset, the similarity between the query and the cases stored in the Case-Base is computed only on the selected features; in this case if the feature subset is not representative enough of the query it will create some noise. Based on this observation we can outline the following conjecture.

When we transpose the problem from the original domain to the *Meta k*-NN domain the new set of features become the way each classifier "sees" the problem domain, through its predictions. Suppose a query case q is given to the *Meta k*-NN. Suppose it is best described by a certain feature subset F_q of the entire feature set F, which may be different from the one that best represent a different query q' (local feature selection). In general the feature subset F_q may not be equal to any of the feature subsets F_j of classifier j, for each j. The feature subset $(F_j \cap F_q)$ is the subset of predictive features for classifier j while $F_j \setminus (F_j \cap F_q)$ contains all the noisy features (the subset $F_q \setminus (F_j \cap F_q)$ of non-selected but predictive features is also an indirect source of noise for the classifier's performance).

If the majority of the voters are right (which is a fundamental hypothesis for the ensemble to work) that means the majority of feature subsets F_j can correctly classify the query, in other words they can represent sufficiently well the feature subset F_q facing the noise introduced by the absence of predictive features and the one introduced by the presence of non-predictive features.

In the Meta Database perspective the nearest neighbours to the query are those cases whose features show the highest similarity to the query features. As the single members class probabilities are the features for the Meta Database it looks quite intuitive that the smaller the distance of a case to the query in the *Meta k*-NN domain, the higher the similarity (in terms of class probability outcome) of the case to the query *over the whole pool* of ensemble members.

The implicit assumption behind the *Meta k*-NN strategy is that two cases that have a high similarity (i.e. very similar class probabilities, at least for the majority of the classifiers) belong to the same class while two having substantially different class

probabilities for the majority of the classifiers belong to different classes. Here below we show in more detail the reasons that support this assumption.

Let's start with an observation on the single classifier "scope". We expect that when a single classifier (single feature subset) can correctly classify the query, in the majority of the cases it happens that the class probability assigned to the query will be closer to the class probability assigned to the cases that correctly classify it (i.e. that have the same class label) than to the class probabilities of the cases that misclassifies it, with the exception, of course, of some noisy examples. Under this perspective a "noisy" example would be a case with an associated probability class similar to the query but with a different associated class label; in the Meta Database representation that would translate as a similar feature value (on the feature corresponding to that single classifier) between the query and the noisy case.

However, as we have seen in Chapter 4, the aggregation of several classifiers in an ensemble would greatly reduce the effect of noisy cases in the final prediction; in particular the ensemble will reduce the effect of a case misclassified by a single classifier if it is correctly classified by a good number of the other classifiers, at least by the majority. In the *Meta k*-NN perspective that would mean the majority of the other classifiers assigns to the misleading case a class probability that is further to the query class probability than the other non-misclassifying cases; in the Meta Database representation that would translate as a substantially different feature value between the query and the noisy case, at least for the majority of the other classifiers, so that the overall distance between the query and the noisy case is sufficiently large to avoid its retrieval.

As the fundamental assumption for *any* ensemble to work is that the majority of the classifiers correctly classify the query (algorithms like *AmbHC* are designed both to fulfil this requirement and to give a "rich" problem representation through diversity) it seems reasonable to adopt an analogue assumption also for the different problem representation embedded in the *Meta k*-NN aggregation process. We will show in next Chapter 6 how the results obtained with the *Meta k*-NN aggregation strategy seem to support the validity of this assumption.

5.4.2 Considerations on possible extensions of *Meta k*-NN

Some points can be made about the extension of *Meta k*-NN to regression and non-binary classification problems.

In the case of a regression problem the *Meta k*-NN aggregation strategy works exactly like in the case of a binary classification problem, with the only difference that the final prediction of the *k*-NN that performs the aggregation step is not mapped into a class but rather kept as a continuous value.

As for the application of *Meta k*-NN to the multiple class problem, there is a variety of ways to deal with it. A first simple way would be to consider all the Meta Database feature values as symbolic, so that in position *(i,j)* instead of writing the class probability (continuous value between 0 and 1) we write the symbol of the predicted class for case *i* by the classifier *j*. This way the probability information is missing as we keep only the class label, but the algorithm gains in simplicity that remains pretty much the same of the binary classification one (after all there are many cases of "real life" database consisting of symbolic values only).

However, as suggested in 5.3.1, a more refined Meta Database representation is highly desirable. If we want to keep the probability information to have a "richer" Meta Database, then we need to expand each feature in a vector of *C* features, where *C* is the number of classes, each one holding the probability of the corresponding class. This way the Meta Database is again with all continuous feature values, representing the class probabilities, and the classification can be done on a higher amount of information, because we can compare the probabilities not only for the chosen class but also for the rejected ones. Obviously this technique can be used only when the number of classes is not too large; for example if we have 4 classes and an ensemble of 25 *k*-NNs we will get a matrix of 100 columns, still acceptable, but if we had a 20 classes problem the number of features would go up to 500, which may turn to be too computationally expensive.

5.4.3 Considerations on Computability Issues

Some observation about the computability cost can also be outlined. The aggregation process made through *Meta k*-NN is computationally heavier than an aggregation process made with a simple weighted average, seen that the computational time requested by a new nearest neighbour layer is certainly higher

than the one necessary to perform simple mathematical operations involved in the computation of a weighted average.

However part of the *Meta* k-NN aggregation process, the building of the Meta Database, will only be performed once after the training of a diverse set of classifiers (see Figure (5.III)) and repeated only in case of a re-training; the remaining part of the aggregation process, performed any time a new query is submitted to the system, would be a simple nearest neighbour retrieval. The computational cost of this operation is surely affordable (in practice, for most of the datasets it would be around a few seconds), especially when compared with the advantages such an aggregation process has.

The relatively higher computational cost required by the Meta k-NN aggregation step, compared to a traditional weighted average step, is also the reason why the ensemble members are still trained with diversity through the *AmbHC* algorithm. On a theoretical level *AmbHC* uses weighted averages (to compute every classifiers' diversity) so it will train ensemble members with an intrinsic different bias from the aggregation step, if performed through *Meta* k-NN. On a more practical level, a Hill Climbing technique that tests the flipping condition (on the feature masks bits) directly in the *Meta* k-NN performance score would have a too large computational increase (the testing conditions are repeatedly performed in any cycle of the algorithm) in the already large time requested to run *AmbHC* with a consistently big dataset. Anyway the important requirement here is that the ensemble members give a sufficiently rich problem representation, i.e. they are trained with sufficient diversity, regardless of how the diversity is computed.

5.5 Summary of the Chapter.

In this chapter we have introduced a new aggregation technique, *Meta* k-NN, that is designed to allow an ensemble of classifiers to rank consistently and retrieve a small set of neighbours in support to query explanation, keeping the performance and stability benefits that are typical of ensembles.

In section 5.1 we have outlined the limits of the use of classic ensemble techniques in reference to explanation. In section 5.2 we have shown an example that illustrates

the inconsistency of any attempt to extract a representative subset of cases from the whole pool of retrieved cases by an ensemble. In section 5.3 we have presented the algorithm of the *Meta k*-NN aggregation strategy. Finally in section 5.4 we have outlined some considerations about theoretical consistency of the *Meta k*-NN technique and some possible extensions to regression and non-binary classification problems together with some observations about the associated computability cost.

In the next Chapter we will show a comprehensive evaluation of the *Meta k*-NN aggregation strategy; analysis of the results will show that ensembles aggregated with the *Meta k*-NN technique have excellent accuracy – at least comparable to that of ensembles aggregated with weighted average – with the added advantage that a small number of cases are identified for use in explanation. Also, the quality of the retrieved cases will prove to be good by some domain experts evaluations.

Chapter 6

Evaluation and Discussion

In this chapter we present an experimental study of the *Meta k*-NN aggregation technique described in the previous one. As there are numerous variables in the process of aggregating ensemble of classifiers through the *Meta k*-NN technique, we will use several different criteria to perform a comprehensive evaluation of the strategy; each one will be directed to show a comparison on a particular side of the problem.

In the following section 6.1 we will concentrate the evaluation study on databases that do not present a skewed class distribution problem (see Zenobi & Cunningham, 2002), in section 6.2 we will propose some strategies, in particular a modification of the *k*-NN algorithm, designed to address the skewed class distribution problem, while in 6.3 we will shift the evaluation to datasets where the class imbalance plays an important role. We have divided the study in this way because simple 0/1 error measures are very questionable for datasets with very unbalanced class distribution. Finally in section 6.4 we will show an evaluation that focuses on the "quality" aspect of the retrieved cases, with the help of two domain experts.

We have focused for simplicity on binary classification problems, except in one case. As we have explained in the previous section, the *Meta k*-NN aggregation technique is easily generalized to the case of C classes or to the case of regression problems.

6.1 Evaluation on Datasets with equally distributed classes

The evaluation performed in this section shows two things:

1) Using the *Meta k*-NN aggregation technique we obtain performance (accuracy) that is at least comparable to that obtained with the standard weighted average technique, provided a sufficient number of neighbours k_M is retrieved. This is shown by a comparison of the generalization errors of both the techniques. Needless to say, the advantage of *Meta k*-NN is the capability to retrieve a consistent set of cases for the explanation.

2) The *Meta k*-NN technique, that is a single classifier working with a transformed representation produced by the ensemble, models the problem domain in a way that is very similar to the same ensemble when aggregated with weighted average. This is shown by measuring the fidelity of the predictions from aggregation with the *Meta k*-NN to the predictions given by aggregation with weighted average.

Since these evaluations are very computationally intensive we present results on only four datasets, three are the same datasets we used in Chapter 4 for the evaluation of the *AmbHC* strategy to train ensembles with diversity (Pima Indians, Heart Disease from the UCI repository, Warfarin described in Byrne et al., 2000), the other datasets (Cylinder) is taken from the UCI repository too. As mentioned in Chapter 4, the Pima Indians datasets is one of those "borderline" cases in regard to class unbalance; the class distribution is (65%; 35%), moderately unbalanced. We will perform a study on Pima Indians in 6.3. when dealing with skewed class datasets; however, similarly to what we have done in Chapter 4, since the dataset is not too computationally expensive (only 8 features) we will perform an analysis also in this section (as non-skewed class). We will compare these results later on with the ones obtained with the skewed class analysis to see if even a small imbalance could affect CBR performances. These are all datasets with a binary classification problem; in the case of Warfarin we have turned the original problem into a 2-class task, with the help of a domain expert.

We have already performed an evaluation on different ensemble sizes for Pima Indians, Heart Disease and Warfarin in Chapter 4, so we will not go any further in this direction and take for each dataset the ensemble size that proved to be the most

accurate (25 in the case of Pima Indians, 23 in the case of Heart Disease and 29 in the case of Warfarin); in the case of Cylinder we have also limited our study to ensembles of size 25 due to the very large amount of data and consequent very big computational time spent to perform all the operations.

6.1.1 Results of the Evaluation

In the following set of four figures we show the first of the two studies mentioned above; each figure refers to a different data set. For a complete comparison we have applied to each dataset both the *HC* and *AmbHC* training algorithms described in section 4.3. The generalization error of each ensemble was determined using 5-fold cross validation. This entire process was repeated 3 times (except for the Cylinder dataset, where it was repeated only 2 times due to its computational heaviness) and the results averaged since the hill-climbing strategy is quite sensitive to the initial condition. We have also performed a paired *t*-test analysis on the average so obtained in the case of k_M equal 15 and 25 and we have obtained confidence of >85% for Warfarin, >95% for Heart and Pima, and >98% for Cylinder.

The diagrams, reported in figure (6.I) to (6.IV), show the generalization error of the *Meta* k-NN aggregation strategy (both for ensembles trained with *HC* and *AmbHC*) plotted against the number of retrieved neighbours k_M. To facilitate comparisons we also show the generalization error of the standard weighted average technique (both for ensembles trained with *HC* and *AmbHC*) and the average generalization of the component classifiers in the ensembles. These four figures appear as horizontal lines as they obviously do not depend on the value of k_M.

Next Figure (6.V) shows the second study mentioned at the beginning of this section. It compares the fidelity of the *Meta k*-NN classifiers with the corresponding *AmbHC* ensemble. We do not consider the *HC* ensemble because it (and the corresponding *Meta k*-NN classifier) has poorer accuracy.

We have plotted for all four datasets the figures for fidelity between the predictions given by the *Meta k*-NN and standard aggregation technique. This is calculated as a binary error (0 if the two predictions match, 1 if they don't) and is plotted against increasing values of k_M.

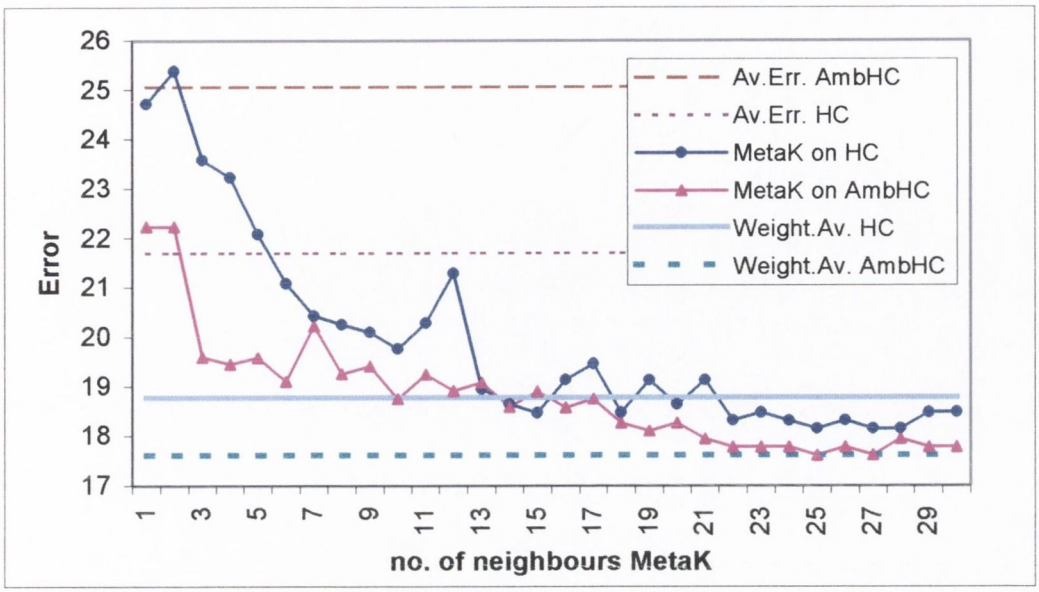


Fig. 6.I Results for Heart data

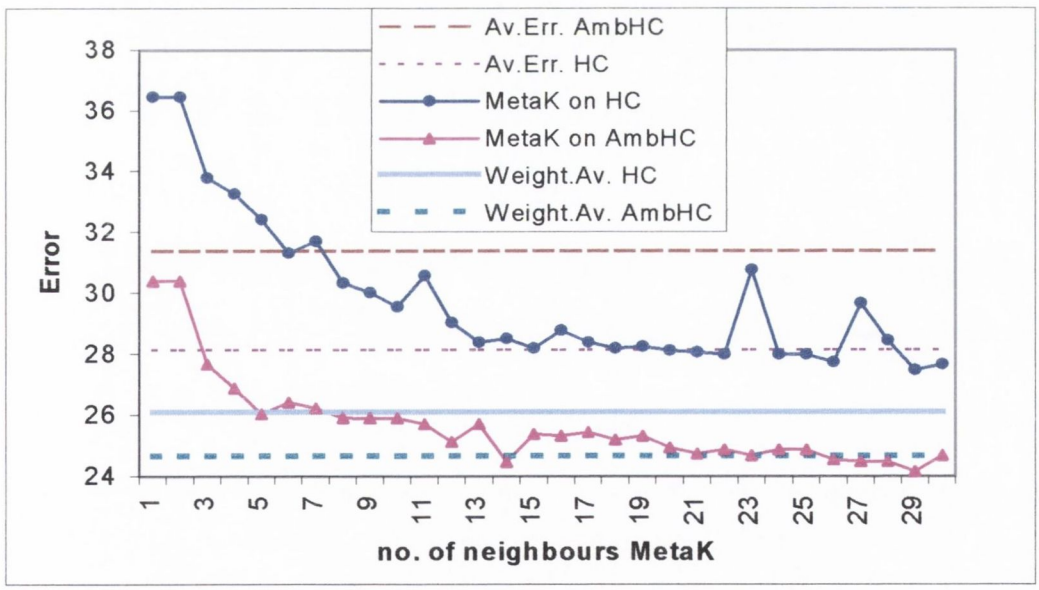


Fig. 6.II Results for Pima data

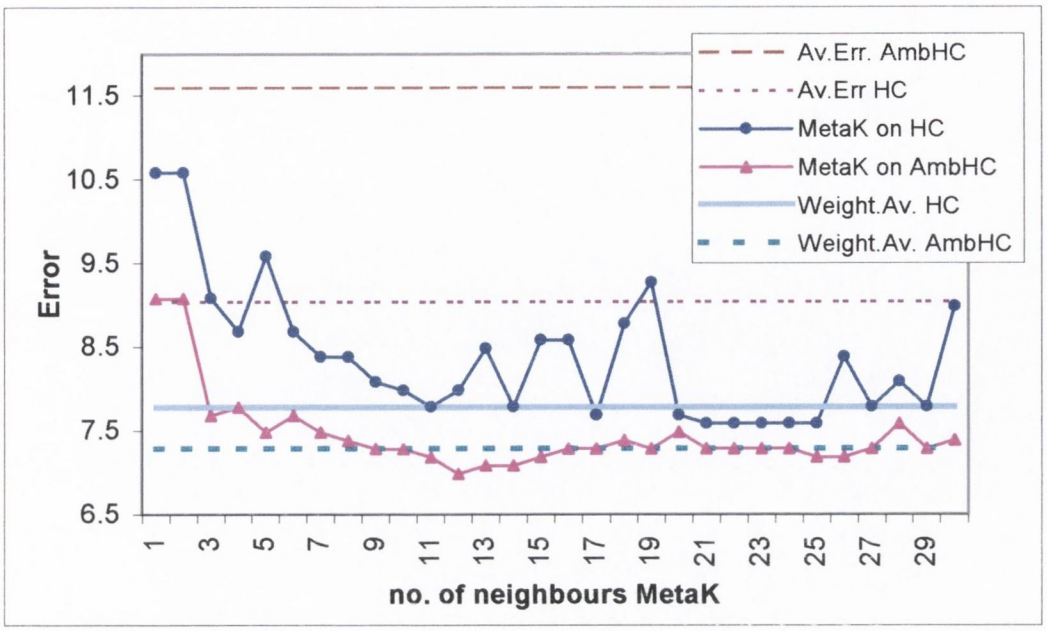


Fig. 6.III Results for Warfarin data

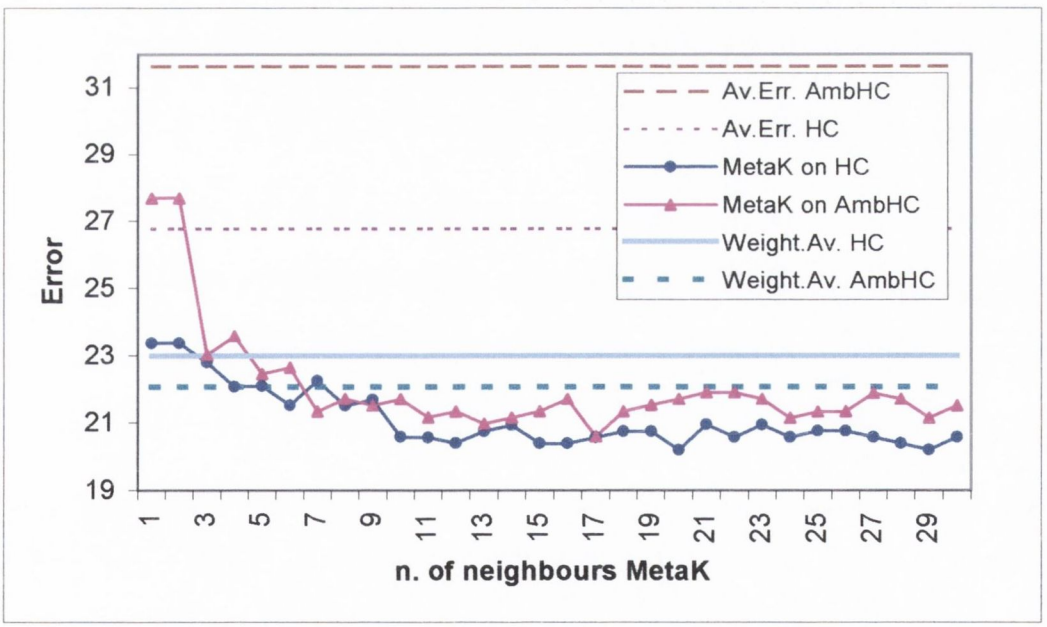


Fig. 6.IV Results for Cylinder data

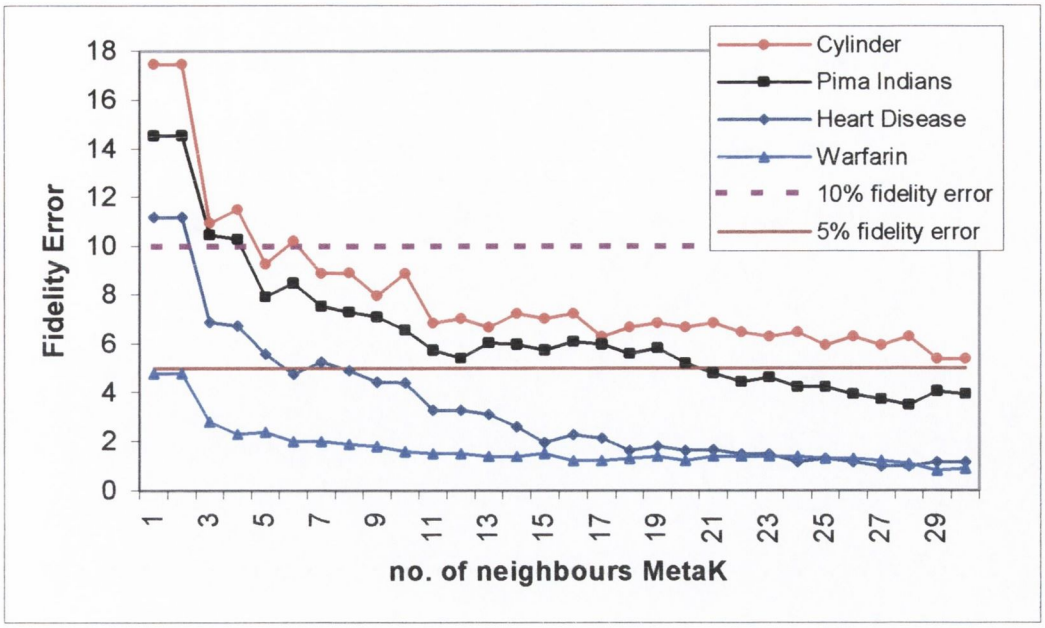


Fig. 6.V. Fidelity error of the *Meta k*-NN classifiers to the ensemble on which they are based. The 10% and 5% error lines are shown for comparison.

6.1.2 Discussion

From the results shown in the figures above we can make a few observations. First, there is a confirmation of the fact that ensembles trained with *AmbHC* and using the standard aggregation approach have a better generalization error that those trained with a simple *HC*, even though the average error of the component classifiers are considerably worse. This is no surprise, and simply shows once again that diversity plays a crucial role in the ensemble performance.

Second, the *Meta* k-NN aggregation technique performed on ensembles trained with *AmbHC* scores either comparable or in some cases better results to the weighted average aggregation technique for larger values of k_M. As a general trend we observe that already for values of k_M between 5 and 7 the result obtained by *Meta* k-NN is comparable to a weighted average on ensembles trained with *HC* (i.e. a score about between 1% and 2.5% worse than the one for *AmbHC* in three datsets and a better score in the remaining one), with the great advantage of retrieving a small set of cases for explanation. For values of k_M between 10 and 13 (still an acceptable number of retrieved cases) the *Meta k*-NN starts to outperform the weighted average aggregation for *AmbHC* strategy in two datasets, and when we increase the value of k_M up to 25

the *Meta* k-NN technique outperforms ensembles trained with *AmbHC* and using the standard aggregation technique except in the case of Heart dataset where it has equal accuracy. Even though in this last case the set of retrieved neighbours is large (but still smaller than that retrieved by a classic aggregation technique, an ensemble of 25 5-nearest neighbours would retrieve a pool of 125 cases!) it has the important capability of giving a coherent *ranking* to the retrieved cases. Furthermore, if we examine carefully the figures above we will notice that the difference in the performance between a low value of k_M (around 5) and a high value (over 20) is at most about 1.5% in the accuracy. That means only 1.5% of the cases will be differently classified by the two *Meta* *k*-NN differing on the value of k_M, for example in the case of Heart Disease it would be 4-5 cases at most over 300.

This observation allows us to introduce a little modification in the *Meta* *k*-NN algorithm in the case the final user prefers anyway to have a (very) small set of retrieved cases even when the dataset "needs" a large k_M to obtain the best performance (like in the case of Heart Disease): with any query it will be sufficient to perform the prediction with the large k_M and then go "backward" decreasing k_M until the *Meta* *k*-NN reaches the point where the final prediction changes. The smallest k_M found that preserves the same prediction of the large one (for that particular query) gives the smallest set of neighbours to be retrieved for an explanation. After all, due to the coherent ranking, if we retrieve k_M= 25 cases the first 5 cases will be anyway the *same* cases retrieved by a *Meta* *k*-NN with k_M equal to 5. Only in a small percentage of the cases (for Heart Disease only on 4-5 cases) the *Meta* *k*-NN will be "forced" to retrieve up to 25 neighbours. In this sense an interesting development for further research would be to perform a study to see if these few "difficult" cases are somehow related to the status of "noisy cases".

Third, the *Meta* *k*-NN aggregation technique performed on ensembles trained with simple *HC* scores generally worse generalization errors than the one performed on ensembles trained with *AmbHC*. A possible explanation for this phenomenon can be the fact that classifiers trained with *HC* have a lower diversity, so they carry less "rich" information about the problem domain, than the ones trained with *AmbHC*; when the *Meta* k-NN case-base is created it is possible that the columns (classifiers' predictions) show a higher dependence in the case of *HC*. The only exception to this is the Cylinder dataset where the *AmbHC* approach has no clear advantage over the *HC*

technique. This is probably due to the large number of features in this dataset (38) and the consequent "natural" diversity that exists even in ensemble members trained using *HC*.

Fourth, from figure (6.V) it is clear that the fidelity is high. After k_M=5 already all the datasets (except Cylinder) score a fidelity over 95% (error less than 5%) and for two of the datasets the fidelity goes up to 98% and more as k_M increases. We can reasonably argue that the problem domain decomposition in the case of the two different aggregation strategies is nearly equivalent.

6.2 The Skewed Class Distribution Problem

In many domains it is frequently the case the database has a very unbalanced class distribution. Think of the example of a medical domain, where the outcome expresses the fact the patient has developed some kind of illnesses or not. In many circumstances it may happen that only a very small fraction of the tested patients has developed the illness in a later stage; in this case the database will show a big imbalance of cases with outcome "No Illness".

Two are the main ways a skewed class distribution affects CBR classification tools like *k*-Nearest Neighbours. The first one regards the way the problem space is "filled" by the sampled cases. The underlying unknown distribution (in the previous example, the dynamic of the illness) may have a quite complex dependence on the feature set. If the areas regarding one or more of the classes in the problem space are undersampled there is a very high risk of wrong retrieval for a tool like nearest neighbour. For example a query case may appear closer to an example of the wrong class only because the class area to which it belongs, is not filled with enough cases. When the case availability of some of the classes is low this may become a very serious problem.

The second one regards the way accuracy is measured. If we perform simple error measures like 0/1 in the case of skewed class distribution we may end up with some absurd results. Consider for example a database with a binary classification problem (classes 0 and 1) whose class distribution is: 90% of the cases are labeled as class "0" and the remaining 10% as class "1". Consider the most banal among all the classifiers:

the one that always replies "0" to any query, without any real inspection of the problem space; with a simple 0/1 measure this classifier will score 90% accuracy! This simple example shows how questionable evaluation performed with simple accuracy becomes when the class imbalance is considerable. Nearly all the domain experts seem to agree it is very much preferable to have in general a balanced error in the various classes and a smaller overall accuracy rather than a big overall accuracy all biased toward the majority class.

In next two subsections we will show two solutions to face both problems; they will be evaluated in section 6.3 when applied to databases that present a considerable skewed class distribution problem.

6.2.1 A Modification of *k*-NN to deal with skewed class

This sub-section will address the first of the two problems mentioned above: the problem space sampling issue. There have been several proposed ways in literature to deal with this problem. Lawrence et al. (1998) introduced algorithms for neural network classifiers that either scale weights updates according to prior class probabilities or alter class frequencies probabilistically, with quite good results even though it proved to be difficult to determine a priori the optimal degree of scaling. Cardie & Howe (1997), Aha & Goldstone (1992) and Domingos (1997) have proposed in a variety of ways the use of local feature weights to assign each case a different value depending on the prior class distribution the case belongs to.

The approach we are concerned with here is different, and has proved to be quite simple and effective. It is basically a variation of the classic nearest neighbours algorithm, called *Duplicate k-NN*, that performs exactly the same operations of a normal *k*-NN until it has retrieved the set of nearest neighbours. At this stage we have all the retrieved cases in a sort of stack of size *k* that would normally be used for the class probability computation and a subsequent query prediction. What *Duplicate k-NN* does instead, it scans the stack from the top (closest case) to the bottom (furthest case) and when it meets a minority class case in position *i*, it "expands" the case creating a number of duplicates that push down all the cases from position *i+1* onwards. Of course the stack size remains *k*, so the cases that have been pushed lower than position *k* (furthest cases) will be removed from the stack. The number of created duplicates is given by the ratio between the classes distribution. Say for example in a

binary classification problem the two classes "Black" and "White" are present with respectively 200 and 600 cases. The ratio between the two classes distribution is 1/3, that means to each sampled "Black" roughly correspond three sampled "White". So the number of duplicates created each time a "Black" is retrieved is three. The figure below illustrates the principle; after the duplicates are created the new stack contains 3 "Black" nearest neighbours against 2 "Whites", so that the final prediction is given as "Black" (a standard *k*-NN would have instead counted 4 "White" against 1 "Black"and replied "White").

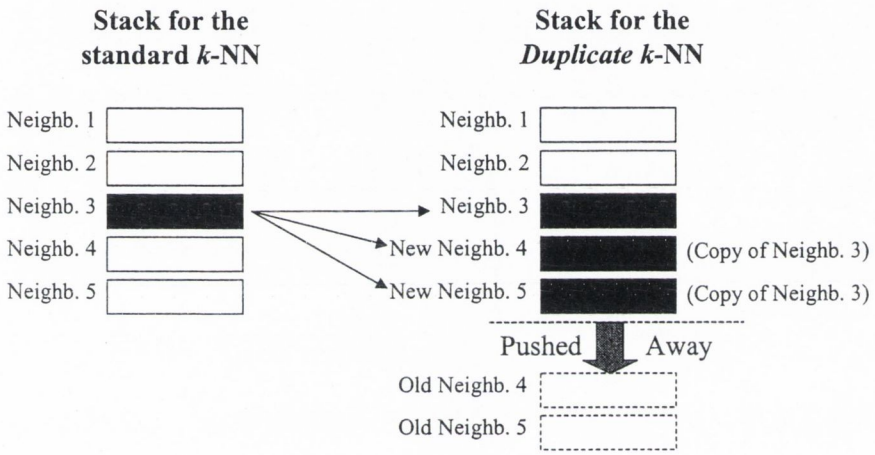


Fig. 6.VI. The *Duplicate k*-NN principle

6.2.2 Use of Harmonic Mean to Evaluate Performance

This section addresses the second problem mentioned at the beginning of section 6.2: the search for an evaluation measure capable to reduce the negative effect a different prior class probability has on simple 0/1 error measures. An interesting way to deal with this problem can be found in Provost & Fawcett (1997, 1998) and Provost et al. (1998). After introducing the idea of *ROC* analyses for skewed class distribution databases, the *ROC convex hull method* is outlined for binary classification problems, making use of computational geometry. The main idea behind this technique is that if we are able to assign a different cost to misclassification for each one of the classes,

we are also able to derive a simple linear equation that relates false positive and false negative cases to the prior probabilities for the single problem; a way to compare different classifiers is then based on a set of parallel lines whose slopes are the same of the one computed for the specific problem, based on the prior class probabilities and the assigned misclassification costs. The optimal classifier will be the point on the convex hull (drawn from all the classifiers' *ROC* curves) that intersects the iso-performance line with the largest intercepts of true positives. This interesting technique proved to be theoretically very sound, however there are several downsides. First, the difficulty to give a misclassification cost (how "much" is worth a misclassification for a patient that is going to develop an illness compared to another one that is not?). Second, it is much less interpretable than a simple accuracy measure. Third, it's computationally a lot more intensive than any accuracy measure; for example it would be impossible to train an ensemble with a *ROC convex hull method* without aggravating in a very significant way the computational time, that is already remarkable.

Some alternative ways to deal with this issue have been proposed by Atkeson et al. (1997), that created different similarity metric for each test case, Hastie & Tibshirani (1994) and Friedman (1994) that were computing test-case-specific metrics based respectively on discriminant analysis and recursive partitioning. Hansen (2000) in his PhD dissertation proposed also the Kullbak-Leibler error function for classification problems, which is theoretically very elegant and keeps into account the prior probability distribution for all the classes; again, interpretability is the downside for this error metric.

The focus of this thesis is on interpretability. It is very frequently the case that in domains where explanation plays a crucial role, like medical or financial ones, the final user is a domain expert (a doctor or a financial advisor) but not a Machine Learning or a Statistics expert. Having a simple and clear index to measure performances (even if it may have some limitations) is in this case preferable than having indexes like ROC or Kullback-Leibler that need each time to be translated into something more "concrete". Furthermore, we have seen that computational time plays a vital role when dealing with ensembles of classifiers and a small computational effort increase may become a crucial characteristic for the choice of a performance index in the context of ensemble learning.

For all the reasons explained above we have chosen to perform error measures with a simple tool, the *Harmonic Mean* of all the single class accuracies. Suppose we are dealing with a classification problem with C classes. The accuracy A_i for class i is defined as:

$$A_i = \frac{\text{n. of cases for class } i \text{ correctly classified}}{\text{Total n. of cases for class } i}$$

Then the Harmonic Mean A_H of all the class accuracies is defined as:

$$\frac{1}{A_H} = \frac{1}{C} \cdot \sum_{i=1}^{C} \frac{1}{A_i} \qquad \textbf{(6.2.i)}$$

In the case of binary classification problem the equation in (6.2.i) becomes simply:

$$A_H = 2 \cdot \frac{A_1 \cdot A_2}{A_1 + A_2} \qquad \textbf{(6.2.ii)}$$

Note that in the case we use the Generalisation Error as measurement index, we need simply to recall that it is the complement to 1 of Accuracy *(A=1-E)* so that the two equations become:

$$\frac{1}{1-E_H} = \frac{1}{C} \cdot \sum_{i=1}^{C} \frac{1}{1-E_i} \qquad \textbf{(6.2.iii)}$$

and in the case of binary classification problem:

$$E_H = 1 - 2 \cdot \frac{(1-E_1) \cdot (1-E_2)}{(1-E_1) + (1-E_2)} \qquad \textbf{(6.2.iv)}$$

The harmonic mean index has some very important properties that make it particularly useful for our task. First, it gives a much better prize to situations where the errors are more homogeneous rather to situations where there is a big imbalance among errors. The following figure shows the trend of the Harmonic Mean in the case of binary classes for accuracies greater than 50%; as we can see the Harmonic Mean has a maximum when the two accuracies are equal, the colours indicate lines of constant Harmonic Mean. If the imbalance between the two accuracies grows we move along a line with constant Harmonic Mean so that, for example, if Accuracy on class1 is 90% and Accuracy on class 2 is 60% that would have the same Harmonic Mean of two equal accuracies of 72%.

In the limit of one of the two accuracies being 0% (error equal to 100%) the Harmonic Mean in 6.2.ii goes down to 0%. This is exactly what we were looking for, the "banal" classifier mentioned in 6.2 would have scored a 0% accuracy in this case.

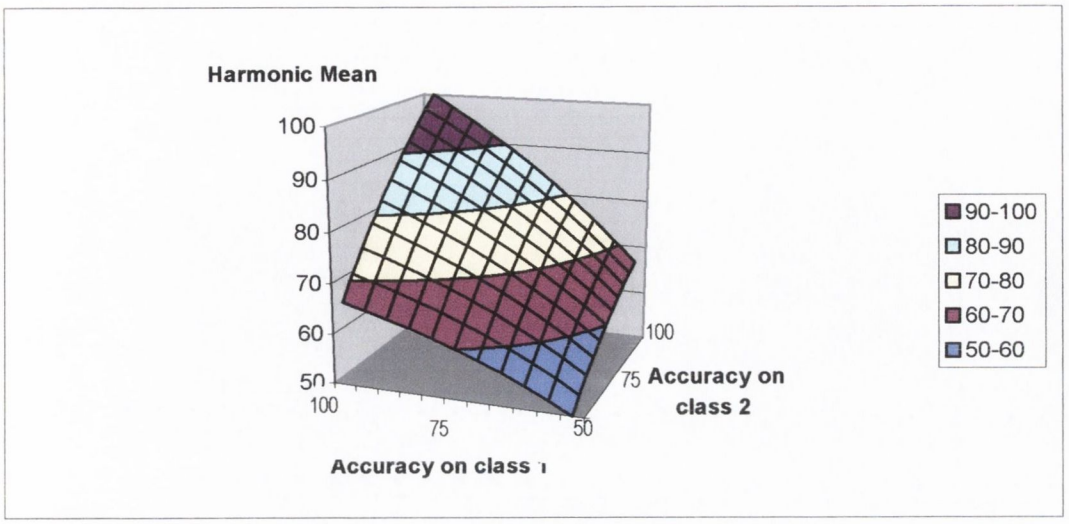


Fig. 6.VII Harmonic Mean of two Accuracies

Second, it has quite a straightforward interpretability. When we say that a certain tool has an Harmonic Mean Accuracy of A_H we know it is *equivalent* to a tool that has *all the single class accuracies* equal to A_H. For example, say a binary classifier applied to a database whose prior class distribution are 90% of "0" and 10% of "1" scores accuracy of 87% for class "0" and 58% for class "1"; its harmonic mean would be of roughly 69.6%, which looks a lot more reasonable than the simple accuracy of 84.1%. We know the classifier is equivalent to one who has both accuracies scores at 69.6%, this gives a much better information about the "goodness" of the classifier.

Third, the increase in computational time is very low. At any step we only need to separate the computation for any class, and this is easily implemented, plus we need only to compute a simple mathematical expression; this takes much less time than other more complicated error measures.

Furthermore, if on one side it is true that Harmonic Mean won't take into consideration any prior class probability but only the class imbalance, on the other one we know we will be using it combined with the *Duplicate k*-NN strategy; this last one will somehow deal with the prior class distribution through the duplication mechanism, while the harmonic mean will give a more sensible measure of the results scored.

6.3 Evaluation on Datasets with Skewed Class Distribution

In this section we will present a complete evaluation of the *Meta* *k*-NN aggregation strategy in the case of databases with skewed class distribution. We will make use both of the *Duplicate* *k*-NN strategy and the Harmonic Mean error measure we explained in the previous section. The main goal of our study remains the comparison between the *Meta* *k*-NN and the classic Weighted Average aggregation strategies. This will be performed through the following evaluation guidelines:

a) we will compare *Meta* *k*-NN and Weighted Average for both ensembles of classic *k*-NN classifiers and *Duplicate* *k*-NN classifiers. It is worth remarking that in the case of *Meta* k-NN, the aggregation step (performed through a nearest neighbour tool) will be also using the *Duplicate* strategy;

b) the comparison in a) will be evaluated both using the simple 0/1 and the Harmonic Mean error measures, to see how much class imbalance affects the two classification strategies;

c) we will perform all the comparisons for each dataset using three different ensemble sizes; they will all be between 15 and 31, chosen after a preliminary study on the datasets;

d) as in 4.3 we have seen *AmbHC* always outperforms the simple *HC* selection strategy, we won't pursue any further this evaluation and leave for simplicity the comparisons only on ensembles trained with *AmbHC*.

The evaluation will be performed on three new datasets taken from the UCI repository, Ionosphere, Horse Colic and Spect, they all present a skewed class distribution problem in different percentage (in the case of Spect it is a remarkable unbalance, see Appendix A). We also used a different version of the Warfarin database where the outcome is mapped with the help of the domain expert into 3 classes instead of 2, the third class being a minority class, and the already mentioned Pima Indians dataset.

6.3.1 Results of the Evaluation

In the next few pages we will show a set of 4 graphics for each dataset. The first two graphics on the top show the comparison between *Meta* *k*-NN and Weighted Average of ensembles trained with *AmbHC* using classic *k*-NN, figure (a), and

Duplicate k-NN, figure (b), evaluated with a simple 0/1 Error. The second two graphics on the bottom show the same comparison, this time evaluated with the Harmonic Mean index, figure (c) for standard k-NN and figure (d) for *Duplicate* k-NN. In each graphic we have printed both *Meta* k-NN and Weighted Average for three different ensemble sizes, for a total of 6 plotted lines. In the y axis it is shown the score through the Generalization Error (simple 0/1 error E for the first two figures, Harmonic Mean E_H for the second two figures) while in the x axis it is shown the value of k_M for the *Meta* k-NN strategy. Obviously, of the six printed lines, the three representing Weighted Average are straight lines as they do not depend on the value of k_M. To facilitate the graphics reading we only remark that in the choice of the colours for the lines we have used the same colour with a slight different intensity for *Meta* k-NN (light blue to dark blue) and Weighted Average (dark yellow to orange); the darker the line the higher the ensemble size it refers to.

6.3.2 Discussion

From the set of figures (6.VIII) to (6.XII) shown below we can draw a couple of general conclusions:

First, in all of the twenty cases shown, with the only exception of the "Duplicate-Simple Error configuration" for the Pima Indians dataset (fig. (6.XI), case (b)), *Meta* k-NN is either comparable or better than the Weighted average technique, provided we use a sufficiently large ensemble size (around 25-27) and in some datasets a sufficiently large k_M (over 10). This is a confirmation of what we found in 6.1.

Second, the use of the Harmonic Mean instead of the simple 0/1 Error can change dramatically the interpretation of the results, while the use of the *Duplicate* k-NN instead of the classic k-NN can change dramatically the error distribution over the classes, and as expected it improves the accuracy on the minority class in the face of a deterioration of the accuracy on the majority class. If we restrict our study to a simple 0/1 Error measure in three datasets over five the *Duplicate* k-NN scores a worse result than the classic k-NN (top couple of graphics) while in the remaining 2 it scores comparable result. If we use instead the Harmonic Mean in four of the five datasets the Duplicate k-NN scores clearly a better result than the classic k-NN, both for Meta k-NN and Weighted Average (bottom couple of graphics), while in the remaining one (Horse) only the Weighted Average scores a worse result using *Duplicate* k-NN.

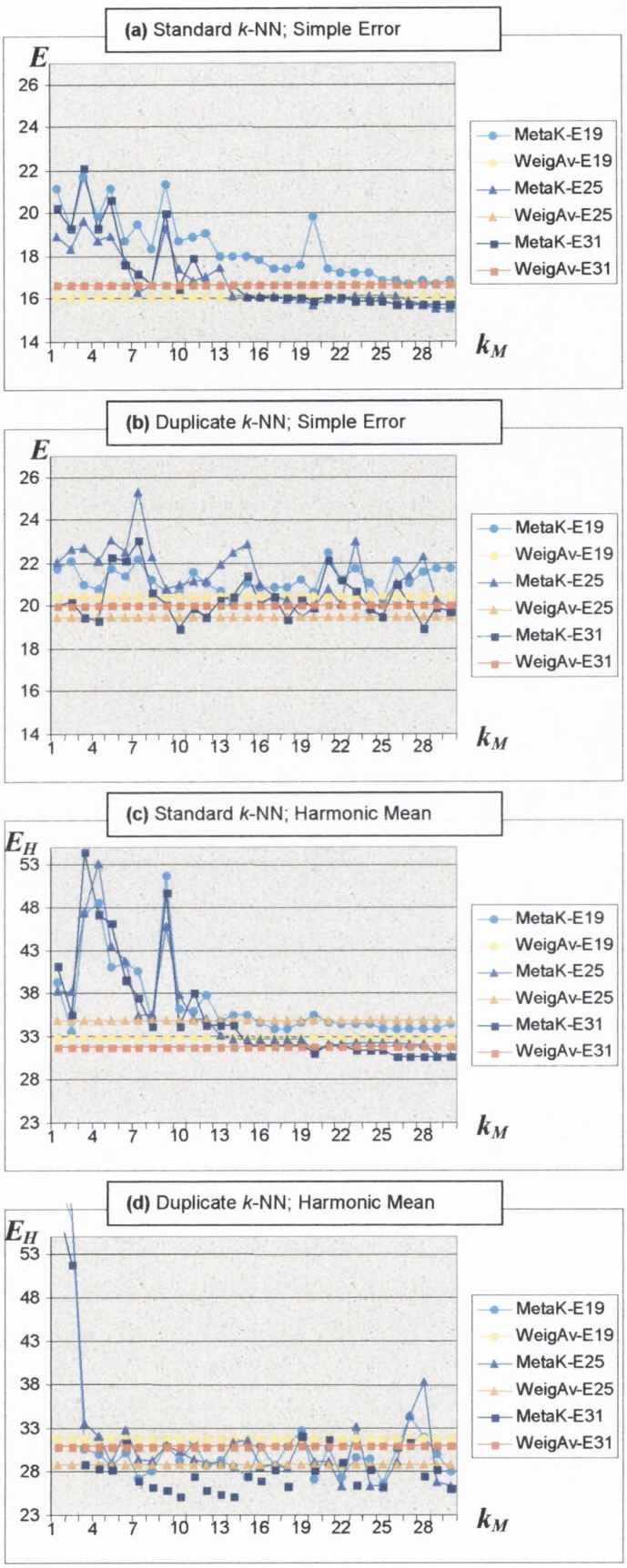


Fig. 6.VIII. Study on the **Spect** dataset

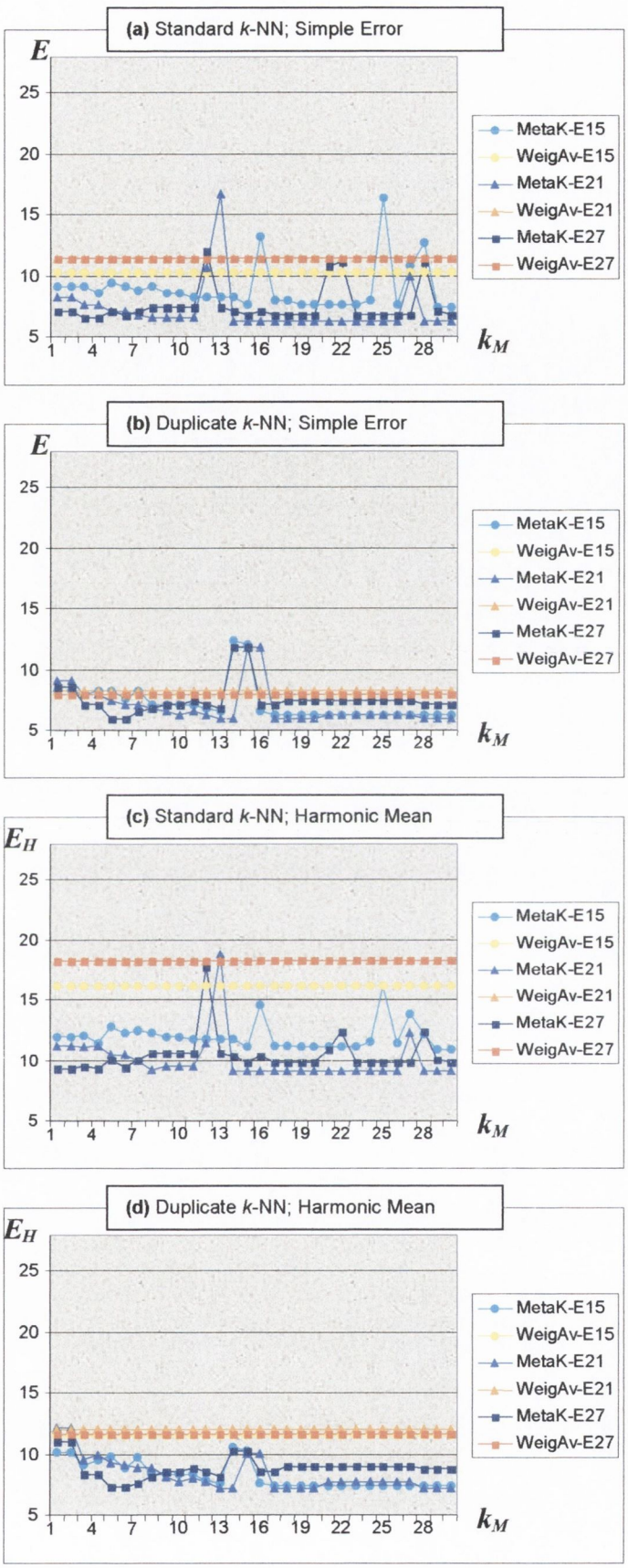


Fig. 6.IX. Study on the **Ionosphere** dataset

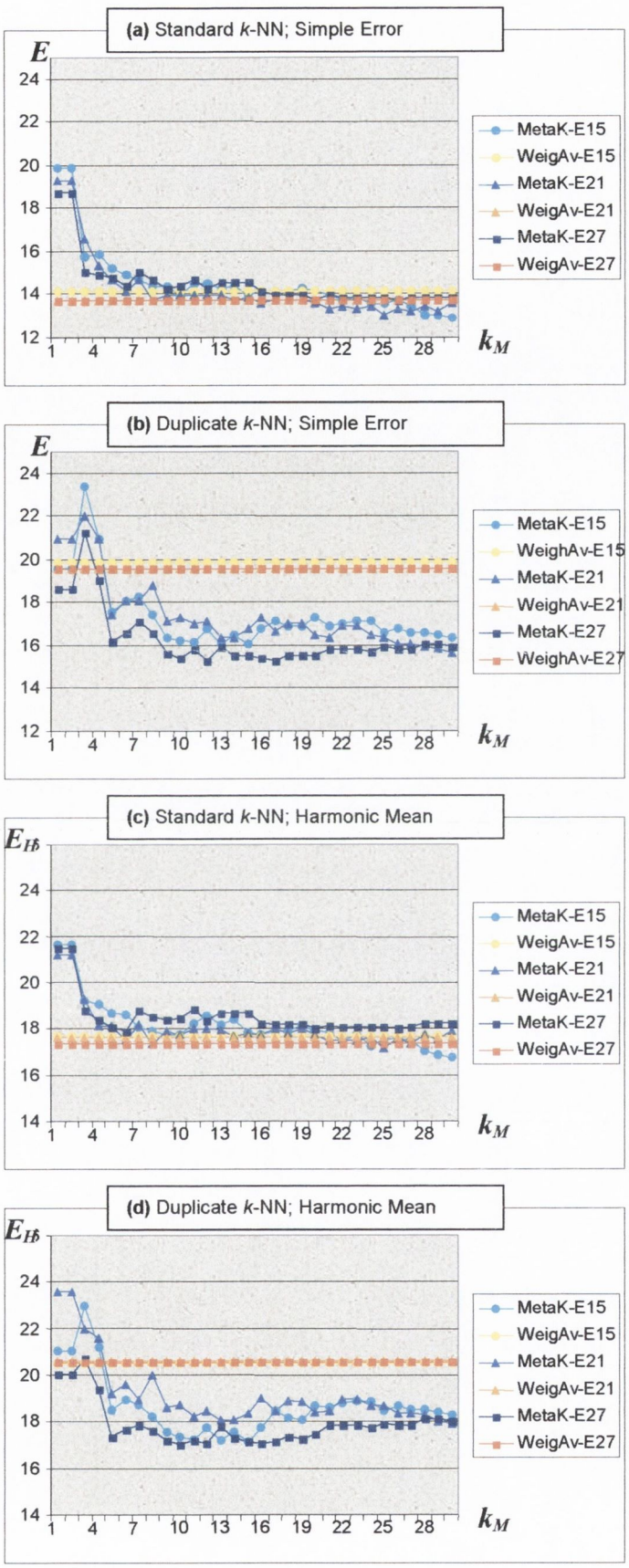


Fig. 6.X. Study on the **Horse Colic** dataset

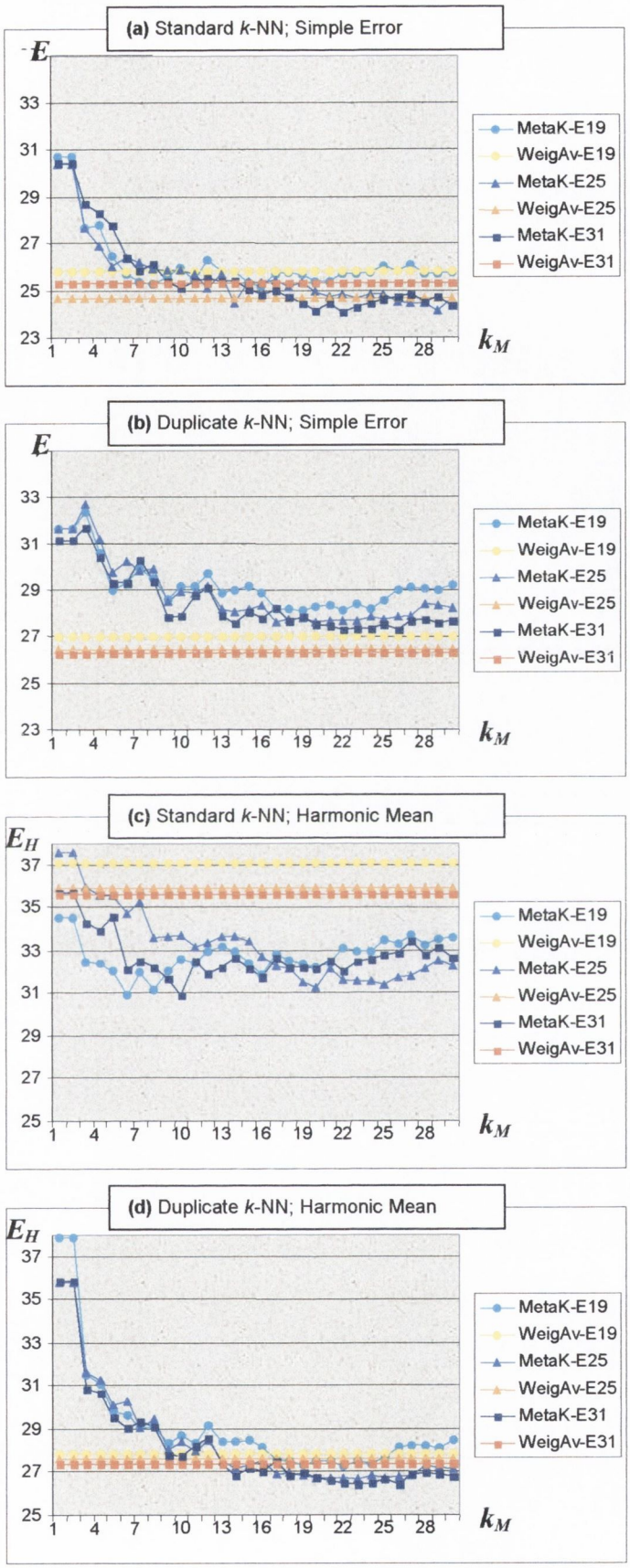


Fig. 6. XI. Study on the **Pima Indians** dataset

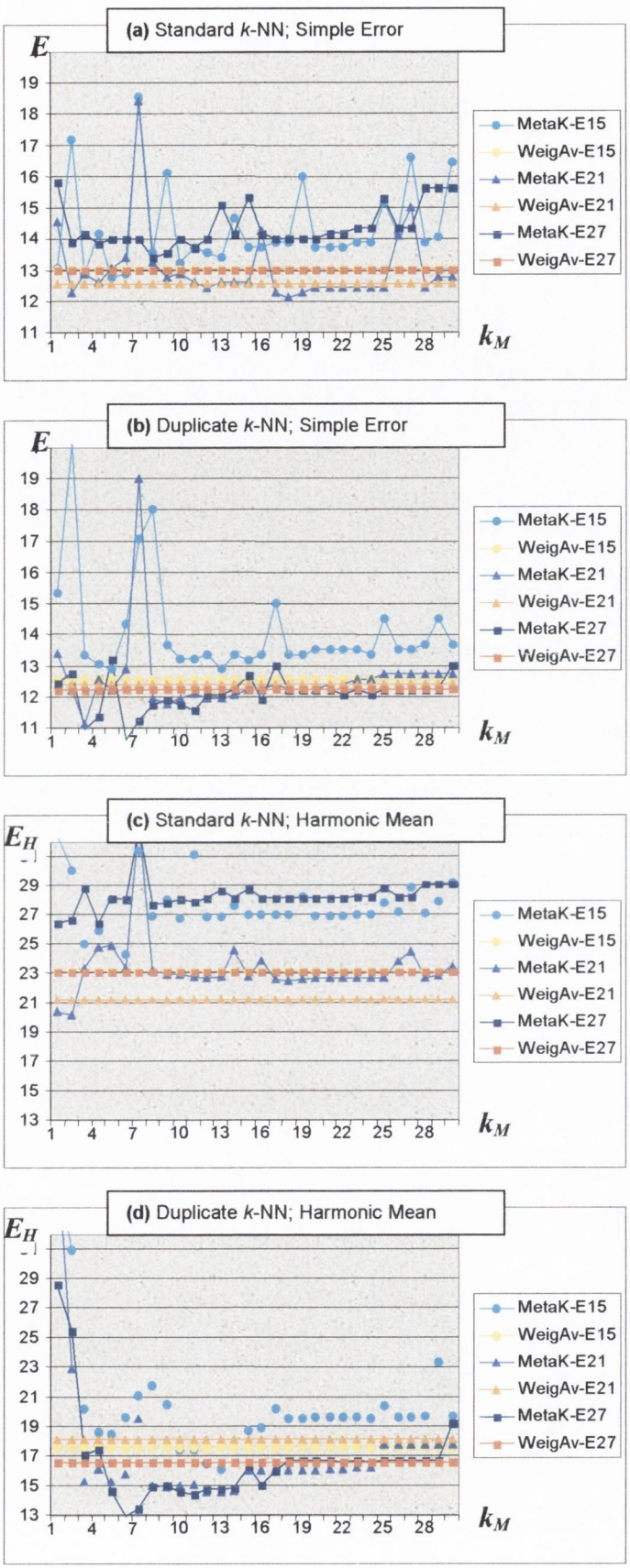


Fig. 6.XII. Study on the **Warfarin (3classes)** dataset

From all the reasons we explained in 6.2.2, when the evaluations performed with a simple 0/1 Error measure and Harmonic Mean differ considerably we are clearly inclined to prefer an evaluation through the Harmonic Mean. Thus, if we focus our attention on the set of graphics that show comparison on the Harmonic Mean (case (c) and (d) on each of the figures (6.VIII) to (6.XII)), two important conclusions can be drawn.

a) the *Duplicate* *k*-NN classification tool outperforms the classic *k*-NN one when dealing with skewed class distribution problems;

b) Second, the *Meta* *k*-NN aggregation strategy either outperforms or is comparable to the Weighted Average one and in 4 cases over 5 already with a small value of k_M (around 5-7 retrieved cases, see bottom-right figure for each dataset).

This is certainly an excellent result both for the *Meta* *k*-NN aggregation strategy and the Duplicate modification of the classic *k*-NN.

As previously mentioned, we have no formal analysis to explain the good performance of *Meta* *k*-NN; in this context only conjectures can be drawn. We will simply conclude this section with an observation on the issue, leaving to further research any possible development.

The weighted average aggregation strategy naturally arises from a statistical analysis when we deal with ensembles of predictors trained on different subsets of the Case-Base (see section 4.2); in this case it leads to a formal equivalence of Bias/Error and Ambiguity decompositions. When we deal instead with ensembles of *k*-NN classifiers trained on different feature subsets there is no longer a reason to prefer the weighted average to aggregate ensemble members. When using *Meta* *k*-NN, it is possible that the aggregation process performed through the Meta Database representation accesses in a better way the model of the problem domain that is implicit in the ensemble.

In the next section we will complete our study. As the main advantage of the *Meta* *k*-NN aggregation strategy remains the capability to retrieve a consistent set of neighbour cases for explanation, we will investigate the "goodness" of the retrieved cases.

6.4 Evaluation of the "quality" of the retrieved cases

So far we have shown a comparison between *Meta k*-NN and Weighted Average aggregation strategies in terms of performance measures, 0/1 Error in the case of equally distributed classes and Harmonic Mean in the case of skewed classes. We have found that *Meta k*-NN is either better or comparable to Weighted Average, depending on the database.

Nevertheless the main advantage of using *Meta k*-NN still remains the capability to retrieve a relatively small set of neighbour cases for explanation. In this section we will perform an evaluation of the "goodness" of the retrieved cases, in terms of utility for an explanation. The only way to perform this evaluation is through the help of a domain expert. Furthermore, as with the Weighted Average there is no clear way to retrieve a small subset of cases for explanation, we will compare the set of neighbours retrieved by *Meta k*-NN with the set of neighbours retrieved by the best single *k*-NN classifier (best feature subset), chosen through a Backward Sequential Selection (see Chapter 3).

We have performed on two datasets this evaluation, with the kind help of Dr. Stephen Byrne for the Warfarin dataset (3 classes version) and of Dr. Paul Walsh for the unpublished Bronchiolites dataset, which is a simple binary classification problem where the outcome is the attribute "Hospital Stay" mapped in 2 classes: "0" for "*Hospital stay*" and "1" for "*NO hospital stay*". The database is actually quite small (only 118 cases), it is described by 22 features, and many cases have more than 1 missing values. Here below are the guidelines for the evaluation.

To perform an accurate and not too expensive (in terms of time taken to the two practitioners!) evaluation we have randomly removed a very small set of n cases, 11 for Warfarin and 7 for Bronchiolites, to be used as queries and done the training of the CBR system on the rest of the database; the removed cases have been chosen following the prior class distribution in both datasets. This process has been repeated three times, each time removing n different cases from the original database and re-doing the training, in order to have a total number of $3n$ original cases, plus a total number of $3nk_M$ retrieved cases (each case has associated the suggested set of closest cases for explanation) to evaluate for *Meta k*-NN and $3nk$ cases to evaluate for the Best Single *k*-NN. This is a better approach than removing directly $3n$ cases because it

would have dropped too much information from the original database. We have chosen to set both k and k_M to 3, in order to minimise the work for the two domain experts.

The percentage of accuracy obtained with the *Meta* k-NN for the entire Warfarin Dataset is about 89.1%, while for the entire Bronchiolitis Dataset is around 83.5%. The percentage of accuracy obtained in the total of the *3n* removed cases is instead, respectively, 90.9% for the Warfarin and 85.7% for the Bronchiolites. As these last two percentages are very close to the ones computed on the entire datasets we can be quite confident the subset of *3n* removed cases are representative of the whole database. A similar thing happens in the case of the Best Single Classifier, whose accuracies on the entire datasets are 86.9% and 81.4% respectively in the Warfarin and Bronchiolites datasets, while accuracies on the total of *3n* removed cases are 87.8% and 80.9%

The two domain experts task was to evaluate how "good" is the retrieved set of nearest case for each query, in terms of *similarity to the query* and also in terms of *amount of extra information* added to explain the query. Let's see more in detail what we mean with this.

Similarity is evaluated considering the influence of the attributes on the output (class) and not the bare similarity among the attributes themselves. Consider the following example with three cases, taken from a database whose output class is whether the observed patient is going to develop a liver disorder or not:

	Age	**Gender**	**Weight**	**Height**	***(LiverDisord)***
Patient 1:	34	Male	150 Kg	182 cm	***(Yes)***
Patient 2:	34	Male	75 Kg	182 cm	***(No)***
Patient 3:	45	Female	100 Kg	154 cm	***(Yes)***

In terms of bare similarity of attributes Patient 1 looks mostly similar to Patient 2: both Male, 34 years old, same height – only a difference in weight (even if substantial!). In terms of similarity considering the influence on the output it is obvious that Patient 1 looks closer to Patient 3 (they're both clearly overweight, considering height and gender, and very likely to develop some liver disorder - if we

give the assumption that being overweight is the main cause of a liver disorder) than to Patient 2. This last type of similarity is the one we refer to.

Amount of extra information is evaluated considering the potential that a case has to give some insights to explain the domain. In several cases it may happen that a single case is clearly related to the query, but has somehow a very banal relationship. The most meaningful example is the case of a medical database where a patient is monitored more than once, so to each patient may correspond more than 1 case. Suppose that a user of a CBR tool needs to get some information on the case "Mr. Jones" and the system retrieves as most similar case to "Mr. Jones" Mr. Jones himself, that wouldn't certainly look like a big domain knowledge discovery from the system, even though nobody can deny the preposition has a big degree of truth! In other words, the fact of retrieving the same patient is certainly not wrong in terms of "how close" a case is to the query, but the "goodness" of neighbours retrieval is measured also quantifying the amount of "extra-information" added by the system for the explanation. The situation becomes even worse when the system retrieves more than 1 neighbour. It looks very obvious that if the CBR tool retrieved as the five closest case 5 times "Mr Jones" that wouldn't certainly look like a big success for the tool in terms of extra amount of information given in support to explanation; at most it may look that the first of the five (identical) cases may be Ok, but as for the other four (somehow copies) they are definitely unnecessary. Smyth & McClave (2001) have pointed out the problem of retrieving a set of cases similar to the query but with a high degree of diversity is very relevant also in the case of recommendation systems.

The evaluation was performed through a careful examination, for each of the n queries of both sets, of the retrieved neighbours subsets and the assignment of a final rating to each one of them, from 1 to 5 (highest rating to the best result, as shown below). The rating mark was taking into account both the similarity to the query and additional information for the domain explanation, as indicated above. In the case of the retrieval of the same patient the domain experts agreed to give a rating of 3 (unless it was the patient monitored on a completely different circumstance that was giving anyhow some extra information). We decided to go this way because if on one side it's true that the prediction cannot be marked as "poor" (rating 1 or 2) it also cannot be marked as "rich" (rating 4 or 5). There's nothing wrong about retrieving the same patient but we felt that a 3 (i.e. case which gives "a moderate amount of extra

information") was more suitable for our evaluation. A possible list of meaning is indicated below.

Rating 5: *The case retrieved has a clear similarity to the query and/or adds a very good piece of information.*

Rating 4: *The case retrieved is quite similar to the query and/or adds a good piece of information.*

Rating 3: *The case retrieved is not particularly related to the query but can still be accepted as an explanation, or alternatively it is very related to the query but adds only a moderate amount of extra information (case of the same patient).*

Rating 2: *The case retrieved is moderately misleading and doesn't provide with any relevant piece of information.*

Rating 1: *The case retrieved is clearly misleading; it has very little in common with the query and doesn't provide any piece of information at all.*

The following two figures (6.XIII) and (6.XIV) show the percentage of ratings given by the two domain experts to the retrieved cases, both for *Meta* *k*-NN and for the best Single *k*-NN.

From the graphics shown below we can see in these two cases the *Meta* *k*-NN strategy proved to be more useful for the explanation (cases marked with 5 and 4) compared to the single best *k*-NN. Particularly in the Warfarin dataset, the single *k*-NN seemed to retrieve in many circumstances a case representing the same patient given as the query, so that it had the great majority of the cases marked with "3". As for the "misleading" cases (the ones marked with 1 and 2) while in the Bronchiolites database *Meta* *k*-NN scores better results than the best single k-NN, in the Warfarin it scores slightly worse results. This is anyway a consequence of the fact that the single k-NN retrieves most of the time the same patient while the *Meta* *k*-NN has a bigger variety in the case retrieval, and this bigger variety has increased a bit the probability to retrieve useless cases for the Warfarin dataset. The domain experts agreed anyway that the balance of the information provided by the *Meta* *k*-NN for the explanation is clearly to be considered bigger than the one provided by the best single *k*-NN. We only remark here the fact that the accuracy scored by each classifier is completely unrelated to the "goodness" of the retrieved cases; for example in the Warfarin dataset the accuracy of the single *k*-NN classifier is quite good, since it tends to retrieve in many occasions the same patients, but the rating is not as good seen that the information provided is small.

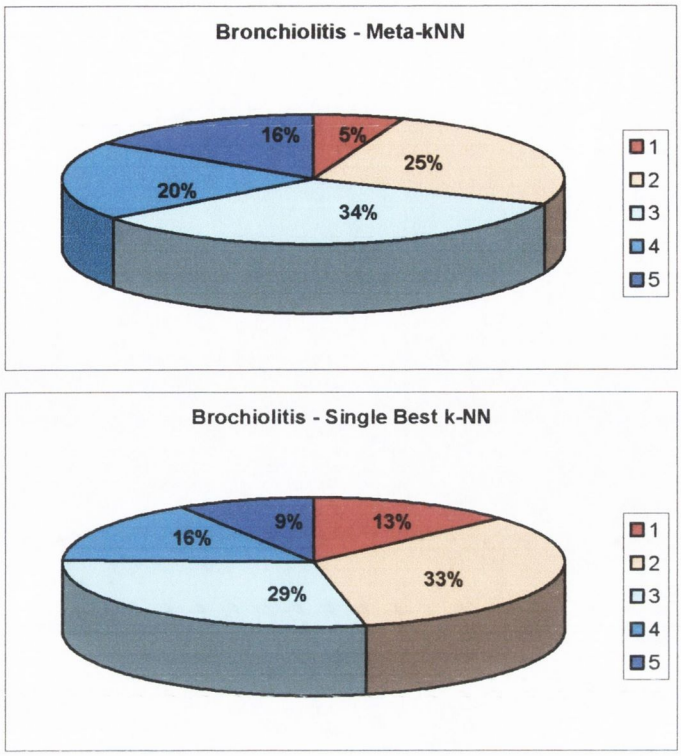


Fig. 6.XIII. Evaluation for the Bronchiolites dataset

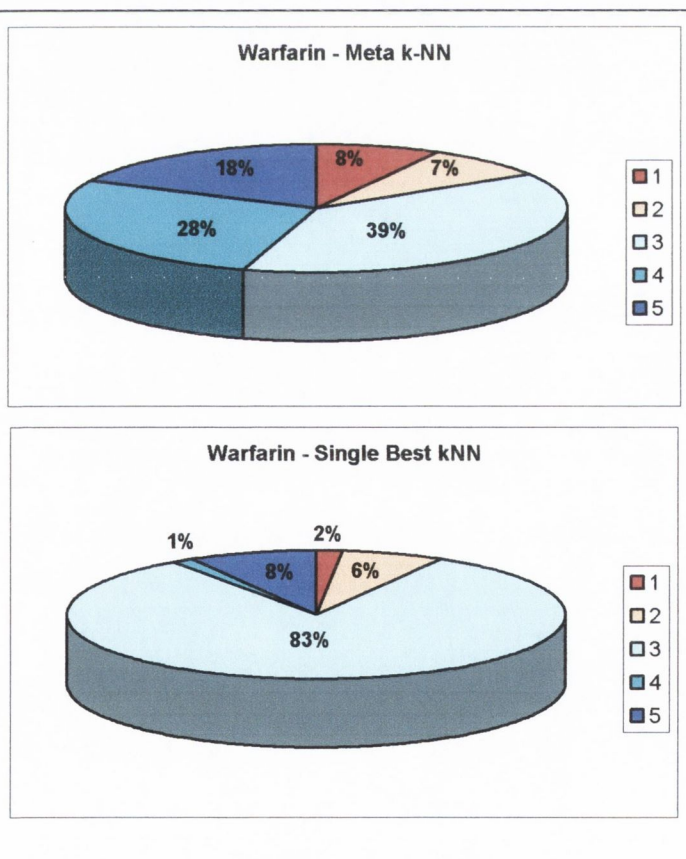


Fig. 6.XIV. Evaluation for the Warfarin dataset

The results are not surprising; the single *k*-NN computes similarity based on a single feature subset, and the similarity is only computed with reference to the feature values, so that given a certain feature subset only the cases that have the same feature subset values will be considered "similar" and the probability to retrieve a case representing the same patient is very high, with a subsequent deficiency of extra information.

In the case of an ensemble aggregated through *Meta k*-NN, the information retrieved is somehow "richer", being the ensemble a pool of experts. The problem space is implicitly represented with a number of sub-domains (each one representing the "expertise" of a single classifier through one feature in the Meta Database) and the risk of getting stuck with neighbours that match only on a certain number of features is much lower. As an effective result there is more variety in the set of cases retrieved for explanation, with a potential information gain (even though there is also a higher risk to retrieve a misleading neighbour) that indicates the *Meta k*-NN strategy as preferable.

As our last observation we just remark that these result stress once again the importance of the ensemble "view" when dealing with weak information domains and, needless to say, they are another "blow" for the information discovery issue applied to a single feature subset selection.

6.5 Summary of the Chapter

In this chapter we have performed a complete evaluation of the *Meta k*-NN aggregation technique; results have shown that it scores accuracies either better or comparable to standard weighted average techniques, provided a sufficiently large value of k_M (number of retrieved nieghbours) is set, with the remarkable advantage that it keeps the CBR characteristic of support for explanation through case retrieval.
In section 6.1 we have shown a comparison of the two aggregation techniques for datasets that present a balanced class distribution. In section 6.2 we have introduced the problem of skewed class distribution and proposed *Duplicate k*-NN, a modification of the standard *k*-NN algorithm designed to deal with minority class retrieval. Also, we have introduced the use of Harmonic Mean Indexes for a better evaluation of performance in the presence of skewed class datasets. In section 6.3 we

have shown a comparison on five skewed class datasets; results largely confirmed the discussion made in 6.1, for datasets with equally distributed classes, when both *Duplicate k*-NN and Harmonic Mean measures are used to perform the comparison. Finally in section 6.4 we have performed an evaluation of the quality of the retrieved cases by *Meta k*-NN in comparison with the ones retrieved by the best single feature subset, with the help of two domain experts.

In the next Chapter 7 we will draw the conclusions of the whole research lead throughout this thesis. Particular emphasis will be given to the conclusions emerging from the evaluation performed in this Chapter and in Chapter 4. Also, we will outline some of the most promising directions for future research arising from this study.

Chapter 7

Conclusions and Future Work

7.1 Thesis Summary and Conclusions

One of the most interesting developments in recent Machine Learning research is the emphasis on interpretability and explanation. The increased interest in Data Mining, where the attention is paid as much on insight as prediction, may be one of the reasons that has focused research this direction (Han & Kamber, 2001). Case-Based Reasoning is a learning paradigm that certainly provides a big support to interpretability through the powerful characteristics of case ranking and retrieval, typical of its nature.

On another side, Ensembles have recently had a big impact on Machine Learning research because they bring significant improvements in accuracy and stability, both to eager and lazy learning systems. If the use of Ensembles may provide Case-Based Reasoning with these benefits, it also reduces its interpretability characteristics; this happens because ensembles introduce an extra layer of complexity that makes explanation much more difficult.

In this thesis, after an introduction on Case-Based Reasoning and the related Feature Selection problem (Chapters 2 and 3), we have focused on ensembles of k-Nearest Neighbours classifiers that are trained over different feature subsets, particularly in relation to the interpretability and explanation issues. The thesis has presented contributions to CBR and ensemble research in a few directions.

The main contribution of this thesis is the proposal of a new aggregation technique, *Meta k*-NN, that allows ensembles to support explanation through coherent case ranking and retrieval. This technique aggregates the Ensemble by building two

layers of Nearest Neighbours classifiers; on the first layer a number of single *k*-Nearest Neighbours classifiers are trained with diversity in the feature subsets, on the second layer a (*Meta*) *k*-Nearest Neighbour classifier performs case retrieval on a different representation of the data, where to each feature is assigned an ensemble member prediction.

The proposed technique obtains good results, either better or comparable to the standard averaging approach to aggregation, in terms of generalization error and allows the identification of a small set of cases for use in explanation. The minimum number of retrieved cases (k_M) to provide an accuracy from the ensemble that is at least as good as a standard weighted average technique varies depending on the database. It is usually a number between 5 and 13, but sometimes a relatively larger value (around 25) is needed. This last situation doesn't represent a problem anyway as the set of cases are ranked and the top ranking cases can be used for explanation. Furthermore, the high "fidelity" between results obtained with the *Meta k*-NN aggregation strategy and the standard weighted average one allows the *Meta k*-NN process to be used in different modes:

a) when the user feels the minimum value of k_M is acceptably small for his goals, *Meta k*-NN can be used *both* to produce the prediction *and* to identify cases for use in explanation. This would be appropriate even in the case when a large k_M can be tolerated for a small number of cases, as we have shown in Chapter 6, *Meta k*-NN needs to use the largest k_M only for a small fraction of the database (between 1 and 2.5%);

b) when the user feels the minimum value of k_M would be too large for his goals, even for a small number of cases, then the standard weighted average aggregation technique can be used to produce predictions and *Meta k*-NN can be then used to identify cases for explanation; this would be allowed by the high fidelity between the two aggregation techniques. This would also be appropriate in the case the accuracy of *Meta k*-NN happened to be poorer than the standard aggregation.

The *Meta k*-NN technique also proved to be very satisfactory concerning the "quality" of the retrieved cases, as it emerged from the domain experts evaluation. It outperformed the best single classifier found, so far the only architecture capable of

retrieving cases in a coherent way, especially with reference to the variety of proposed solutions.

A related contribution was also given to Performance and Evaluation in the case of databases with very unbalanced class distribution. In this thesis we have introduced the *Duplicate k-Nearest Neighbours* technique, a modification of the standard *k*-NN algorithm, that addresses the skewed class problem by creating duplicates of the retrieved cases that belong to a 'minority class', a number of times which is directly proportional to the class unbalance. We have also argued that the Harmonic Mean of the single class Accuracies is a more suitable evaluation measure than the simple overall Accuracy, as it gives a high penalty to performances that are biased toward one class. When using the Harmonic Mean of the accuracies the *Meta k*-NN based on the *Duplicate k*-NN paradigm outperforms *Meta k*-NN based on standard k-NN, and confirmed to be either better or comparable to the Standard Weighted Average. The *Duplicate k*-NN algorithm can obviously be applied also to single classifiers; however in this research we have only evaluated it in the context of ensembles of *k*-Nearest Neighbours.

A second major contribution to ensemble research was given through the proposal of a Hill-Climbing algorithm, named *AmbHC*, to train ensembles of classifiers with a high degree of Diversity (Ambiguity); the results obtained have also raised some considerations about case-representation with feature vectors.

The *AmbHC* algorithm performs at each step an evaluation of both the Error of each classifier and its degree of Ambiguity in comparison to the rest of the ensemble; only changes that lead to a state with a relatively small Error coupled with a relatively good degree of Diversity are accepted. The evaluation showed that ensembles trained with the *AmbHC* algorithm have lower generalization Errors than those trained with a simple Hill Climbing Algorithm provided the size of the ensemble is sufficiently large, even though the single ensemble members score higher errors. All the evaluations of the *Meta k*-NN aggregation technique mentioned above were performed using ensembles trained with *AmbHC*.

There are two main issues emerging from the results obtained with Ensembles. The first one is the fact an ensemble of classifiers that differ on the feature subsets is able to obtain a better performance than any possible single classifier; this poses serious questions about the feature vector representation of cases. A related evaluation

was also shown in Chapter 3 when a high potential for instability was stressed for the feature selection paradigm. The second aspect is the fact that ensemble members produced using ambiguity have fewer features on average than the ones produced without the use of ambiguity.

These issues support the conjecture that ensembles are a set of local specialists who perform a problem space decomposition.

7.2 Suggestions for Future Work

The results provided clearly show that aggregating a set of *k*-Nearest Neighbours to form an ensemble through the *Meta k*-NN technique make them more accurate, more stable and keep the desirable property of explanation support through case ranking and retrieval.

These results have also suggested some insights for interesting future research developments; here below we outline some of the most promising ones, apart from the fact that an obvious extension could be to perform a *Meta k*-NN analysis on some other datasets, especially in relation to a regression problem that has not been analysed in this thesis.

7.2.1 Research Suggestions to develop the *Meta k*-NN approach

A first interesting development would be to perform a feature selection (or a feature weighting) to the *Meta k*-NN representation and see if there is any improvement both to the performance and to the goodness of retrieved cases. The goal would be to discover if some of the ensemble members give a higher contribution to the classification problem than the others. As in the Meta Database each feature represents a single classifier (its predictions), this feature selection/weighting approach can be seen, in a more traditional Weighted Average perspective, as the analogue of assigning binary/continuous different weights to the classifiers in the weighted average computation.

Another interesting development would be to examine more in detail what are the cases that are differently classified by the *Meta k*-NN and the Weighted Average aggregation strategies (the cases than are not included in the "fidelity" set), to see if they are in any way related to the status of "noisy cases", both in the case of equally

distributed classes and skewed class distribution. Following the same line of research, it would be interesting to perform a similar type of analysis, this time identifying the cases that are differently classified by the *Meta k*-NN when the value of k_M is increased, and see how they relate to noise.

7.2.2 Research Suggestions to develop Ensemble analysis

Following the conjecture that considers an Ensemble as a set of local specialists, outlined at the end of Chapter 4, an interesting line of research for future work would be to discover a problem space decomposition through the use of an ensemble of lazy learners, and see if it would allow for a hierarchical CBR solution with higher performance than a single CBR system. A potential mean to do it would be to use clustering and see if meaningful regions of the problem space can be identified in the areas where ensemble members specialise.

A very interesting development would also be to examine how the potential space decomposition would relate to the *Meta k*-NN problem representation.

7.2.3 Research Suggestions for the Feature Relevance Analysis

The use of an ensemble of classifiers differing on the feature subsets has posed serious questions about the feature vector representation of single cases (Chapter 4). When the similarity of two cases is computed by the *Meta k*-NN, the features represent the single classifiers' predictions; since every classifier differs in the chosen feature subset, the relationship between the query and the returned case is not so clearly expressed anymore in the domain of the original feature set (local relevance).

Thus, when a case x is returned to the user in reply to a query q it can be interpreted in terms of the transformed feature set (i.e. *"q and x are similar because Classifiers 1, 4, 5 and 6 see them as similar"*). Nevertheless query and nearest neighbour may also show a similarity in terms of the original feature set (i.e. *"q and x are more similar on features 1, 3, 7, 8 than on features 2, 4, 5, 6"*); it can be reasonably argued that for the *local* region where the retrieved neighbour x belongs to, those matching features may be the predictive ones.

Following this perspective, it would be interesting to perform a statistical analysis of the feature subsets selected by the Ensemble members representing the most similar features in the *Meta k*-NN representation (in the previous example the feature subsets chosen by Classifiers 1, 4, 5 and 6) and see how they relate to the set of most similar

features in the Original representation (in the previous example features 1, 3, 7 and 8). The goal of such an analysis would be to discover a potential relationship between the two case representations (the Original and the *Meta* *k*-NN ones) and offer some insights for the Feature Relevance issue.

Bibliography

Aamodt, A. (1994). Explanation-driven case-based reasoning, in: *S. Wess, K. Althoff, M. Richter (eds.):Topics in Case-based reasoning.* Springer Verlag, 1994, pp 274-288.

Aamodt, A (1990). Knowledge intensive case-based reasoning and sustained learning. In: ECAI-90; *Proceedings of the ninth European Conference on Artificial Intelligence,* Stockholm, August, 1990

Aha, D.W. (1998a), "The Omnipresence of Case-Based Reasoning in Science and Application", *Knowledge-Based Systems, Vol. 11 (5-6),* pp. 261-273.

Aha, D.W. (1998b) Feature weighting for lazy learning algorithms. In: *H.Liu and H. Motoda (Eds.) Feature Extraction, Construction and Selection: A Data Mining Perspective.* Norwell MA: Kluwer.

Aha D.W. (1997). Lazy Learning. Kluwer, Norwell, MA.

Aha, D.W. (1992). Tolerating noisy, irrelevant and novel attributes in Instance-Based learning algorithms. *International Journal of Man-Machine Studies 36,* pp. 267-287.

Aha, D. W., Bankert, R. L. (1994). Feature selection for case-based classification of cloud types: An empirical comparison. *In D. W. Aha (Ed.) Case-Based Reasoning: Papers from the 1994 Workshop (Technical Report WS-94-01).* Menlo Park, CA: AAAI Press.

Aha, D. W., Bankert, R. L. (1995), A comparative evaluation of sequential feature selection algorithms, in D. Fisher & H. Lenz, eds, Proceedings of the Fifth

International Workshop on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, pp. 1—7

Aha, D.W., Goldstone, R.L. (1992). Concept learning and flexible weighting. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society,* Indiana University, Bloomington, IN. Lawrence Erlbaum Associates. 534-539.

Aha D.W., Kibler D., Albert M. (1991). Instance based learning algorithm. *Machine Learning* 6, 37-66.

Alpaydin, E. (1997) Voting over Multiple Condensed Nearest Neighbors. in *Lazy Learning*, (D. Aha, ed.), Kluwer, 115-132.

Armengol, E., Palaudàries, A., Plaza, E., (2001). Individual Prognosis of Diabetes Long-term Risks: A CBR Approach. *Methods of Information in Medicine. Special issue on prognostic models in Medicine.* vol. 40, pp. 46-51.

Atkenson, C.G., Moore, A.W., Shaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review.* 11:11-73.

Bartsch-Sporl, B., Lenz, M., Hubner, A. (1999). Case-Based Reasoning – Survey and Future Directions. In *Lecture Notes in Artificial Intelligence,* vol. 1570, Springler-Verlag, Berlin, Heidelberg, pp.67-89.

Bay, S. (1998). Combining Nearest Neighbour Classifiers Through Multiple Feature Subsets. *Proceeding of the 17th Intl. Conf. on Machine Learning*, pp. 37--45, Madison, WI, 1998.

Bergmann, R., Pews, G., and Wilke, W (1994). Explanation-based similarity: A unifying approach for integrating domain knowledge into case-based reasoning for Diagnosis and Planning tasks. *In S. Wess, K.-D. Althoff, and M.M. Richter, editors, Topics in Case-Based Reasoning, volume 837 of Lecture Notes on Artificial Intelligence,* pp. 182--196. Springer, 1994.

Boese, K.D., Kahng, A.B. (1994) "Best-So-Far vs. Where-You-Are: Implications for Optimal Finite-Time Annealing", *Systems and Control Letters, 22(1), Jan. 1994*, pp. 71-8.

Boland, P. J. (1989) Majority Systems and the Condorcet Jury Theorem. In *The Statistician n. 38.* Pp. 181-189.

Breiman, L., (1996) Bagging predictors. *Machine Learning,* 24:123-140.

Breslow, L., Aha, D.W. (1997). NaCoDAE: Navy Conversational Decision Aids Environment *(Technical Report AIC-97-018). Washington, DC: Naval Research Laboratory,* Navy Center for Applied Research in Artificail Intelligence.

Byrne, S., Cunningham, P., Barry, A., Graham, I., Delaney T., Corrigan, O.I., (2000) Using Neural Nets for Decision Support in Prescription and Outcome Prediction in Anticoagulation Drug Therapy, N. Lavrac, S. Miksch (eds.): *The Fifth Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2000).*

Cardie, C., Howe, N. (1997)Improving Minority Class Prediction using Case-Specific Feature Weights. In *Proceedings of the 14th International Conference on Machine Learning.* Morgan Kaufmann, pp. 57-65.

Caruana, R., and Freitag, D. (1994). Greedy attribute selection. *In Proceedings of ML-94.* Morgan Kaufmann.

Chenoweth, T., Obradovic, Z. (1995) An Explicit Feature Selection Strategy for Predictive Models of the S&P 500 Index. *Stuffing Mind into Computer: Knowledge and Learning for Intelligent Systems. Informatica* 19:4 (501-511) Nov. 1995

Cherkauer, K.J. (1996). Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In P. Chan, editor, Working Notes of the AAAI Workshop on Integrating Multiple Learned Models, pages 15--21, 1996.

Condorcet, Marquis J. A. (1781) Sur les elections par scrutiny, *Histoire de l'Academie Royale des Sciences,* 31-34.

Cover, T.M. (1968) Estimation by the nearest neighbor rule. *IEEE Trans. Inrom. Theory,* vol. IT-14, pp. 50-55.

Cover, T.M., Hart, P.E. (1967). Nearest Neighbor pattern classification. *IEEE Trans. Inform. Theory,* vol. IT-13, pp. 21-27.

Cunningham, P., Carney, J., (2000) Diversity versus Quality in Classification Ensembles based on Feature Selection, *11th European Conference on Machine Learning (ECML 2000)*, Lecture Notes in Artificial Intelligence, R. López de Mántaras and E. Plaza, (eds) pp109-116, Springer Verlag.

Cunningham, P., Doyle, D., Loughrey, J. (2003). An Evaluation of the Usefulness of Case-Based Explanation. To be presented at The *5th International Conference on Case-Based Reasoning. Trondheim,* Norway, June 2003. Available at http://www.cs.tcd.ie/publications/tech-reports/reports.03/TCD-CS-2003-04.pdf

Cunningham, P., Zenobi, G., (2001) Case Representation Issues for Case-Based Reasoning from Ensemble Research, in *Proceedings of 4th International Conference on Case-Based Reasoning* eds D. W. Aha, I. Watson, LNAI 2080, pp146-157, Springer Verlag.

Dietterich, T. (2000a) Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, Cagliari, Italy, 2000.

Dietterich , T.G. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning,* 40(2), 139-158, 2000.

Dietterich, T. (1997) Machine Learning Research: Four Current Directions. In *AI Magazine*, 18(4):97--136, 1997.

Dietterich, T.G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2, pp. 263-286.

Doak J (1992). – An Evaluation of Feature Selection Methods and their Application to Computer Security. *University of California at Davis.* Technical Report CSE-92-18, Davis, CA.

Domingos, P. (2000). A Unified Bias-Variance decomposition for Zero-One and Squared Loss. In *Proceedings of the 17 th National Conference on Artificial Intelligence*, Austin, TX, 2000. AAAI Press.

Domingos, P. (1997). Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review* 11:227-253.

Duda, R.O., Hart, P.E. (1973). Pattern Classification and Scene Analysis. *New York: John Wiley & Sons.*

Dunne, K., Cunningham, P, Azuaje, F. (2002). Solutions to Instability Problems with Sequential Wrapper-based Approaches to Feature Selection. *Technical Report no. TCD-CS-2002-28. Trinity College Dublin.* Available at http://www.cs.tcd.ie/publications/tech-reports/reports.02/TCD-CS-2002-30.pdf

Freund, Y., Schapire, R.E. (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences,* 55(1):119--139, August 1997.

Freund, Y. & Schapire, R.E. (1996). Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning,* pp. 148-146. Morgan Kaufmann.

Freund, Y., Mansour, Y.,& Schapire, R.E. (2001). Why Averaging Classifiers can protect against Overfitting. In *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics.*

Friedman, J. H. (1996) On bias, variance, 0/1-loss and the course-of-dimensionality. *Tech. Rep., Department of Statistics and Stanford Linear Accelerator Center,* Stanford University.

Friedman, J. H. (1994). Flexible metric nearest neighbor classification. *Technical Report 113,* Stanford University Statistics Department.

Friedman, J., Hastie, T. and Tibshirani, R. (1998). Additive logistic regression: a statistical view of boosting. *Tech. Rep., Department of Statistics, Sequoia Hall, Stanford University,* California 94305.

Gates, G.W. (1972). The reduced nearest neighbor Rule. *IEEE Trans. Inform. Theory* (Corresp.), vol. IT-18, pp. 431-433.

Gierl, L., Bull, M., Schmidt, R. (1998). CBR in Medicine. *Case-Based Reasoning Technology – From Foundations to Applications*, chapter 11, pp. 273-298.

Gierl, L., Stengel-Rutkowski, S. (1994). Integrating consultation and semi-automatic knowledge acquisition in a prototype-based architecture: Experiences with dysmorphic syndromes. *Artificial Intelligence in Medicine,* 6:29-49.

Guerra-Salcedo, C., Whitley, D., (1999a). Genetic Approach for Feature Selection for Ensemble Creation. *in GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (eds.). Orlando, Florida USA, pp236-243, San Francisco, CA: Morgan Kaufmann.

Guerra-Salcedo, C., Whitley, D., (1999b). Feature Selection Mechanisms for Ensemble Creation: A Genetic Search Perspective, in *Data Mining with Evolutionary Algorithms: Research Directions. Papers from the AAAI Workshop.* Alex A. Freitas (Ed.) Technical Report WS-99-06. AAAI Press, 1999.

Han, J., Kamber, M. (2001). Data Mining: concepts and Techniques. Morgan Kaufmann.

Hansen, J.V. (2000) Combining Predictors: Meta Machine Learning Methods and Bias/Variance & Ambiguity Decomposition. *PhD dissertation. Department of Computer Science, University of Aarhus, Denmark.*

Hansen, J.V (2000)., Heskes, T. General Bias/Variance decomposition with Target Independent Variance of Error Functions derived from the Exponential Family of Distributions. In *proceedings of the 15th International Conference on pattern recognition, 2000.*

Hansen, J.V (2000)., Krogh, A. A General Method for Combining Predictors Tested on Protein Secondary Structure Prediction. In *Proceedings of Artificial Neural Networks in Medicine and Biology* (Goteborg, Sweden), H. Malmgren, M.Borga, and L. Niklasson, Eds. Springler-Verlag, London. Pp. 259-264.

Hansen, L.K., Salamon, P., (1990) Neural Network Ensembles, *IEEE Pattern Analysis and Machine Intelligence*, 1990. 12, 10, 993-1001.

Hart, P.E. (1968). The Condensed Nearest Neighbor Rule. *IEEE Trans. Inform. Theory* (Corresp.), vol. IT-14, pp. 515-516.

Hastie, T.J., Tibshirani, R.J. (1996). Discriminant adaptive nearest neighbor classification. *IEEE PAMI*, 18, 607-616.

Heider, R. (1996). Troubleshooting CFM 56-3 Engines for Boeing 737 Using CBR and Data Mining. *In Smith and Faltings*, pp. 512-518.

Heskes, T.M. (1998a). Selecting weighting factors in logarithmic opinion pools. *Advances in Neural Information Processing Systems, 10*, 266-272. M.I. Jordan, M.J. Kearns and S.A. Solla, Eds., vol.10, The MIT Press.

Heskes, T.M. (1998b). Bias/Variance decompositions for likelihood-based estimators. *Neural Computation 10, 6, 1425-1433.*

Ho, T.K., (1998a) The Random Subspace Method for Constructing Decision Forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, 8, 832-844.

Ho, T.K., (1998b) Nearest Neighbours in Random Subspaces, *Proc. Of 2nd International Workshop on Statistical Techniques in Pattern Recognition,* A. Amin, D. Dori, P. Puil, H. Freeman, (eds.) pp640-648, Springer Verlag LNCS 1451.

Howe, N., Cardie, C. (1997). Examining Locally Varying Weights for Nearest Neighbor Algorithms. In *Proceedings of the Second International Conference on Case-Based Reasoning,* Brown Univeristy, Providence, RI.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J., & Hinton, G.E., (1991) Adaptive mixtures of local experts, *Neural Computation*, 3, 79-98.

James, G, Hastie, T.(1996) Generalizations of the bias/variance decomposition for prediction error. *Tech. Rep., Dept. of Statistics, Stanford University.* Link: http://www-stat.stanford.edu/gareth/ftp/papers/bv.ps.

Jarmulak, J., Craw, S. "Genetic Algorithms for Feature Selection and Weighting", *IJCAI-99 Workshop on Automating the Construction of Case-Based Reasoners*, Stockholm, Sweden, 2 August 1999

Kira, K., Rendell, L.A. (1992). A practical approach to feature selection. In *Proceedings of the 9th International Conference on Machine Learning,* pp. 249-256. Aberdeen, Scotland: Morgan Kaufmann.

Kohavi, R., John, G.H., (1998) The Wrapper Approach, in *Feature Selection for Knowledge Discovery and Data Mining*, H. Liu & H. Motoda (eds.), Kluwer Academic Publishers, pp33-50.

Kohavi, P. Langley, Y. Yun, (1997) The Utility of Feature Weighting in NearestNeighbor Algorithms, *European Conference on Machine Learning, ECML'97*, Prague, Czech Republic, 1997, poster.

Kohavi, R., Sommerfield, D. (1995). Feature subset selection using the wrapper method: Overfitting and dynamic search space. pages 192--197. AAAI Press.

Kohavi,R., Wolpert, D.H. (1996). Bias plus Variance Decomposition for Zero-One Loss Functions. In *Proceedings of the 13th International Conference in Machine Learning.*

Kolen, J.F., Pollack, J.B. (1991). Back propagation is sensitive to initial conditions. In *Advances in Neural Information Processing Systems,* Vol. 3, pp. 860-867. San Francisco, CA. Morgan Kaufmann.

Kolodner, J. Case-Based Reasoning, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

Krogh, A., Vedelsby, J., (1995) Neural Network Ensembles, Cross Validation and Active Learning, in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretsky, T. K. Leen, eds., pp231-238, MIT Press, Cambridge MA.

Kwok, S.W., Carter, C. (1990). Multiple decision trees. In Schachter, R.D., Levitt, T.S., Kannal, L.N., & Lemmer, J.F. (eds.), *Uncertainty In Artificial Intelligence 4*, pp.327-335. Elsevier Science, Amsterdam.

Lawrence, S., Burns, I., Back, A., Tsoi, A.C., Giles, C.L., (1998). Neural network classification and Prior Class Probabilities. In *Trick of the trade, Lecture Notes in Computer Science State-of-theArt Surveys,* edited by G. Orr and R. Caruana, Springler Verlag, pp. 299-314.

Leake, D.B., (1996) CBR in Context: The Present and Future, in Leake, D.B. (ed) Case-Based Reasoning: Experiences, Lessons and Future Directions, MIT Press.

Leake, D.B. (1995). Adaptive similarity assessment for case-based explanation. *International Journal of Expert Systems*, 8 (2), 165--194.

Leake, D.B. (1994). Issues in goal-driven explanation. In desJardins, M. and Ram, A., editors, *Working notes of the AAAI Spring Symposium in goal driven learning*, pages 72—79.

Lenz, M. (1994). Case-Based Reasoning for holiday planning. In Schertler, W., Schmid, B., Tjoa, A.M., Werthner, H. editors, *Information and Communications Technologies in Tourism,* pp.126-132. Springler Verlag.

Lenz, M., Auriol, E., Manago, M. (1998). Diagnosis and Decision Support. *Case-Based Reasoning Technology – From Foundations to Applications*, chapter 3, pp. 51-90.

Liu Y., Yao X. (1999) Ensemble learning via negative correlation, *Neural Networks* 12, 1999.

Marling, C.R., Whitehouse, P.J., Fioritto, P.A., & Bendis, J.E. (1999). Knowledge Sharing and Case-Based Reasoning in Geriatric Care. *In Aha & Muñoz-Avila, 1999.*

McSherry, D. (2001). Interactive Case-Based Reasoning in Sequential Diagnosis. *Applied Intelligence* 14, pp. 65-76.

Mitchell T. (1997). Machine Learning. McGraw Hills.

Mitchell, T., Keller, R., Kedar-Cabelli, S. (1986). Explanation-Based Generalization: A Unifying View. *Machine Learning*, vol. 1, issue 1, Jan. 1986.

Neal, R. (1993). Probabilistic inference using Markov chain Monte Carlo methods. *Tech. Rep. CRG-TR-93-1, Department of Computer Science, University of Toronto*, Toronto, CA.

Newell, A., & Simon, H.A., (1976) Computer Science as Empirical Enquiry: Symbols and Search. *Communications of ACM*, 19(3), 1976, pp.113-126.

Nitzan, S.I., Paroush, J., (1985) *Collective Decision Making*. Cambridge: Cambridge University Press.

Ong, L.S., Sheperd, B., Tong, L.C., Seow-Choen, F., Ho, Y.H., Tong, L.C., Ho Y.S, Tan, K. (1997) The Colorectal Cancer Recurrence Support (CARES) System. *Artificial Intelligence in Medicine* 11(3): 175-188.

Opitz D., Shavlik J., (1996) Generating Accurate and diverse members of a Neural Network Ensemble, *Advances in Neural Information Processing Systems*, pp. 535-543, Denver, CO. MIT Press. 1996.

Parmanto B., Munro P.W., Doyle H.R. (1996). Improving committee diagnosis with resampling techniques. In Touretzky D.S., Mozer M.C. & Hesselmo M.E. (Eds.), *Advances in Neural Information Processing Systems*, Vol.8, pp. 882-888. Cambridge, MA. MIT Press.

Pews, G., Wess, S. (1993). Combining Case-Based and Model-Based Approaches for Diagnostic Applications in Technical Domains, in *Proceedings of EWCBR 93.*

Provost, F., Fawcett, T. (1998). Robust classification systems for imprecise environments. In *Proceedings of AAAI-98.* AAAI press.

Provost, F., Fawcett, T. (1997). Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97).* Huntington Beach, CA, 1997. pp. 43-48. AAAI Press.

Provost, F., Fawcett, T., Kohavi, R. (1998). The Case against Accuracy Estimation for Comparing Induction Algorithms. In *Proceedings of the 15th International Conference on Machine Learning (IMLC-98),* Madison, WI, 1998.

Ram, A. (1993). Indexing, Elaboration and Refinement: Incremental Learning of Explanatory Cases. *Machine Learning,* Vol. 10, No. 3, pp. 201--248, 1993.

Ramberg, R. (1996). Constructing and Testing Explanations in a Complex Domain. In *Computers in Human Behaviour,* Vol. 12, No 1, pp. 29-48.

Raviv, Y., Intrator, N. (1996). Bootstrapping with noise: An effective regularization technique. *Connection Science, 8* (3-4), 355-372.

Ricci, F., Aha, D. W. (1998). Error-correcting output codes for local learners. *Proceedings of the Tenth European Conference on Machine Learning* (280-291). Chemnitz, Germany: Springer.

Ricci, F., Aha, D. W. (1997). Extending local learners with error-correcting output codes. *Tech. Rep., Naval Center for Applied Research in Artificial Intelligence*, Washington, D.C.

Richter, M. M. (1998). Introduction (to Case-Based Reasoning). in *Case-based reasoning technology: from foundations to applications,* Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D. & Wess, S. (eds.) (1998). Springer-Verlag, LNAI 1400, pp1-16.

Richter, M. M.(1991). Similarity, Uncertainty and Case-Based Reasoning in PATDEX, *Automated Reasoning: Essays in honor of Woody Bledsoe (ed. R. S. Boyer),* Kluwer Acad. Publ., pp. 249-265

Salzberg, S.L. (1991). A Nearest Hyper-rectangle learning Method. *Machine Learning* 6, pp. 251-276.

Salzberg, S., Delcher, A., Heath, D., and Kasif, S (1995). Best-case results for nearest neighbor learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):599--610, 1995.

Schapire, R.E. (1997). Using output codes to boost multiclass learning problems. In *Proceedings of the 14th International Conference on Machine Learning,* pp. 313-321. San Francisco, CA. Morgan Kaufmann.

Skalak, D. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the Eleventh International Machine Learning Conference* (pp. 293--301). New Brunswick, NJ: Morgan Kaufmann.

Smyth B., Cunningham P., (1992) Déjà Vu: A Hierarchical Case-Based Reasoning System for Software Design, in Proceedings of *European Conference on Artificial Intelligence*, ed. Bernd Neumann, John Wiley, pp587-589, Vienna Austria, August 1992.

Smyth, B., Keane, M., Cunningham, P., (2000) Hierarchical Case-Based Reasoning: Integrating Case-Based and Decompositional Problem-Solving Techniques for Plant-Control Software Design, *to appear in IEEE Transactions on Knowledge and Data Engineering.*

Smyth, B., McClave, P. (2001) "Similarity vs. Diversity", *Proceedings of the 4th International Conference on Case-Based Reasoning.* Vancouver, Canada.

Sycara, K., Guttal, R. Koning, J. Narasimhan, S., Navinchandra, D. (1992) CADET: A Case-Based Synthesis tool for Engineering Design. *International Journal of Expert Systems,* 4(2), 157-188.

Sutter J., Dixon S., Jurs P. (1995) – Automated Descriptor Selection for Quantitative Structure – Activity Relationship Using Generalized Simulated Annealing. *J. Chem. Inf. Comput. Sci.* 35, pp. 77-84.

Tibshirani, R. (1996) Bias, Variance and prediction error for classification rules. *Technical Report, Statistics Department,* University of Toronto.

Tomek, I. (1976). Two modifications of CNN. *IEEE Trans. Syst. Mach. Cybern.,* vol. SMC-6, pp. 769-772.

Tumer, K., and Ghosh, J., (1996) Error Correlation and Error Reduction in Ensemble Classifiers, *Connection Science, Special issue on combining artificial neural networks: ensemble approaches*, Vol. 8, No. 3 & 4, pp 385-404.

Van de Laar, P., Heskes, T., (2000) Input selection based on an ensemble, *Neurocomputing*, 34:227-238.

Yang, J., Honavar, V. (1997). Feature subset selection using a genetic algorithm. In *Proceedings of the Genetic Programming Conference*, GP'97, pages 380--385, Stanford University, CA, 1997.

Wettschereck, D., Aha, D.W. (Eds.) (1997). ECML-97 MLNet Workshop Notes: Case-Based Learning: Beyond Classification of Feature Vectors *(Technical Report AIC-97-005). Washington, DC: Naval Research Laboratory,* Navy Center for Applied Research in Artificial Intelligence.

Wettschereck, D., Aha, D.W., Mohri, T. (1997). A review and comparative evaluation of feature weighting methods for lazy learning algorithms. *Artificial Intelligence Review, 11*, 273-314.

Wilke, W., Lenz, M., Wess, S. (1998). Intelligent Sales Support with CBR. In *Case-Based Reasoning Technology – From Foundations to Applications,* chapter 4, pp. 91-114.

Wilson, D.R., Martinez, T.R. (1997). Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research* 6. 1-34. AI Access Foundation and Morgan Kauffmann Publishers.

Zenobi, G., Cunningham, P., (2002) An Approach to Aggregating Ensembles of Lazy Learners that Supports Explanation, *5th European Conference on Case-Based Reasoning (ECCBR 2002)*, eds S. Craw & A. Preece, LNAI 2416, pp436-447, Springer Verlag.

Zenobi, G., Cunningham, P., (2001) Using Diversity in Preparing Ensembles of Classifiers Based on Different Feature Subsets to Minimize Generalization Error, *12th European Conference on Machine Learning (ECML 2001),* eds L. De Raedt & P. Flach, LNAI 2167, pp576-587, Springer Verlag.

Appendix A

Datasets used in the Evaluations

NAME OF DATASET	SOURCE	TYPE OF PROBLEM	N. OF FEAT.	N. OF CASES	CLASS IMBALANCE
Abalone	UCI repository	Mapped into Binary Classification	8	4177	[48%; 52%]
Bronchiolites	Unpublished, given by Dr. Paul Walsh	Binary Classification	22	118	[69%; 31%]
Cylinder	UCI repository	Binary Classification	38	539	[58%; 42%]
Heart Disease	UCI repository	Binary Classification	13	303	[54%; 46%]
Horse Colic	UCI repository	Binary Classification	27	368	[65%; 35%]
Ionosphere	UCI repository	Binary Classification	34	351	[35%; 65%]
IVF	Described in Cunningham & Carney (2000)	Binary Classification (Balanced Reduction)	53	614	[52%; 48%]
Pima Indians	UCI repository	Binary Classification	8	768	[65%; 35%]
Spect	UCI repository	Binary Classification	22	267	[21%; 79%]
Warfarin	Described in Byrne et al. (2000)	Mapped into Binary Classification	22	291	[52%; 48%]
Warfarin (3 classes)	Described in Byrne et al. (2000)	Mapped into 3 Classes Classification	22	314	[48%; 45%; 7%]

Appendix B

Meta k-NN: an example of Implementation

In this Appendix we give a brief description of a practical implementation of the algorithms that we have presented in this thesis (*Meta-k NN, AmbHC, Duplicate k-NN*). These algorithms have been embedded in a unique CBR tool, designed to assist medical practitioners for decision support through the all the CBR potentialities; the benefits that CBR may provide to a medical domain have already been widely described.

There are different phases in a Decision Support cycle. The tool was designed to address only some of them; the other specific tasks that are not particularly related to the CBR domain are left as possible implementations for some other components and will not be described here.

The first phase is the task selection: the user specifies which decision (task) he is interested in having a support; he also makes a pre-selection of the features in the case he already knows some of them are not useful for that particular task. This was not included in the CBR tool. A second phase is given by a conversion of the database from the relational architecture to the "flat table" one; this phase also was not addressed by the CBR tool.

After these two preliminary steps are carried out a Case-Base is available in the form of a flat file and, subsequently, a CBR algorithm can start to operate.

B.1 CBR Tool Description

As shown in Chapter 5, there are two distinctive phases for the CBR tool that makes use of Ensembles aggregated through *Meta k*-NN.

A - the Training Phase, which takes place only once in a while, usually only when a considerable number of new cases is added to the database. During this phase, that can be computationally very expensive, the tool selects and trains the ensemble to be used in the prediction phase (B). The number of classifiers is automatically selected by the tool, through a number of tests on the source Database. The classifiers differ on the feature subsets chosen among the whole set of features describing the cases (for example in a medical dataset typically to a case corresponds a patient) and are trained with *AmbHC*. Once the Training is over, the set of classifiers composing the ensemble is stored and will be used everytime the user wants to query the system on a new case (Phase B)). Before proceeding for the Training Phase the tool performs simple tests to evaluate the class imbalance and in case sets the *k*-NN classifiers to the *Duplicate k*-NN variation in order to minimize the impact of skewed class in the error.

B – *the Performing (Prediction) Phase*, which takes place every time the user wants to query the system about the prediction of a field-value or the retrieval of the nearest cases on a new case. The tool retrieves the stored set of classifiers and to perform predictions the trained ensemble uses *Meta-k* NN, that gives extremely good accuracy results and retrieves a small set of neighbours to provide the user with an explanation for the query case.

The CBR tool was written in ANSI/ISO standard C++ and is in the form of a Dynamic Linked Library (DLL) that makes it usable in a very wide variety of platforms. The Dynamic Linked Library is provided with set of methods to perform all the previously described CBR tasks. For portability reasons the set of methods was designed using plane C interface, so that the library can be used both in a C and C++ environment.

The interactions between the CBR component and any other component that will want to use the CBR DLL happens through ASCII text files. In the training phase the CBR component requires a data source provided by the user. The source file

has to follow the structure indicated below and is automatically normalized by the CBR component before the training phase starts. The normalization generates a reference file to the source (locally stored) that will be used in the prediction phase to reconvert the normalized values into their original range. If the source is updated the CBR needs to re-perform a training phase.

B.2 Data Format

B2.1 Training Phase

The source will have to follow the following structure:

a: Rows Order

The first row contains two integers: the number of data rows (the first three rows are not included) and the number of columns. The second row contains all the name of the columns (features). The third row contains a description of data type: either numerical ("N") or symbolic ("S"). The following rows contain the data. The data on every row are separated by the "|" delimiter.

b: Columns Order

The first column is always the *"Patient Id"* column and is not computed during the training but only used as a reference. The last column contains the Task attribute. The remaining columns contain all the attribute descriptions (useful features) and they have to be sorted with the following order: first all the numerical attributes, then all the symbolic attributes (except obviously for the *Patient ID* and Task columns). If the last column (Task) is indicated as "S" then the CBR will perform a classification task, if it is instead indicated as "N" then the CBR will perform a regression task.

c: Missing Values

The PatientID and Task columns must be filled with non-empty fields. If some cases contain empty fields in the Task column they must be pre-processed and discarded before the source is given to the CBR tool, as all the cases must be associated to a task-label (supervised learning) during the training phase. The non-task field can instead be empty (missing values), the CBR tool is designed to deal with them. In this case the empty fields must be filled with the constant *"?"*.

Below is an example of data source:

```
6|15
PatientID|Age|Weight|INR|PrevDose|Duration|INRTarget|Alcohol|INRDelta|NoADR|Gender|CurrentMed|OTC|Compl|SubseqDose
S|N|N|N|N|N|N|N|S|S|S|S|S|S
No.XX6|56|57|2.4|3.5714286|107|3|7|0.6|NoAdverse|F|None|?|NoChange|2->5
No.XX8|51|70.3|1.3|4|160|3.75|8|2.45|NoAdverse|M|MinorInc|None|TooLittle|2->5
No.XX4|58|51.8|2.6|5.8571429|10|2.75|10|0.15|NoAdverse|F|None|None|NoChange|5->10
No.XX2|61|?|3.1|6.7142857|12|2.75|1|-0.35|NoAdverse|F|None|None|NoChange|5->10
No.XX5|52|73.7|1.8|3|13|3.25|8|1.45|NoAdverse|M|None|None|NoChange|2->5
No.XX3|51|72|1.8|5|164|3.75|0|1.95|NoAdverse|M|None|None|NoChange|5->10
```

There are 6 data rows of 15 columns.

The columns are: *PatientID, Age, Weight, INR, PrevDose, Duration, INRTarget, Alcohol, INRDelta, NoADR, Gender, CurrentMed, OTC, Compl, SubseqDose.*

PatientID is symbolic and not computed, all columns from *Age* to *INRDelta* included are numerical, all columns from *NoADR* to *Compl* included are symbolic. *SubseqDose* is the task, it is a symbolic attribute so we are in the presence of a classification problem.The two classes are *"2->5"* and *"5->10"*. Patient No.XX6 has a missing value on the *OTC* field, while Patient No.XX2 has a missing value on the *Weight* field.

B.2.2 Performing (Prediction) Phase

During the Prediction phase the tool is asked to predict the value of the task field on a given query. This will be passed CBR following the same structure of one of the data rows provided during the training phase, except for the unknown last attribute (task). Here is an example of query:

```
No.XX1|47|52.2|2.1|8.5714286|20|3.5|0|1.4|NoAdverse|F|None|None|NoChange
```

B.3 Description of the Methods

B.3.1 Training Phase

Here below is the prototype of the function that computes all the training steps necessary to the CBR system. The user has to provide two strings, through character pointers: the first one is a unique identifier (description) for the process, the second (filename) is the complete pathname where the data source, following the format previously described, is physically stored, for example *"c:\\Sources\\Database1.txt"*.

```
void CBRtrain(char* description, char* filename)
```

Let's see more in detail the `description` string. This label will uniquely identify the particular task that the CBR is training for. Every time the user wants to run any of the Performing Phase Methods on the same task he will have to refer to this description name. It is good practice to give the description field a meaningful name; for example a good name for the task prediction of the Insuline Level on the Database given by Dr. John Murphy could be something like: *JohnMurphy_InsulineLevel*. The user must also keep into account that if there are further updates to the source (in which case the CBR system must be re-trained) any calls to `CBRtrain()` with an already existing name will overwrite the data. In the case the user feels he needs to have the previous version not deleted, it might be useful to include the current date in the description.

The system is designed to make an automated choice of all the parameters defining the ensemble. Such a selection is made also keeping into account the Maximum Amount of Time that the user can approximately wait to have the completion of the CBR training phase; this is obviously only an approximate estimation as it's not possible to know a priori the exact number of cycles the system will need to perform before the algorithm stops.

B.3.1 Prediction Phase

Here below is the function prototype for the main method used in the performing phase, the call for a prediction on a given query:

```
void CBRpredict(char* reply, char* description, char* query)
```

The `description` field is necessary to identify the normalized source and references. The `query` follows the format specified in section 3.2. The `reply` is returned as an array of characters and it is not automatically appended to the source database. Let's spend a few more words on the `reply` field. This is a character pointer that holds the reply given by the CBR system to the query; this pointer has to be allocated in the main program by the user of the CBR tool. This strategy has been chosen to avoid having memory allocation and de-allocation made by two different executable (the user `main()` and the library). The only difficulty that arises with this strategy is that the user has to allocate a sufficient number of characters, otherwise the CBR tool may try to write on a non-allocated memory area. An easy solution to this inconvenient is reached by having a further method that computes automatically and returns the maximum number of characters that the CBR system may need to allocate. The function prototype is illustrated below; the only parameter to be passed is obviously the `description`:

```
int CBRmaxSize(char* description)
```

The structure of the reply to `CBRpredict()` is the following:

Row 1: Predicted Class | Error on Previous Data | Number of retrieved Neighbours

Row 2: Patient ID of Neighbour no.1 | Class of Neighbours no.1 | Matching Percentage

Following Rows: same as Row 2 for each retrieved neighbour

Here below is an example of how the reply to the query given in section 2.3 may look like:

```
2->5|8.5|3
No.XX8|2->5|97.83
No.XX2|5->10|95.24
No.XX5|2->5|92.38
```

In the case of a classification problem the error is simply a 1/0 type of measurement (i.e. *Accuracy = 100-Error* and in the example above the accuracy would be 91.5%), in the case of a regression problem the error is a squared error. The number of retrieved cases varies from query to query; the system keeps retrieving cases until it "feels confident" about the final prediction, through some tests on the *Meta k*-NN parameters that won't be described here.

Here is an example of a piece of C code that uses `CBRpredict()` and `CBRmaxSize()`:

```
#include <CBR>
.....
void main()
{
  .....
  char* description="JohnMurphy_InsulineLevel";
  char* result= (char*)malloc(CBRmaxSize(description)*sizeof(char)); //allocation of memory for the pointer
  char* query="No.1258|47|52.2|2.1|8.5714286|20|3.5|0|1.4|NoAdverse|F|None|None|NoChange";
  CBRpredict(result, description, query);
  printf("\nReply to query %s  is: %s",query, result);
  free(result); //the user frees the memory previously allocated
  .....
}
```