

Departamento de Ciencias de la Computación e Inteligencia Artificial
Konputazio Zientziak eta Adimen Artifiziala Saila
Department of Computer Science and Artificial Intelligence



Universidad del País Vasco
Euskal Herriko Unibertsitatea
University of the Basque Country

Advances in Supervised Classification based on Probabilistic Graphical Models

by

Iñaki Inza

Dissertation submitted to the Department of Computer Science and Artificial Intelligence
of the University of the Basque Country in partial fulfillment of the requirements for the
PhD degree in Computer Science

Donostia - San Sebastián, February 2002

Departamento de Ciencias de la Computación e Inteligencia Artificial
Konputazio Zientziak eta Adimen Artifiziala Saila
Department of Computer Science and Artificial Intelligence



Universidad del País Vasco
Euskal Herriko Unibertsitatea
University of the Basque Country

Advances in Supervised Classification based on Probabilistic Graphical Models

by

Iñaki Inza

Dissertation submitted to the Department of Computer Science and Artificial Intelligence
of the University of the Basque Country in partial fulfillment of the requirements for the
PhD degree in Computer Science

Donostia - San Sebastián, February 2002

Contents

Acknowledgments	ix
List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Overview of the dissertation	2
2 The Supervised Classification Task	5
2.1 Overview of the chapter	5
2.2 Definitions and terminology	5
2.3 Statistic-based paradigms	6
2.4 Second-generation paradigms: Machine Learning	7
2.5 Datasets	10
2.6 Accuracy estimation: evaluating and comparing classification models	14
2.6.1 k -fold cross-validation	15
2.6.2 5 times 2-fold cross-validation	16
2.7 Summary	16
3 Probabilistic Graphical Models	17
3.1 Overview of the chapter	17
3.2 Terminology and basic concepts	17
3.3 Bayesian networks	20
3.3.1 Introduction	20
3.3.2 Basic notation for Bayesian networks	21
3.3.3 Model induction	22
3.3.4 Simulation	27
3.4 Gaussian networks	28
3.4.1 Introduction	28
3.4.2 Basic notation for Gaussian networks	28
3.4.3 Model induction	29
3.4.4 Simulation	31
3.5 Summary	31

4	Estimation of Distribution Algorithms	33
4.1	Overview of the chapter	33
4.2	The need of the EDA paradigm	33
4.3	EDA approaches in discrete domains	35
4.3.1	Without dependencies	37
4.3.2	Bivariate dependencies	39
4.3.3	Multivariate dependencies	43
4.4	EDA approaches in continuous domains	46
4.4.1	Without dependencies	46
4.4.2	Bivariate dependencies	47
4.4.3	Multivariate dependencies	47
4.5	Summary	49
5	Representing the Joint Behaviour of Supervised Learning Algorithms by Bayesian Networks	51
5.1	Overview of the chapter	51
5.2	Motivation: accuracy versus class label	51
5.3	Methodology	52
5.3.1	Learning algorithms and datasets	52
5.3.2	Modelization by Bayesian networks and the simplification process	52
5.3.3	Concepts for interpreting the joint behaviour of classifiers	54
5.4	Results from induced Bayesian networks	56
5.4.1	Hard conditional independence	56
5.4.2	Conditional independence in a proposed family of algorithms	57
5.4.3	Conditional independence between proposed families	58
5.5	Summary and future work	59
6	Feature Subset Selection for Supervised Classification by Estimation of Distribution Algorithms	61
6.1	Overview of the chapter	61
6.2	Motivation	61
6.3	Feature Subset Selection: basic components	62
6.3.1	The starting point in the space	63
6.3.2	The organization of the search	63
6.3.3	Evaluation strategy for feature subsets	64
6.3.4	Criterion for halting the search	66
6.4	FSS by EDA in small and medium scale domains	66
6.4.1	Characteristics of the evaluation function	68
6.4.2	The ‘overfitting’ problem and the relevance of the stopping criteria	68
6.4.3	Experiments in real domains	71
6.4.4	Experiments in artificial domains	74
6.5	FSS by EDA in large scale domains	75
6.5.1	Experiments in real domains	76
6.5.2	Experiments in artificial domains	80
6.6	Summary and future work	81
7	Feature Weighting for Nearest Neighbor Classifier by Estimation of Distribution Algorithms	83

<i>Contents</i>	vii
7.1 Overview of the chapter	83
7.2 Motivation	83
7.3 Review of related works	84
7.4 Learning weights for Nearest Neighbor by Bayesian and Gaussian networks	86
7.5 Experimental comparison	88
7.6 Summary and future work	93
8 Conclusions	95
8.1 Summary of contributions	95
8.2 Future work	97
References	99
Index	111

Acknowledgments

The course of time is the teacher which has shown me all I owe to several persons. All they support me, all I learn with them, all I imitate them, is part of me.

The generosity and faith of Pedro Larrañaga, my thesis advisor, have been the guide of this long trip. He has been an example of work and constancy. It has been a very valuable adventure to share with Pedro many experiences in the field of research.

I remember with affection the altruism of Basilio Sierra: always ready to aid, always ready to listen, always with a smile.

I was very lucky to find Pedro and Basi.

I am grateful to the Department of Education, Universities and Research of the Basque Government for the financial support during these years under the program of Formation of Researchers (grant PI 96/12). I would like to thank Marisa Merino for all the things she taught me about the application of Machine Learning in the field of Medicine. I also remember the persons who have accompanied me during these years at the Computer Science and Artificial Intelligence Department of the University of the Basque Country.

The life has met me with three persons who have taught me many things about it, enjoying many and many rich and sincere conversations that I have maintained with them: Emeterio Abete, Javi Ruiz and Txiki Zapiain.

Finally, all my deepest appreciation and affection to my family: my parents and my sister. I have no words to acknowledge all my parents have done for me: their example, their humility and their generosity have been a faithful and beloved reference. This dissertation is dedicated to them.

List of Figures

3.1	Structure for a probabilistic model defined over $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5, X_6)$.	19
3.2	Checking conditional independencies in a probabilistic graphical model by means of the u -separation criterion for undirected graphs.	19
3.3	Degrees of complexity in the structure of probabilistic graphical models.	20
3.4	Structure, local probabilities and resulting factorization for a Bayesian network with four variables (X_1, X_3 and X_4 with two possible values, and X_2 with three possible values).	21
3.5	The K2 algorithm.	26
3.6	Pseudocode for the PLS method.	27
3.7	Structure, local densities, and resulting factorization for a Gaussian network with four variables.	28
4.1	Main scheme of the EDA approach.	35
4.2	Illustration of the EDA approach to optimization.	36
4.3	Graphical representation of the probability model proposed by EDAs without dependencies.	37
4.4	Pseudocode for the cGA.	40
4.5	The MIMIC approach to estimation of the joint probability distribution at generation l . The symbols $\hat{h}_l(X)$ and $\hat{h}_l(X Y)$ denote the empirical entropy of X and the empirical entropy of X given Y respectively. Both are estimated from D_l^{Se} .	41
4.6	The TREE approach to estimation of the joint probability distribution at generation l .	42
4.7	Graphical representation of the probability models proposed by EDAs with pairwise dependencies.	43
4.8	Graphical representation of the probability models proposed by EDAs with multiple dependencies.	45
4.9	EBNA basic scheme.	45
4.10	EGNA basic scheme.	48

5.1	Proposed modelization process for each dataset. We start from the class predictions of Machine Learning inducers for the test instances. Let assume that N is the number of instances in test set, from which the Bayesian network is induced.	54
5.2	Simplified Bayesian network for the <i>Breast cancer</i> domain.	54
5.3	Simplified Bayesian network for the <i>Cleveland</i> domain.	55
6.1	In this 3-feature (F1,F2,F3) problem, each individual in the space represents a feature subset, a possible solution for the FSS problem. In each individual, a feature's rectangle being filled, indicates that it is included in the subset.	63
6.2	Summarization of the whole FSS process for filter and wrapper approaches.	66
6.3	FSS-EBNA method.	67
6.4	Internal and external loop accuracy values in FSS-EBNA for different training sizes with <i>Waveform-40</i> dataset and NB algorithm. The internal loop accuracy 10-fold cross-validation is multiple times repeated until the standard deviation of the accuracy estimation drops below 1%. Dotted-lines show the internal-loop accuracy estimation and solid-lines the external-loop one. Both loop accuracies for the best solution of each search generation are represented. '0' represents the initial generation of the search.	70
6.5	Relations between relevant concepts to estimate a reliable Bayesian network.	76
7.1	FW-EBNA method.	88

List of Tables

2.1	An overview of the needed dataset for Supervised Classification task.	5
2.2	Details of experimental domains. C: continuous. N: nominal.	11
3.1	Variables (X_i), number of possible values of variables (r_i), set of variable parents of a variable (\mathbf{Pa}_i), number of possible instantiations of the parent variables (q_i).	22
6.1	Details of small and medium dimensionality experimental domains.	71
6.2	Accuracy percentages of the NB classifier on real datasets without feature selection and using the five FSS methods. The last row shows the average accuracy percentages for all six domains.	72
6.3	Cardinalities of finally selected features subsets for the NB classifier on real datasets without feature selection and using the five FSS methods. It must be taken into account that when no FSS is applied to NB, it uses all the features.	73
6.4	Mean stop-generation for GAs and FSS-EBNA. The standard deviation of the mean is also reported. The initial generation is considered to be the zero generation.	74
6.5	Number of generations needed on average (and their standard deviation) by FSS-GA-o, FSS-GA-u and FSS-EBNA to discover the optimum feature subset in artificial domains. The initial generation is considered as generation zero.	75
6.6	Details of large-dimensionality experimental domains.	76
6.7	Accuracy percentages of the NB classifier on real datasets without feature selection and using FSS-GA-o and FSS-GA-u. The last row shows the average accuracy percentages for all six domains.	77
6.8	Accuracy percentages of the NB classifier on real datasets using FSS-PBIL, FSS-BSC, FSS-MIMIC and FSS-TREE. The last row shows the average accuracy percentages for all six domains.	78
6.9	Cardinalities of finally selected feature subsets for the NB classifier on real datasets without feature selection and using FSS-GA-o and FSS-GA-u. It must be taken into account that when no FSS is applied to NB, it uses all the features.	78
6.10	Cardinalities of finally selected features subsets for the NB classifier on real datasets using FSS-PBIL, FSS-BSC, FSS-MIMIC and FSS-TREE.	78

6.11	Mean stop-generation for FSS algorithms. The standard deviation of the mean is reported. The initial generation is considered to be the zero generation.	79
6.12	Average CPU times (in seconds) for the induction of different probabilistic models (standard deviations are nearly zero) in each generation of the EDA search. The last column shows the average CPU time to estimate the predictive accuracy of a feature subset by the NB classifier.	79
6.13	Number of generations needed on average (and their standard deviation) by FSS-GA-o, FSS-GA-u, FSS-PBIL, FSS-BSC, FSS-MIMIC and FSS-TREE to discover a feature subset that equalizes or surpasses the estimated accuracy level of the feature subset which induces the domain. The initial generation is considered to be the zero generation.	80
7.1	Details of experimental domains.	89
7.2	Accuracy percentages of the Nearest Neighbor algorithm using the 5 FW methods shown and without FW. The standard deviation of the estimated percentage is also reported.	90
7.3	Mean stop-generation for FW-GA-o, FW-EBNA and FW-EGNA. The standard deviation of the mean is also reported. The initial generation is considered to be the zero generation.	91
7.4	Average CPU times (in seconds) for the induction of different probabilistic models (standard deviations are nearly zero) in each generation of the EDA search. The last column shows the average CPU time to estimate the predictive accuracy of a feature weight set.	92

Chapter 1

Introduction

Empirical Learning is ‘accomplished by reasoning from externally supplied examples to produce general rules’ (Dietterich and Shavlik, 1990). Closely related to Empirical Learning, the Machine Learning discipline tries to construct computer programs (or learning programs) that, starting from a set of examples, automatically produce the desired general rule. The basic source of this process is the set of examples (or training experience), that is, the input of the Machine Learning program.

An example describes the basic entity of the reality to be studied, a part of a known experience, such as a medical patient, a credit request or a handwritten letter. A feature describes a specific characteristic of an example, such as the output of a medical test, the age of the credit requester or the width of the handwritten letter.

In Supervised Classification, which can be considered as a subfield of Machine Learning, every example has a special feature, known as the class label, which describes the value of the phenomenon of interest, *i.e.*, the diagnostic of the medical patient, the possible fraud of the credit requester or the specific letter of the handwritten character. It can be considered that the class label of each example is provided by a *supervisor* or teacher. Unsupervised Classification (Peña, 2001) deals with the discovering of the underlying group structure when the true class label membership of the supplied examples is unknown.

In Supervised Classification, the task of a learning algorithm is to learn a general rule (or classifier). Starting from the dataset of labelled examples, the class label *supervises* the induction process of the classifier, which is used in a second step to predict, the most accurately as possible, the class label for further unlabelled examples. Thus, a bet is performed, classifying the unlabelled example with the label that the classifier predicts for it.

The following real medical example (Inza et al., 2001d) could aid to understand the role of the Supervised Classification. The problem is tackled with success in conjunction with a group of physicians from the Basque Health Service - Osakidetza and the University Clinic of Navarra. The Transjugular Intrahepatic Portosystemic Shunt (TIPS) is an interventional treatment for cirrhotic patients. In the light of our medical staff’s experience, the consequences of TIPS are not homogeneous for all the patients and a subgroup of them dies in the first months after the TIPS placement.

The training experience of our study arises from a prospective study which results in a set of 107 examples-patients with liver cirrhosis who underwent TIPS from May 1991 to September 1998 in the University Clinic of Navarra. Each example is characterized by 77 features (or medical findings)

that arise from the history of the patient, results of laboratory tests, Doppler sonography, endoscopy, hemodynamic parameters and angiography. Taking into account that the average waiting time on a list for a liver transplant at the University Clinic of Navarra is approximately six months, it was decided that the class label should reflect whether the patient died in the first six months after the placement of the TIPS or not. In this way, based on the input dataset of 107 examples, the output of a supervised learning algorithm is a classifier which predicts the survival within six months after the TIPS placement. This classifier can be used to aid physicians in the decision to choose a cirrhotic patient for liver transplantation.

1.1 Overview of the dissertation

This dissertation is comprised of seven chapters. Chapter 2 formalizes Supervised Classification, introducing its basic concepts and terminology. The algorithms and datasets used in the dissertation are described, coupled with the employed accuracy estimation methods.

Probabilistic graphical models are presented in Chapter 3. Special emphasis is put on Bayesian networks, which are more used in the dissertation than Gaussian networks, the other graphical model used in this work. Special attention is paid to the score+search procedure to induce graphical models, reviewing the principal score metrics and search mechanisms. Other related issues are also described, such as the conditional independence concept and the simulation process. The existing literature dedicated to probabilistic graphical models is reviewed.

Chapter 4 presents the novel EDA (Estimation of Distribution Algorithms) paradigm, closely related with the Probabilistic Graphical Model topic introduced in the previous chapter. The principal motivations that generate the birth of the EDA paradigm are exposed, so related with Genetic Algorithms. Principal approaches for discrete and continuous domains are reviewed, organizing the approaches with respect to the order of the interactions covered among problem variables.

Chapter 5 shows how Bayesian networks can be used to study the nature of the classification models induced by a set of Supervised Classification algorithms. Bayesian networks, and its associated conditional independence concept, make possible to perform this study in a joint manner, based on the class label predicted by the classifiers. The study is performed in a set of eleven medical supervised domains.

Chapter 6 presents and experimentally evaluates the application of the EDA paradigm to solve the well known Feature Subset Selection task in domains of different dimensionalities. Different EDA approaches (from univariate probabilistic models until Bayesian networks) are proposed, depending on the dimensionality of the studied domain. The basic components of the Feature Subset Selection problem are studied and the principal existing literature works are reviewed. The ‘overfitting’ problem is studied in depth, which has a large impact in Feature Subset Selection task with domains of few examples. The capability of EDA inspired approaches is compared with sequential search engines and Genetic Algorithms in a set of natural and artificial domains.

Chapter 7 shows how Bayesian and Gaussian networks are used, within the EDA paradigm, to solve the Feature Weighting task for the Nearest Neighbor supervised algorithm. A large survey of previous approaches in the field is carried out. An empirical evaluation of our proposal is included, performing a comparison with sequential search engines and Genetic Algorithms in a set of natural and artificial domains.

Finally, Chapter 8 lists the main contributions of this dissertation, coupled with future lines of work also exposed in the final part of each chapter.

After the list of all the literature references of the dissertation, an index is included. The index encloses the principal topics of the dissertation. For each concept of the index, the page where it

appears for the first time is referenced. Each concept is also referenced when it actively participates in the explained issue.

Chapter 2

The Supervised Classification Task

2.1 Overview of the chapter

In this chapter the principal components and characteristics of the supervised classification task are formalized. First, the principal motivations for the birth of the Machine Learning paradigm are exposed, coupled with its antecessors: Discriminant Analysis and Logistic Regression. Then, Machine Learning classifiers and datasets used in this dissertation are presented. The chapter is finished with a section about accuracy estimation techniques, where the employed validation techniques and statistical tests are studied more in depth.

2.2 Definitions and terminology

The main task in Supervised Classification is the application of a learning algorithm (or inducer) to obtain a classifier (or classification model) (Kohavi, 1995d). The learning algorithm needs a dataset of labelled N examples (or instances), $E = \{\mathbf{e}_1, \dots, \mathbf{e}_N\}$, each one characterized by n descriptive features (attributes or variables), $\mathbf{X} = \{X_1, \dots, X_n\}$, and the class label, $C = \{c_1, \dots, c_N\}$, to which they belong, where the class label of each instance is part of a discrete set of R values: $c_j \in \{c^1, \dots, c^R\}$. The learning algorithm uses the set of labelled examples to induce a classifier which will be used to classify unlabelled examples. An overview of the needed dataset of cases can be seen in Table 2.1.

A structured supervised learning algorithm is able to induce a classifier, which can be seen as a function that maps an unlabelled example to a specific class label. Nowadays, this process can be

Table 2.1. An overview of the needed dataset for Supervised Classification task.

	X_1	\dots	X_n	C
\mathbf{e}_1	x_1^1	\dots	x_1^n	c_1
\mathbf{e}_2	x_2^1	\dots	x_2^n	c_2
\dots	\dots	\dots	\dots	\dots
\mathbf{e}_N	x_N^1	\dots	x_N^n	c_N

done automatically by a computer program (Mitchell, 1997). In order to assess the quality of the induced classifier, the following concepts are usually taken into account:

- the accuracy of the classifier assesses its ability to correctly predict the class label of unlabelled examples;
- the computational cost or computer speed of the learning algorithm to induce the classifier;
- the comprehensibility and interpretability of the classifier to humans. This characteristic is highly desired in some domains: for instance, in medical environments, where the reliability of the medical staff with respect to the classifier is judged as crucial;
- the simplicity and compactness of the classifier are characteristics related with the previous one. However, both ideas must not be considered as equal: although a Naive Bayes classifier (see Section 2.4) induced with 3 variables can be compact, its comprehensibility of conditional probabilities and independence concepts could not be comprehensible for certain experts not habitué with probability concepts. In this way, two statistical concepts, the *Occam's Razor* and *KISS* ('Keep It as Simple as possible Stupid') (Kohavi, 1995d), are related with the desired property of the preference for simple classifiers, when other characteristics can not make a decision to select among several classifiers.

The concepts and definitions exposed in this section can be presented and extended in several ways. Depending on the specific classification task, the literature presents other ideas. However, as the exposed concepts could be the core of the Supervised Classification, it will be used for the rest of this dissertation.

2.3 Statistic-based paradigms

Through the history of Supervised Classification, the first supervised learning algorithms were heavily influenced by Statistics. The construction of this family of classifiers is also based on the idea of maximizing the likelihood of the data, given the classifier. Prior hypothesis were taken into account about the data distribution and the free-parameters of the classifier for its construction. Thus, these learning algorithms have a profound theoretic and Statistics basis. Discriminant Analysis and Logistic Regression are the most known examples of this kind of algorithms.

- Discriminant Analysis (Fisher, 1936) constructs a linear combination of the predictive features to induce a classifier which separates between classes. The classifier is constructed under the assumption that each feature follows a normal probability distribution. The induced classifier can be seen as a n -dimensional hyperplane. Although the assumption of normality and the classes of many real problems are not separable by an hyperplane and a non-linear combination of the features is needed, it has demonstrated a good classification accuracy in many tasks. In a 2-class problem, its general formula which discriminates between both classes has the following form:

$$w_1X_1 + w_2X_2 + \dots + w_nX_n - w_0$$

where each constant value w_i assesses the discriminatory power of the i -th feature and w_0 is another constant value. Coupled with this process, a selection of features is performed (Inza et al., 2001b), ejecting a subset of features from the classifier. In a problem with more than two

classes, an hyperplane must be constructed for each class pair or the whole problem must be reformulated as a 2-class problem;

- the Logistic Regression (Hosmer and Lemeshow, 1989) arises as an alternative classifier which is not based on the normality assumption of Discriminant Analysis. Logistic Regression is based on the next formula for a 2-class problem:

$$P(C = c^i | X_1 = x_1, \dots, X_n = x_n) = \frac{1}{1 + e^{-b_0 + \sum_{i=1}^n b_i x_i}}$$

which estimates the probability of a new example belongs to class c^i . Maximum likelihood estimation is used to calculate the b_i constant values. Closely related with Discriminant Analysis, Logistic Regression also uses an hyperplane expression to separate between 2 classes. In the statistical literature several different procedures have been suggested for variable selection in Logistic Regression. The most well known are based on forward selection and backward elimination, together with some kind of decision rule for the number of variables (Urban, 1994).

2.4 Second-generation paradigms: Machine Learning

In the 80s and 90s there has been a big growth in the accumulation of information in Economic, Marketing, Finance, etc. databases. The larger size of this databases inspired a set of techniques that are grouped under the *Machine Learning* term and that discover and extract knowledge in a more automated way than classical Statistic-based paradigms. These techniques use a ‘data-driven’ process in the sense that less emphasis is put on prior hypothesis about data probability distribution and classifier’s free parameters than is the case with classical Statistic-based paradigms. By approaching an analysis as a search for knowledge, rather than to test a hypothesis about the nature of the data, the discovery of previously unknown relationships in the data is the core of the process for the induction of the classifier.

In order to construct a classification model, this kind of paradigms make assumptions, which are called *biases* in Machine Learning. By approaching the analysis as a search for knowledge, one requires to shrink or *prune* the search space. Induction without bias is imposible, it is the *sine que non* condition for induction (Dutton and Conroy, 1996). Apart from the data, biases are the principal builders of the classification models. Mitchell (Mitchell, 1982) resumes the need of biases in Machine Learning in the following form: *Although removing all biases from a generalization system may seem to be a desirable goal, in fact the result is nearly useless... An unbiased system is one whose inferences logically follow from the training instances, whereas classification of new instances do not logically follow from the classification of the training instances.* In this way, the definition of biases of existing algorithms and how to find out when a bias is appropriate are crucial question for Machine Learning researchers. Biases can be divided into two types (Kohavi, 1995d):

- *restricted space bias*: this bias assumes that the classifier belongs to some restricted space of classification models, typically defined in terms of their representation. For example, most decision tree algorithms (see Section 2.4) restrict the space of considered classifiers to the space of finite trees with univariate splits in its nodes, assuming that classes are separated by line segments parallel to the coordinate axes;
- *preference bias*: this bias places a preference ordering on the considered classification models. Many times the preference ordering is defined by how the search through the space of classification models is conducted. Most preference biases attempt to minimize some measure of syntactic complexity, following Occam’s Razor principle of preferring simpler classifiers.

The literature is plenty of algorithms that propose different biases to construct the classifier. Discovering new learning algorithms (or versions thereof) has occupied much of the research of the past decade in Machine Learning with reasonable success. However, much remains to be learned about what makes a particular algorithm work well (or not) in a particular domain.

Regarding this large amount of available algorithms, the user is frequently faced with the problem of selecting the ideal algorithm for a specific classification task, trying to select the learning algorithm which biases are best suited to the data. The ambitious objective of generating and selecting a unique winner algorithm for all datasets has been rejected by the empirical evidence of the ‘No Free Lunch Theorem’ (Kohavi and John, 1997). In the next lines we present the Machine Learning algorithms that are used in this dissertation. Although several details are left for future chapters, more emphasis will be put on the algorithms that will be more frequently used in this dissertation. The basic lines of the principal families of algorithms will also be explained. As a depth study about these algorithms is out of the objectives of this dissertation, other classic works in Machine Learning are recommended (Mitchell, 1997) (Michalski et al., 1998). This is the list of the algorithms:

- ID3 (Quinlan, 1986) classification tree algorithm. Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node of the tree performs a test over the value of a specific attribute. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, until a leaf node is reached. The gain-ratio measure, inspired in the Information Theory of Shannon (Shannon, 1948), is used to select the attribute which performs the split in each node of the tree. It does not incorporate a post-pruning strategy in the construction of the tree and it tries to construct leaves with instances of an unique class (in spite of very few training instances appear in the leaves). ID3 incorporates a pre-pruning strategy, using the chi-square statistic to guarantee a minimum correlation between the proposed split-attribute and the class;
- C4.5 algorithm (Quinlan, 1993) is inspired in ID3. In contrast to ID3, C4.5 carries out a post-pruning phase, based on the error-based-pruning algorithm, which replaces a node by one of its children if the accuracy of the child is considered better. It is usually employed in the literature as a reference algorithm to perform comparisons with novel supervised algorithms;
- OC1 can be categorized as an oblique decision tree algorithm (Murthy et al., 1994). By the combination of deterministic hill-climbing with two forms of randomization, OC1 builds an hyperplane (considering all the features) at each tree node to perform a split. It also incorporates a post-pruning strategy;
- HOODG builds oblivious graphs bottom-up (Kohavi, 1995a). It can be seen like a decision tree that tests the same single attribute in all the nodes of the same level. It is coupled with feature subset selection and discretization of data. Oblivious graphs try to overcome the replication and fragmentation problems (Kohavi, 1995d), inherent to decision trees;
- T2 (Auer et al., 1995) builds the two level decision tree that minimizes the number of errors in the training set and it is inspired in the PAC-learning theory (Valiant, 1984). This algorithm is categorized in the literature under the ‘simple decision tree’ term (Auer et al., 1995);
- OneR (Holte, 1993) is another ‘simple decision tree’ algorithm inspired in Occam’s razor and PAC-learning theory. It computes the one-level decision tree that minimizes the number of errors in the training set;

- Naive Bayes (NB) (Cestnik, 1990) learning algorithm uses a variation of the Bayes rule to predict the class for a test instance, assuming that features are independent to each other for predicting the class. NB applies the following rule:

$$c_{NB} = \arg \max_{c^j \in C} p(c^j) \prod_{i=1}^n p(x_i | c^j)$$

where c_{NB} denotes the category value predicted by the NB classifier for a test instance. The probability for discrete features is estimated from data using maximum likelihood estimation and applying the Laplace correction. A normal distribution is assumed to estimate the class conditional probabilities for continuous attributes. Unknown values in the test instance are skipped. Although its simplicity and its independence assumption among variables, the literature shows that the NB classifier gives remarkably high accuracies in many domains (Langley and Sage, 1994), specially in medical ones. Many researchers think that the success of the NB rule is based on the idea that doctors, in order to make a diagnosis, collect the attributes in the same way as the NB rule uses them to classify: that is to say, independently with respect to the class (Inza et al., 2001d);

- Naive-Bayes Tree (NBTree) algorithm (Kohavi, 1996) combines the ideas of decision trees and NB, executing NB at the leaves of an univariate-split decision tree. Normalized mutual-information (Shannon, 1948) is used to select split attributes in tree nodes and no post-pruning strategy is employed. Cross-validation is employed to decide when the tree expansion is terminated and the NB rule is applied;
- IB1 (Aha et al., 1991) is an instance based inducer that uses homogeneous weights for all attributes to compute the dissimilarity function, assuming that all features have the same relevance to define that class. The overlap metric (Salzberg, 1991) is used for symbolic features and the Euclidean metric for continuous ones. A 1-NN (Nearest Neighbor) schema is employed, predicting for a test instance the class of its nearest instance in the training set;
- IB4 (Aha et al., 1991) is the successor of IB1, incorporating a weight learning capability: different weights are computed for each feature, according to its relevance, assuming that all features should not have the same importance to define that class. IB4 assigns weights to features in the $[0,1]^n$ continuous weight space by means of a hill-climbing, sequential, incremental and on-line strategy, with only one pass through the training data;
- PEBLS is an instance based inducer (Cost and Salzberg, 1993) that incorporates the MVDM distance metric to deal with symbolic features, a modification of the VDM (Stanfill and Waltz, 1986) metric. It also attaches weights to the training instances according to their performance history. PEBLS was inspired on the need of a special distance metric to deal with Molecular Biology extracted features;
- Table-Majority is based on decision table paradigm (Kohavi, 1995c), storing a table of all instances and predicting according to it. If an instance is not found, it predicts the majority class. As for HOODG, to make the algorithm useful, it is coupled with feature subset selection and discretization of data. Given an unlabelled test instance, Table-Majority predicts the majority class of the training set instances whose values on the selected set of features match the test instance; if no instances are found, the classifier ‘gives up’ and predicts the majority of the whole training set;

- CN2 (Clark and Nibblet, 1989) induces a set of ordered IF-THEN decision rules. The condition of the rule appears in the IF part of the rule. The class predicted for the unlabelled examples that match the condition (IF) part appears in the THEN part of the rule. To classify a test example, each rule is tried in order until one is found whose conditions are satisfied by the example being classified. When the example does not match any rule, the most frequent class in the training set is assigned for it: this action is known as the ‘default-rule’. Although CN2 does not incorporate a classic post-pruning mechanism as C4.5, the rules are generated and expanded by the supervision of statistical tests that guarantee their significance degree (pre-pruning mechanism);
- Ripper (Cohen, 1995) is an IF-THEN decision rule algorithm, similar to CN2. Apart from other minor variations, its major apportionment with respect to CN2 is that Ripper post-prunes the generated rules by the error-based-pruning technique employed in C4.5 (Quinlan, 1993).

2.5 Datasets

In order to evaluate the claims and contributions that will be made in this dissertation, an extensive group of benchmark datasets is needed. The datasets are independently selected and they come from very different areas of the reality. The datasets have an interesting balanced coverage of a wide range of statistical criteria: number of cases, frequency of missing values, level of noise, class distribution, real nature of the domain, number of attributes, number of classes, etc. All real-world selected datasets, except *Cloud* (Aha and Bankert, 1994) domain, come from the UCI repository (Murphy, 1995), which is considered the principal reference for the election of datasets for Machine Learning comparisons. All of them are well known datasets with a long tradition in Machine Learning studies. Researchers have actively contributed to compile the near 120 domains that form the repository, covering wide areas of the reality and many different statistical characteristics. The characteristic features of the domains are summarised in Table 2.2.

Both real-world and artificial datasets are included in the different comparisons of this dissertation. Kohavi (1995d)(Kohavi, 1995d) resumes the need of this duality for Machine Learning comparisons in the following form: *Artificial domains are useful because they allow us to vary parameters, understand the specific problems that algorithms exhibit, and test conjectures. Real-world domains are useful because they come from real-world problems that we do not always understand and are therefore actual problems on which we would like to improve performance.* In this way, the apparition of artificial datasets are closely related with the hypotheses and conjectures of a specific comparison. Following these ideas, it is preferred to introduce the principal characteristics of the artificial datasets when their specific comparison is performed.

The following paragraphs provide a short description of the nature and reality aspects covered by each domain.

- *Echocardiogram* dataset’s task is to predict whether a patient survived for at least one year following a heart attack (Clark and Boswell, 1991). The most difficult part of this problem is correctly predicting that the patient will not survive (part of the difficulty seems to be the size of the data set);
- *Lymphography* dataset is obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Slovenja. Thanks go to M. Zwitter and M. Soklic for providing the data. The task is to distinguish between patient that are healthy and those with metastases or malignant lymphoma;

Table 2.2. Details of experimental domains. C: continuous, N: nominal.

<i>Domain</i>	<i>Number of instances</i>	<i>Number of features</i>	<i>Number of classes</i>
<i>Echocardiogram</i>	131	6 C, 1N	2
<i>Lymphography</i>	147	19 N	4
<i>Hepatitis</i>	155	6 C, 13 N	2
<i>Glass</i>	214	9 C	7
<i>Audiology</i>	226	69 N	24
<i>Heart disease</i>	270	6 C, 7 N	2
<i>Breast cancer</i>	286	9 N	2
<i>Hungarian</i>	294	6 C, 7 N	2
<i>Cleveland</i>	303	6 C, 7 N	2
<i>Liver (BUPA)</i>	345	6 C	2
<i>Ionosphere</i>	351	34 C	2
<i>Horse-colic</i>	368	7 C, 15 N	2
<i>Arrhythmia</i>	452	279 C	16
<i>Soybean-large</i>	683	35 N	19
<i>CRX</i>	690	6 C, 9 N	2
<i>Breast cancer (Wisconsin)</i>	699	10 C	2
<i>Diabetes (Pima)</i>	768	8 C	2
<i>Vehicle</i>	846	18 C	4
<i>Anneal</i>	898	9 C, 29 N	6
<i>Contraceptive</i>	1,473	4 C, 5 N	3
<i>Cloud</i>	1,834	204 C	4
<i>Image</i>	2,310	19 C	7
<i>Sick-euthyroid</i>	3,163	7 C, 18 N	2
<i>Hypothyroid</i>	3,163	7 C, 18 N	2
<i>DNA</i>	3,186	180 N	3
<i>Internet advertisements</i>	3,279	3 C, 1,555 N	2
<i>Spambase</i>	4,601	57 C	2

- *Hepatitis* dataset's task is to predict whether a patient will die from hepatitis. It has a moderate number of missing values compared to the size of the dataset and the number of attributes. Close to 80% of the data are the 'alive' patients' records;
- *Glass* identification dataset represents the measurements of chemical compounds extracted from the glass. The study of classification of types of glass was motivated by criminological investigation. The glass left at the scene of the crime can be used as evidence, if it is correctly identified. The instance identifier number of the original dataset is removed. The dataset has a skewed class distribution. Out of the total 214 instances, the class that has highest number of instances constitutes 35.7% and the lowest constitutes 4.2%;
- *Audiology* dataset is obtained from the Baylor College of Medicine. Thanks go to Professor Jergen for providing the data. As each example identifies a patient, the 24 classes are diagnoses of hearing disorders;
- *Heart disease* dataset comes from the Cleveland Clinic Foundation (USA) and was supplied by R. Detrano. It is a part of the *Cleveland* dataset and it was employed in the STATLOG project (Michie et al., 1994). The original *Cleveland* dataset contains 303 examples but 6 of these

contained missing values and were discarded for *Heart disease* dataset. Other 27 examples were retained in case of dispute, leaving the final total of 270. The original dataset *Cleveland* dataset has 76 attributes, but all published experiments refer to using a subset of 14 of them. It contains cardiological diagnoses. As each instance describes a patient's medical data, the task is to determine whether the patient suffers from heart disease;

- *Breast cancer* dataset is obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Slovenia. Thanks go to M. Zwitter and M. Soklic for providing the data. For about 30% of patients that undergo a breast cancer operation, the illness reappears after 5 years: thus prognosis of recurrence is important. Thus, 2 possible classes are 'no recurrence' and 'recurrence';
- *Hungarian* dataset was collected at the Hungarian Institute of Cardiology, Budapest (Hungary). Thanks go to A. Janosi. It has the same classification task and collects features as *Heart disease* and *Cleveland* domains. As *Cleveland* domain, the original dataset has 76 attributes, but all published experiments refer to using a subset of 14 of them;
- *Cleveland* dataset comes from the Cleveland Clinic Foundation (USA) and was supplied by R. Detrano. In previous paragraphs we have cited its relation with *Heart disease* and *Hungarian* datasets. Less than 2% of the total attribute values are missing in this dataset;
- *Liver (BUPA)* dataset was created at BUPA Medical Research Ltd. As each instance describes a male patient's medical data, the task is to determine whether the patient suffers from liver disorder, that might arise from excessive alcohol consumption. All except one attribute are blood tests which are thought to be sensitive to liver disorders;
- *Ionosphere* radar dataset was collected by a system in Goose Bay, Labrador, USA. The system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. The classes of the problem are the following: 'good' radar returns are those showing evidence of some type of structure in the ionosphere and 'bad' returns are those that do not; their signals pass through the ionosphere;
- *Horse-colic* dataset was created by M. McLeish and M. Cecile from the University of Guelph. The task is to determine whether a horse lesion is surgical. The features describe whether the horse had surgery, whether it is young or old, rectal temperature, pulse, etc.;
- *Arrhythmia* dataset was created by H. A. Guvenir, at Bilkent University, Turkey. The aim is to distinguish between the presence and absence of cardiac arrhythmia and to classify it in one of the 15 classes of arrhythmia;
- *Soybean-large* dataset's task is to diagnose soybean diseases. The 19 classes are described by 34 features collecting leaf properties and various abnormalities. There is a considerable number of missing values;
- *CRX* dataset concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. Both classes labels are 'yes' and 'no'. The dataset is interesting because there is a good mix of attributes (continuous, discrete with small number of values, and discrete with larger number of values). There are also a few missing values;

- *Breast cancer (Wisconsin)* dataset concerns medical diagnosis applied to breast cytology. The task is to predict whether a breast tumour is benign or malignant. The dataset was collected over a period of two and a half years by the University of Wisconsin Hospitals, Madison, USA. Thanks go to Dr. W.H. Wolberg;
- *Diabetes (Pima)* dataset was collected by the USA National Institute of Diabetes and Digestive and Kidney Diseases. The task is to determine whether the patient shows signs of diabetes according to World Health Organization Criteria. All patients are females who live near Phoenix, Arizona, USA. They are at least 21 years old and of Pima Indian heritage;
- *Vehicle* dataset's purpose is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles. Thus, images were acquired by a camera looking downwards at the model vehicle from a fixed angle of elevation (34.2 degrees to the horizontal). All images were captured with a spatial resolution of 128×128 pixels quantised to 64 greylevels;
- *Anneal* dataset is donated by D. Sterling and W. Buntine. The dataset has 38 features and 898 which belong to a steel-domain supervised task. It has a considerable number of missing values;
- *Contraceptive* dataset is donated by T.S. Lim and it is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The examples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics;
- *Cloud* dataset, provided by D.W. Aha and R.L. Bankert, was collected at the US Naval Research Laboratory and presents cloud features and classifications (Aha and Bankert, 1994). Each example represents a cloud belonging to one of eleven cloud types;
- *Image* segmentation dataset instances were drawn randomly from a database of 7 outdoor colour images. These were hand segmented to create a classification for every pixel as one of brickface, sky, foliage, cement, window, path or grass. There were 19 features for each 3×3 region, for example summary measurements of contrast in the vertical and horizontal directions;
- *Sick-euthyroid* dataset was collected at the Garavan Institute in Sidney, Australia. Its task is to identify a patient as having sick-euthyroid disease or not;
- *Hypothyroid* dataset was collected at the Garavan Institute in Sidney, Australia, and it has the same data format and attributes as *Sick-euthyroid*. Thus, *Hypothyroid* task is to identify a patient as having hypothyroid disease or not;
- *DNA* domain is drawn from the field of Molecular Biology. Splice junctions are points on a DNA sequence at which 'superfluous' DNA is removed during protein creation. The task is to recognize exon-intron boundaries, referred to as EI sites; intron-exon boundaries, referred to as IE sites; or neither. The features of each instance provide a window of 60 nucleotides, each represented as 3 binary indicator features that represent the value 'a', 'c', 'g' or 't', thus giving 180 binary features (using 0-1 codification). The classification is the middle point of the window, thus providing 30 nucleotides at each side of the junction;
- *Internet advertisements* is donated by N. Kushmerick. The dataset represents a set of possible advertisements on Internet pages (Kushmerick, 1999). The features encode the geometry of the

image (if available) as well as phrases occurring in the URL, the image's URL and alt text, the anchor text, and words occurring near the anchor text. The task is to predict whether an image is an advertisement or not;

- *Spambase* dataset tries to identify the 'spam' concept in personal e-mails: advertisements for products/web sites, make money fast schemes, chain letters, pornography, unsolicited commercial e-mail... The dataset was created by collecting a set of personal e-mails received at Hewlett-Packard Laboratories, Palo Alto, USA. The task is to determine whether a given email is spam or not. Most of the attributes indicate whether a particular word or character was frequently occurring in the e-mail.

2.6 Accuracy estimation: evaluating and comparing classification models

Accuracy estimation is fundamental in any Machine Learning work (it is also known as model validation). The evaluation of the performance of the learned classification model is important: one reason for this is simply to understand whether to use the induced model in the real life to classify new unlabelled case. For instance, it is important to acceptably assess the effectiveness of a proposed Machine Learning model in medical diagnosis (Mitchell, 1997). In the same way, it is the key and crucial feature in many Machine Learning experimentations.

The accuracy estimation, seen as the p parameter of a Bernoulli random variable ($Y \rightarrow B(1, p)$), has an intrinsic uncertainty (Kohavi, 1994). This estimation is somewhat straightforward and has a low uncertainty degree when data is plentiful. However, as we normally must estimate the accuracy of a learned model with a limited number of cases, we must have in mind the following two difficulties, inherent to accuracy estimation (Mitchell, 1997):

- in testing the accuracy of a classifier, its error rate estimate tends to be *biased* if they are assessed from the same set of examples that was used to construct the classification model. This is especially critical when the classification algorithm considers a very large space of possible models, enabling it to overfit the training cases. To obtain an unbiased estimate of the accuracy, the classifier should be tested on a set of examples chosen independently of examples that built it;
- the measured accuracy can still vary from the true accuracy, depending on the specific makeup of the particular test examples. This is especially critical when a small number of test examples is provided: in this case, the error rate estimate tends to have a large *variance*.

The literature includes plenty of works presenting different accuracy estimation techniques, exposing their strengths and weakness with respect to the previous *bias* and *variance* concepts (Kohavi and Wolpert, 1996).

Depending on the employed accuracy estimation method, the use of an hypothesis testing approach enables us to determine the significance degree of the estimated accuracy difference between two algorithms.

Although an active debate exists within the Machine Learning community with respect to the best estimation method and coupled hypothesis testing comparison (Dietterich, 1998), the question remains unanswered.

The *resubstitution estimate* and *holdout* methods are two classic accuracy estimation procedures. Many estimation techniques are inspired on them. They work as follows:

- the *resubstitution estimate* procedure, also called apparent accuracy, estimates the accuracy on the same data used to construct the classifier. As learning algorithms try to minimize it, this measure is a highly optimistic estimate of the accuracy. For learning procedures that perfectly fit the data, such as nearest-neighbor or decision trees without a pruning phase, the optimistic bias is really high;
- the *holdout* method, or test-sample estimation, randomly partitions the dataset into two mutually exclusive subsets called the training subset and the test subset, or holdout set. Usually $\frac{2}{3}$ of data form the training subset and the rest $\frac{1}{3}$ forms the test subset. Then, the learning algorithm is run using the training subset and the induced classifier's accuracy is estimated on the test subset. This measure is used as the accuracy estimate of the classifier built with the entire dataset. *Random subsampling* appears as an improvement of the holdout method: as the holdout procedure is repeated k times, the estimated accuracy is derived by averaging the runs and the standard deviation is estimated as the standard deviation of the accuracy estimations from each holdout run.

In spite of both methods are not employed in this dissertation, they are relevant to understand more elaborate validation procedures. Although several details are left for future chapters, in the next subsections the accuracy estimation methods and coupled hypothesis testing approaches that are employed in this dissertation are presented.

2.6.1 k -fold cross-validation

In *k-fold cross-validation* (Stone, 1974), the dataset is randomly split into k mutually exclusive subsets (or folds) of approximately equal size. The supervised learning algorithm is trained k times: each time, the training process is performed in $k - 1$ folds and testing in the remaining fold. The k -fold cross-validation accuracy estimate is the average accuracy measure from these testing k folds. In the same way, the standard deviation of this average can be reported. The most common value in the literature of the k parameter is 10. This value will be employed in this dissertation.

A method to improve the quality of a cross-validation estimate is to repeat the exposed process multiple times. In this way, the intrinsic uncertainty and standard deviation of the cross-validation estimate can be reduced. Although it is preferred to leave the details for further chapters, an heuristic based on this idea will be used in this dissertation. Another improvement of the cross-validation is the *stratified cross-validation* (Breiman et al., 1984), where the folds are stratified so that they contain approximately the same proportions of classes as the original dataset.

A particular case of k -fold cross-validation is the *leave-one-out cross-validation error* (LOOCE) procedure (Lachenbruch and Mickey, 1968). In the LOOCE technique, the learning algorithm is run k times, where k is the number of examples of the dataset. In this way, each time $k - 1$ instances are used for training and the remaining example is used for testing, where each case is used only once for testing. The LOOCE estimate of accuracy can be also seen as the overall number of correct classifications, divided by k , the number of cases in the dataset. LOOCE gives almost unbiased accuracy estimations (Lachenbruch, 1967).

In this dissertation, when the k -fold cross-validation procedure is used to estimate the accuracy of two algorithms in the same dataset, the k -fold cross-validates paired t test (Dietterich, 1998) is used to study the significance degree of the accuracy differences between both algorithms. It works as follows:

Suppose that the dataset is divided into k disjoint sets of equal size, T_1, T_2, \dots, T_k . We then conduct k trials for learning algorithms A and B . In the i -th trial, the test set is T_i for both learning

algorithms and the training set if the union of all of the other $T_j, j \neq i$. Let p_A^i (respectively p_B^i) be the observed proportion of test examples misclassified by algorithm A (respectively B) during trial i . If we assume that the k differences $p^i = p_A^i - p_B^i$ are drawn independently from a normal distribution, then a Student's t test can be applied, by computing the statistic

$$t = \frac{\bar{p} \cdot \sqrt{k}}{\sqrt{\frac{\sum_{i=1}^k (p^i - \bar{p})^2}{k-1}}} \quad (2.1)$$

where $\bar{p} = \frac{1}{k} \sum_{i=1}^k p^i$. Under the null hypothesis, this statistic has a t distribution with $k - 1$ degrees of freedom.

2.6.2 5 times 2-fold cross-validation

A novel approach to estimate the accuracy of a learning algorithm is to perform 5 replications of 2-fold cross-validation (5x2cv). In each replication, available dataset is randomly partitioned into two equal-sized sets: the algorithm is trained on each set and tested on the other set. In this way, the reported accuracy is the mean of 10 accuracy values. In the same way, the standard deviation of this average can be reported.

This validation method is well-suited when randomized learning algorithms are used. As 5 replications of the 2-fold cross-validation scheme are carried out, the randomness of the learning algorithm is automatically taken into account.

When the 5x2cv procedure is used to estimate the accuracy of two algorithms in the same dataset, the 5x2cv F test (Alpaydin, 1999) is used to study the significance degree of the accuracy differences between both algorithms. It works as follows:

Suppose that p_i^j is the difference between the error rates of the two classifiers on fold $j = 1, 2$ of replication $i = 1, \dots, 5$. The average on replication i is $\bar{p}_i = \frac{p_i^1 + p_i^2}{2}$ and the estimated variance is $s_i^2 = (p_i^1 - \bar{p}_i)^2 + (p_i^2 - \bar{p}_i)^2$. We assume that the p_i^j differences are drawn independently from a normal distribution with zero mean and unknown variance σ^2 (Dietterich, 1998) and then $\frac{s_i^2}{\sigma^2}$ is chi-square with one degree of freedom. Under the null hypothesis the statistic

$$f = \frac{\sum_{i=1}^5 \sum_{j=1}^2 (p_i^j)^2}{2 \sum_{i=1}^5 s_i^2} \quad (2.2)$$

is approximately F distributed with 10 and 5 degrees of freedom.

2.7 Summary

In this chapter the principal roots of Supervised Classification are exposed. Special emphasis is focused on the supervised procedures employed in this dissertation. First supervised algorithms, Discriminant Analysis and Logistic Regression are introduced: they are the inspiration for the birth of the learning algorithms grouped under the Machine Learning term. The Machine Learning algorithms and real datasets employed in this dissertation are studied. The last section has described the importance of the accuracy estimation in the Supervised Classification. The accuracy estimation procedures, coupled with their respective statistical hypothesis testing techniques for accuracy differences, are studied in detail.

Chapter 3

Probabilistic Graphical Models

3.1 Overview of the chapter

In this chapter Bayesian and Gaussian networks will be introduced. In Chapters 5, 6 and 7 of this dissertation, in conjunction with the Estimation of Distribution Algorithms (EDAs, see Chapter 4) paradigm, both Probabilistic Graphical Models will be used in order to improve certain aspects of Supervised Classification.

The probabilistic graphical model paradigm (Pearl, 1988; Lauritzen, 1996), which has become a popular representation for encoding uncertain knowledge in expert systems over the last decade, will be introduced. Once a general notation and the probabilistic graphical model paradigm are presented in an abstract way, the emphasis of the chapter will be put in two well known probabilistic graphical models that will be employed in further chapters of this dissertation: Bayesian networks and Gaussian networks. For these paradigms, the structural learning from data, the conditional (in)dependence concept and the simulation tasks will be studied with attention.

This chapter is an adaptation of the work by Larrañaga (2001a).

3.2 Terminology and basic concepts

In the next lines, a general notation that will be used in the rest of this dissertation, useful for Bayesian and Gaussian probabilistic graphical models, is presented.

The symbol X_i is used to represent a random variable. A possible instance of X_i is denoted x_i . $\rho(X_i = x_i)$ (or simply $\rho(x_i)$) represents the *generalized probability distribution* (DeGroot, 1970) over the point x_i . Similarly, $\mathbf{X} = (X_1, \dots, X_n)$ is used to represent an n -dimensional random variable, and $\mathbf{x} = (x_1, \dots, x_n)$ to represent one of its possible instances. The *joint generalized probability distribution* of \mathbf{X} is denoted $\rho(\mathbf{X} = \mathbf{x})$ (or simply $\rho(\mathbf{x})$). The *generalized conditional probability distribution* of the variable X_i given the value x_j of the variable X_j is represented as $\rho(X_i = x_i | X_j = x_j)$ (or simply by $\rho(x_i | x_j)$). The symbol D represents a dataset, i.e. a set of N instances of the variable $\mathbf{X} = (X_1, \dots, X_n)$.

If the variable X_i is discrete, $\rho(X_i = x_i) = p(X_i = x_i)$ (or simply $p(x_i)$) is called the *mass probability* for the variable X_i . If all the variables in \mathbf{X} are discrete, $\rho(\mathbf{X} = \mathbf{x}) = p(\mathbf{X} = \mathbf{x})$ (or simply $p(\mathbf{x})$) is the *joint probability mass*, and $\rho(X_i = x_i | X_j = x_j) = p(X_i = x_i | X_j = x_j)$ (or simply $p(x_i | x_j)$) is the *conditional mass probability* of the variable X_i given that $X_j = x_j$.

In the case that X_i is continuous, $\rho(X_i = x_i) = f(X_i = x_i)$ (or simply $f(x_i)$) is the *density function* of X_i . If all the variables in \mathbf{X} are continuous, $\rho(\mathbf{X} = \mathbf{x}) = f(\mathbf{X} = \mathbf{x})$ (or simply $f(\mathbf{x})$) is the *joint density function*, and $\rho(X_i = x_i | X_j = x_j) = f(X_i = x_i | X_j = x_j)$ (or simply $f(x_i | x_j)$) is the *conditional density function* of the variable X_i given that $X_j = x_j$.

Let $\mathbf{X} = (X_1, \dots, X_n)$ be a vector of random variables. The symbol x_i is used to denote a value of X_i , the i^{th} component of \mathbf{X} , and $\mathbf{y} = (x_i)_{X_i \in \mathbf{Y}}$ to denote a value of $\mathbf{Y} \subseteq \mathbf{X}$. A *probabilistic graphical model* for \mathbf{X} is a graphical factorization of the joint generalized probability distribution, $\rho(\mathbf{X} = \mathbf{x})$ (or simply $\rho(\mathbf{x})$). The representation consists of two components: a structure and a set of local generalized probability distributions. The structure S for \mathbf{X} is a Directed Acyclic Graph (DAG) that represents a set of conditional (in)dependence¹ (Dawid, 1979) assertions on the variables on \mathbf{X} , and each node in the structure is associated with each variable of \mathbf{X} .

The structure S for \mathbf{X} represents for each variable the assertions that X_i and $\{X_1, \dots, X_n\} \setminus \mathbf{Pa}_i^S$ ² are independent given \mathbf{Pa}_i^S , $i = 2, \dots, n$. Thus, by the application of the chain-rule, the factorization can be simplified as follows:

$$\begin{aligned} \rho(\mathbf{x}) &= \rho(x_1, \dots, x_n) \\ &= \rho(x_1) \cdot \rho(x_2 | x_1) \cdot \dots \cdot \rho(x_i | x_1, \dots, x_{i-1}) \cdot \dots \cdot \rho(x_n | x_1, \dots, x_{n-1}) \\ &= \rho(x_1) \cdot \rho(x_2 | \mathbf{pa}_2^S) \cdot \dots \cdot \rho(x_i | \mathbf{pa}_i^S) \cdot \dots \cdot \rho(x_n | \mathbf{pa}_n^S) = \prod_{i=1}^n \rho(x_i | \mathbf{pa}_i^S). \end{aligned} \quad (3.1)$$

It is assumed that, for each variable, the local generalized probability distributions depend on a finite set of parameters $\boldsymbol{\theta}_S \in \Theta_S$. Thus, the previous equation is rewritten as follows:

$$\rho(\mathbf{x} | \boldsymbol{\theta}_S) = \prod_{i=1}^n \rho(x_i | \mathbf{pa}_i^S, \boldsymbol{\theta}_i) \quad (3.2)$$

where $\boldsymbol{\theta}_S = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n)$. Taking both components of the probabilistic graphical model into account, this will be represented by $M = (S, \boldsymbol{\theta}_S)$. That is, a couple formed by the structure and their associated parameters.

Example 3.1 *The structure of the probabilistic graphical model represented in Figure 3.1 provides us the following factorization of the joint generalized probability distribution:*

$$\begin{aligned} &\rho(x_1, x_2, x_3, x_4, x_5, x_6) \\ &= \rho(x_1) \cdot \rho(x_2) \cdot \rho(x_3 | x_1, x_2) \cdot \rho(x_4 | x_3) \cdot \rho(x_5 | x_3, x_6) \cdot \rho(x_6). \end{aligned} \quad (3.3)$$

Informally, an arc between two nodes relates the two nodes so that the value of the variable corresponding to the ending node of the arc depends on the value of the variable corresponding to the starting node.

The conditional independence and separation criterion concepts are basic for the understanding of the underlying semantic of probabilistic graphical models. To check the conditional independence—see Figure 3.2—between variables \mathbf{Y} and \mathbf{Z} given \mathbf{W} , the smallest subgraph containing \mathbf{Y} , \mathbf{Z} , \mathbf{W} and their ancestors must be considered. This subgraph must be moralized. To carry out this

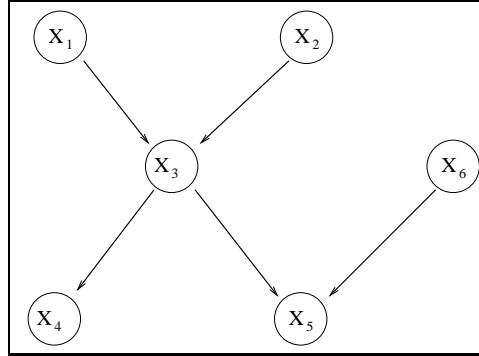


Figure 3.1. Structure for a probabilistic model defined over $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5, X_6)$.

-
- Step 1. Obtain the smallest subgraph containing \mathbf{Y} , \mathbf{Z} , \mathbf{W} and their ancestors
 - Step 2. Moralize the obtained subgraph
 - Step 3. If every path between the variables in \mathbf{Y} and the variables in \mathbf{Z} in the obtained undirected graph is blocked by a variable in \mathbf{W} , then $I(\mathbf{Y}, \mathbf{Z} \mid \mathbf{W})$
-

Figure 3.2. Checking conditional independencies in a probabilistic graphical model by means of the u -separation criterion for undirected graphs.

moralization it is mandatory to add an edge between parents with common children and, then to delete the direction of the arcs, that is to transform the arcs in edges. If every path between the variables in \mathbf{Y} and variables in \mathbf{Z} in the obtained undirected graph is blocked by a variable in \mathbf{W} , then it is said that in the original graph the variables \mathbf{Y} and \mathbf{Z} are conditionally independent given \mathbf{W} and then $I(\mathbf{Y}, \mathbf{Z} \mid \mathbf{W})$ is written. Otherwise \mathbf{Y} and \mathbf{Z} are said to be conditionally dependent given \mathbf{W} , and then $D(\mathbf{Y}, \mathbf{Z} \mid \mathbf{W})$ is written.

Depending on the connectivity of the model structure, different degrees of complexity in probabilistic graphical models can be considered. This can be a form to carry out this classification:

- Without dependencies

In this type of structures, each variable has no parent parent variable. As a consequence, the following factorization is followed:

$$\rho(\mathbf{x} \mid \boldsymbol{\theta}_S) = \prod_{i=1}^n \rho(x_i \mid \boldsymbol{\theta}_i). \quad (3.4)$$

- Tree

In this type of structures, each variable can have at the most one parent variable. As a consequence, the following factorization is followed:

$$\rho(\mathbf{x} \mid \boldsymbol{\theta}_S) = \prod_{i=1}^n \rho(x_i \mid x_{j(i)}, \boldsymbol{\theta}_i) \quad (3.5)$$

where $X_{j(i)}$ is the (possibly empty) parent of variable X_i .

■ Multiply connected

While in tree structures, given two nodes in the DAG, there is an unique path connecting them at the most, in multiple connected structures, every two nodes in the DAG can be connected by more than one path. As a result, the factorization is as follows:

$$\rho(\mathbf{x} \mid \boldsymbol{\theta}_S) = \prod_{i=1}^n \rho(x_i \mid \mathbf{pa}_i^S, \boldsymbol{\theta}_i). \quad (3.6)$$

However, both previous structure types can be considered as a particular case of multiply connected ones.

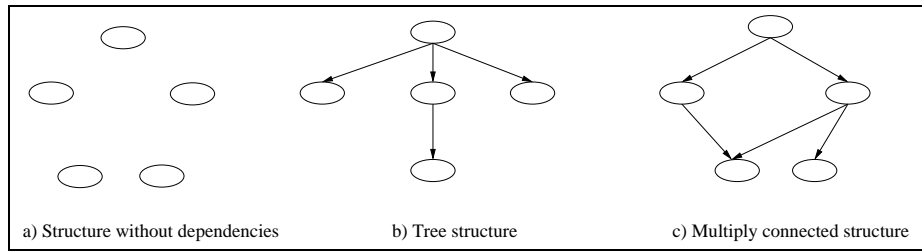


Figure 3.3. Degrees of complexity in the structure of probabilistic graphical models.

Figure 3.3 shows a graphical representation of the different types of structures introduced in this section.

3.3 Bayesian networks

3.3.1 Introduction

When the variables of the problem are discrete, Bayesian networks are an interesting probabilistic graphical model that have been surrounded by a growing interest in the recent years. From an informal perspective, Bayesian networks are DAGs where the nodes are random variables with a discrete set of values and the arcs specify the (in)dependence assumptions that must be held between the random variables. The literature shows a large number of dedicated books and a wide range of theoretical and practical publications in the field. The classic and well-known book of Pearl (1988) must be mentioned. Neapolitan (1990) explains the basics of propagation algorithms and these are studied in detail by Shafer (1996). Jensen (2001) is a recommended tutorial introduction while in Castillo et al. (1997) another sound introduction with many worked examples can be found. Lauritzen (1996) provides a mathematical analysis of graphical models. More recently, Cowell et al. (1999) is an excellent compilation material covering recent advances in the field.

Bayesian networks constitute a probabilistic framework for reasoning under uncertainty. The reasoning inside the model, that is, the propagation of the evidence through the model, depends on the structure reflecting the conditional (in)dependencies between the variables. Cooper (1990) prove that this task is NP-hard in the general case of multiply connected Bayesian networks. The most popular algorithm to accomplish this task is proposed by Lauritzen and Spiegelhalter (1988) –later improved by Jensen et al. (1990)– and is based on a manipulation of the Bayesian network which starts with the moralization of the graphical model.

3.3.2 Basic notation for Bayesian networks

In a Bayesian network structure, when the discrete variable $X_i \in \mathbf{X}$ has r_i possible values, $x_i^1, \dots, x_i^{r_i}$, the local distribution, $p(x_i | \mathbf{pa}_i^{j,S}, \theta_i)$ is an unrestricted discrete distribution:

$$p(x_i^k | \mathbf{pa}_i^{j,S}, \theta_i) = \theta_{x_i^k | \mathbf{pa}_i^j} \equiv \theta_{ijk} \quad (3.7)$$

where $\mathbf{pa}_i^{1,S}, \dots, \mathbf{pa}_i^{q_i,S}$ denotes the values of \mathbf{Pa}_i^S , the set of parents of the variable X_i in the structure S . The term q_i denotes the number of possible different instances of the parent variables of X_i . Thus, $q_i = \prod_{X_g \in \mathbf{Pa}_i} r_g$.

The local parameters are given by $\theta_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i}$. In others words, the parameter θ_{ijk} represents the conditional probability of variable X_i being in its k^{th} value, knowing that the set of its parent variables is in its j^{th} value.

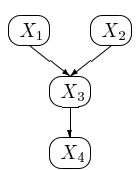
Structure	Local probabilities	
	$\theta_1 = (\theta_{1-1}, \theta_{1-2})$	$p(x_1^1), p(x_1^2)$
	$\theta_2 = (\theta_{2-1}, \theta_{2-2}, \theta_{2-3})$	$p(x_2^1), p(x_2^2), p(x_2^3)$
	$\theta_3 = (\theta_{311}, \theta_{321}, \theta_{331},$ $\theta_{341}, \theta_{351}, \theta_{361},$ $\theta_{312}, \theta_{322}, \theta_{332},$ $\theta_{342}, \theta_{352}, \theta_{362})$	$p(x_3^1 x_1^1, x_2^1), p(x_3^1 x_1^1, x_2^2), p(x_3^1 x_1^2, x_2^1),$ $p(x_3^1 x_1^2, x_2^2), p(x_3^2 x_1^1, x_2^1), p(x_3^2 x_1^1, x_2^2),$ $p(x_3^2 x_1^2, x_2^1), p(x_3^2 x_1^2, x_2^2), p(x_3^3 x_1^1, x_2^1),$ $p(x_3^3 x_1^1, x_2^2), p(x_3^3 x_1^2, x_2^1), p(x_3^3 x_1^2, x_2^2)$
	$\theta_4 = (\theta_{411}, \theta_{421}, \theta_{412}, \theta_{422})$	$p(x_4^1 x_3^1), p(x_4^1 x_3^2), p(x_4^2 x_3^1), p(x_4^2 x_3^2)$
	Factorization of the joint mass-probability	
	$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3 x_1, x_2)p(x_4 x_3)$	

Figure 3.4. Structure, local probabilities and resulting factorization for a Bayesian network with four variables (X_1, X_3 and X_4 with two possible values, and X_2 with three possible values).

Example 3.2 In order to understand the introduced notation, Table 3.1 and Figure 3.4 are introduced. It must be taken into account that the values expressed in Table 3.1 are obtained from Figure 3.4. Notice also that from the factorization of the joint probability mass derived from the structure of the Bayesian network in Figure 3.4, the number of parameters needed to specify the joint probability mass is reduced from 23 to 11.

From Figure 3.4, it can be seen that to assess a Bayesian network, the user needs to specify:

- a structure by means of a DAG which reflects the set of conditional (in)dependencies among the variables;

Table 3.1. Variables (X_i), number of possible values of variables (r_i), set of variable parents of a variable (\mathbf{Pa}_i), number of possible instantiations of the parent variables (q_i).

<i>variable</i>	<i>possible values</i>	<i>parent variables</i>	<i>possible values of the parents</i>
X_i	r_i	\mathbf{Pa}_i	q_i
X_1	2	\emptyset	0
X_2	3	\emptyset	0
X_3	2	$\{X_1, X_2\}$	6
X_4	2	$\{X_3\}$	2

- the unconditional probabilities for all root nodes (nodes with no predecessors), $p(x_i^k \mid \emptyset, \theta_i)$ (or θ_{i-k});
- the conditional probabilities for all other nodes, given all possible combinations of their direct predecessors, $\theta_{ijk} = p(x_i^k \mid \mathbf{pa}_i^{j,S}, \theta_i)$.

3.3.3 Model induction

Once the Bayesian network is built, it constitutes an efficient device to perform probabilistic inference. Nevertheless, the problem of building such a network remains. The structure and conditional probabilities necessary for characterizing the Bayesian network can be provided either externally by experts –time consuming and subject to mistakes– or by automatic learning from a dataset of cases. On the other hand, the learning task can be separated into two subtasks: *structure learning*, that is, to identify the topology of the Bayesian network, and *parametric learning*, the numerical parameters (conditional probabilities) for a given network topology.

The easier accessibility to huge databases during the recent years has led to a big number of model learning algorithms had been proposed. Different approaches to Bayesian network model induction can be classified according to the nature of the modeling (detecting conditional (in)dependencies versus score+search methods). The principal approaches will be introduced in the following lines.

The reader can consult some good reviews on model induction in Bayesian networks in Heckerman (1995), Buntine (1996), Sangüesa and Cortés (1998) and Krause (1998).

3.3.3.1 Detecting conditional (in)dependencies

Every algorithm that tries to recover the structure of the Bayesian network by detecting (in)dependencies has some conditional (in)dependence relations between some subset of variables of the model as input, and a DAG that represents a large percentage (and even all of them if possible) of these relations as output. Once the structure has been learnt, the conditional probability distributions required to completely specify the model are estimated from the database –using some of the different approaches to parameter learning– or are given by an expert. The works by de Campos (1998) and Spirtes et al. (1993) provide good reviews of this kind of algorithms.

The PC algorithm of Spirtes et al. (1991) is possibly the most known algorithm for constructing Bayesian networks by detecting conditional (in)dependencies. It starts by forming the complete undirected graph and then it tries to ‘thin’ it. First, edges with zero order conditional independence relations are removed, then edges with first order conditional independence relations, and so on.

3.3.3.2 Score+search methods

Although the approach to the Bayesian network construction based on detecting conditional (in)dependencies is quite appealing due to its closeness to the semantic of Bayesian networks, the major part of the developed structure learning algorithms belongs to the category of score+search methods.

For this kind of methods, a metric that measures the goodness of every candidate Bayesian network with respect to a dataset of cases is needed. In addition, a search procedure, to intelligently move through the space of possible network structures is also necessary.

Search approaches

As the problem of finding the best network with respect to some criterion from the set of all networks in which each node has no more than k parents ($k > 1$) is NP-hard (Chickering et al., 1994), the use of search heuristics is justified. The search is carried out in the space of DAGs that represent the feasible Bayesian network structures. The cardinality of this space is huge and it is given by the recursive formula obtained by Robinson in 1977:

$$f(n) = \sum_{i=1}^n (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i); \quad f(0) = 1; \quad f(1) = 1. \quad (3.8)$$

The literature includes plenty of works proposing different search strategies to find good network structures: greedy searches (Buntine, 1991; Cooper and Herskovits, 1992), simulated annealing (Chickering et al., 1995), tabu search (Bouckaert, 1995), genetic algorithms (Larrañaga et al., 1996; Myers et al., 1999) and evolutionary programming (Wong et al., 1999).

Due to their speed and demonstrated efficiency, greedy strategies are widely used. As they will be used in further chapters of this dissertation, the principal techniques of this types are exposed in the next lines:

- Algorithm B (Buntine, 1991) starts with an arc-less structure and, at each step it adds the arc with the maximum improvement in the used score. It stops when adding an arc does not increase the scoring measure;
- another possibility is, starting with a given structure, carry out the addition or deletion of the arc with the maximum increase in the scoring measure at every step. The search stops when no modification of the structure improves the scoring measure. The main drawback of this strategy is that it heavily depends on the initial structure.

Score metrics

- *Penalized maximum likelihood*

Given a dataset D with N cases, $D = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, one might calculate for any structure S the maximum likelihood estimate, $\hat{\boldsymbol{\theta}}$, for the parameters $\boldsymbol{\theta}$ and the associated maximized log likelihood of the data, given the structure and the parameters: $\log p(D \mid S, \hat{\boldsymbol{\theta}})$. This can be used as a crude measure of the success of the structure S to describe the observed data D . It seems appropriate to score each structure by means of its associated maximized log likelihood and thus, to search for the structure that maximizes $\log p(D \mid S, \hat{\boldsymbol{\theta}})$. Following the proposed steps, the following formulation can be obtained:

$$\begin{aligned}
\log p(D \mid S, \boldsymbol{\theta}) &= \log \prod_{w=1}^N p(\mathbf{x}_w \mid S, \boldsymbol{\theta}) = \log \prod_{w=1}^N \prod_{i=1}^n p(x_{w,i} \mid \mathbf{pa}_i^S, \boldsymbol{\theta}_i) \\
&= \sum_{i=1}^n \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \log(\theta_{ijk})^{N_{ijk}}
\end{aligned} \tag{3.9}$$

where N_{ijk} denotes the number of cases in D in which the variable X_i has the value x_i^k and \mathbf{pa}_i has its j^{th} value, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. It must be remembered (see Section 3.2) that each case in D is characterized by n variables.

Taking into account that the maximum likelihood estimate for θ_{ijk} is given by $\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$ (Cooper and Herskovits, 1992), we obtain:

$$\log p(D \mid S, \hat{\boldsymbol{\theta}}) = \sum_{i=1}^n \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}. \tag{3.10}$$

The monotonicity of the likelihood with respect to the complexity of the structure usually leads the search through complete networks. The incorporation of a penalty term for the model complexity into the maximized likelihood could reduce this drawback, resulting in the following formula for a penalized maximum likelihood:

$$\sum_{i=1}^n \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - f(N) \dim(S) \tag{3.11}$$

where $\dim(S)$ is the dimension –number of parameters needed to specify the model– of the Bayesian network with a structure given by S . Thus, the dimension is specified by $\dim(S) = \sum_{i=1}^n q_i(r_i - 1)$ and $f(N)$ is a non negative penalization function. The most known penalization functions of the literature are the following:

- the Akaike’s Information Criterion (AIC) (Akaike, 1974) where $f(N) = 1$;
- the Jeffreys-Schwarz criterion, also known as the Bayesian Information Criterion (BIC) (Schwarz, 1978) where $f(N) = \frac{1}{2} \log N$.

■ *Bayesian scores. Marginal likelihood*

Our uncertainty on the proposed Bayesian network model (structure and parameters) can be expressed by the Bayesian approach for score metrics. As $p(\mathbf{x}|D)$ is the probability distribution we want to find, if \mathcal{S} represents the possible DAGs, then from probability theory we obtain:

$$p(\mathbf{x}|D) = \sum_{S \in \mathcal{S}} p(\mathbf{x}|S, D) p(S|D) \tag{3.12}$$

This equation is known as *Bayesian model averaging* (Madigan and Raftery, 1994). As it requires the summing of all possible structures, its use is unfeasible and usually two approximations are used instead of the afore mentioned approach.

The first is known as *selective model averaging* (Madigan and Raftery, 1994), where only a reduced number of promising structures is taken into account. In this case, denoting the set of promising structures by \mathcal{S}^{sel} , we have:

$$p(\mathbf{x}|D) \approx \sum_{S \in \mathcal{S}^{sel}} p(\mathbf{x}|S, D)p(S|D) \quad (3.13)$$

$$\text{where } \sum_{S \in \mathcal{S}^{sel}} p(S|D) \approx 1.$$

Although this approach could obtain good results, it has a large computational cost. The second approximation is known as *model selection* where $p(\mathbf{x}|D)$ is approximated in the following manner:

$$p(\mathbf{x}|D) \approx p(\mathbf{x}|\hat{S}, D) \quad (3.14)$$

$$\text{where } \hat{S} = \arg \max_{S \in \mathcal{S}^{sel}} p(S|D).$$

This means that only the structure with the maximum posterior likelihood is used, knowing that for large enough D , we have $p(\hat{S}|D) \approx 1$.

Obviously, better results must be obtained using model averaging, but due to its easier application and lower cost, model selection is preferred most of the times. As a quick estimation of $p(\mathbf{x}|D)$ is needed in subsequent parts of this dissertation, this second approximation will be used one.

A score commonly used in Bayesian model selection is the *logarithm of the relative posterior probability* of the model:

$$\log p(S | D) \propto \log p(S, D) = \log p(S) + \log p(D | S). \quad (3.15)$$

Under the assumption that the prior distribution over the structure is uniform, an equivalent criterion is the *log marginal likelihood* of the data given the structure, $\log p(D | S)$.

It is possible –see Cooper and Herskovits (1992) and Heckerman et al. (1995) for details– to compute the marginal likelihood efficiently and in closed form under some general assumptions.

As it will be used in Chapter 5 of this dissertation, K2 (Cooper and Herskovits, 1992), a well known algorithm for the induction of network structures, will be introduced in the following lines. It incorporates a Bayesian inspired score metric, coupled with a greedy strategy for the search process of the network structure. The search process starts by assuming that a node does not have parents, after which in every step it adds incrementally that parent whose addition most increases the probability of the resulting structure. The K2 algorithm stops adding parents to the nodes when the addition of a single parent can not increase the probability. Its basic scheme can be seen in Figure 3.5.

Algorithm K2

INPUT: A set of n nodes, an ordering on the nodes, an upper bound u on the number of parents a node may have, and a database D containing N cases
 OUTPUT: For each node, a printout of the parents of the node
 BEGIN K2
 FOR $i := 1$ TO n DO
 BEGIN
 $\mathbf{Pa}_i := \emptyset$
 $p_{\text{old}} := g(i, \mathbf{Pa}_i)$
 OKToProceed := TRUE
 WHILE OKToProceed AND $|\mathbf{Pa}_i| < u$ DO
 BEGIN
 Let Z be the node in $\text{Pred}(X_i) \setminus \mathbf{Pa}_i$ that
 maximizes $g(i, \mathbf{Pa}_i \cup \{Z\})$
 $p_{\text{new}} := g(i, \mathbf{Pa}_i \cup \{Z\})$
 IF $p_{\text{new}} > p_{\text{old}}$ THEN
 BEGIN
 $p_{\text{old}} := p_{\text{new}}$
 $\mathbf{Pa}_i := \mathbf{Pa}_i \cup \{Z\}$
 END
 ELSE OKToProceed := FALSE
 END;
 WRITE('Node:', X_i , 'Parents of this node:', \mathbf{Pa}_i)
 END
 END K2

Figure 3.5. The K2 algorithm.

Given a Bayesian network model, if the examples of the dataset occur independently, there are not missing values, and the density of the parameters given the structure is uniform, then the previous authors show that

$$p(D \mid S) = \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \quad (3.16)$$

The uniformity assumption of the parameters can be relaxed, allowing the user to employ Dirichlet distributions to specify prior probabilities of the parameters (Cooper and Herskovits, 1992). Apart from the algorithm, this metric is also known as the K2 metric for the scoring of Bayesian networks. The K2 algorithm assumes that an ordering on the variables is available and that, a priori, all structures are equally likely. It searches, for every node, the set of parent nodes that maximizes the following function (see Figure 3.5):

$$g(i, \mathbf{Pa}_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \quad (3.17)$$

In a generalization of the K2 metric, Heckerman et al. (1995) derive the BD (Bayesian Dirichlet) metric. It allows the prior probability functions for the Bayesian network parameters given the

```

PLS
Find an ancestral ordering,  $\pi$ , of the nodes in the Bayesian network
For  $j = 1, 2, \dots, N$ 
  For  $i = 1, 2, \dots, n$ 
     $x_{\pi(i)} \leftarrow \text{generate a value from } p(x_{\pi(i)} | \mathbf{pa}_{\pi(i)})$ 
    
```

Figure 3.6. Pseudocode for the PLS method.

structure to be Dirichlet distributions. In the same work the authors also present the Bayesian Dirichlet equivalent metric (BDe), which has the property that the score of two structures that reflect the same set of conditional (in)dependencies is the same.

■ Scores based on information theory

The literature includes several scores that, inspired in the Information Theory (Shannon, 1948), they are able to compare two probability distributions ($p(\mathbf{x})$ and $p'(\mathbf{x})$). The Kullback-Leibler cross-entropy measure (Kullback, 1951) is possibly the most frequently used information-inspired score. It is computed as follows:

$$\begin{aligned}
 D_{K-L}(p(\mathbf{x}), p'(\mathbf{x})) &= \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{p'(\mathbf{x})} \\
 &= \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) - \sum_{\mathbf{x}} p(\mathbf{x}) \log p'(\mathbf{x}).
 \end{aligned} \tag{3.18}$$

It can be considered that the Kullback-Leibler measure represents the difference in the information contained in the actual distribution $p(\mathbf{x})$ and the information contained in the approximate distribution $p'(\mathbf{x})$.

Other works that propose scores based on information theory can be found in Chow and Liu (1968) (Maximum Weight Spanning Tree: it will be introduced in Chapter 4), Herskovits and Cooper (1990), Lam and Bacchus (1994) and Bouckaert (1995).

3.3.4 Simulation

The simulation of Bayesian networks can be considered as an alternative to the exact propagation methods developed in order to reason with the networks. The literature includes plenty works proposing different methods to perform this simulation. Several interesting references can be found in Bouckaert et al. (1996), Jensen et al. (1993) and Salmeron et al. (2000).

One of the most known and intuitive simulation procedures is the Probabilistic Logic Sampling (PLS) method proposed by Henrion (1988). PLS (see Figure 3.6) takes advantage of how the Bayesian network defines the probability distribution. The values for each variable are generated in a forward way following their ancestral ordering, that is, a variable is sampled after all its parents have been already sampled. An ancestral ordering of the variables guarantees that the values for $\mathbf{Pa}_{\pi(i)}$ are assigned before $X_{\pi(i)}$ is sampled. Once the values of $\mathbf{Pa}_{\pi(i)}$ are assigned, the simulation for the value of $X_{\pi(i)}$ is performed using the distribution $p(x_{\pi(i)} | \mathbf{pa}_{\pi(i)})$.

3.4 Gaussian networks

3.4.1 Introduction

In this section a probabilistic graphical model paradigm called Gaussian networks (Shachter and Kenley, 1989) will be introduced. Instead of Bayesian networks, this kind of networks are used when the variables of the problem are continuous. The joint density function is factorized following a multivariate Gaussian density (Whittaker, 1990), reducing the number of parameters needed to specify this multivariate Gaussian density.

3.4.2 Basic notation for Gaussian networks

When each variable of the problem $X_i \in \mathbf{X}$ is continuous, it is assumed that it follows a local density function by the linear-regression model:

$$f(x_i | \mathbf{pa}_i^S, \boldsymbol{\theta}_i) \equiv \mathcal{N}(x_i; m_i + \sum_{x_j \in \mathbf{pa}_i} b_{ji}(x_j - m_j), v_i) \quad (3.19)$$

where $\mathcal{N}(x; \mu, \sigma^2)$ is a univariate normal distribution with mean μ and variance σ^2 . Given this form, a missing arc from X_j to X_i implies that $b_{ji} = 0$ in the former linear-regression model. The local parameters are given by $\boldsymbol{\theta}_i = (m_i, \mathbf{b}_i, v_i)$, where $\mathbf{b}_i = (b_{1i}, \dots, b_{i-1i})^t$ is a column vector. A probabilistic graphical model constructed by these local density functions is known as a Gaussian network (Shachter and Kenley, 1989).

The interpretation of the components of the local parameters is as follows: m_i is the unconditional mean of X_i , v_i is the conditional variance of X_i given \mathbf{Pa}_i , and b_{ji} is a linear coefficient reflecting the strength of the relationship between X_j and X_i . See Figure 3.7 for an example of a Gaussian network.

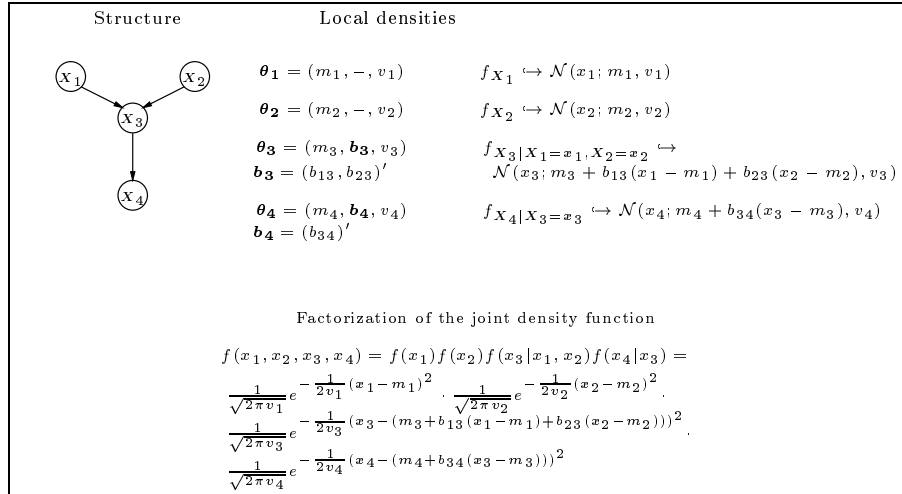


Figure 3.7. Structure, local densities, and resulting factorization for a Gaussian network with four variables.

Close to Gaussian networks, the multivariate normal density paradigm appears. The joint probability density function of the continuous n -dimensional variable \mathbf{X} is a multivariate normal distribution if:

$$f(\mathbf{x}) \equiv \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) \equiv (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})} \quad (3.20)$$

where $\boldsymbol{\mu}$ is the vector of means, Σ is an $n \times n$ covariance matrix, and $|\Sigma|$ denotes the determinant of Σ . The inverse of this matrix, $W = \Sigma^{-1}$, whose elements are denoted by w_{ij} , is referred to as the precision matrix. In this way, this density function can be written as a product of n conditional densities by the following formula:

$$f(\mathbf{x}) = \prod_{i=1}^n f(x_i | x_1, \dots, x_{i-1}) = \prod_{i=1}^n \mathcal{N}(x_i; \mu_i + \sum_{j=1}^{i-1} b_{ji}(x_j - \mu_j), v_i) \quad (3.21)$$

where μ_i is the unconditional mean of X_i , v_i is the variance of X_i given X_1, \dots, X_{i-1} , and b_{ji} is a linear coefficient reflecting the strength of the relationship between variables X_j and X_i (DeGroot, 1970). This notation gives us the possibility of interpreting a multivariate normal distribution as a Gaussian network where there is an arc from X_j to X_i whenever $b_{ji} \neq 0$ with $j < i$.

3.4.3 Model induction

In the next lines the basic automatic approaches for Gaussian network construction are introduced. Similarly to Bayesian networks, the methods to induce the structure and density functions are classified into two groups: methods based on the testing for the edge exclusion versus score+search methods. Among score+search approaches, two different score types are presented: as in the case of Bayesian networks, one penalizes the maximum likelihood metric and the other is a Bayesian inspired score.

3.4.3.1 Edge exclusion tests

This approach is based on the idea of testing whether the edge connecting the vertices corresponding to X_i and X_j in the conditional independence graph can be eliminated. Hence, such tests are known as edge exclusion tests. Many graphical model selection procedures start by making the $\binom{n}{2}$ single edge exclusion tests evaluating the likelihood ratio statistic and comparing it to a χ^2 distribution. Interesting works in this field appear in Dempster (1972), Speed and Kiiveri (1986) and Smith and Whittaker (1998).

3.4.3.2 Score+search methods

Following the idea of Bayesian networks, the main idea under this approach consists of having a measure for each candidate Gaussian network in combination with an intelligent search through the space of different network structures.

Search approaches

All the comments expressed for Bayesian networks are also valid in the case of Gaussian networks. In fact, Algorithm B or variants of the general local search strategy introduced in the Bayesian network section can be also applied in this case.

Score metrics

■ *Penalized maximum likelihood*

Denoting by $L(D \mid S, \boldsymbol{\theta})$ the likelihood of the database $D = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ given the Gaussian network model $M = (S, \boldsymbol{\theta})$, we have that:

$$L(D \mid S, \boldsymbol{\theta}) = \prod_{r=1}^N \prod_{i=1}^n \frac{1}{\sqrt{2\pi v_i}} e^{-\frac{1}{2v_i}(x_{ir} - m_i - \sum_{x_j \in \mathbf{pa}_i} b_{ji}(x_{jr} - m_j))^2}. \quad (3.22)$$

The maximum likelihood estimates for $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n)$, denoted as $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\theta}}_1, \dots, \hat{\boldsymbol{\theta}}_n)$, are obtained by maximizing $L(D \mid S, \boldsymbol{\theta})$ or equivalently maximizing $\ln L(D \mid S, \boldsymbol{\theta})$. The expression for the latter is:

$$\ln L(D \mid S, \boldsymbol{\theta}) = \sum_{r=1}^N \sum_{i=1}^n \left[-\ln(\sqrt{2\pi v_i}) - \frac{1}{2v_i} (x_{ir} - m_i - \sum_{x_j \in \mathbf{pa}_i} b_{ji}(x_{jr} - m_j))^2 \right]. \quad (3.23)$$

In this way, the maximum likelihood estimate can be penalized as follows:

$$\sum_{r=1}^N \sum_{i=1}^n \left[-\ln(\sqrt{2\pi v_i}) - \frac{1}{2v_i} (x_{ir} - m_i - \sum_{x_j \in \mathbf{pa}_i} b_{ji}(x_{jr} - m_j))^2 \right] - f(N) \dim(S). \quad (3.24)$$

The number of parameters, $\dim(S)$, needed to specify a Gaussian network model with a structure given by S can be obtained with the following formula:

$$\dim(S) = 2n + \sum_{i=1}^n |\mathbf{pa}_i|. \quad (3.25)$$

For each variable, X_i , it is needed to specify its mean, m_i , its conditional variance, v_i , and its regression coefficients, b_{ji} . The comments done about $f(N)$ in Section 3.3.2 are also valid in this case.

■ *Bayesian scores. Marginal likelihood*

BGe (Bayesian Gaussian equivalent) is a continuous version of the BDe metric (Heckerman et al., 1995) for Gaussian networks. A Bayesian approach is used to induce the network from data, expressing our uncertainty on the model (structure and parameters). As BDe, this metric also verifies the interesting property of score equivalence. This means that two Gaussian networks that represent the same set of conditional (in)dependence assertions receive the same score. The metric is based in the fact that the prior joint density of the network parameters is the normal-Wishart density. This fact allows us to obtain a closed formula for the computation of the log marginal likelihood of the data given the proposed DAG structure of the network in the following form:

$$L(D \mid S) = \prod_{i=1}^n \frac{L(D^{X_i \cup \mathbf{pa}_i} \mid S_c)}{L(D^{\mathbf{pa}_i} \mid S_c)} \quad (3.26)$$

where each term is of the form given in Equation 3.27, $D^{X_i \cup \mathbf{Pa}_i}$ is the database D restricted to the variables $X_i \cup \mathbf{Pa}_i$, and S_c represents a complete network structure. Combining the results provided by the theorems given in DeGroot (1970) and Geiger and Heckerman (1994), we obtain:

$$L(D \mid S_c) = (2\pi)^{-\frac{nN}{2}} \left(\frac{\nu}{\nu + N} \right)^{\frac{n}{2}} \frac{c(n, \alpha)}{c(n, \alpha + N)} |T_0|^{\frac{\alpha}{2}} |T_N|^{-\frac{\alpha+N}{2}} \quad (3.27)$$

where ν , α and T_0 are implicit functions of the user's background knowledge. T_0 and T_N can be considered as precision matrices when no cases and N cases are available, respectively, and both matrices follow a normal-Wishart density function. ν and α parameters also reflect the user's prior knowledge. While ν can be thought of as being an equivalent sample size for the vector of means, α can be interpreted as the user's equivalent sample size for the precision matrix T_0 (being $\alpha > n - 1$).

On the other hand, the $c(n, \alpha)$ function is defined as follows:

$$c(n, \alpha) = \left[2^{\frac{\alpha n}{2}} \pi^{\frac{n(n-1)}{4}} \prod_{i=1}^n \Gamma\left(\frac{\alpha + 1 - i}{2}\right) \right]^{-1}. \quad (3.28)$$

This result yields a metric for scoring the marginal likelihood of any Gaussian network. The reader interested in more details can consult Geiger and Heckerman (1994).

3.4.4 Simulation

Close to the idea of the ancestral ordering proposed in the PLS method (see Section 3.3.4), the book by Ripley (1987) proposes a method for the simulation of Gaussian networks known as the conditioning method. This procedure generates instances of \mathbf{X} by computing X_1 , then X_2 conditional on X_1 , and so on. For the simulation of an univariate normal distribution, a simple and well-known method based on the sum of 12 uniform variables can be applied.

3.5 Summary

In this chapter two probabilistic graphical models –Bayesian networks and Gaussian networks– have been introduced. Both paradigms are basic for the development of Estimation of Distribution Algorithms, which will be presented in the next chapter of this dissertation.

The most known model induction methods for both paradigms have been presented. For both types of networks, induction methods are divided into two types: procedures based on detecting conditional (in)dependencies and procedures based on a score+search point of view. Within the second group, principal search strategies and score metrics for model induction have been studied. The simulation task for both kinds of networks have been also reviewed.

Notes

- 1 Given $\mathbf{Y}, \mathbf{Z}, \mathbf{W}$ three disjoint sets of variables, it is said that \mathbf{Y} is *conditionally independent* of \mathbf{Z} given \mathbf{W} if for any $\mathbf{y}, \mathbf{z}, \mathbf{w}$ we have $\rho(\mathbf{y} \mid \mathbf{z}, \mathbf{w}) = \rho(\mathbf{y} \mid \mathbf{w})$. If this is the case, we denote $I(\mathbf{Y}, \mathbf{Z} \mid \mathbf{W})$. In the case of the conditional dependence $D(\mathbf{Y}, \mathbf{Z} \mid \mathbf{W})$ is denoted.
- 2 \mathbf{Pa}_i^S represents the set of parents –variables from which an arrow that ends in X_i comes out– of the variable X_i in the probabilistic graphical model with structure given by S .

Chapter 4

Estimation of Distribution Algorithms

4.1 Overview of the chapter

In this chapter a novel population-based, randomized, evolutionary paradigm, called Estimation of Distribution Algorithms (EDAs) will be introduced. This paradigm will be used in Chapters 6 and 7 to solve two classic optimization problems that arise in Supervised Classification.

As EDAs are based on probabilistic graphical models, this chapter is strongly related with the previous one. Once the motivations for the birth of EDAs are exposed, the principal EDA approaches that appear in the literature for discrete and continuous domains are explained. As discrete EDA approaches are more extensively used than continuous ones in this dissertation, more emphasis is put on discrete procedures.

This chapter is an adaptation and extension of the work by Larrañaga (2001b).

4.2 The need of the EDA paradigm

Many combinatorial optimization algorithms have no mechanism for capturing the relationships among the variables of the problem. The related literature has many papers proposing different heuristics in order to implicitly capture these relationships. Genetic Algorithms (GAs) tries to implicitly capture these relationships by a semi-blind process, concentrating samples on combinations of high-performance members of the current population through the use of the recombination (crossover and mutation) operators (Michalski, 2000):

- the crossover operator semi-randomly combines the information contained within pairs of selected ‘parent’ solutions by placing random subsets of each parent’s bits into their respective positions in a new ‘child’ solution;
- the mutation operator is a random modification of a current solution.

In GAs no explicit information is kept about which groups of variables jointly contribute to the quality of candidate solutions. It is often assumed that GAs are an efficient method to solve problems where variables interact, finding independently these groups of variables and properly mixing them afterwards (van Kemenade, 1998). However, as crossover and mutation operations are randomized, they could disrupt many of these ‘desired’ relationships among the variables (Inza et al., 2001a). Although the search process could produce an individual which covers an optimal relation among a subset of variables, a crossover or mutation operator could break this. Therefore,

most of the crossover and mutation operations yield unproductive results and the discovery of the global optima could be delayed.

On the other hand, GAs are also criticized in the literature for three aspects (Larrañaga et al., 2000b):

- the large number of parameters and their associated referred optimal selection or tuning process. If the researcher does not have experience in using this type of approach for the resolution of a concrete optimization problem, then the choice of suitable values for these parameters is itself converted into an optimization problem, as was shown by Grefenstette (1986);
- the extremely difficult prediction of the movements of the populations in the search space;
- their incapacity to solve the well known deceptive problems (Goldberg, 1989).

The identification of the groups of variables (or Building Blocks) that are related to produce ‘good’ solutions is an interesting way to improve the results of GAs.

De Bonet (1997) describes the need to take into account the variable relations during the optimization process: *‘In many optimization problems, the structure of solutions reflects complex relationships between the different input parameters. For example, experience may tell us that certain parameters are closely related and should not be explored independently... Any search of the cost landscape should take advantage of these relationships’*.

In this way, a group of techniques grouped under the Messy Genetic Algorithm (MGA) (Goldberg et al., 1989) term appears in the late 1980s. They are based on manipulating the representation of solutions in the search algorithm to make the found relationships less likely to be broken by recombination operators. The values of the variables in the search process are not the only thing that is optimized as it was with classic GAs. The representation of the problem (the positions of the variables) and other features are also optimized in the mGA process. For this purpose, a large set of new recombination operators and a not fixed-length representation are used to tightly code related variables. In this way, the objective is to reduce the chance for the disruption of these relationships during the recombination process. This approach works quite well for decomposable problems although they require some prior knowledge about the problem and they are usually very memory and time consuming (Pelikan and Mühlenbein, 1999). Nevertheless, this kind of genetic approaches have been often designed considering the particular characteristics of the problem to be optimized, constraining its use to narrow domains.

A different way to cope with the disruption of partial solutions is to change the basic principle of recombination. Instead of extending the GA, the idea of the explicit discovery of these relationships during the optimization process itself has emerged from the roots of the GA community. One way to discover these relationships is to estimate the joint distribution of promising solutions and use this estimate in order to generate new individual. A general scheme of the algorithms based on this principle is called the Estimation of Distribution Algorithms (EDAs) (Mühlenbein and Paaß, 1996) (Larrañaga and Lozano, 2001). In EDAs, there are no crossover nor mutation operators, and the new population is sampled from a probability distribution which is estimated from the selected individuals. A basic scheme of the EDA approach can be seen in Figure 4.1 and Figure 4.2. At the beginning M individuals are generated at random, for example, from a uniform distribution (discrete or continuous) for each variable. These M individuals constitute the initial population, D_0 , and each of them is evaluated. In a first step, a number N ($N \leq M$) of individuals are selected (usually those with the best objective function values). Next, induction of the n -dimensional probabilistic model that best reflects the interdependencies between the n variables is carried out. In a third step, M

new individuals (the new population) are obtained from simulation of the probability distribution learned in the previous step. The previous three steps are repeated until a stopping condition is verified. Examples of stopping conditions are: when a fixed number of populations or a fixed number of different evaluated individuals are achieved, uniformity in the generated population, no improvement with respect to the best individual obtained in the previous generation, etc.

EDA

$D_0 \leftarrow$ Generate M individuals (the initial population) randomly
repeat for $l = 1, 2, \dots$ **until** a stop criterion is met
 $D_{l-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{l-1} according to a selection method
 $\rho_l(\mathbf{x}) = \rho(\mathbf{x}|D_{l-1}^{Se}) \leftarrow$ Estimate the generalized joint density of selected individuals being among the selected individuals
 $D_l \leftarrow$ Sample M individuals (the new population) from $\rho_l(\mathbf{x})$

Figure 4.1. Main scheme of the EDA approach.

Before the explanation of principal EDA approaches, it is necessary to mention the novel research line proposed under the Learnable Evolution Model (LEM) (Michalski, 2000). LEM employs Machine Learning to guide the generation of new populations in an evolutionary process. In each generation of the search, a classification model that discriminates among highly and lowly fitted individuals is induced. Then, by an instantiation process of the induced rules, this classification model is used to create a new generation of individuals.

As the motivations for the birth of this novel approach, they argue the semi-blind nature of the GAs and that *‘in GAs, the generation of new individuals is not guided by lessons learned from the past generations, but is a form of a trial and error process executed in parallel. Consequently, computational processes based on Darwinian evolution tend to be inefficient’*. The similarities between the reasons that motivate the birth of LEM and EDAs seem very similar. In contrast to GAs, both paradigms not only take into consideration the ‘experience’ of single individuals for the generation of new ones: the experience of an entire population or a collection of populations is the engine that guides the search and the creation of new individuals. Both paradigms try to outperform the semi-blind nature of GAs by means of the explicitly expression of the experience learned during the optimization process. As LEM tries to express this knowledge by a supervised classification model, EDAs use a probabilistic-based framework.

4.3 EDA approaches in discrete domains

As we have seen in the previous section, the estimation of the probability distribution of best individuals in each generation of the search is a critical task for the EDA’s success. As in this section the principal EDA approaches that appear in the literature for discrete domains are reviewed, we organize them with respect to the complexity of the probabilistic model used to learn the interdependencies among the variables of the domain. In this way, the literature includes approaches that assumes the independence among the features of the problem, approaches that cover pairwise interactions and those that reflect multivariate dependencies. Apart from the essence of the probability estimation, in spite of all the approaches have minor variations with respect to the basic EDA scheme presented in the Figure 4.1, all presented algorithms can be grouped under the EDA paradigm. For each specific algorithm, these differences with respect to the main EDA scheme are

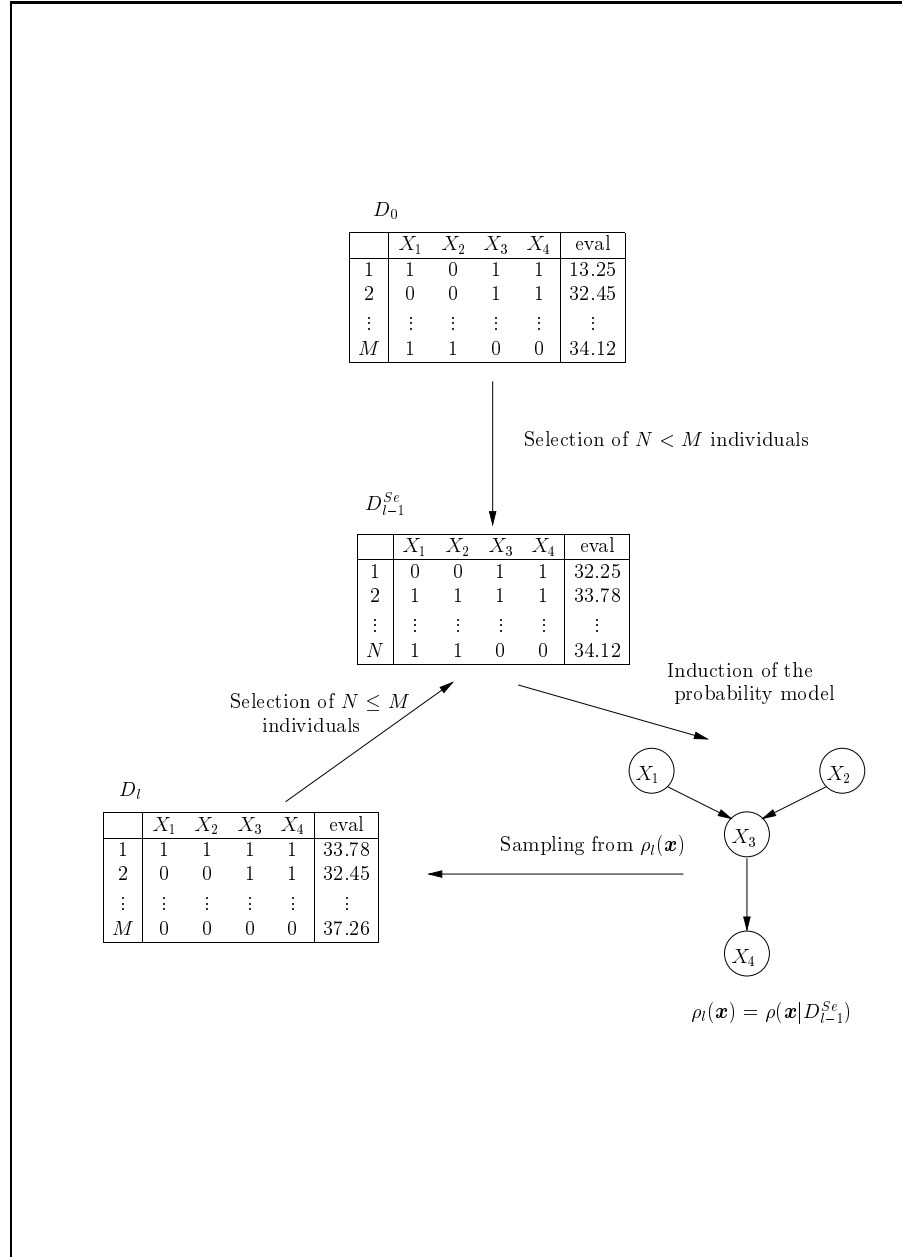


Figure 4.2. Illustration of the EDA approach to optimization.

marked: differences in the selection scheme, differences in the stop criterion, etc. As these variations can be modified and adapted to the basic EDA scheme of Figure 4.1, for each algorithm our main interest resides in the way they learn the probability distribution of best individuals from which the next population is sampled.

It must be noted that the sampling of new individuals from the learned probability distribution in discrete domains is performed by the use of Probabilistic Logic Sampling (PLS) (Henrion, 1988). It finds an ancestral ordering of the variables in the learned probabilistic structure, generating the values for the variables following this order.

4.3.1 Without dependencies

The simplest way to estimate the distribution of good solutions assumes the independence between the features of the problem (the terms ‘variable’ and ‘feature’ are used indistinctly). New candidate solutions are sampled by only regarding the proportions of the values of the variables independently to the remaining ones. It is assumed that the n -dimensional joint probability distribution factorizes as a product of n univariate and independent probability distributions. That is $p_l(\mathbf{x}) = \prod_{i=1}^n p_l(x_i)$ (see Figure 4.3 for a graphical representation). Obviously this assumption is not real in many optimization problems, where interdependencies between the variables usually exist.

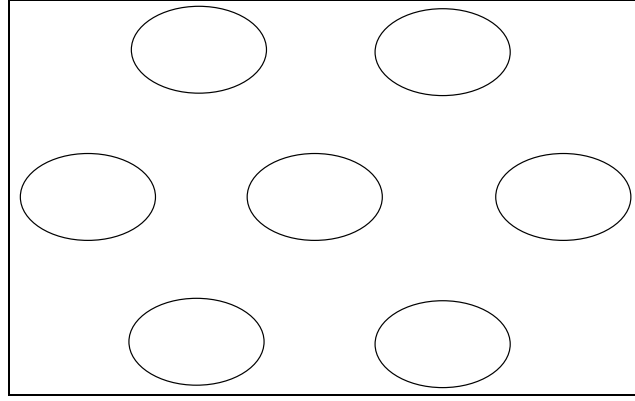


Figure 4.3. Graphical representation of the probability model proposed by EDAs without dependencies.

4.3.1.1 UMDA

UMDA (Univariate Marginal Distribution Algorithm) (Mühlenbein, 1998) estimates the joint probability distribution of selected individuals in the simplest possible way. That is:

$$p_l(\mathbf{x}) = p(\mathbf{x} | D_{l-1}^{Se}) = \prod_{i=1}^n p_l(x_i). \quad (4.1)$$

Each univariate marginal distribution is estimated from marginal frequencies:

$$p_l(x_i) = \frac{\sum_{j=1}^N \delta_j(X_i = x_i | D_{l-1}^{Se})}{N} \quad (4.2)$$

where

$$\delta_j(X_i = x_i | D_{l-1}^{Se}) = \begin{cases} 1 & \text{if in the } j^{th} \text{ case of } D_{l-1}^{Se}, X_i = x_i \\ 0 & \text{otherwise.} \end{cases} \quad (4.3)$$

Applications and works about UMDA appear in:

- Alba et al. (2000) in a Feature Subset Selection problem;
- Blanco et al. (2001) in a Bioinformatics problem for the selection of accurate genes;
- Rivera (1999) in the construction of a classifier system;
- Santana and Ochoa (1999) and Santana et al. (2000) modify the simulation phase;
- Mahnig and Mühlenbein (2000) and Mühlenbein and Mahnig (2000) carry out a mathematical analysis of UMDA.

4.3.1.2 BSC

In order to estimate the probability distribution of selected individuals, BSC (Bit-Based Simulated Crossover) (Syswerda, 1993) takes into account their evaluation function values. As in the case of UMDA, the joint probability distribution is factorized as a product of independent univariate marginal distributions (see Equation 4.1). BSC estimates each marginal distribution proportionally with respect to the evaluation function of those selected individuals with the specific marginal value. That is:

$$p_l(x_i) = \frac{\sum_{\{\mathbf{x} | \delta_j(X_i=x_i | D_{l-1}^{se})=1\}} ef(\mathbf{x})}{\sum_{\{\mathbf{x} \in D_{l-1}^{se}\}} ef(\mathbf{x})}. \quad (4.4)$$

Taking into account the meaning of the δ_j function explained in Equation 4.3, $\sum_{\{\mathbf{x} | \delta_j(X_i=x_i | D_{l-1}^{se})=1\}} ef(\mathbf{x})$ represents the sum of the evaluation function values of the individuals with value x_i in the variable X_i . On the other hand, the term $\sum_{\{\mathbf{x} \in D_{l-1}^{se}\}} ef(\mathbf{x})$ is the sum of the evaluation value of all selected individuals.

Applications and works about BSC appear in:

- Inza et al. (2001d) in a Feature Subset Selection problem within a medical domain;
- Inza et al. (2002a) in Feature Subset Selection problems of large dimensionality;
- Roure et al. (2001) in partitional clustering task.

4.3.1.3 PBIL

PBIL (Population Based Incremental Learning) (Baluja, 1994; Baluja and Caruana, 1995) represents the population of individuals by a vector of probabilities:

$$p_{l-1}(\mathbf{x}) = (p_{l-1}(x_1), \dots, p_{l-1}(x_i), \dots, p_{l-1}(x_n)) \quad (4.5)$$

where $p_{l-1}(x_i)$ refers to the probability of obtaining a value of 1 in the i^{th} component of D_{l-1} , the population of individuals in the $(l-1)^{th}$ generation. The vector of probabilities is randomly initialized in the first search generation.

At each generation, by simulation of the probability vector $p_{l-1}(\mathbf{x})$, a set of M individuals, which can be considered as the new population, is obtained. Each of these M individuals is evaluated and the N best of them ($N \leq M$) are selected. Selected individuals are denoted them by:

$$\mathbf{x}_{1:M}^{l-1}, \dots, \mathbf{x}_{i:M}^{l-1}, \dots, \mathbf{x}_{N:M}^{l-1}.$$

These selected individuals are used, by a Hebbian inspired rule, to update the probability vector from which the next population of individuals will be sampled:

$$p_l(\mathbf{x}) = (1 - \alpha)p_{l-1}(\mathbf{x}) + \alpha \frac{1}{N} \sum_{k=1}^N \mathbf{x}_{k:M}^{l-1} \quad (4.6)$$

where $\alpha \in (0, 1]$ is a parameter of the algorithm. In the case that $\alpha = 1$, PBIL coincides with UMDA. This process of adaptation of the vector of probabilities continues until the vector of probabilities has converged. Equation 4.6 is used when problem variables are binary.

In order to estimate the joint probability distribution from which the next population of individuals is sampled, the major part of EDA approaches only takes into account the subset of selected individuals and discard the subset of not selected individuals. In contrast to this framework, in order to calculate the probability vector from which the l -th generation of individuals is simulated, $p_l(\mathbf{x})$, it must be noted that PBIL takes into account the probability vector of the $l - 1$ -th generation, $p_{l-1}(\mathbf{x})$ (see Equation 4.6).

Applications and works about PBIL appear in:

- Inza et al. (2001d) in a Feature Subset Selection problem within a medical domain;
- Höhfeld and Rudolph (1997) and González et al. (2000) carry out theoretical studies about PBIL;
- Maxwell and Anderson (1999) and Gallagher (2000) apply PBIL to the problem of obtaining optimal weights between nodes in a neural network architecture.

4.3.1.4 cGA

Similarly to PBIL, the essence of the cGA (compact Genetic Algorithm) (Harik et al., 1998) resides in a vector of probabilities, which maintains the way in which the search will be directed. The cGA algorithm begins by initializing the vector of probabilities for each component following a Bernoulli distribution with parameter 0.5. Next, two individuals are generated randomly from this vector of probabilities. After the individuals are evaluated, a competition between them is carried out. The competition is held at the level of each of the unidimensional variables, in such a way that if for the i^{th} position the conquering individual takes a value different from the loser, then the i^{th} component of the vector of probabilities increases or diminishes by a constant amount which depends on whether the i^{th} position of the conquering individual was a one or a zero. This process of adaptation of the vector of probabilities towards the winning individual continues until the vector of probabilities has converged. Figure 4.4 shows a pseudocode for the cGA in the case of binary representations (it can be easily adapted to larger representation spaces).

4.3.2 Bivariate dependencies

Although the estimation of the probability distribution of best individuals can be quickly done without assuming dependencies among the variables of the domain, this assumption is far from the reality in many problems. This can be improved, by the aid of second-order statistics, trying to cover interactions between pairs of variables.

Step 1. Initialize the probability vector $p_0(\mathbf{x})$
 $p_0(\mathbf{x}) = p_0(x_1, \dots, x_i, \dots, x_n) = (p_0(x_1), \dots, p_0(x_i), \dots, p_0(x_n)) = (0.5, \dots, 0.5, \dots, 0.5)$

Step 2. $l = l + 1$. Sampling $p_l(\mathbf{x})$ with $l = 0, 1, 2, \dots$ obtain two individuals: $\mathbf{x}_1^l, \mathbf{x}_2^l$

Step 3. Evaluate and rank \mathbf{x}_1^l and \mathbf{x}_2^l obtaining:
 $\mathbf{x}_{1:2}^l$ (the best of both) and $\mathbf{x}_{2:2}^l$ (the worst of both)

Step 4. Update the probability vector $p_l(\mathbf{x})$ towards $\mathbf{x}_{1:2}^l$
 for $i = 1$ to n
 if $x_{i,1:2}^l \neq x_{i,2:2}^l$ then
 if $x_{i,1:2}^l = 1$ then $p_l(x_i) = p_{l-1}(x_i) + \frac{1}{K}$
 if $x_{i,1:2}^l = 0$ then $p_l(x_i) = p_{l-1}(x_i) - \frac{1}{K}$

Step 5. Check if the probability vector $p_l(\mathbf{x})$ has converged
 for $i = 1$ to n do
 if $p_l(x_i) > 0$ and $p_l(x_i) < 1$ then
 return to Step 2

Step 6. $p_l(\mathbf{x})$ represents the final solution

Figure 4.4. Pseudocode for the cGA.

While for EDA approaches that do not cover dependencies among the variables of the domain the model structure is fixed, for EDA approaches that try to capture pairwise dependencies a structure learning is needed.

4.3.2.1 MIMIC

In MIMIC (Mutual Information Maximization for Input Clustering) (De Bonet et al., 1997), the joint probability distribution is factorized by a chain structure. In order to carry out this chain-scheme factorization in each generation, MIMIC searches for the best permutation between the variables in order to find the probability distribution, $p_l^\pi(\mathbf{x})$, that is closest to the empirical distribution of the set of selected individuals, $p_l(\mathbf{x})$, when using the Kullback-Leibler distance, where

$$p_l^\pi(\mathbf{x}) = p_l(x_{i_1} | x_{i_2}) \cdot p_l(x_{i_2} | x_{i_3}) \cdots p_l(x_{i_{n-1}} | x_{i_n}) \cdot p_l(x_{i_n}) \quad (4.7)$$

and $\pi = (i_1, i_2, \dots, i_n)$ denotes a permutation of the indexes of the problem variables $(1, 2, \dots, n)$.

However, the problem of finding the optimal permutation π^* among the variables remains unanswered. An interesting property to reply to this question is that the Kullback-Leibler divergence between two probability distributions, $p_l(\mathbf{x})$ and $p_l^\pi(\mathbf{x})$, as expressed in the previous equation, is a function of:

Step 1. Choose $i_n = \arg \min_j \hat{h}_l(X_j)$

Step 2. **for** $k = n - 1, \dots, 1$

Choose $i_k = \arg \min_{j \neq i_{k+1}, \dots, i_n} \hat{h}_l(X_j \mid X_{i_{k+1}})$

Step 3. $p_l^\pi(\mathbf{x}) = p_l(x_{i_1} \mid x_{i_2}) \cdot p_l(x_{i_2} \mid x_{i_3}) \cdots p_l(x_{i_{n-1}} \mid x_{i_n}) \cdot p_l(x_{i_n})$

Figure 4.5. The MIMIC approach to estimation of the joint probability distribution at generation l . The symbols $\hat{h}_l(X)$ and $\hat{h}_l(X \mid Y)$ denote the empirical entropy of X and the empirical entropy of X given Y respectively. Both are estimated from $D_l^{S^e}$.

$$H_l^\pi(\mathbf{x}) = h_l(X_{i_n}) + \sum_{j=1}^{n-1} h_l(X_{i_j} \mid X_{i_{j+1}}) \quad (4.8)$$

where $h(X)$ denotes the Shannon entropy of the X variable and $h(X \mid Y)$ denotes the Shannon entropy of the variable X when the values of the variable Y are known. In this way, the problem of searching for the best $p_l^\pi(\mathbf{x})$ becomes equivalent to the problem of searching for the permutation π^* that minimizes $H_l^\pi(\mathbf{x})$. To find π^* , De Bonet et al. (1997) propose a greedy algorithm. The idea is to select X_{i_n} as the variable with the smallest entropy then, at every following step, to pick the variable –from the set of variables not chosen so far– whose average conditional entropy with respect to the variable selected in the previous step is the smallest. A pseudocode that facilitates the understanding of the exposed probability estimation algorithm can be shown in Figure 4.5.

Applications and works about MIMIC appear in:

- Robles et al. (2001) in the Traveling Salesman Problem (TSP);
- Inza et al. (2002a) in Feature Subset Selection problems of large dimensionality;
- de Campos et al. (2001) in the partial abductive inference problem in Bayesian networks.

4.3.2.2 TREE

We denote by TREE the EDA inspired algorithm that factorizes the probability distribution of selected individuals using a tree structured Bayesian network learned using the algorithm proposed by Chow and Liu (1968). In the original work, the authors call the probability estimation algorithm the Maximum Weight Spanning Tree (MWST). In each generation, TREE induces the optimal dependency tree structure in the sense that among all possible trees, it learns the probabilistic tree structure, $p_l^t(\mathbf{x})$, that best approximates the probability distribution of the set of selected individuals, $p_l(\mathbf{x})$. To obtain $p_l^t(\mathbf{x})$, the authors use the Kullback-Leibler cross-entropy measure as a distance criterion between $p_l(\mathbf{x})$ and $p_l^t(\mathbf{x})$, that is, $D_{K-L}(p_l(\mathbf{x}), p_l^t(\mathbf{x}))$. This distance is minimized by projecting $p_l^t(\mathbf{x})$ on any MWST, where the weight on the branch (X_i, X_j) is defined by the mutual information measure:

-
- Step 1. Compute the pairwise distributions, $p_l(x_i, x_j)$, for all variable pairs
They are estimated from D_l^{Se}
- Step 2. Using the pairwise distributions, compute all $n(n-1)/2$ branch weights and order them by magnitude
- Step 3. Assign the largest two branches to the tree to be constructed
- Step 4. Examine the next-largest branch, and add it to the tree unless it forms a loop, in which case discard it and examine the next-largest branch
- Step 5. Repeat Step 4 until $n-1$ branches have been selected (at this point the spanning tree has been constructed)
- Step 6. $p_l^t(\mathbf{x})$ can be computed by selecting an arbitrary root node and forming the product: $p_l^t(\mathbf{x}) = \prod_{i=1}^n p_l(x_i | x_{j(i)})$
-

Figure 4.6. The TREE approach to estimation of the joint probability distribution at generation l .

$$I_l(X_i, X_j) = \sum_{x_i, x_j} p_l(x_i, x_j) \log \frac{p_l(x_i, x_j)}{p_l(x_i)p_l(x_j)}. \quad (4.9)$$

Mutual information measures and univariate and bivariate distributions are estimated from the set of selected individuals, D_l^{Se} . A pseudocode that facilitates the understanding of the exposed probability estimation algorithm can be shown in Figure 4.6.

Applications and works about TREE appear in:

- Inza et al. (2001d) in a Feature Subset Selection problem within a medical domain;
- Inza et al. (2002a) in Feature Subset Selection problems of large dimensionality;
- Sierra et al. (2001a) in a rule induction task.

4.3.2.3 COMIT

Baluja and Davies (1998) propose the COMIT (Combining Optimizers with Mutual Information Trees) algorithm, which hybridizes the bivariate EDA approach with other optimizers. Estimation of the probability distribution of the selected individuals in each generation is done using the tree structure approach proposed by Chow and Liu (1968) and exposed for the TREE algorithm. A set of new individuals is sampled from the learned probabilistic model. The best of these individuals are considered as initial points for other fast-search procedure: Baluja and Davies (1998) use the hill-climbing and PBIL procedures. Some of the best individuals obtained during these fast-searches are added to the individuals selected in the previous generation, to create the new population of individuals. The authors show good results for COMIT in different well known problems, such as the Traveling Salesman Problem, the Jobshop Scheduling Problem and the Knapsack Problem.

4.3.2.4 BMDA

Pelikan and Mühlenbein (1999) propose an interesting EDA approach that captures bivariate dependencies. Their approach, BMDA (Bivariate Marginal Distribution Algorithm), is based on the construction of a dependency graph, which is always acyclic but in contrast to the TREE

algorithm, it does not necessarily have to be a connected graph. This dependency graph can be seen as a set of trees that are not mutually connected (a forest of trees). First, an arbitrary variable is chosen and added as a node of the graph. Second, in a similar way to Chow and Liu (1968) MWST algorithm, it is needed to add to the graph the variable with the greatest dependency –measured by Pearson’s χ^2 statistic– between any of the previously incorporated variables and the set of not yet added variables. This last step is repeated until there is no dependency surpassing a previously fixed threshold between already added variables and the rest. If this is the case, a variable is chosen at random from the set of variables not yet used. The whole process is repeated until all variables are added into the dependency graph.

In each generation the factorization obtained with the BMDA is given by:

$$p_l(\mathbf{x}) = \prod_{X_r \in R_l} p_l(x_r) \prod_{X_i \in V \setminus R_l} p_l(x_i \mid x_{j(i)}) \quad (4.10)$$

where V denotes the set of n variables, R_l denotes the set containing the root variable –in generation l – for each of the connected components of the dependency graph, and $X_{j(i)}$ returns the variable connected to the variable X_i and added before X_i .

The probabilities for the root nodes, $p_l(x_r)$, as well as the conditional probabilities, $p_l(x_i \mid x_{j(i)})$, are estimated from the database, D_{l-1}^{Se} , containing the selected individuals.

Pelikan and Mühlenbein (1999) show good results for BMDA in different benchmark fitness functions.

Figure 4.7 shows a graphical representation of the probabilistic models proposed by EDA approaches with pairwise dependencies.

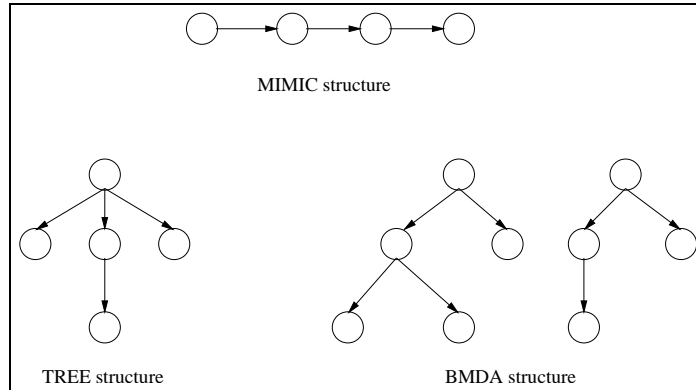


Figure 4.7. Graphical representation of the probability models proposed by EDAs with pairwise dependencies.

4.3.3 Multivariate dependencies

Covering only bivariate interactions has shown to be insufficient to properly solve problems where interactions of order greater than two exist among the variables of the task (Bosman and Thierens, 1999; Pelikan et al., 1999). Figure 4.8 shows the basic graphical representation of the principal EDA multivariate approaches.

In a similar way to bivariate approaches, the structure learning process to capture these multivariate dependencies is a crucial task. However, taking into account the evolutionary nature of

the EDA process, the complexity of the structure learning procedure must be controlled to avoid a computational overload.

4.3.3.1 EcGA

Harik (1999) presents the EcGA (Extended compact Genetic Algorithm). In this algorithm, the variables are divided into a number of intact clusters which are manipulated as independent variables in the UMDA. Each cluster is taken as a whole and different clusters are considered to be mutually independent (see Figure 4.8). It means that, in each generation, the factorization of the joint probability distribution is done as a product of marginal distributions of variable size. These marginal distributions of variable size are related to the variables that are contained in the same cluster and to the probability distributions associated with them. The grouping is carried out using a greedy forward algorithm that obtains a partition between the n variables.

The greedy algorithm that carries out the grouping begins with a partition with n clusters (a variable in each cluster). Then, the algorithm performs the union of the two variables that results in the greatest reduction of a measure that conjugates the sum of the entropies of the marginal distributions with a penalty for the complexity of the model based on the minimum description length principle (MDL) (Rissanen, 1978).

Harik (1999) shows good results for EcGA in different benchmark fitness functions.

4.3.3.2 FDA

Mühlenbein et al. (1999) present the FDA (Factorized Distribution Algorithm), which uses a factorized distribution as a fixed model throughout the whole computation. The FDA is not capable of learning the structure of a problem in the fly, and it needs the factorization and decomposition of the task given by an expert. This is usually not available when solving real problems, and therefore the use of FDA is limited to problems where the structure of the task can be at least accurately approximated. Distributions contain marginal and conditional probabilities which are updated according to the currently selected subset of best solutions. The authors show the superiority of the FDA in problems where the given model is correct and decomposable in the exposed form.

4.3.3.3 BOA

Pelikan et al. (1999) present the BOA (Bayesian Optimization Algorithm) procedure, which uses Bayesian networks to estimate the probability distribution of selected individuals. A score+search procedure is used to construct the Bayesian network structure by means of the BDe (Heckerman et al., 1995) score metric and the B search algorithm (Buntine, 1991) (starting in each generation from scratch). A complexity control factor is used, constraining the maximum number of parents of a node in the Bayesian network.

Several works (Pelikan et al., 1999; Pelikan and Goldberg, 2000; Schwarz and Ocenasek, 1999) show good results for BOA in different kinds of domains.

4.3.3.4 EBNA

Etzeberria and Larrañaga (1999) propose the EBNA (Estimation of Bayesian Networks Algorithm), where the factorization of selected individuals in each EDA generation is performed by a Bayesian network. In the first EBNA implementation a score+search procedure was used to construct the Bayesian network structure by means of the BIC (Schwarz, 1978) score metric and the B search algorithm (Buntine, 1991) (starting in each generation from scratch). Figure 4.9 shows the general procedure of EBNA.

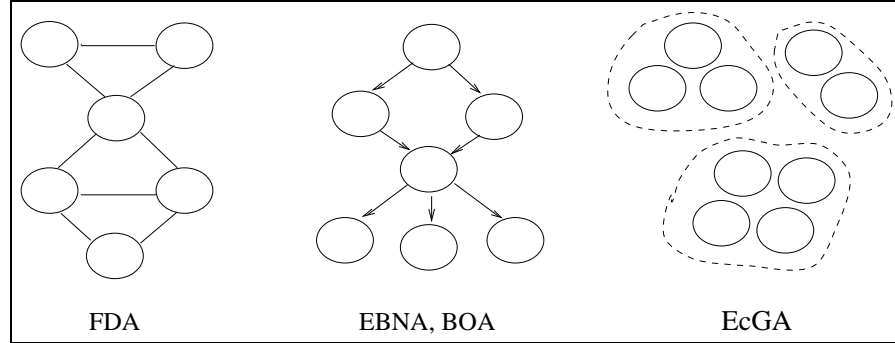


Figure 4.8. Graphical representation of the probability models proposed by EDAs with multiple dependencies.

EBNA

$BN_0 \leftarrow (M_0, \theta_0)$

$D_0 \leftarrow$ Sample N individuals from BN_0

repeat for $l = 1, 2, \dots$ **until** a stop criterion is met

$D_{l-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{l-1} according to a selection method

$\hat{M}_l \leftarrow$ Find the structure which maximizes $BIC(M_l, D_{l-1}^s)$

$\theta_l \leftarrow$ Calculate $\{\theta_{ijk}^l = \frac{N_{ijk}^{l-1} + 1}{N_{ij}^{l-1} + r_i}\}$ using D_{l-1}^s as the data set

$BN_l \leftarrow (\hat{M}_l, \theta_0)$

$D_l \leftarrow$ Sample M individuals from BN_l using PLS

Figure 4.9. EBNA basic scheme.

Two variations of this basic EBNA scheme appear in the literature, proposing other approaches for the construction of the Bayesian networks:

- instead of the BIC metric, a penalization of the K2 score is proposed (Larrañaga et al., 2000b). In this way, the number of parents that one node can have is bound;
- by the testing of conditional (in)dependencies the PC algorithm (Spirtes et al., 1991) is proposed to construct the Bayesian network.

Applications and works about EBNA appear in:

- Inza et al. (2000) in Feature Subset Selection problems of medium dimensionality;
- Inza et al. (2001c) in Feature Weighting problems for K-NN;
- Sierra et al. (2001a) in a rule induction task;
- Robles et al. (2001) in the Traveling Salesman Problem.
- Lozano et al. (2001) propose a parallel version of EBNA;
- Bengoetxea et al. (2002) in the graph matching task.

4.4 EDA approaches in continuous domains

Although the EDA paradigm in continuous domains is not extensively used in this dissertation, the basic and principal EDA approaches that appear in the literature for continuous domains are reviewed in this section. The section is similarly organized to the previous one, grouping different approaches by the complexity of the covered interdependencies between the variables of the domain. In this way, the literature includes approaches where the density function is factorized as a product of n univariate marginal densities, approaches that use two order densities and those that consider multivariate dependencies.

It must be noted that the sampling of new individuals from the learned density function in continuous domains is performed by the use of the continuous version of Probabilistic Logic Sampling (PLS) (Henrion, 1988).

4.4.1 Without dependencies

The simplest way to estimate the distribution of selected solutions does not take into account dependencies between the variables of the problem. In this kind of domains with continuous variables, it is assumed that the joint density function follows a n -dimensional normal distribution, which is factorized by a product of unidimensional and independent normal densities. Using the mathematical notation, $\mathbf{X} \equiv \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma)$, this can be formulated in the following form:

$$f_{\mathcal{N}}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \prod_{i=1}^n f_{\mathcal{N}}(x_i; \mu_i, \sigma_i^2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}\left(\frac{x_i - \mu_i}{\sigma_i}\right)^2}. \quad (4.11)$$

4.4.1.1 UMDA_c

Larrañaga et al. (1999) and Larrañaga et al. (2000) introduce the UMDA_c (Univariate Marginal Distribution Algorithm) procedure for continuous domains. In order to find the density function that best fits each variable of the domain, in each EDA generation and for every variable a set of statistical tests is carried out. In spite of the discrete case of the UMDA procedure, in UMDA_c a structure identification approach is carried out, identifying the density components of the model via hypothesis tests. Once the density functions are identified, the estimation of the associated parameters is performed by their maximum likelihood estimates. In the particular case that all univariate distributions are normal, the procedure is known as UMDA_c^G (Univariate Marginal Distribution Algorithm for Gaussian models). González et al. (2002) present a theoretic study of this algorithm.

4.4.1.2 SHCLVND

SHCLVND (Stochastic Hill Climbing with Learning by Vectors of Normal Distributions) (Rudlof and Köppen, 1996) estimates the joint density function as a product of unidimensional and independent normal densities. The vector of means $\boldsymbol{\mu} = (\mu_1, \dots, \mu_i, \dots, \mu_n)$ is adapted by means of the Hebbian rule:

$$\boldsymbol{\mu}^{(l+1)} = \boldsymbol{\mu}^l + \alpha \cdot (\mathbf{b}^l - \boldsymbol{\mu}^l) \quad (4.12)$$

where $\boldsymbol{\mu}^{(l+1)}$ denotes the vector of means in the generation $l + 1$, α denotes the learning rate, and \mathbf{b}^l denotes the baricenter of the N (an amount fixed at the beginning) best individuals in the l^{th} generation. Adaptation of the vector of variances $\boldsymbol{\sigma}$ is carried out using a reduction policy in the following way:

$$\sigma^{(l+1)} = \sigma^l \cdot \beta \quad (4.13)$$

where β denotes a previously fixed constant ($0 < \beta < 1$).

4.4.1.3 PBIL_c

Sebag and Ducoulombier (1998) present the continuous version of the PBIL algorithm for continuous spaces, named PBIL_c. As in the case of SHCLVND, PBIL_c also assumes that the joint density function follows a Gaussian distribution factorizable as a product of unidimensional and independent marginal densities. The adaptation of each component of the vector of means is carried out using the following formula:

$$\mu_i^{(l+1)} = (1 - \alpha) \cdot \mu_i^{(l)} + \alpha \cdot (x_{i,1:N}^l + x_{i,2:N}^l - x_{i,N:N}^l) \quad (4.14)$$

where $\mu_i^{(l+1)}$ represents the i^{th} component of the mean, $\mu^{(l+1)}$, at generation $(l + 1)$, and $x_{i,1:N}^l ((x_{i,2:N}^l)(x_{i,N:N}^l))$ denote the i -th component of the best ((second best) (worst)) individual of the generation l , and α is a constant. See the work of Sebag and Ducoulombier (1998) for the details of four heuristics for the adaptation of the vector of variances.

4.4.1.4 Servet et al

Servet et al. (1997) introduce a progressive approach to estimate the univariate density functions. At each generation, and for each dimension i , an interval (a_i^l, b_i^l) and a real number z_i^l ($i = 1, \dots, n$) are stored. z_i^l represents the probability that the i^{th} component of a solution is on the right half of the previous interval. In every generation the probabilities, z_i^l , for each dimension are calculated, and when z_i^l is closer to 1(0) than a previously fixed quantity, the interval is reduced to its right (left) half.

4.4.2 Bivariate dependencies

4.4.2.1 MIMIC_c^G

As far we know, the unique bivariate approach in continuous domains is the MIMIC_c^G (Mutual Information Maximization for Input Clustering) (Larrañaga et al., 1999; Larrañaga et al., 2000) procedure. It constitutes an adaptation of the MIMIC algorithm proposed by De Bonet et al. (1997) for this kind of problems. The underlying probability function for every pair of variables is assumed to follow a bivariate Gaussian function.

As in the case of MIMIC algorithm for discrete domains, the joint density function is factorized by a chain structure, fitting the model as closely as possible to the empirical data by using only one univariate marginal density and $n - 1$ pairwise conditional density functions. See Larrañaga and Lozano (2001) for details in the structure learning of MIMIC_c^G.

4.4.3 Multivariate dependencies

The literature shows two approaches for continuous domains in which the density function learnt at each EDA generation is not restricted: multivariate normal density functions and Gaussian networks.

EGNA

For $l = 1, 2, \dots$ until the stopping criterion is met

$D_{l-1}^{Se} \leftarrow$ Select N individuals from D_{l-1}

(i) $\hat{S}_l \leftarrow$ Structural learning via:

Bayesian score+search \rightarrow EGNA

(ii) $\hat{\theta}^l \leftarrow$ Calculate the estimates for the parameters of \hat{S}_l

(iii) $M_l \leftarrow (\hat{S}_l, \hat{\theta}^l)$

(iv) $D_l \leftarrow$ Sample M individuals from M_l using the continuous version of the PLS algorithm

Figure 4.10. EGNA basic scheme.

4.4.3.1 EMNA

Larrañaga et al. (1999) present the EMNA (Estimation of Multivariate Normal Algorithm) procedure, where the factorization of selected individuals in each EDA generation is performed by a multivariate normal density function. The estimation of the vector of means and the variance-covariance matrix is performed by their maximum likelihood estimates.

The authors propose two variations of this basic scheme for the construction of the multivariate normal density function:

- in an adaptive version of the basic EMNA, the flow of the population resembles a steady-state GA. After a new individual is simulated from the current multivariate normal density model, its evaluation function is compared with the goodness of the worst individual of the population. If the fitness of the new individual is better than the fitness of the worst one, this worst individual is replaced in the population by the newly simulated one. When a replacement occurs, the associated parameters of the multivariate normal density function must be updated;
- in an incremental approach of the basic EMNA one individual is also generated from each model. The difference with respect to the adaptive approach is that each simulated individual is added to the population without any restriction. In this way, the amount of individuals in the population increases as the algorithm evolves.

4.4.3.2 EGNA

Larrañaga et al. (1999) and Larrañaga et al. (2000) propose the EGNA (Estimation of Gaussian Networks Algorithm), where the factorization of selected individuals in each EDA generation is performed by a Gaussian network. In the first EGNA implementation a Bayesian score+search procedure was used to construct the Gaussian network structure by means of the BGe (Geiger and Heckerman, 1994) score metric and the B search algorithm (Buntine, 1991) (starting in each generation from scratch). Figure 4.10 shows the general procedure of EGNA.

Two variations of this basic EGNA scheme appear in the literature, proposing other approaches for the construction of the Gaussian networks:

- instead of the BGe metric, a penalized maximum likelihood score based on the BIC criterion (Schwarz, 1978) is used;

- the use of edge exclusion tests, as explained in Chapter 3, is proposed to construct the Gaussian network.

Applications and works about EGNA appear in:

- Cotta et al. (2001) in the task of adjusting weights for artificial neural networks;
- Inza et al. (2001c) in Feature Weighting problems for K-NN;
- Mendiburu and Lozano (2001) in the Job Shop scheduling problem.

4.5 Summary

In this chapter the EDA paradigm has been presented, a novel tool for evolutionary computation. First, the principal reasons that motivated its apparition have been introduced, so closely related with an attempt to overcome several known drawbacks of GAs.

The principal EDA approaches for discrete and continuous domains that appear in the literature have been presented, with special emphasis on discrete ones. Approaches have been divided with respect to the order of the variable relationships they cover. For each approach, theoretical and practical references have been pointed out.

Chapter 5

Representing the Joint Behaviour of Supervised Learning Algorithms by Bayesian Networks

5.1 Overview of the chapter

In this chapter, an approach to study, in a joint manner, the nature of the classification models induced by a large set of well known supervised Machine Learning algorithms is proposed. As the objective is not to discover which classifier induces the most accurate classification model in a specific domain, the classifier's accuracy will not be used to characterize the nature of the induced classification model. Instead of the predictive accuracy, with the use of Bayesian networks, the values of the class labels predicted by a set of supervised classifiers will be jointly studied. Using the *conditional independencies* expressed by the Bayesian networks, assertions about the joint behaviour and nature of classification models induced by a large set of supervised Machine Learning paradigms will be extracted.

After a brief introduction about the motivations of this chapter, the characteristics of the proposed modelization process will be presented. By the use of the conditional independence concept, three kinds of relations among the classification models will be studied. After the presentation of these relations, the results obtained over a set of medical datasets will be exposed.

This chapter is an adaptation and extension of the work by Inza et al. (1999).

5.2 Motivation: accuracy versus class label

Within the Machine Learning discipline, in the 80's and 90's, there has been a considerable growth in the number of available supervised classification algorithms (Mitchell, 1997). Nowadays, in an attempt to detect the classification procedure which biases are best suited to the data, the user is frequently faced with the problem of selecting the ideal algorithm for a specific dataset. However, the ambitious objective of generating or selecting a unique winner algorithm for all tasks has been rejected by the empirical evidence of the 'No Free Lunch Theorem' (Kohavi and John, 1997). The selection of the 'best' algorithm, usually only based on the error percentage, had the effect of mainly focusing the attention of the Machine Learning community on predictive accuracy. A no-end career can be felt in Machine Learning, with the aim of constructing the algorithm with the highest predictive accuracy for each dataset. In this way, Clark (1998) mentioned us the obsession of the Machine Learning community with summary statistics, and accuracy in special.

Our objective is not to compare or study supervised learning algorithms from the point of view of their goodness. Our aim is to study the nature of the models induced by supervised learning

algorithms. In this way, the error or the accuracy percentage of the generated model is not the appropriate reference to guide the study. Instead of the predictive accuracy, the class predictions are used as the external expression of the model induced by the supervised learning algorithm. The probability distribution of the class predictions of a set of well known learning algorithms over a dataset are studied in a joint manner by means of a Bayesian network, displaying the joint behaviour and the relations between the models induced by the algorithms.

5.3 Methodology

5.3.1 Learning algorithms and datasets

Fourteen well known supervised learning algorithms are used in the experimentation process. Algorithms with different biases, many of them arising from traditional Machine Learning paradigm families. This is the list of selected algorithms (see Chapter 2 for details about them): ID3, C4.5, OC1, T2, NB, NBTree, HOODG, IB1, IB4, PEBLS, Table-Majority, OneR, CN2 and Ripper. Some of them can be grouped in the following Machine Learning paradigm families (Mitchell, 1997):

- Decision-Trees (DT): Quinlan's decision tree classic algorithms, ID3 and C4.5;
- Simple-Trees (ST): one-depth (OneR) and two-depth (T2) simple classification trees, both inspired in the PAC-learning theory. They minimize the number of errors in the training set, based on their constraints;
- K-NN (K-NN): algorithms based on the Nearest Neighbor paradigm, IB1 and IB4;
- Decision-Rules (DR): classic IF-THEN decision rule algorithms, Ripper and CN2;
- simple Bayesian Classifiers (BC) that assume the independence of the features to predict the class: NB and NBTree.

Although the rest of the algorithms (OC1, HOODG, PEBLS, Table-Majority) could be related in some way with several of the exposed families, based on their biases and classic Machine Learning review texts (Michie et al., 1994; Michalski et al., 1998; Mitchell, 1997), these relations are not considered 'solid' or 'strong' enough to include the algorithms in a specific family or to form a new one.

In an attempt to perform an homogeneous study, eleven medical databases from the UCI Machine Learning Repository (Murphy, 1995) are selected. As in this work there is not an 'accuracy' or 'goodness' measure of the obtained results, the selection of the datasets from a specific domain (medical, in this case) could be a way to increase the coherence and robustness of the study and its results. Selected datasets are: *Breast cancer*, *Breast cancer (Wisconsin)*, *Cleveland*, *Diabetes (Pima)*, *Echocardiogram*, *Heart disease*, *Hepatitis*, *Hungarian*, *Hypothyroid*, *Liver (BUPA)* and *Lymphography*. See Chapter 2 for more details about the domains.

5.3.2 Modelization by Bayesian networks and the simplification process

Once the problem domains and learning algorithms to be studied are fixed, the process to extract assertions about the joint behaviour and nature of learning procedures can be explained.

All datasets are split in training and test data in a 2 : 1 proportion. For each domain and algorithm, a classification model is induced using the training set. The model induced from the training set is used to classify the instances of the test set, and then, the class label predictions for

these test instances are saved in a new working file (or matrix). In this way, when all proposed algorithms are run over a dataset, a new file can be constructed where the rows represent the instances of the test set and the columns represent the learning algorithms: thus, the (i, j) position of the constructed file shows the class label prediction made by the j -th learning algorithm for the i -th test instance.

For a domain, each column of this file shows the class labels predicted by a specific classification model for all test instances. On the other hand, each row of the file shows the class labels predicted by all the classification models for a test instance. It must be noted that the real class label of each test instances does not appear in this file; neither does any accuracy measure of each model. This file can be a solid point to start the study of the joint behaviour of proposed supervised learning algorithms. In an intuitive way, it can be assumed that this file saves the nature and the behaviour of the studied classifiers in the proposed files.

The MLC++ library of programs (Kohavi et al., 1997b) is used to execute the supervised learning algorithms. All algorithms are run with their default parameters. It must be noted that when an algorithm execution is performed with a set of fixed parameters, this process is deterministic for all algorithms except for OC1. No special treatment for missing values is carried out, exploiting for each algorithm its own characteristics. As PEBLS and HOODG algorithms are not able to handle missing values, they are only executed in the four datasets without missing values (*Diabetes*, *Heart disease*, *Liver* and *Lymphography*).

The constructed file is used as the input to automatically induce a Bayesian network. The network structure is induced by a score+search procedure. The implementation proposed in Larrañaga et al. (1996) is used: with the Bayesian K2 metric, the search is conducted by an elitist Genetic Algorithm, hybridized with a local optimizer.

After the induction of the Bayesian networks, we discovered that the network structures had a extremely high level of connectivity, presumably indicating an ‘overfitting’ to the data (Quinlan, 1989). This could be due to the fact that ten of eleven experimented datasets (all except *Hypothyroid*) have less than 300 test instances (this absence of data is a typical characteristic in medical domains). As these highly connected structures (we will know it as the ‘overfitted’ structure) are not an appropriate starting point to extract assertions about the possible conditional independence incidences among the learning algorithms, an intuitive simplification process of the network structure is conducted in each domain (due to the large number of instances of the *Hypothyroid* domain, this simplification procedure is not needed for this dataset):

- an ordered list of the arcs appearing in the ‘overfitted’ network is assembled. The order reflects the influence of the arc in the overall K2 metric of the network structure: the influence of each arc is calculated by the difference of subtracting to the K2 score of the ‘overfitted’ structure the K2 score value of the ‘overfitted’ structure when the studied arc is removed;
- half of the most influential arcs are maintained. Half of the least influential arcs are removed;
- then, a *Recover procedure* is started for the removed arcs: taking into account two of the least important arcs maintained in the network, we consider recovering the removed most important arc. If the influence of this removed arc is smaller than half of the mean of the influence of the two least important arcs maintained then, the *Recover procedure* is stopped. Otherwise, the removed arc is recovered in the network and the recovering of the next removed arc is considered (taking into account the arc recently recovered in the network structure). Continue until stop.

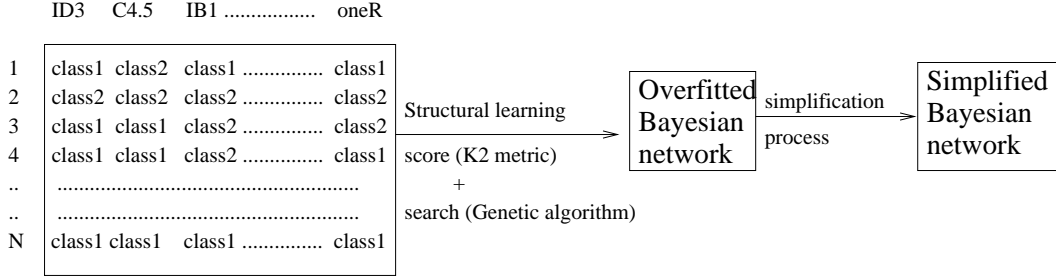


Figure 5.1. Proposed modelization process for each dataset. We start from the class predictions of Machine Learning inducers for the test instances. Let assume that N is the number of instances in test set, from which the Bayesian network is induced.

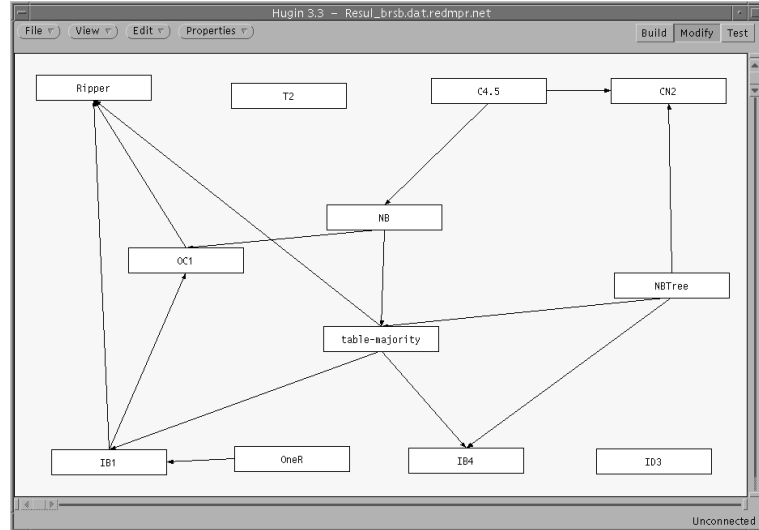


Figure 5.2. Simplified Bayesian network for the *Breast cancer* domain.

With this simplification procedure, interpretable structures are achieved. Intuitively testing the significance degree of each arc in the network structure, the *Recover procedure* can be seen as a heuristic of the likelihood ratio statistic test. Starting from the file with the predictions of the classifiers for the test set of a specific domain, Figure 5.1 summarizes the explained process. It must be noted that this process is individually performed for each domain. As two examples, the induced simplified Bayesian network graphical structures for *Breast cancer* and *Cleveland* domains can be seen in Figures 5.2 and 5.3, respectively.

Experiments are run on a SUN-SPARC machine and Hugin software (Andersen et al., 1989) is used for the management of Bayesian networks.

5.3.3 Concepts for interpreting the joint behaviour of classifiers

Once the Bayesian networks are induced, our aim is to extract assertions and guidelines about the joint behaviour of Machine Learning inducers: assertions on the similarities and dissimilarities

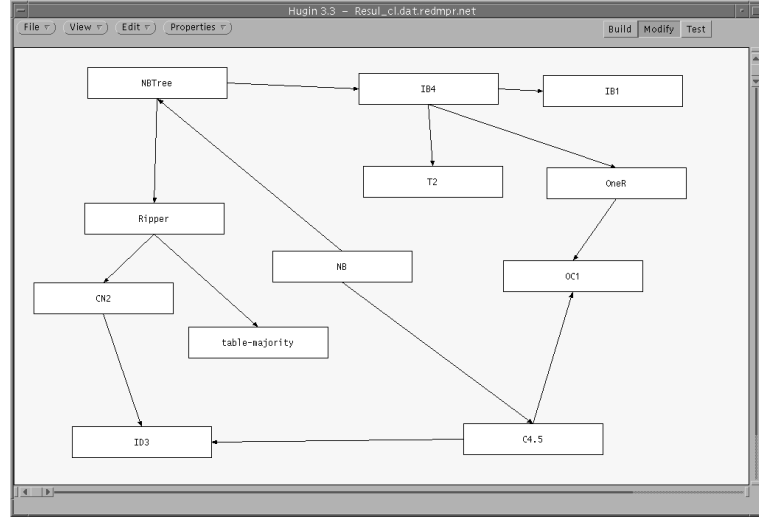


Figure 5.3. Simplified Bayesian network for the *Cleveland* domain.

among classification models, using the possibility of studying the models in the joint manner offered by Bayesian networks. As Bayesian networks' semantic is based on the conditional (in)dependence concept, three variations or adaptations of this concept are used to extract assertions about the relations between the nature of the classification models induced by studied supervised learning algorithms. In the next lines these concepts are explained.

5.3.3.1 Hard conditional independence

As *hard conditionally independent* we consider a learning algorithm that in a Bayesian network, given another algorithm, is conditionally independent of the rest of the algorithms. That is:

Let \mathbf{W} represent the whole set of algorithms. Classifier X is hard conditionally independent given algorithm Y if and only if $I(X, \mathbf{W} \setminus \{X, Z\} | Y)$ occurs for any algorithm Z .

This condition does not easily occur in a Bayesian network. It implies that, giving only another algorithm, the hard conditionally independent one has a different behaviour regarding the other algorithms. From the point of view of Bayesian networks and probability distributions, it can be considered that the hard conditionally independent algorithm creates an 'original' classification model regarding the other algorithms: it has little relation to the models generated by the major part of the other algorithms, reflecting a different behaviour with respect to them. The probability distribution of its classification model needs just another algorithm to be explained in the whole representation of probability distributions induced by the Bayesian network.

5.3.3.2 Conditional independence in a proposed family of algorithms

Our aim is also to study, in a joint manner, the 'union-degree' of a proposed family of learning algorithms. Five family sets proposed in this chapter are studied: Decision-Trees (DT), Simple-Trees (ST), K-NN, Decision-Rules (DR) and simple Bayesian classifiers (BC). The conditional independence between two algorithms of the same family given another algorithm from a different family is studied in the induced Bayesian networks. That is:

Let X, Y and Z be three different algorithms and X and Y be from the same family and Z from another family. Is $I(X, Y|Z)$ true or is $D(X, Y|Z)$ true?

This type of conditional independence implies a serious difference between the probability distributions of both algorithms of the same family. From the point of view of Bayesian networks and probability distributions, when this property occurs in a family, it can be considered that the classification models of both algorithms of the family show significant differences. This kind of relation gives us an idea about the ‘compactness’ (or similarity degree) between algorithms of a proposed family. It can be a way to analyze the ‘solidity’ of the families of algorithms traditionally proposed in the literature.

5.3.3.3 Conditional independence between proposed families

Using the proposed family definitions, our aim is to study the conditional independence of all the learning algorithms of a family on all the algorithms of another family, given any group of algorithms which do not belong to any of the compared families. That is:

Let \mathbf{X} and \mathbf{Y} be two families of algorithms and let \mathbf{Z} represent the rest of the algorithms. The question is whether \mathbf{X} and \mathbf{Y} are conditionally independent, given any subset of \mathbf{Z} .

This conditional independence between proposed families means that given any set of algorithms that do not belong to any of the compared families, the probability distribution of each component of a family does not need the probability distribution of any component of the other family to be explained. From the point of view of Bayesian networks and probability distributions, this kind of relation gives us an idea about the ‘dissimilarity’ and ‘similarity’ between proposed algorithms’ families.

5.4 Results from induced Bayesian networks

These three types of conditional (in)dependences are studied in the set of induced Bayesian networks, one for each of the proposed eleven medical datasets. As we do not work with accuracy percentages (or any other kind of metric), conclusions can not be extracted in the form of comparisons between accuracy percentages. Based on the amount of domains a learning algorithm or a set of algorithms presents one of the explained conditional independence concepts, assertions and tendencies about different types of behaviours among studied algorithms can be extracted. In this way, these assertions can be formulated in a comparative manner between (sets of) algorithms, based on the amount of datasets (Bayesian networks) that a (set) of learning algorithm(s) shows the proposed conditional (in)dependence relation concepts. It must be noted that the extracted conclusions and assertions are restricted to the experimented domains.

5.4.1 Hard conditional independence

The next list shows the number of Bayesian networks where each learning algorithm shows the hard conditionally independent property:

- ID3: 7;
- Table-Majority: 6;
- IB1, OneR, IB4, T2: 5;
- NBTree, C4.5, Ripper: 3;
- CN2, NB: 2;

- OC1, PEBLS, HOODG: 1.

Focusing our attention on the extremes of the list, two types of behaviours with respect to the exposed property can be marked:

- the trend of perfect memorizer algorithms¹(ID3 decision tree, IB1 and IB4 nearest neighbor algorithms and Table-Majority) and ST algorithms (OneR and T2) to show more times than the rest of the algorithms the hard conditionally independent property must be noted. These algorithms tend to appear with few connections in the learned Bayesian networks.

Perfect memorizers tend to form ‘original’ classification models with respect to the major part of algorithms. It seems that the absence of a pruning mechanism to ‘smooth’ the model differentiates these algorithms from the rest.

Similar assertions can be extracted from the behaviour of ST algorithms. These algorithms construct very simple models (OneR with a single feature and T2 with a pair of attributes), pruning and smoothing their classification rules in a large degree. It seems that the use of a powerful pruning mechanism, inducing very simple classification rules, differentiates these algorithms from the rest;

- the previous behaviour contrast with the nature of the models induced by BC classifiers (NB and NBTree) and algorithms that incorporate classic pruning strategies (OC1, C4.5, Ripper and CN2), pruning procedures not as strong as ST algorithms’ ones. These algorithms show the hard conditionally independent property in a lesser degree than perfect memorizers and ST procedures. They mainly appear with a higher number of connections than perfect memorizers and ST algorithms in the induced Bayesian networks.

They do not tend to form ‘original’ classification models with respect to the major part of algorithms. It seems that the Bayesian nature and the pruning mechanism of these procedures do not differentiate them from the rest.

5.4.2 Conditional independence in a proposed family of algorithms

The next lists shows the number of Bayesian networks where both algorithms of a family are conditionally independent, given at most one algorithm of another family:

- DT: 7;
- ST: 7;
- DR: 7;
- K-NN: 3;
- BC: 3.

Focusing our attention on the extremes of the list, two types of behaviours with respect to the exposed property can be marked:

- DT, ST and DR families show a lower degree of ‘compactness’ or similarity between their component algorithms than K-NN and BC families.

ID3 and C4.5 algorithms (DT family) tend to induce significantly different classification models in many datasets. As the main difference between both algorithms is the existence of a pruning

mechanism (C4.5 uses it and ID3 not), it seems that the use of the pruning procedure is the fundamental reason to differentiate the behaviour of this pair of familiar algorithms.

OneR and T2 algorithms (ST family) tend to induce significantly different classification models in many datasets. As the main difference between both algorithms is that OneR uses a single feature to induce the model (one-depth tree) and T2 selects a pair of attributes (two-depth tree), it seems that the use of the second variable (second stage in the classification tree) is the fundamental reason to differentiate the behaviour of this pair of familiar algorithms.

CN2 and Ripper algorithms (DR family) tend to induce significantly different classification models in many datasets. Although both algorithms are very similar and both incorporate a pruning strategy, their principal difference is that CN2 pre-prunes the rules and Ripper post-prunes them. In the greedy construction of each IF-THEN rule, CN2 analyzes the significance degree of a new attribute in the condition part by the use of statistical tests. On the other hand, Ripper largely expands the condition part of the rules in a first stage, post-pruning them in a second stage. It seems that these pre or post pruning mechanisms are the fundamental reason to differentiate the behaviour of this pair of familiar algorithms;

- previous families contrast with K-NN and BC families, which show a more ‘compact’ or similar behaviour between their component algorithms.

Although the main difference between NB and NBTree algorithms (BC family) is that NBTree first segments the set of instances by a tree structure, it does not seem that this segmentation is enough to differentiate in a large part of the datasets the behaviour of both Bayesian algorithms.

Although the main difference between IB1 and IB4 algorithms (K-NN) family is that IB4 incorporates a feature weighting mechanism, it does not seem that this weighting procedure is enough to differentiate in a large part of the datasets the behaviour of both K-NN algorithms.

5.4.3 Conditional independence between proposed families

The next lists shows the number of Bayesian networks where conditional independences between families of algorithms appear²:

- $I(BC, ST \mid -)$: 7
- $I(DT, K-NN \mid -)$: 6
- $I(ST, DT \mid -)$: 5
- $I(ST, DR \mid -)$: 5
- $I(BC, DT \mid -)$: 5
- $I(BC, K-NN \mid -)$: 4
- $I(DR, K-NN \mid -)$: 3
- $I(ST, K-NN \mid -)$: 3
- $I(DR, BC \mid -)$: 3
- $I(DT, DR \mid -)$: 2

Focusing our attention on the extremes of the list, the following behaviours can be marked:

- the notable similarity degree between the behaviour of DT and DR families can be pointed out. This related behaviour can have its roots in the top-down way where all DT and DR algorithms split the training instances to form the classification model.

The related behaviours between the DR vs. BC, ST vs. K-NN and DR vs. K-NN pairs of families can be also marked;

- the notable different behaviour between the BC vs. ST and DT vs. K-NN pairs of families can be marked.

Regarding the former list, for each family, assertions about its most similar and dissimilar families can be directly extracted.

5.5 Summary and future work

From an homogeneous set of databases, a study of the joint behaviour of the predictions made by a set of Machine Learning algorithms is carried out. Bayesian networks, induced from the learning algorithms class label predictions, were used to study the behaviour of a set of known algorithms. From the obtained Bayesian networks, guided by the conditional independence concept, properties of the algorithms and relations between the probability distributions of the classification models induced by different algorithms were found. Three different types of relations, adaptations of the conditional independence concept, have been studied:

- given an algorithm, the conditional independence of another algorithm with the rest of algorithms;
- conditional independence of two algorithms of the same family, given another algorithm from another family;
- conditional independence between families of algorithms, given any set of algorithms that do not belong to any of the considered families.

As future work, the use of unsupervised hierarchical classification (Lozano, 1998) can be used over the set of predictions of classifiers to determine clusters or new families of algorithms. Another interesting way could be the use of statistical tests to study the relations among the induced models. In this way, the induction of the Bayesian networks by the detection of conditional independencies (de Campos, 1998) could be an interesting approach.

Another exciting research way could be to study the relation between these differences in the joint behaviour of the learning algorithms and the success of the combination of classifiers (Dietterich, 1997; Sierra, 2000), which usually improve the classification accuracy percentage of single classifiers. Based on the a-priori studied joint behaviour of supervised algorithms, it could be an interesting way to decide which combinations of single learning algorithms could form a successful ensemble classifiers.

The use of this type of relationships in the field of genetic networks could be an exciting way. As Bayesian networks, genetic networks show the probabilistic relationships of a set of genes, which are represented by variables (or nodes) in the network structure. In the Bioinformatics discipline, the discover of the interactions and relations among genes is a crucial point.

Notes

1. OC1 incorporates a randomized component to form splitting hyperplanes: in the datasets tested, predictions of different 10 runs are nearly the same. The algorithm is run 10 times for each datasets, and the predictions of a randomly selected run are used.
2. Perfect memorizer algorithms' principal property is that they are very faithful with respect to the training set: in order to induce the model, pruning or smoothing mechanisms are used in a small degree (or they do not exist). K-NN algorithms are the classic representants of perfect memorizers: a model itself does not exist and training instances are maintained to be matched with further test instances. Table-majority operates in a similar way than K-NN algorithms. Although ID3 has a soft pre-pruning mechanism, it tends to form large trees, constructing leaves with instances of an unique class (in spite of very few training instances appear in the leaves).
3. By the '–' symbol, any set of algorithms that do not belong to any of the compared families is represented.

Chapter 6

Feature Subset Selection for Supervised Classification by Estimation of Distribution Algorithms

6.1 Overview of the chapter

Feature Subset Selection is a well known task in the Machine Learning, Data Mining, Pattern Recognition and Text Learning paradigms. In this chapter, we present a set of EDA inspired techniques to tackle the Feature Subset Selection problem in Machine Learning supervised tasks. Bayesian networks are used to factorize the probability distribution of best solutions in small and medium dimensionality datasets, and simpler probabilistic models are used in larger dimensionality domains. In a comparison with different sequential and traditionally used genetic-inspired algorithms in natural and artificial datasets, EDA-based approaches have encouraging accuracy results and a smaller number of evaluations than genetic approaches.

After an introduction, the Feature Subset Selection task and its basic components will be studied, performing a survey of previous approaches. The specific application of Bayesian networks to solve the Feature Subset Selection problem within the EDA paradigm for small and medium dimensionality scale domains will be presented. PBIL, BSC, MIMIC and TREE probabilistic algorithms will be employed in large dimensionality scale domains. For all approaches, results over natural and artificial domains will be presented. The chapter includes a study about the ‘overfitting’ problem when the Feature Subset Selection process is carried out, obtaining a basis to define the stopping criteria of the presented search algorithms.

This chapter is an adaptation of the works of Inza et al. (2000), Inza et al. (2001a), Inza et al. (2001b) and Inza et al. (2002a).

6.2 Motivation

In supervised Machine Learning and Data Mining processes, the goal of a supervised learning algorithm is to induce a classifier that allows us to classify new examples $E^* = \{\mathbf{e}_{N+1}, \dots, \mathbf{e}_{N+Q}\}$ that are only characterized by their n descriptive features. To generate this classifier we have a set of N samples $E = \{\mathbf{e}_1, \dots, \mathbf{e}_N\}$, each characterized by n descriptive features $\mathbf{X} = \{X_1, \dots, X_n\}$ and the class label $C = \{c_1, \dots, c_N\}$ to which they belong. The Machine Learning and Data Mining communities have formulated the following question: *Are all of these n descriptive features useful for learning the ‘classification rule’?* The Feature Subset Selection (FSS) approach tries to respond to this question and reformulates it as follows: *Given a set of candidate features, select the best subset under some learning algorithm.*

This dimensionality reduction produced by a FSS process has several advantages for a classification system on a specific task:

- a reduction in the cost of data acquisition;
- an improvement in the comprehensibility of the final classification model;
- faster induction of the final classification model;
- an improvement in classification accuracy.

The attainment of higher classification accuracies is the usual objective of supervised Machine Learning processes. It has long been proved that the classification accuracy of supervised classification algorithms is not monotonic with respect to the addition of features. Irrelevant or redundant features, depending on the specific characteristics of the learning algorithm, may degrade the predictive accuracy of the classification model. In this chapter, the objective of FSS will be maximization of the performance of the classification algorithm. In addition, with the reduction in the number of features, it is more likely that the final classifier is less complex and more understandable by humans (Inza et al., 2001d).

Once its objective is fixed, FSS can be viewed as a search problem, with each state in the search space specifying a subset of the possible features of the task. Exhaustive evaluation of possible feature subsets is usually infeasible in practice because of the large amount of computational effort required. Many search techniques have been proposed for solving the FSS problem when there is no knowledge about the nature of the task, by carrying out an intelligent search in the space of possible solutions. As they are randomized, evolutionary and population-based search algorithms, Genetic Algorithms (GAs) are possibly the most commonly used search engine in the FSS process.

As an alternative paradigm to GAs, in this chapter we propose the use of EDA-inspired techniques for the FSS task. The choice of the specific EDA-inspired algorithm which performs FSS depends on the initial dimensionality (number of features) of the domain. A FSS problem is considered small scale, medium scale or large scale (Kudo and Sklansky, 2000) if the number of features is less than 20, 20 – 49 or greater than 49, respectively. For small and medium scale domains we propose using the most attractive probabilistic paradigm, Bayesian networks, to factorize the probability distribution of the best solutions. For large scale domains, a large number of solutions is needed to induce a reliable Bayesian network, and we propose using simpler probabilistic structures: PBIL, BSC, MIMIC and TREE.

6.3 Feature Subset Selection: basic components

Our work is located in Machine Learning and Data Mining, but FSS literature includes numerous works from other fields such as Pattern Recognition (Ferri et al., 1994; Jain and Chandrasekaran, 1982; Jain et al., 2000; Kittler, 1978), Statistics (Miller, 1990; Narendra and Fukunaga, 1997), Bioinformatics' microarray gene expression (Blanco et al., 2001; Golub et al., 1999) unsupervised classification (Peña et al., 2001; Talavera, 2000) and Text-Learning (Mladenić, 1998; Yang and Pedersen, 1997). Thus, different research communities have exchanged and shared ideas on dealing with the FSS problem. A good review of FSS methods can be found in Liu and Motoda (1998).

The objective of FSS in a supervised Machine Learning or a Data Mining classification framework (Aha and Bankert, 1994) is to *reduce the number of features used to characterize a dataset so as to improve a learning algorithm's performance on a given task*. Our objective is the maximization of classification accuracy in a specific task for a specific learning algorithm; as a side effect, we will have

ward elimination (Kittler, 1978), floating selection methods (Pudil et al., 1994) and best-first search (Kohavi and John, 1997). They are deterministic in the sense that their runs always obtain the same solution. Vafaie and De Jong (1993) results suggest that classic greedy hill-climbing approaches, tend to get trapped on local peaks caused by interdependencies among features. In this sense the work of Pudil et al. (1994) is an interesting idea in an attempt to avoid this phenomenon. *Non-deterministic heuristic* search is used to escape from local maxima. Randomness is used for this purpose and this implies that one should not expect the same solution from different runs. Up until now, the next non-deterministic search engines have been used in FSS: Genetic Algorithms (Eberhardt et al., 2001; Ferri et al., 1993; Guerra-Salcedo and Whitley, 1998; Kuncheva, 1993; Siedelecky and Sklansky, 1988; Vafaie and Jong, 1993; Yang and Honavar, 1998), Simulated Annealing (Doak, 1992), Las Vegas Algorithm (Liu and Setiono, 1996; Skalak, 1994). See Jain and Zongker (1997) or Liu and Motoda (1998) for other kinds of classifications of FSS search algorithms.

6.3.3 Evaluation strategy for feature subsets

The evaluation function identifies promising areas of the search space by calculating the goodness of each proposed feature subset. It measures the effectiveness of a particular subset after the search algorithm has chosen it for examination. Being the objective of the search its maximization, the search algorithm utilizes the value returned by the evaluation function to help guide the search. Many measures carry out this objective regarding only the characteristics of the data, capturing the relevance of each feature or set of features to define the target concept. As reported by John et al. (1994), when the goal of FSS is the maximization of the accuracy, the features selected should depend not only on the features and the target concept to be learned, but also on the learning algorithm. Kohavi and John (1997) report domains in which a feature, although being in the target concept to be learned, does not appear in the optimal feature subset that maximizes the predictive accuracy for the specific learning algorithm used. This occurs due to the intrinsic characteristics and limitations of the classifier used: feature relevance and accuracy optimality are not always coupled in FSS. The idea of using the error reported by a classifier as the feature subset evaluation criterion appears in previous works such as Stearns (1976) or Siedelecky and Sklansky (1988). However, the machine computation limitations of those years forced to stop its application. Doak in 1992 already used the classification error rate to guide non-large searches that did not require demanding machine computation. In John et al. (1994), the *wrapper* concept definitively appears. This implies that the FSS algorithm conducts a search for a good subset of features using the induction algorithm itself as a part of the evaluation function, the same algorithm that will be used to induce the final classification model. Once the classification algorithm is fixed, the idea is to train it with the feature subset encountered by the search algorithm, estimating the error percentage, and assigning it as the value of the evaluation function of the feature subset. In this way, representational biases of the induction algorithm used to construct the final classifier are included in the FSS process. The role of the supervised classification algorithm is the principal difference between the filter and wrapper approaches. The wrapper strategy needs a high computational cost, but technical computer advances in the last decade have made the use of this wrapper approach possible, calculating an amount of accuracy estimations (training and testing on significant amounts of data) not envisioned in the 80's.

Before applying the wrapper approach, an enumeration of the available computer resources is critical. Two different factors become a FSS problem 'large' (Liu and Setiono, 1998): the number of features and the number of instances. One must bear in mind that the learning algorithm of the wrapper scheme is trained for every solution visited by the FSS search engine. Many approaches have

been proposed in the literature to alleviate the loading of the training phase, such as Caruana and Freitag (1994) (avoiding the evaluation of many subsets taking advantage of the intrinsic properties of the used learning algorithm) or Moore and Lee (1994) (reducing the burden of the cross-validation technique for model selection).

When the learning algorithm is not used in the evaluation function, the goodness of a feature subset can be assessed regarding only the intrinsic properties of the data. The learning algorithm only appears in the final part of the FSS process to construct the final classifier using the set of selected features. The Statistics literature proposes many measures for evaluating the goodness of a candidate feature subset. See Ben-Bassat (1982) for a review of these classic measures. These statistical measures try to detect the feature subsets with higher discriminatory information with respect to the class (Kittler, 1978) regarding the probability distribution of data. These measures are usually monotonic and increase with the addition of features that afterwards can hurt the accuracy classification of the final classifier. In Pattern Recognition FSS works, in order to recognize the forms of the task, from a full set of n features, it is so common to fix a positive integer number d and select the best feature subset of d -cardinality found during the search. When this d parameter is not fixed a examination of the slope of the curve –value of the proposed statistical measure vs. cardinality of the selected feature subset– of the best feature subsets is required to select the cardinality of the final feature subset. This type of FSS approach, which ignores the induction algorithm to assess the merits of a feature subset, is known as the *filter* approach. Mainly inspired on these statistical measures, in the 90's and in our decade, more complex filter measures which do not use the final induction algorithm in the evaluation function generate new FSS algorithms, such as FOCUS (Almuallin and Dietterich, 1991), RELIEF (Kira and Rendell, 1992), Cardie's (1993) algorithm, Koller and Sahami's (1996) work with probabilistic concepts, the named 'Incremental Feature Selection' (Liu and Setiono, 1998), Hall's (1999) work with correlation concepts, the Bell and Wang's (2000) interesting 'Relevance Formalism' idea, the Information-Theoretic Algorithm of Last et al. (2001) or Perner's CM procedure (2001). Nowadays, the filter approach is receiving considerable attention from the Data Mining community to deal with huge databases when the wrapper approach is unfeasible (Liu and Motoda, 1998). Figure 6.2 locates the role of filter and wrapper approaches within the overall FSS process.

Interesting efforts have been made (Blanco et al., 2001; Das, 2001; Xing et al., 2001) to combine-hybridize the goodness of both filter and wrapper approaches.

When the size of the problem allows the application of the wrapper approach, works in the 90's have noted its superiority, in terms of predictive accuracy over the filter approach. Doak (Doak, 1992) in the early 90's, also empirically showed this superiority of the wrapper model. However, due to computational availability limitations, he could only apply Sequential Feature Selection with the wrapper model, discarding the use of computationally more expensive global search engines (Best-First, Genetic Algorithms, etc.). More recent studies (Kohavi, 1995d; Hall and Holmes, 2000) have also empirically studied the more accurate behaviour of the wrapper evaluation scheme with respect to the filter one.

Blum and Langley (1997) also present another type of FSS, known as *embedded*. This concept covers the feature selection process made by the induction algorithm itself. For example, both Decision Tree and Decision Rule methods implicitly select features for inclusion in a branch or rule in preference to other features that appear less relevant, and in the final model some of the initial features might not appear. On the other hand, some induction algorithms (i.e., Naive Bayes or IB1) include all the presented features in the model when no FSS is executed. This FSS approach is done within the learning algorithm, preferring some features instead of others and possibly not

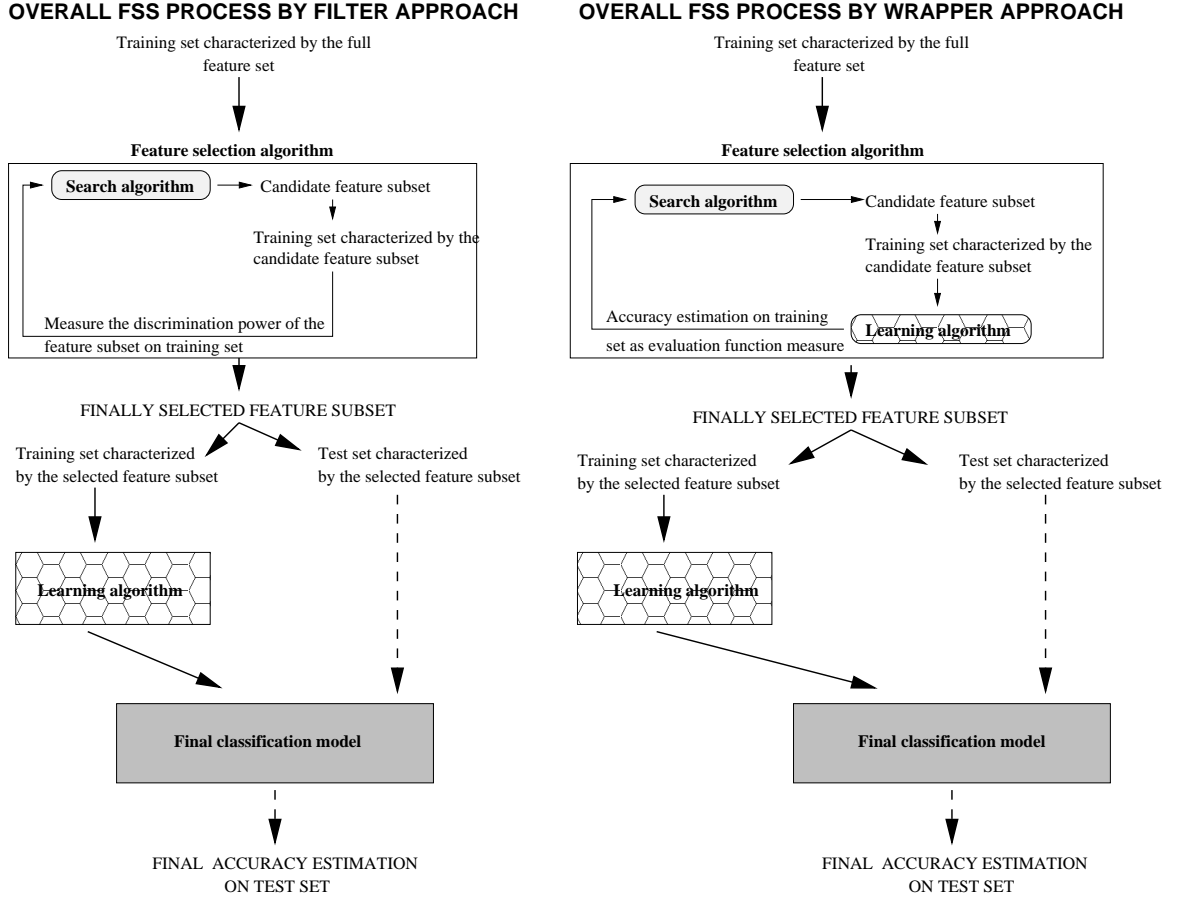


Figure 6.2. Summarization of the whole FSS process for filter and wrapper approaches.

including all the available features in the final classification model. However, filter and wrapper approaches are located one abstraction level above the embedded approach, performing a feature selection process for the final classifier apart from the embedded selection done by the learning algorithm itself.

6.3.4 Criterion for halting the search

An intuitive criterion for stopping the search is the non-improvement of the evaluation function value of alternative subsets. Another classic criterion is to fix a limit on the number of possible solutions to be visited during the search.

6.4 FSS by EDA in small and medium scale domains

For small and medium dimensionality domains, we use the search scheme provided by the EBNA algorithm (Etxeberria and Larrañaga, 1999). Using an intuitive notation to represent each individual (there are n bits in each individual, each bit indicating whether a feature is present (1) or absent (0)), Figure 6.3 shows an overview of the application of the EBNA search engine to the FSS problem:

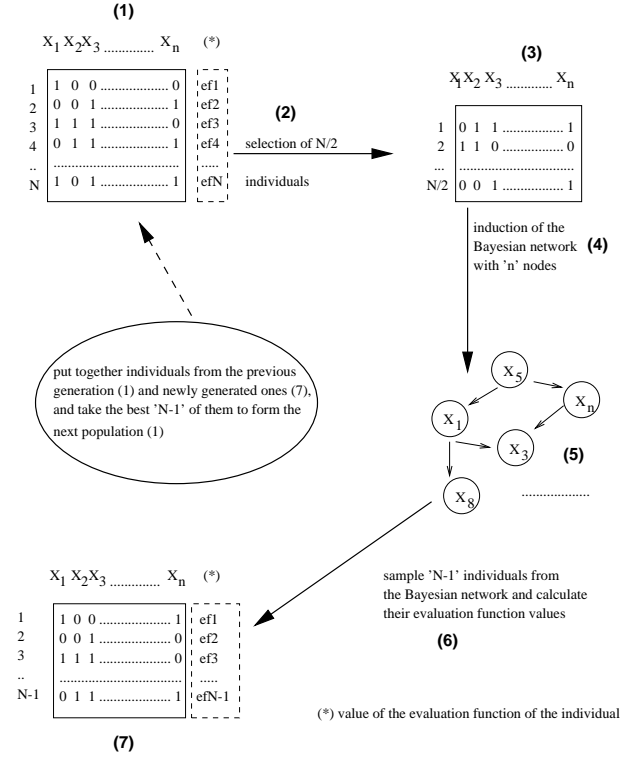


Figure 6.3. FSS-EBNA method.

thus, the algorithm is called FSS-EBNA. In each generation of the search, the induced Bayesian network will factorize the probability distribution of selected solutions. The Bayesian network contains n nodes, where each node represents one feature of the domain.

In our specific implementation of the EBNA algorithm, instead of better (but slow) techniques, a fast ‘score + search’ procedure is used to learn the Bayesian network in each generation of the search. Algorithm B (Buntine, 1991) is used for learning Bayesian networks from data. Algorithm B is a greedy search heuristic which starts with an arc-less structure and at each step, adds the arc which gives the maximum increase in the score: here, the score used is the BIC approximation (Schwarz, 1978). The algorithm stops when adding an arc does not increase this score.

Determination of a minimum population size to reliably estimate the parameters of a Bayesian network is not an easy task (Friedman and Yakhini, 1996). This difficulty is greater for real world problems where the true probability distribution is not known. Taking the dimensionality of our problems into account, we consider that a population size of 1,000 individuals is enough to reliably estimate the Bayesian network model.

In our approach, the best individual of the previous generation is maintained and $N-1$ individuals are created as offspring. An elitist approach is used then to form iterative populations. Instead of directly discarding the $N-1$ individuals from the previous generation and replacing them with $N-1$ newly generated ones, the $2N-2$ individuals are put together and the best $N-1$ chosen from them. These best $N-1$ individuals together with the best individual of the previous generation

form the new population. In this way, the populations converge faster to the best individuals found, but, this also carries a risk of losing diversity within the population.

6.4.1 Characteristics of the evaluation function

A wrapper approach is used here to calculate the evaluation function value of each proposed individual or feature subset. The value of the evaluation function of a feature subset found by the EBNA search technique, once the supervised classifier is fixed, is given by an accuracy estimation on training data. The accuracy estimation, seen as a random variable, has an intrinsic uncertainty. A 10-fold cross-validation multiple times, combined with a heuristic proposed by Kohavi and John (1997), is used to control the intrinsic uncertainty of the evaluation function. The heuristic works as follows:

- if the standard deviation of the accuracy estimate is above 1%, another 10-fold cross-validation is executed;
- this is repeated until the standard deviation drops below 1%, up to a maximum of five times.

In this way, small datasets will be cross-validated many times, but larger ones may only be once.

Although FSS-EBNA is independent of the specific supervised classifier used within its wrapper approach, in our set of experiments we use the NB (Cestnik, 1990) supervised classifier. Due to its simplicity and fast induction, this classifier is commonly used on Data Mining tasks of high dimensionality (Kohavi and John, 1997; Mladenić, 1998). Despite its good scaling with irrelevant features, NB can improve its accuracy level by discarding correlated or redundant features. Because of its independence assumption of features to predict the class, NB is degraded by correlated features which violate this independence assumption. Thus, FSS can also play a ‘normalization’ role that discards these groups of correlated features, and ideally selects just one of them in the final model.

6.4.1.1 Internal loop and external loop of the evaluation process

FSS-EBNA, as any Machine Learning algorithm to assess its accuracy, must be tested on unseen instances which do not participate in the selection of the best feature subset. Two accuracy estimation loops can be seen in the FSS process (see Figure 6.2):

- the *internal-loop* 10-fold cross-validation accuracy estimation that guides the search process is explained in the previous point. The internal-loop represents the evaluation function of the proposed solution;
- the *external-loop* accuracy estimation, reported as the final ‘goodness’ of FSS-EBNA, obtained by testing on unseen instances not used in the search process the feature subset selected by the internal-loop. Due to the non-deterministic nature of FSS-EBNA, 5 replications of 2-fold cross-validation (5x2cv) are applied to assess its predictive generalization accuracy. In each replication, available data is randomly partitioned into two equal-sized sets S_1 and S_2 . The FSS algorithm is trained on each set and tested on the other set. In this way, the reported accuracies are the mean of 10 accuracies.

6.4.2 The ‘overfitting’ problem and the relevance of the stopping criteria

In the initial stages of the definition of FSS-EBNA, we hypothesized to report the accuracy of the internal-loop as the final performance. However, Aha (1999) and Kohavi (1999), in personal

communications, alerted us of the overly optimistic nature of the cross-validation estimates which guide the search. Due to the search nature of FSS-EBNA, it is possible that one feature subset (from the big amount of subsets visited) could be very well adapted to the training set, but when presented to new instances not presented in the training process, the accuracy could dramatically decay: an ‘overfitting’ (Quinlan, 1989) can occur internally in the FSS process. Although it was not done by some authors, we recommend not to report the accuracy used to guide the search as the final accuracy of a FSS process.

Jain and Zongker (1997) report for a non-deceptive function in a Pattern Recognition problem that the quality of selected feature subsets for small training sets was poor; however, improved as the training set increased. Kohavi (1995b) also notes in a wrapper Machine Learning approach that the principal reason of ‘overfitting’ is the low amount of training instances.

To study this issue for FSS approach, a set of experiments with different training sizes of the noisy *Waveform-40* dataset (Breiman et al., 1984) with the NB supervised algorithm (Cestnik, 1990) is carried out: training sizes of 100, 200, 400, 800 and 1,600 samples and tested over a fixed test set with 3,200 instances. Figure 6.4 summarizes the set of experiments.

For 100, 200 and 400 training sizes, although the internal-loop cross-validation was repeated multiple times, differences between internal and external-loop accuracies are high (greater than twice the standard deviation of the internal-loop). However, when the training size is increased, the fidelity between internal and external loop accuracies increases, and the accuracy estimation of the external-loop appears in the range formed by the standard deviation of the internal-loop accuracy.

Apart from these accuracy estimation differences between both loops, a serious ‘overfitting’ risk arises for small datasets: as the search process advances, the internal-loop’s improvement deceives us, as posterior performance on unseen instances does not improve. The difference between internal and external estimations would not be so important if both estimations had the same behaviour: that is, an improvement in the internal estimation coupled with an external improvement and a decrease in internal estimation coupled with an internal improvement. However, it clearly seems that this can not be guaranteed for small size training sets, where two curves show an erratic relation. Thus, FSS generalization results must be done with high care for small datasets.

It seems obvious that for small datasets, it is not possible to see any FSS process as an ‘anytime algorithm’ (Boddy and Dean, 1994), where the quality of the result is a nondecreasing function in time. Looking at Figure 6.4, we discard this ‘monotonic-anytime idea’ (more time \leftrightarrow better solution) for small training set sizes. Our findings follow the work of Ng (1997), who in an interesting work on the ‘overfitting’ problem, demonstrates that when cross-validation is used to select from a large pool of different classification models in a noisy task with too small training set, it may not be advisable to pick the model with minimum cross-validation error, and a model with higher cross-validation error will have better generalization error over novel test instances.

However, Kohavi (1999), in personal communication, alerts us about the difficulty to determine the stopping point of a FSS search process to avoid this obvious ‘overfitting’ risk.

Regarding this behaviour, so related with the number of instances in training set, the next heuristic as stopping criteria is adopted for FSS-EBNA:

- for datasets with more than 2,000 instances (more than 1,000 instances in each training subset for the 5x2cv external loop accuracy estimation), the search is stopped when in a sampled new generation no feature subset appears with an evaluation function value improving the best subset found in the previous generation. Thus, the best subset of the search, found in the previous generation, is returned as FSS-EBNA’s solution;

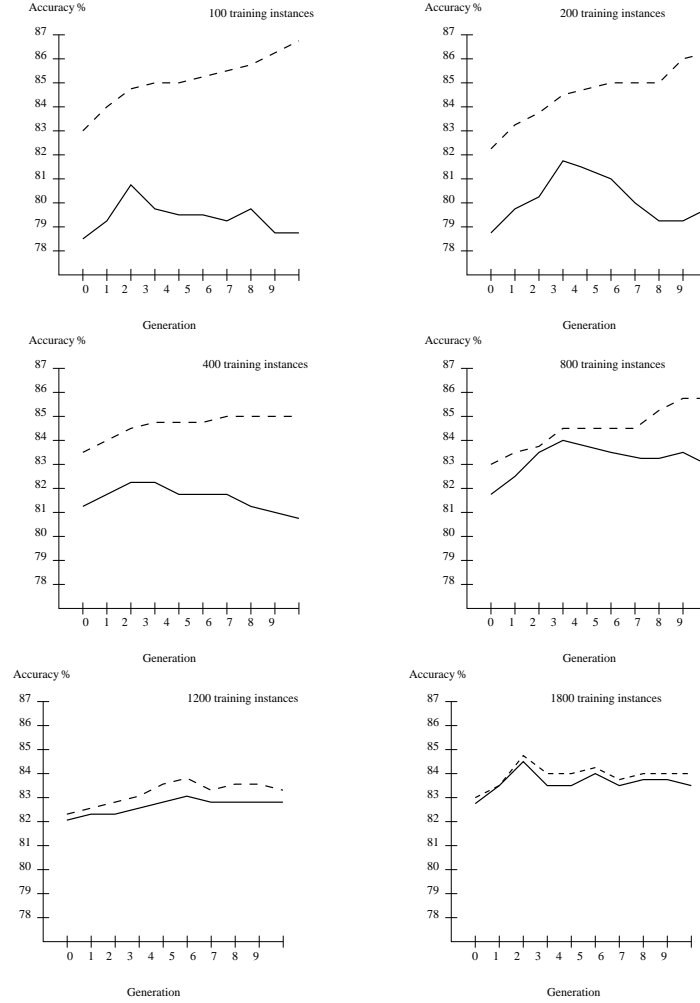


Figure 6.4. Internal and external loop accuracy values in FSS-EBNA for different training sizes with *Waveform-40* dataset and NB algorithm. The internal loop accuracy 10-fold cross-validation is multiple times repeated until the standard deviation of the accuracy estimation drops below 1%. Dotted-lines show the internal-loop accuracy estimation and solid-lines the external-loop one. Both loop accuracies for the best solution of each search generation are represented. '0' represents the initial generation of the search.

- for smaller datasets (less than 1,000 instances in each training subset for the 5x2cv external loop accuracy estimation), the search is stopped when in a sampled new generation no feature subset appears with an evaluation function value improving, at least with a p -value smaller than 0.1 (using a 10-fold cross-validated paired t test between the folds of both estimations, taking only the first run into account when 10-fold cross-validation is repeated multiple times), the value of the evaluation function of the feature subset of the previous generation. Thus, the best subset of the previous generation is returned as FSS-EBNA's solution.

An improvement in the internal loop estimation is not the only measure to take into account to allow the continuation of the search in FSS-EBNA. The number of instances of the dataset is also

Table 6.1. Details of small and medium dimensionality experimental domains.

<i>Domain</i>	<i>Number of instances</i>	<i>Number of features</i>
<i>Ionosphere</i>	351	34
<i>Horse-colic</i>	368	22
<i>Soybean-large</i>	683	35
<i>Anneal</i>	898	38
<i>Image</i>	2,310	19
<i>Sick-euthyroid</i>	3,163	25

critical for this permission. For larger datasets the ‘overfitting’ phenomenon has a lesser impact and we hypothesize that an improvement in the internal-loop estimation will be coupled with an improvement in generalization accuracy on unseen instances. Otherwise, for smaller datasets the ‘overfitting’ phenomenon has a greater risk in occurring and the continuation of the search is only allowed when a significant improvement in the internal-loop accuracy estimation of best individuals of consecutive generations appears. We hypothesize that when this significant improvement appears, the ‘overfitting’ risk decays and there is a basis for further generalization accuracy improvement over unseen instances.

Another concept to consider in this stopping criteria is the wrapper nature of the proposed evaluation function. As we will see in the next section, the evaluation function value of each visited solution (the accuracy estimation of the NB classifier on the training set by 10-fold cross-validation multiple times, using only the features proposed by the solution) needs several seconds to be calculated. As the creation of a new generation of individuals implies the evaluation process of each of them, we only allow the search to continue when it demonstrates that it is able to escape from the local optimas, and is able to discover new ‘best’ solutions in each generation. When the wrapper approach is used, CPU time must also be controlled: we hypothesize that when the search is allowed to continue by our stopping criteria, the CPU times to evaluate a new generation of solutions are justified.

6.4.3 Experiments in real domains

We test the power of FSS-EBNA in six real small and medium-dimensionality datasets. Although they have been presented in the Chapter 2 of this dissertation, Table 6.1 gives the principal characteristics of these datasets. All the datasets have been frequently used in the FSS literature. As an exhaustive search for all feature combinations is infeasible, a heuristic search is needed.

To test the power of FSS-EBNA, a comparison with the following well known FSS algorithms is carried out:

- Sequential Forward Selection (SFS) is a classic hill-climbing search algorithm (Kittler, 1978) which starts from an empty subset of features and sequentially selects features until no improvement is achieved in the evaluation function value. It performs the major part of its search near the empty feature set;
- Sequential Backward Elimination (SBE) is another classic hill-climbing algorithm (Kittler, 1978) which starts from the full set of features and sequentially deletes features until no improvement is achieved in the evaluation function value. It performs the major part of its search near the

Table 6.2. Accuracy percentages of the NB classifier on real datasets without feature selection and using the five FSS methods. The last row shows the average accuracy percentages for all six domains.

<i>Domain</i>	<i>Without FSS</i>	<i>SFS</i>	<i>SBE</i>	<i>FSS-GA-o</i>	<i>FSS-GA-u</i>	<i>FSS-EBNA</i>
<i>Ionosphere</i>	$84.84 \pm 3.12\dagger$	$90.25 \pm 1.58^*$	91.39 ± 2.68	91.17 ± 3.19	$90.97 \pm 2.56^*$	92.40 ± 2.04
<i>Horse-colic</i>	$78.97 \pm 2.98\dagger$	83.31 ± 1.98	$82.12 \pm 2.41^*$	83.43 ± 2.82	83.51 ± 1.47	83.93 ± 1.58
<i>Soybean-large</i>	$81.96 \pm 3.46\dagger$	$86.38 \pm 3.30^*$	$87.78 \pm 3.90^*$	$85.64 \pm 4.06\dagger$	$86.09 \pm 4.37\dagger$	88.64 ± 1.70
<i>Anneal</i>	$93.01 \pm 3.13^*$	$86.72 \pm 2.09\dagger$	$92.49 \pm 2.94^*$	$92.95 \pm 2.67^*$	93.13 ± 2.56	94.10 ± 3.00
<i>Image</i>	$79.95 \pm 1.52\dagger$	88.65 ± 1.21	88.82 ± 1.74	88.67 ± 2.48	89.12 ± 1.56	88.98 ± 0.98
<i>Sick-euthyroid</i>	$84.77 \pm 2.70\dagger$	$90.73 \pm 0.55\dagger$	95.57 ± 0.16	95.97 ± 0.58	95.90 ± 0.43	96.14 ± 0.65
Average	83.91	87.67	89.69	89.63	89.78	90.69

full feature set. Instead of working with a population of solutions, SFS and SBE try to optimize a single feature subset;

- GA with one-point crossover (FSS-GA-o);
- GA with uniform crossover (FSS-GA-u);
- FSS-EBNA.

For all the FSS algorithms the wrapper evaluation function explained in the previous section is used. SFS and SBE algorithms stop deterministically, and the GA algorithms apply the same stopping criteria as FSS-EBNA.

Although the optimal selection of parameters is still an open problem for GAs (Grefenstette, 1986), for both these GA algorithms, guided by the recommendations of Bäck (1996), the probability of crossover is set to 1.0 and the mutation probability to $1/(\text{problem} - \text{dimensionality})$ (these values are so common in the literature). Fitness-proportional selection is used to select individuals for crossover. In order to avoid any bias in the comparison, the remaining GA parameters are the same as FSS-EBNA's: the population size is set to 1,000 and the new population is formed from the best members of both the old population and the offspring. In order to avoid any randomized effect due to a low population size, we fix it to 1,000.

The comparison is extended by running the NB classifier without feature selection. Table 6.2 shows accuracy (and its standard deviation) results for real datasets. Apart from a high accuracy level, we will also focus our attention on the achieving a reduced number of features: a good tradeoff between a high-accuracy and a low-cardinality of the selected feature subset is required.

A deeper analysis of the accuracy results is carried out by using statistical tests. The 5x2cv F test (Alpaydin, 1999) is performed to determine the significance degree of accuracy differences between each algorithm and FSS-EBNA. Thus, in Table 6.2 the symbol ' \dagger ' denotes a statistically significant difference to FSS-EBNA at the 0.05 confidence level, and ' * ', denotes significant difference at the 0.1 confidence level. The meaning of these symbols is the same in all the tables of this dissertation. Table 6.3 shows the average (and its standard deviation) number of features selected by each approach. Experiments are executed on a SGI-Origin 200 computer using the NB algorithm's implementation of the MLC++ (Kohavi et al., 1997b) software.

All FSS algorithms help NB to reduce the number of features needed to induce the final models. This dimensionality reduction is coupled with considerable accuracy improvements in all datasets

Table 6.3. Cardinalities of finally selected features subsets for the NB classifier on real datasets without feature selection and using the five FSS methods. It must be taken into account that when no FSS is applied to NB, it uses all the features.

Domain	Without FSS	SFS	SBE	FSS-GA-o	FSS-GA-u	FSS-EBNA
<i>Ionosphere</i>	34	6.00 ± 1.41	21.30 ± 3.80	15.00 ± 2.36	12.66 ± 1.03	13.40 ± 2.11
<i>Horse-colic</i>	22	6.00 ± 2.74	11.20 ± 2.65	5.00 ± 2.82	4.60 ± 1.75	6.10 ± 1.85
<i>Soybean-large</i>	35	12.70 ± 2.71	23.50 ± 2.75	19.00 ± 2.09	19.16 ± 2.31	18.90 ± 2.76
<i>Anneal</i>	38	5.50 ± 2.32	33.60 ± 2.91	21.66 ± 2.66	19.50 ± 2.25	20.50 ± 3.13
<i>Image</i>	19	5.60 ± 1.57	9.40 ± 1.95	8.00 ± 1.41	8.00 ± 1.09	8.00 ± 0.66
<i>Sick-euthyroid</i>	25	0	13.83 ± 1.32	10.66 ± 2.58	10.16 ± 1.72	9.80 ± 2.09

except *Anneal*, where FSS-EBNA is the only algorithm able to significantly improve the accuracy of the NB model without feature selection. Although accuracy differences are not statistically significant across datasets between FSS algorithms for most domains, FSS-EBNA has the best average accuracy of the compared methods.

Although SBE achieves similar accuracy results in many datasets relative to randomized algorithms, its major disadvantage is the small reduction that it produces in the number of features. In all domains, SBE is the algorithm with the lowest feature reduction, and this reduction is nearly insignificant in the *Anneal* dataset. Although the other sequential algorithm, SFS, returns the subsets with the smallest number of features in all datasets, its accuracy results in all except one dataset are significantly inferior to FSS-EBNA.

Although both GA approaches and FSS-EBNA obtain similar accuracy results in many datasets, we note that GA approaches need more generations than FSS-EBNA to arrive at similar (or lower) accuracy levels. Table 6.4 shows which generations GA approaches and FSS-EBNA stop in, using the explained stopping criteria.

Starting from the fact that accuracy differences between both GA approaches are not statistically significant, it seems that FSS-GA-o is better suited for *Horse-colic* and *Soybean-large* datasets than FSS-GA-u, and while the opposite behaviour is achieved in *Ionosphere* and *Anneal*. However, the results show that FSS-EBNA arrives faster to similar or better accuracies (see also Table 6.2) than both GA approaches: it seems that FSS-EBNA, by using Bayesian networks, is able to capture the underlying structure of the problem faster than GAs. Only in the *Image* dataset, the domain of lowest dimensionality, does using FSS-EBNA not give an advantage. This superiority of EDA approaches that use Bayesian networks over GAs in domains with interacting variables is also noted in the literature (Pelikan et al., 1998; de Campos et al., 2001).

When the wrapper approach is used to calculate the evaluation function value of each subset found, then faster discovery of similar or better accuracies is a critical task. Although NB is a fast supervised classifier, it needs several CPU seconds to estimate the predictive accuracy (by 10-fold cross-validation multiple times, as explained) of a feature subset on the training set: depending on the number of features selected by the subset, around 1 CPU second is needed in *Ionosphere* (the domain with fewest instances), while around 3 CPU seconds are needed in *Image* (the domain with most instances). Since the generation of a new population of solutions implies the evaluation of 1,000 new individuals, stopping earlier without damage to the accuracy level is highly desirable in order to save CPU time. On the other hand, the times for the induction of the Bayesian networks over the best individuals in each EDA generation are insignificant in all domains and the EDA

Table 6.4. Mean stop-generation for GAs and FSS-EBNA. The standard deviation of the mean is also reported. The initial generation is considered to be the zero generation.

<i>Domain</i>	<i>FSS-GA-o</i>	<i>FSS-GA-u</i>	<i>FSS-EBNA</i>
<i>Ionosphere</i>	$3.50 \pm 0.84\dagger$	$3.10 \pm 0.56\dagger$	1.80 ± 0.42
<i>Horse-colic</i>	$3.20 \pm 1.13^*$	$3.40 \pm 0.51\dagger$	2.40 ± 0.69
<i>Soybean-large</i>	$3.30 \pm 0.82^*$	$3.60 \pm 0.51\dagger$	2.50 ± 0.70
<i>Anneal</i>	$3.80 \pm 0.42\dagger$	$3.20 \pm 0.44\dagger$	1.80 ± 0.42
<i>Image</i>	3.60 ± 0.84	3.70 ± 0.48	3.50 ± 0.42
<i>Sick-euthyroid</i>	$4.50 \pm 0.70^*$	$4.80 \pm 0.42^*$	3.50 ± 0.97

savings in CPU time relative to GA approaches are maintained: 3 CPU seconds are needed on average in the *Image* domain to induce a Bayesian network (the domain with fewest features) and 14 CPU seconds in *Anneal* (the domain with most features).

6.4.4 Experiments in artificial domains

In order to enrich this comparison among GA approaches and FSS-EBNA, we have designed three artificial datasets of 2,000 instances each, where we know the feature subset which induces each domain: *Redundant-order-3*, *Redundant-order-5* and *Redundant-order-7* all have 21 continuous features in the range $[3, 6]$. The target concept in all three domains is to determine whether an instance is nearer (using Euclidean distance) to $(0, 0, \dots, 0)$ or $(9, 9, \dots, 9)$. At first, all 21 features participate in the distance calculation and they are conditionally independent given the class. In this way, the predictive features respect NB's classification scheme. As NB's predictive power is heavily damaged by redundant features, we decide to generate groups of repeated features:

- there are 3 groups of 3 repeated features each in *Redundant-order-3* while the remaining 12 features are not repeated. The class of the domain is induced by these 12 individual features and by one feature from each of the 3 groups;
- there are 3 groups of 5 repeated features each in *Redundant-order-5*, while the remaining 6 features are not repeated. The class of the domain is induced by these 6 individual features and by one feature from each of the 3 groups;
- there are 2 groups of 7 repeated features each in *Redundant-order-7*, while the remaining 7 features are not repeated. The class of the domain is induced by these 7 individual features and by one feature from each of the 2 groups.

The size of the relations between the features of these domains is well suited to be covered by Bayesian networks rather than probabilistic approaches that are only able to cover interactions of order one or two: not only conditional probabilities for a position having given the value for another one but for a position having given values for a set of some of other positions as well must be taken into account. Maintaining the framework given in the previous section, Table 6.5 shows the generation in which GA approaches and FSS-EBNA discover a feature subset that equalizes or surpasses the estimated accuracy level of the feature subset which induces the domain. This stopping criteria is also used by Rana et al. (1996).

Although no statistically significant differences are achieved in the stop generations of different algorithms, it seems that for artificial datasets, FSS-EBNA needs fewer generations than GA ap-

Table 6.5. Number of generations needed on average (and their standard deviation) by FSS-GA-o, FSS-GA-u and FSS-EBNA to discover the optimum feature subset in artificial domains. The initial generation is considered as generation zero.

<i>Domain</i>	<i>FSS-GA-o</i>	<i>FSS-GA-u</i>	<i>FSS-EBNA</i>
<i>Redundant-order-3</i>	2.50 ± 1.76	3.33 ± 1.63	1.33 ± 0.81
<i>Redundant-order-5</i>	3.83 ± 2.04	3.33 ± 2.16	1.83 ± 0.75
<i>Redundant-order-7</i>	2.00 ± 1.67	2.50 ± 1.76	1.00 ± 1.09

proaches to arrive at similar fitness solutions. We therefore hypothesize that the superior behaviour of FSS-EBNA with respect to GAs in natural domains is due to the existence of interacting features in these tasks. Table 6.5's results are achieved when the interacting variables of the same group are mapped-coded together in the individual's representation. When we perform the same set of experiments, but randomly separate the interacting features in the individual's representation, FSS-GA-o needs the following number of generations to discover a feature subset which equalizes or surpasses the estimated accuracy level of the feature subset which induces the domain:

- 4.00 ± 1.67 in *Redundant-order-3*;
- 4.66 ± 0.94 in *Redundant-order-5*;
- 3.50 ± 1.87 in *Redundant-order-7*.

While FSS-GA-u and FSS-EBNA are not influenced by the positions of variables in the individual's representation, FSS-GA-o suffers notably when interacting features are not coded together. This phenomenon is noted in the GA literature by many authors (Harik and Goldberg, 1996; Thierens and Goldberg, 1993).

6.5 FSS by EDA in large scale domains

Although the FSS literature contains many papers, few of them tackle the task of FSS in domains with more than 50 features (Aha and Bankert, 1994; Kudo and Sklansky, 2000; Li et al., 2001; Mladenić, 1998). In this section we propose several EDA-inspired approaches to this kind of task.

For large dimensionality domains, instead of Bayesian networks, we propose the use of four simpler probabilistic models to perform FSS. It is well known in the Bayesian network literature (Friedman and Yakhini, 1996) that a large number of individuals is needed to induce a reliable Bayesian network in domains of large dimensionality. The possibility of obtaining a large number of individuals is not a problem in certain environments, but calculation of the evaluation function of an individual takes several CPU seconds when a wrapper evaluation function is used. Before the execution of a FSS wrapper approach, we must take into account the quantity of available computational resources in order to fix the following parameters for the estimation of a reliable Bayesian network: number of instances and features of the dataset, characteristics of the evaluation function (computational speed of the wrapper classification algorithm) and number of solutions in the population. Fig 6.5 shows the relations between these concepts.

In this way, for large dimensionality problems, despite losing the capability of Bayesian networks to factorize multiple order interactions among the variables of the problem, we prefer to use simpler

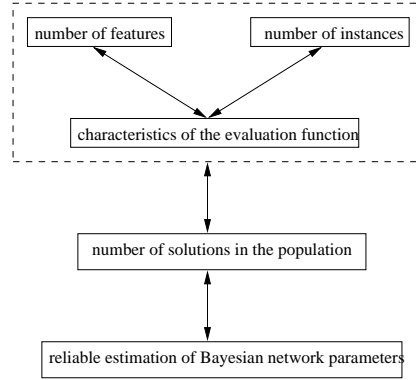


Figure 6.5. Relations between relevant concepts to estimate a reliable Bayesian network.

Table 6.6. Details of large-dimensionality experimental domains.

<i>Domain</i>	<i>Number of instances</i>	<i>Number of features</i>
<i>Audiology</i>	226	69
<i>Arrhythmia</i>	452	279
<i>Cloud</i>	1,834	204
<i>DNA</i>	3,186	180
<i>Internet advertisements</i>	3,279	1558
<i>Spambase</i>	4,601	57

probabilistic models that avoid an increase in the number of individuals in the population. In this chapter we use the following four probabilistic algorithms:

- PBIL (Baluja, 1994) (using $\alpha = 0.5$) and BSC (Syswerda, 1993) univariate distribution models;
- MIMIC (De Bonet et al., 1997) chain distribution model;
- the TREE algorithm proposed by Chow and Liu (1968).

We therefore have the following FSS algorithms: FSS-PBIL, FSS-BSC, FSS-MIMIC and FSS-TREE. These algorithms only differ from FSS-EBNA in the probabilistic model employed to factorize the probability distribution of selected solutions: instead of FSS-EBNA's Bayesian network, they use the referred probabilistic algorithm. They employ the same evaluation function scheme, basic wrapper classifier (NB), stopping criteria, population size and rule for forming successive populations as FSS-EBNA.

6.5.1 Experiments in real domains

We test the power of FSS-PBIL, FSS-BSC, FSS-MIMIC and FSS-TREE on six real, large-dimensionality datasets. Although they have been presented in the Chapter 2 of this dissertation, Table 6.6 shows the principal characteristics of these datasets.

Due to the large dimensionality of the datasets, we do not include in the comparison sequential algorithms such as SFS and SBE: we just compare our four FSS EDA-inspired algorithms with

Table 6.7. Accuracy percentages of the NB classifier on real datasets without feature selection and using FSS-GA-o and FSS-GA-u. The last row shows the average accuracy percentages for all six domains.

<i>Domain</i>	<i>Without FSS</i>	<i>FSS-GA-o</i>	<i>FSS-GA-u</i>
<i>Audiology</i>	$52.39 \pm 5.56^\dagger$	68.29 ± 2.98	68.44 ± 4.46
<i>Arrhythmia</i>	$39.91 \pm 8.50^\dagger$	63.23 ± 3.95	64.73 ± 3.52
<i>Cloud</i>	$68.18 \pm 2.09^\dagger$	74.49 ± 1.93	75.17 ± 1.22
<i>DNA</i>	93.93 ± 0.67	94.00 ± 0.75	95.01 ± 0.56
<i>Internet advertisements</i>	$95.23 \pm 0.40^*$	96.10 ± 0.12	96.38 ± 0.47
<i>Spambase</i>	$81.71 \pm 0.92^\dagger$	88.92 ± 1.45	88.77 ± 1.28
Average	71.88	80.83	81.41

FSS-GA-o and FSS-GA-u. While Kudo and Sklansky (2000) recommend the use of sequential FSS algorithms in small and medium dimensionality scale problems, they do not advise using them in large scale domains, and they consider GAs the only practical way to get reasonable feature subsets over this kind of domains. The main reason for this is that sequential algorithms exhaustively search a specific part of the solution space (Doak, 1992) (SFS near the empty feature set and SBE near the full feature set), leaving remaining large parts of the solution space unexplored. Sequential algorithms, which are prone to get stuck in local peaks (Rozsypal and Kubat, 2001), do not have a mechanism for jumping from a subset to another very different subset, but trace a sequence of subsets in which adjacent subsets differ by only one feature (Kudo and Sklansky, 2000). On the other hand, population-based algorithms make use of their randomized nature, and allow a search with a larger degree of diversity. Another reason to avoid using sequential algorithms is that the application of SBE within a wrapper evaluation scheme is computationally prohibitive for large dimensionalities.

With medium dimensionality datasets, the probability of selecting and discarding a feature in a solution of the initial population is the same for all the domains. However, in the huge search space of large scale domains, an adequate bias towards specific areas of the search space could notably improve the operation of population-based algorithms (GAs and EDAs), and avoid visiting a large number of solutions. Taking into account the expert considerations in the *Cloud* dataset (Aha and Bankert, 1994), the probability of selecting a feature in a solution of the initial population is biased to 0.1. The *Audiology*, *Arrhythmia* and *Internet advertisements* datasets suggest to applying a similar bias: in these datasets the probability of selecting a feature in a solution of the initial population is biased to 0.05. By the introduction of this bias, the nature of the comparison between GAs and EDAs is not altered and a large amount of CPU time is saved.

Tables 6.7 and 6.8 show accuracy results (and their standard deviation) for the example domains. For each domain, statistically significant differences relative to the algorithm with the best estimated accuracy are also noted in Tables 6.7 and 6.8. Tables 6.9 and 6.10 show the average number of features selected by each approach (and its standard deviation). For this comparison, the GA parameters from the previous section and the 5x2cv cross-validation procedure are used.

With the use of FSS approaches, statistically significant accuracy improvements and notable dimensionality reductions are achieved relative to the no-FSS approach in all except the *DNA* dataset. All six FSS algorithms obtain similar accuracy results and dimensionality reductions in all the domains. However, as in the case of small and medium dimensionality datasets, we note

Table 6.8. Accuracy percentages of the NB classifier on real datasets using FSS-PBIL, FSS-BSC, FSS-MIMIC and FSS-TREE. The last row shows the average accuracy percentages for all six domains.

<i>Domain</i>	<i>FSS-PBIL</i>	<i>FSS-BSC</i>	<i>FSS-MIMIC</i>	<i>FSS-TREE</i>
<i>Audiology</i>	70.22 ± 2.78	68.29 ± 3.18	68.88 ± 3.93	70.09 ± 4.12
<i>Arrhythmia</i>	64.62 ± 2.70	65.01 ± 2.22	64.33 ± 1.82	64.51 ± 2.59
<i>Cloud</i>	75.18 ± 1.30	76.24 ± 1.25	76.31 ± 0.95	75.84 ± 0.98
<i>DNA</i>	94.86 ± 0.64	95.40 ± 0.40	95.53 ± 0.29	95.40 ± 0.28
<i>Internet adv.</i>	96.49 ± 0.21	96.37 ± 0.41	96.46 ± 0.46	96.69 ± 0.63
<i>Spambase</i>	88.63 ± 1.36	89.52 ± 1.38	89.80 ± 0.79	89.60 ± 0.93
Average	81.66	81.80	81.88	82.02

Table 6.9. Cardinalities of finally selected feature subsets for the NB classifier on real datasets without feature selection and using FSS-GA-o and FSS-GA-u. It must be taken into account that when no FSS is applied to NB, it uses all the features.

<i>Domain</i>	<i>Without FSS</i>	<i>FSS-GA-o</i>	<i>FSS-GA-u</i>
<i>Audiology</i>	69	14.00 ± 3.68	15.33 ± 3.50
<i>Arrhythmia</i>	279	15.40 ± 3.02	18.30 ± 4.71
<i>Cloud</i>	204	26.40 ± 4.45	27.60 ± 3.86
<i>DNA</i>	180	59.00 ± 8.35	55.80 ± 6.46
<i>Internet advertisements</i>	1,558	113.10 ± 7.52	108.00 ± 5.35
<i>Spambase</i>	57	29.20 ± 3.88	29.00 ± 4.24

Table 6.10. Cardinalities of finally selected features subsets for the NB classifier on real datasets using FSS-PBIL, FSS-BSC, FSS-MIMIC and FSS-TREE.

<i>Domain</i>	<i>FSS-PBIL</i>	<i>FSS-BSC</i>	<i>FSS-MIMIC</i>	<i>FSS-TREE</i>
<i>Audiology</i>	10.66 ± 2.50	14.33 ± 4.67	13.33 ± 3.14	12.50 ± 2.34
<i>Arrhythmia</i>	13.60 ± 1.95	13.40 ± 2.36	17.60 ± 2.83	20.50 ± 6.13
<i>Cloud</i>	26.40 ± 3.47	30.00 ± 3.59	29.50 ± 4.83	30.60 ± 4.08
<i>DNA</i>	56.90 ± 5.83	56.90 ± 5.89	57.40 ± 7.04	59.40 ± 5.10
<i>Internet adv.</i>	114.30 ± 5.65	120.25 ± 18.00	122.25 ± 8.88	125.00 ± 17.60
<i>Spambase</i>	28.80 ± 3.82	29.10 ± 3.78	29.10 ± 3.41	30.50 ± 3.40

differences in the number of generations needed to achieve given accuracy levels. Table 6.11 shows which generation FSS algorithms halt in, when the stopping criteria described earlier is used.

Table 6.11 shows two notably different kinds of behaviour. For each domain in Table 6.11 statistically significant differences relative to the algorithm which needs the lowest number of generations are noted. The results show that FSS-BSC, FSS-MIMIC and FSS-TREE arrive faster to similar fitness areas than FSS-PBIL and both of the GA approaches in all the domains. As in the case of medium dimensionality datasets, the capture of the underlying structure of the problem seems to be essential: as FSS-MIMIC and FSS-TREE are able to cover interactions of order-two among the features of the task, this could be the reason for their good behaviour. Note the good behaviour of

Table 6.11. Mean stop-generation for FSS algorithms. The standard deviation of the mean is reported. The initial generation is considered to be the zero generation.

<i>Domain</i>	<i>FSS-GA-o</i>	<i>FSS-GA-u</i>	<i>FSS-PBIL</i>	<i>FSS-BSC</i>	<i>FSS-MIMIC</i>	<i>FSS-TREE</i>
<i>Audiology</i>	$5.80 \pm 0.42\dagger$	$4.60 \pm 0.96*$	$5.20 \pm 1.03*$	2.50 ± 0.70	2.80 ± 0.78	2.80 ± 0.78
<i>Arrhythmia</i>	$8.70 \pm 0.48\dagger$	$8.80 \pm 0.42\dagger$	$8.30 \pm 0.48*$	7.10 ± 0.73	7.00 ± 0.66	7.20 ± 0.78
<i>Cloud</i>	$10.50 \pm 0.52*$	$10.60 \pm 1.07*$	10.40 ± 0.84	8.40 ± 0.51	8.40 ± 0.69	8.30 ± 0.82
<i>DNA</i>	$12.80 \pm 0.91\dagger$	$11.80 \pm 0.42\dagger$	$11.30 \pm 0.48\dagger$	8.70 ± 0.82	8.10 ± 0.73	8.40 ± 0.69
<i>Int. adv.</i>	4.70 ± 1.41	5.00 ± 1.41	5.00 ± 0.66	4.40 ± 1.26	4.30 ± 0.67	4.00 ± 1.63
<i>Spambase</i>	4.80 ± 1.03	5.20 ± 0.63	5.50 ± 1.17	4.20 ± 0.91	3.70 ± 0.82	4.20 ± 1.22

Table 6.12. Average CPU times (in seconds) for the induction of different probabilistic models (standard deviations are nearly zero) in each generation of the EDA search. The last column shows the average CPU time to estimate the predictive accuracy of a feature subset by the NB classifier.

<i>Domain</i>	<i>PBIL</i>	<i>BSC</i>	<i>MIMIC</i>	<i>TREE</i>	<i>NB</i>
<i>Audiology</i>	1.2	1.3	1.8	2.2	1.0
<i>Arrhythmia</i>	4.0	4.2	12.2	25.3	2.6
<i>Cloud</i>	2.3	2.4	6.5	14.6	7.2
<i>DNA</i>	1.8	2.0	4.8	10.9	5.3
<i>Internet advertisements</i>	101.1	106.4	808.5	1,945.6	9.8
<i>Spambase</i>	0.8	0.9	1.2	1.8	8.2

FSS-BSC, a probabilistic algorithm which does not cover interactions among domain features: the explanation of these FSS-BSC results could be its direct use of the accuracy percentages to estimate the univariate probabilities, probabilities which are simulated to generate the new solutions of each EDA-generation. On the other hand, the behaviour of FSS-PBIL, the other order-one probabilistic algorithm, is similar to that of the GA approaches. We suspect that the explanation of this result is the absence of a tuning process to select a value for the α parameter: previous studies indicate that a good selection of the PBIL α parameter is a critical task (González et al., 2001).

Because of the large dimensionality of the datasets, when the wrapper approach is employed to estimate the goodness of a feature subset, faster discovery of similar fitness solutions becomes a critical task. Despite the faster nature of the NB classifier, a large amount of CPU time is saved by avoiding the simulation of several generations of solutions. In order to understand the advantages of the EDA approach relative to the GA approach, CPU times for the induction of the probabilistic models must be studied: the EDA approach has the added overhead of the calculation of probabilistic models in each EDA-generation. Table 6.12 shows, for each domain, the average CPU times to induce the associated probabilistic model in each generation. The last column also shows the average CPU times needed to estimate the predictive accuracy of a single feature subset by the NB classifier: note that the times in the last column are not comparable with the previous columns, but they help to understand the magnitude of the CPU time savings when fewer generations are needed to achieve similar accuracy results.

As CPU times for the induction of probabilistic models are insignificant in all domains except *Internet advertisements*, the CPU time savings relative to GA approaches shown in Table 6.11 are

Table 6.13. Number of generations needed on average (and their standard deviation) by FSS-GA-o, FSS-GA-u, FSS-PBIL, FSS-BSC, FSS-MIMIC and FSS-TREE to discover a feature subset that equalizes or surpasses the estimated accuracy level of the feature subset which induces the domain. The initial generation is considered to be the zero generation.

Domain	FSS-GA-o	FSS-GA-u	FSS-PBIL	FSS-BSC	FSS-MIMIC	FSS-TREE
<i>Red60of1</i>	$6.70 \pm 0.48^\dagger$	4.10 ± 0.31	$12.80 \pm 0.91^\dagger$	$7.60 \pm 0.51^\dagger$	$8.60 \pm 0.51^\dagger$	$8.00 \pm 0.47^\dagger$
<i>Red30of2</i>	$22.40 \pm 4.22^\dagger$	$73.50 \pm 5.73^\dagger$	$66.30 \pm 7.52^\dagger$	$36.40 \pm 3.13^\dagger$	15.10 ± 2.33	10.90 ± 1.52
<i>Red30of3</i>	$21.00 \pm 2.26^\dagger$	$119.00 \pm 5.27^\dagger$	$113.80 \pm 8.76^\dagger$	$89.50 \pm 17.60^\dagger$	18.90 ± 2.13	16.30 ± 1.33

maintained. In the case of the *Internet advertisements* domain, as order-two probabilistic approaches (MIMIC and TREE) need a large amount of CPU time in each generation, the advantage of using them (in CPU time savings) relative to GA approaches is considerably reduced. It must be noted that GAs' CPU times for recombination operations in each generation are nearly zero.

6.5.2 Experiments in artificial domains

As in the case of small and medium scale domains, we have designed three artificial datasets of 2,000 instances each, where the feature subset which induces each domain is known; *Red60of1* and *Red30of3* have 100 and *Red30of2* 80 continuous features in the range $[3, 6]$, and the target concept is the same as the artificial datasets in the previous section. The description of the three databases is as follows:

- no interactions appear among the features of the *Red60of1* domain. While 60 features induce the class of the domain, the remaining of 40 features are irrelevant;
- there are 30 groups of 2 repeated features each in the *Red30of2* domain while the remaining 20 features are not repeated. The class of the domain is induced by these 20 individual features and one feature from each of the 30 groups;
- there are 30 groups of 3 repeated features each in the *Red30of3* domain while the 10 individual features are not repeated. The class of the domain is induced by these single 10 features and one feature from each of the 30 groups.

While the degree of the relations among the features of *Red60of1* is well suited to be covered by probabilistic algorithms of order-one, approaches of order-two are needed for *Red30of2* and approaches of order three (i.e. Bayesian networks) for *Red30of3*. Conditional probabilities for a variable given the value of another variable and also for a variable given values of a set of other variables should be considered in *Red30of3*. Maintaining the framework given in the previous section, Table 6.13 shows the generation where GA and EDA approaches discover a feature subset that equalizes or surpasses the estimated accuracy level of the feature subset which induces the domain. For each domain, statistically significant differences relative to the algorithm which needs the lowest number of generations are also shown in Table 6.13.

In *Red60of1*, a domain with no interactions among the variables of the problem, the good behaviour of GA approaches relative to EDA order-two approaches must be noted. We think that the absence of a tuning process (González et al., 2001) to fix the α parameter of PBIL is critical to understanding its behaviour in this domain. However, with the appearance of interacting features

in the tasks *Red30of2* and *Red30of3*, the performance of order-two probabilistic approaches (MIMIC and TREE) is noticeably different from the remaining algorithms: this superiority of EDA order-two approaches relative to GAs and order-one approaches in domains with interacting features is also noted in the literature (De Bonet et al., 1997; Pelikan and Mühlenbein, 1999). In this way, we hypothesize that in natural domains, the superior behaviour of order-two probabilistic approaches relative to order-one approaches and GAs is due to the existence of interacting features in these tasks.

As in the case of medium dimensionality datasets, Table 6.13's results are achieved when the interacting variables of the same group are mapped-coded together in the individual's representation. When we perform the same set of experiments but randomly separate the interacting features in the individual's representation, FSS-GA-o needs the following number of generations to discover a feature subset which equalizes or surpasses the estimated accuracy level of the feature subset that induces the domain:

- 46.70 ± 6.53 in *Red30of2*;
- 48.00 ± 5.88 in *Red30of3*.

We note again that while FSS-GA-u and EDA approaches are not influenced by the positions of features in the individual's representation, FSS-GA-o noticeably suffers when interacting features are not coded together. As the *Red60of1* domain has no interactions among the features of the task, it is not included in this comparison.

6.6 Summary and future work

The application of the EDA paradigm to solve the well known FSS problem has been studied. While the most powerful probabilistic model (Bayesian networks) is used with small and medium dimensionality scale datasets, simpler probabilistic models are used with large dimensionalities. Making use of an appropriate probabilistic model, we note that GA approaches need more generations than an adequate EDA approach to discover similar fitness solutions. We show this behaviour on a set of natural and artificial datasets. We also show that while the performance of GA with uniform crossover and EDA approaches are not influenced by the bit positioning in the individual's representation, GA with one-point crossover shows a noticeable decay in its performance when interacting bits are not coded together.

While Bayesian networks are an adequate and non-CPU expensive probabilistic tool for small and medium dimensionality datasets, PBIL, BSC, MIMIC and dependency trees seem suitable for large dimensionality ones. However, because of the high CPU times needed for the induction of order-two algorithms in the *Internet advertisements* domain, the CPU time saving produced by this reduction in the number of solutions relative to GA approaches is noticeably reduced.

As future work, we envision the use of other probabilistic models with large dimensionality datasets, models which assume few or no dependencies among the variables of the domain. Another interesting possibility is the use of parallel algorithms to induce Bayesian networks in these kinds of tasks (Lozano et al., 2001; Sangüesa et al., 1998; Xiang and Chu, 1999). When dimensionalities are higher than 1,000 variables, research is needed on the reduction of CPU times associated with the use of probabilistic order-two approaches.

The advances in the last decade in genome sequencing have lead to the spectacular development of a new technology, named DNA microarray (Golub et al., 1999). DNA microarray allows the monitoring and measurement of the expression levels of many genes simultaneously, obtaining datasets

of thousands of genes. Although several experimentation processes have been carried out (Blanco et al., 2001; Inza et al., 2002b), the application of FSS EDA-inspired approaches in this kind of domains is an exciting area of research.

Chapter 7

Feature Weighting for Nearest Neighbor Classifier by Estimation of Distribution Algorithms

7.1 Overview of the chapter

Nearest Neighbor is a well known paradigm, largely used to solve many classification problems. Its accuracy depends heavily on the weight of each feature in its distance metric calculation. In this chapter, we present two EDA inspired techniques to learn accurate feature weights for the Nearest Neighbor algorithm. The first method, with the use of Bayesian networks within the EDA scheme, performs for each variable a search in a set of three discrete weights. The second one, with the employment of Gaussian networks, works in a continuous range of weights. Both methods are compared with two sequential and one genetic-inspired algorithm in natural and artificial datasets. A wrapper procedure is used to evaluate the sets of weights proposed by all the compared algorithms.

After an introduction, where the Feature Weighting problem for Nearest Neighbor classifier is formalized, a survey of previous approaches in the field will be performed. The specific application of Bayesian and Gaussian networks, within the EDA paradigm, to solve the proposed problem will be exposed. Results over natural and artificial domains and ways for future research will also be presented.

This chapter is an adaptation of the works of Inza (1999) and Inza et al. (2001c).

7.2 Motivation

The K-Nearest Neighbor (K-NN) classifier has long been used by the Pattern Recognition and Machine Learning communities (Dasarathy, 1991) in supervised classification tasks. The basic approach involves storing all training instances and then, when a test instance is presented, retrieving the training instances nearest (least distant) to this test instance and using them to predict its class. Distance is classically defined as follows in a domain with n descriptive features:

$$distance(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n w_i \times difference(x_i, y_i)^2}$$

where $\mathbf{x} = (x_1, \dots, x_i, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_i, \dots, y_n)$ are instances and w_i is a weight value assigned to feature i -th feature. To compute the *difference* between two values, the overlap metric (Salzberg, 1991) is normally used for symbolic features and the absolute difference (after being normalized) for numeric ones.

Dissimilarities among values of the same feature are computed and added together to obtain a representative value of the dissimilarity (distance) between the compared instances. As this distance calculation process is highly CPU time consuming, the Nearest Neighbor algorithm is usually applied over datasets with few instances (normally less than 1,000).

In the basic Nearest Neighbor approach, dissimilarities in each dimension are added in a naive manner, weighting equally dissimilarities in each dimension (for all features: $w_i = 1$). This approach is unrealistic, allowing irrelevant features to influence the distance computation and treating equally features with different degrees of relevance. With this handicapped approach, each time an unimportant feature is added to the feature set and a weight similar to the weight for an important feature is assigned to it, the number of training instances needed to maintain the predictive accuracy is increased exponentially (Lowe, 1995). This phenomenon is known as the *curse of dimensionality*. In order to find a realistic weight for each feature of the problem, several approaches have been proposed by the Pattern Recognition and Machine Learning communities, under the heading of *Feature Weighting for Nearest Neighbor*. Hill-climbing and genetic approaches are possibly the most commonly used search techniques for this Feature Weighting (FW) task.

In this chapter we propose the use of EDA-inspired techniques to tackle the FW task. We present two novel approaches that search for a set of appropriate weights for the Nearest Neighbor algorithm. The first approach, called FW-EBNA (Feature Weighting by Estimation of Bayesian Network Algorithm), which uses Bayesian networks, performs for each variable a search in a set of three discrete weights. The second one, FW-EGNA (Feature Weighting by Estimation of Gaussian Network Algorithm), which uses Gaussian networks, allows the search to be carried out in the $[0, 1]$ continuous interval of weights.

7.3 Review of related works

Since the Pattern Recognition and Machine Learning communities frequently use the Nearest Neighbor classifier, they have also proposed many variants of it to address the FW problem. A complete review of these efforts was done by Wettschereck et al. (1997), which classified FW algorithms along five different dimensions. In this section, in order to properly locate both EDA-inspired approaches, and assuming that their principal contribution is the search mechanism, we organize the principal FW algorithms by the search strategy that they use to optimize the set of weights. The major part of approaches state the FW task as a classic NP-hard optimization problem (Wettschereck et al., 1997), which justifies the use of search heuristics.

Several well-known FW methods learn the set of weights by means of a hill-climber, an incremental and on-line strategy, which makes only one pass through the training data. Applying the nearest neighbor's distance function, they iteratively adjust the feature weights after one or several classifications (on the training set) are made. Weight adjustment is computed to take account of whether the classification given was correct or incorrect. Considering each training instance once, weight adjustment has the purpose of decreasing the distance similarity metric among instances of the same class and increasing the distance among instances of different classes. These types of algorithms can be seen in Salzberg (1991), Aha (1992) and Kira and Rendell (1992).

Lowe (1995) and Scherf and Brauer (1997) have proposed another local search mechanism known as gradient descent optimization that optimizes a set of continuous weights. Lowe (1995) applies gradient descent over the distance similarity metric to optimize feature weights so as to minimize the leave-one-out cross-validation error (LOOCE) on the training set. Lowe (1995) tries to prevent large weight changes in the optimization process which are not statistically reliable on datasets with few examples. In the other work, Scherf and Brauer (1997) apply gradient descent to optimize feature

weights so as to minimize a function which reinforces the distance similarities between all training instances of the same class while decreasing the similarities between instances of different classes. Instead of being incremental on-line optimizers, these two approaches, repeatedly pass through the training set: each time a new weight set of weights is calculated, they use the training set to measure the value of the function to be optimized.

We can qualify hill-climbing and gradient descent optimization as local search engines in the sense that they can not escape from local optima. Kohavi et al. (1997a) propose a best-first search which has this capability of escape from local optima. Using the wrapper approach, each time a weight set is found, the training set is used to estimate the accuracy of the proposed set by 10-fold cross-validation. In order to avoid the overfitting risk, rather than considering a continuous weight space, they restrict the possible weights to a small, finite set.

All the algorithms reviewed so far are deterministic in the sense that all runs over the same data will always give the same result. As far as we know, GAs, EDAs and random sampling have been the only non-deterministic search engines applied to the FW problem. With the non-deterministic approach, randomness is used to avoid getting stuck in local optimas: this implies that one should not expect the same weight set solution from different runs.

GAs are used in much work in this area (Kelly and Davis, 1991; Punch et al., 1993; Rozsypal and Kubat, 2001; Wilson and Martínez, 1996). Proposed GA works use the wrapper approach to guide the search, with access to the training set each time a new weight set is found, but they differ slightly in the way that they apply it. Kelly and Davis (1991) measure the five-fold cross-validation accuracy on training data. Punch et al. (1993) use a 5-NN approach, with LOOCE used on the training data, and propose a mixed fitness function which combines the LOOCE with the number of neighbors that were not used in the final classification of each training instance. Rozsypal and Kubat (2001), at the same time they select the proper features (only two discrete weights are used: 0.0 and 1.0), they also try to reduce the number of training prototypes, and propose a mixed fitness function which involves a trade-off between the LOOCE value and the number of retained examples. Wilson and Martínez (1996) just measure the LOOCE on the training data.

Sierra et al. (2001), by the use of univariate EDAs, also try to simultaneously select features and prototypes for the Nearest Neighbor classifier in a medical problem. A wrapper approach is used, measuring the LOOCE on the training set.

Skalak (1994) uses Monte Carlo sampling to simultaneously select features and prototypes for Nearest Neighbor. He also utilizes random mutation hill climbing (Papadimitriou and Steiglitz, 1982), a local search method that has a stochastic component: changing at random one point in the solution until an improvement is achieved, with a bound on the maximum number of iterations. The wrapper approach is used, with a hold-out estimate to measure the feature and prototype subset performance on the training data.

The feature selection algorithm for Nearest Neighbor proposed by Aha and Bankert (1994), in a domain with 204 features, also has a random sampling component: they randomly sample a specific part of the feature space for a fixed number of iterations and then begin a beam search with the best (by the ten-fold cross-validation wrapper approach) feature subset found during those iterations.

All the reviewed algorithms state the FW task as a search problem. They are grouped under the term wrapper because they use feedback from the Nearest Neighbor classifier itself during training to learn weights. In order to measure the value of the wrapper function to be optimized, while on-line weighting algorithms only use the training set once, the rest of the algorithms presented use the training set each time a new weight set is found.

Other well-known approaches do not state the FW task as a search problem and learn feature weights from the intrinsic characteristics of the data. These approaches do not make use of the Nearest Neighbor classifier itself to learn the weights, and, as in the case of FSS task, they can therefore be grouped under the term filter (Kohavi and John, 1997). To learn a set of weights, these classic approximations make use of conditional probabilities (Crecy et al., 1992), class projections (Stanfill and Waltz, 1986) (Howe and Cardie, 1997)), mutual-information (Wettschereck and Dietterich, 1995) and information-gain (van den Bosch and Daelemans, 1993). In other interesting work, Cardie and Howe (1997) first build a decision tree to select features and then weight each feature according to its information-gain score. The algorithm proposed by Güvenir and Akkus (1997) assumes that the weight of a feature is proportional to the accuracy that will be obtained by considering only that feature for classification when using the OneR (Holte, 1993) classifier.

7.4 Learning weights for Nearest Neighbor by Bayesian and Gaussian networks

We use the search mechanism provided by EBNA (Etzeberria and Larrañaga, 1999) to search a set of appropriate weights for the FW task: the algorithm is called FW-EBNA. In order to specify the nature of our search space for FW-EBNA, we restrict the space of possible weights to a set of discrete weights. Thus, we follow the findings of Kohavi et al. (1997a) to determine the number of possible discrete weights. Applying a wrapper approach (Kohavi and John, 1997), the authors found that considering only a small set of weights gave better results than using a larger set: increasing the number of possible weights greatly increased the variance in their FW algorithm, which induced a deterioration in the overall performance. However, when the wrapper approach was applied, they concluded that the overall utility of increasing the number of possible weights above two or three is negative. In FW-EBNA the search is also performed by a wrapper procedure in a discrete space of weights: we restrict our search space to three possible weights for each feature, i.e. $\{0, 0.5, 1\}$. If n is the number of features in a domain, then the cardinality of the search space for FW-EBNA is 3^n . With this restricted set of possible feature weights, a common notation is used to represent each individual: for a full n feature problem, there are n bits in each proposed solution, where each bit indicates whether a feature has 0.0 weight, 0.5 weight or 1.0 weight.

The same implementation of the EBNA algorithm proposed for FSS-EBNA is used for FW-EBNA: a fast score + search procedure is used to learn the Bayesian network in each generation of the search, where Algorithm B (Buntine, 1991) constitutes the search part and the BIC metric (Schwarz, 1978) the score part.

The set of possible weights can be extended to a continuous space of weights by the use of the EGNA (Larrañaga et al., 2000a) search engine. By using Gaussian networks, we consider a continuous space of weights for each feature in the $[0, 1]$ range. In our specific implementation of the EGNA algorithm a fast score + search procedure is also preferred for learning the Gaussian network in each search generation. The search method is also Algorithm B (Buntine, 1991) and the BGe (Geiger and Heckerman, 1994) is the applied scoring metric.

As the execution of the Nearest Neighbor algorithm has a high computational cost, it is traditionally applied over datasets with a low number of instances and features. Taking into account this low dimensionality, Bayesian and Gaussian networks are attractive paradigms within the EDA paradigm to discover the relationships among the moderated amount of variables of the domain. In this way, the use of simpler probabilistic models which are not able to reflect multiple order relationships paradigm among domain variables is discarded.

Taking the moderated dimensionality of our problems into account, as in the case of FSS, we consider that a population size of 1,000 individuals within the EDA paradigm is enough to reliably estimate the network parameters.

Since our objective is to research FW rather than the correct amount of neighbors (K) to be considered for classification, the number of neighbors for FW-EBNA and FW-EGNA is fixed to one. Thus, the 1-NN classifier is used.

The same scheme proposed for FSS is used to configure subsequent populations. The best individual of the previous generation is maintained and $N - 1$ individuals are created as offspring. An elitist approach is used then to form iterative populations. Instead of directly discarding the $N - 1$ individuals from the previous generation and replacing them with $N - 1$ newly generated ones, the $2N - 2$ individuals are put together and the best $N - 1$ chosen from them. These best $N - 1$ individuals together with the best individual of the previous generation form the new population. In this way, the populations converge faster to the best individuals found, but, this also carries a risk of losing diversity within the population.

The wrapper schema (Kohavi and John, 1997) is applied to assess the evaluation function of each proposed solution, by calculating the LOOCE of the 1-NN classifier applied over the found set of weights. Following the basic EDA scheme, the initial population of weights is randomly created.

In order to adopt a stopping criteria, we are based on the study about the overfitting problem and election of an appropriate stopping criteria carried out in Chapter 6. As we said in the second section of this Chapter, the Nearest Neighbor algorithm is normally applied over small (and usually noisy) training sets, and this increases the risk of impact of the overfitting phenomenon. In an attempt to reduce this overfitting risk, we adopt the criteria to stop the search when in a sampled new generation no feature weight set appears with a LOOCE improvement, with a p -value smaller than 0.1 (applying a cross-validated paired t test), over the lowest LOOCE of the previous generation. Thus, the feature weight set with the lowest LOOCE of the previous generation is returned as FW-EBNA's or FW-EGNA's solution. This criteria was also adopted to tackle the FSS task for datasets with less than 1,000 instances (see Chapter 6).

As in the case of FSS, adopting this stopping criterion, we aim to avoid the overfitting risk of the wrapper process, by only allowing the search to continue when a significant improvement appears in the accuracy estimation of the best feature weights of consecutive generations. We hypothesize that when this significant improvement appears, the overfitting risk decays and there is a basis for further generalization accuracy improvement over unseen instances. When this improvement is absent, we hypothesize that the search is getting stuck in an area of the search space without statistically significant better feature weights (compared to already found ones). Therefore, it is best to stop the search to avoid the risk of overfitting.

Another consideration in this stopping criterion is the wrapper nature of the proposed evaluation function, the LOOCE estimate of the 1-NN classifier using the found weights. As we will see in the next section, the evaluation function value of each visited solution needs several seconds to be calculated. As the simulation of a new generation of individuals implies the evaluation of 1,000 new individuals, the search is only allowed to continue when it demonstrates that it is able to escape from local optima, and can discover new best feature weights in each generation. When the wrapper approach is used, the CPU time must be also controlled: we hypothesize that when the search is allowed to continue by our stopping criterion, the CPU times to evaluate a new generation of feature weights are justified.

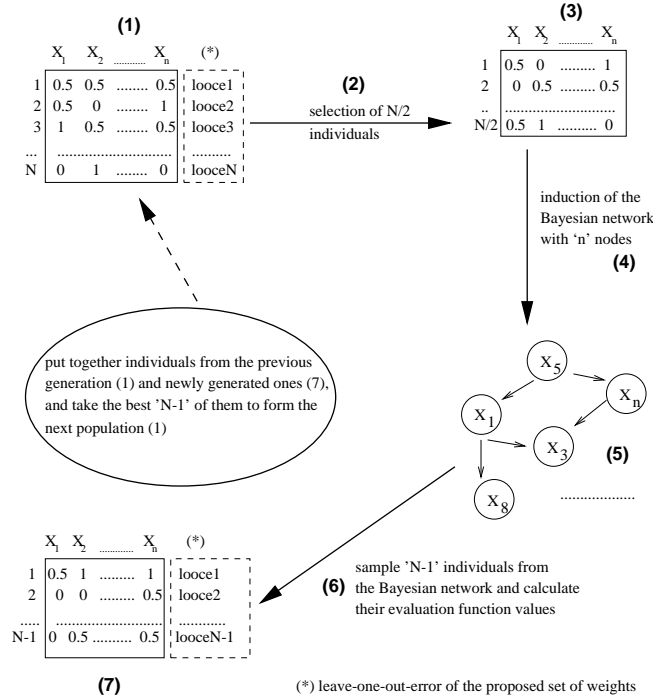


Figure 7.1. FW-EBNA method.

Figure 7.1 gives an overview of the FW-EBNA method. FW-EGNA only differs from FW-EBNA in the probabilistic model employed to factorize the probability distribution of selected solutions and in the set of possible weights.

7.5 Experimental comparison

The power of FW-EBNA and FW-EGNA is tested on four artificial and four real domains. All real datasets are presented in Chapter 2 of this dissertation. Table 7.1 summarizes the characteristics of these domains.

LED24 is a well known artificial dataset with 7 equally relevant and 17 irrelevant binary features.

Waveform-21 is another well known artificial dataset. All the features have different degrees of relevance. Both datasets have a significant degree of noise.

The *3-Weights* domain has 12 continuous features in the range $[3, 6]$. Its target concept is to define whether the instance is closer (using the Euclidean metric in all dimensions and summing them) to $(0, 0, \dots, 0)$ or $(9, 9, \dots, 9)$. In the distance computation 1.0 weight is assigned to 4 features, 0.5 weight to another 4 features and 0.0 to the 4 remaining ones. The *3-Weights* dataset is inspired by the *W* domain proposed by Kohavi et al. (1997a).

The *C-Weights* domain has 10 continuous features in the range $[3, 6]$ and its target concept is the same as the *3-Weights* dataset. However, randomly created weights in the continuous range $[0, 1]$ are assigned for the distance computation of the features.

We hypothesize that, while *LED24* and *3-Weights* domains are properly designed for the FW-EBNA approach, the continuous nature of *Waveform-21* and *C-Weights* should be better for FW-EGNA. These four domains arise from natural tasks for which the true weights are unknown.

Table 7.1. Details of experimental domains.

<i>Domain</i>	<i>Number of instances</i>	<i>Number of features</i>
(1) <i>LED24</i>	600	24
(2) <i>Waveform-21</i>	600	21
(3) <i>3-Weights</i>	600	12
(4) <i>C-Weights</i>	600	10
(5) <i>Glass</i>	214	9
(6) <i>CRX</i>	690	15
(7) <i>Vehicle</i>	846	18
(8) <i>Contraceptive</i>	1,473	9

Due to the large number of instances in the *Contraceptive* domain, instead of a LOOCE evaluation scheme, a multiple-times 10-fold cross-validation procedure is used (Kohavi and John, 1997). The 10-fold cross-validation is repeated until the standard deviation drops below 1%, up to a maximum of five times.

To test the power of FW-EBNA and FW-EGNA, a comparison with two sequential and one genetic FW algorithm is carried out. These FW algorithms are:

- Genetic Algorithm with one-point crossover (FW-GA-o). This performs the search for each variable in the same set of 3 discrete weights as FW-EBNA;
- a improved variation of the sequential DIET (called DIET-10) algorithm proposed by Kohavi et al. (1997a). This also performs the search in the same 3 discrete weight space as above. DIET and DIET-10 use the best-first (Russell and Norvig, 1995) algorithm to guide the search. The search starts with the solution $(0.5, 0.5, \dots, 0.5)$ and is stopped when it encounters 10 consecutive nodes with no children having scores more than 0.1% better than their parent. The improvement introduced relative to Kohavi et al.'s (1997a) implementation is that the original DIET algorithm stops the search when the number of consecutive nodes with worse child scores is only 5;
- IB4 (Aha, 1992) assigns weights to features by means of a hill-climbing, sequential, incremental and on-line strategy, with only one pass through the training data. As with FW-EGNA, IB4 performs the search in the $[0, 1]^n$ continuous weight space.

FW-GA-o and DIET-10 use the same wrapper evaluation function as the FW EDA approaches (FW-EBNA and FW-EGNA). Although DIET-10 and IB4 stop deterministically, FW-GA-o applies the same halting criteria as FW EDA algorithms. While DIET-10, FW-GA-o and FW-EBNA perform the search in 3-weight discrete space, IB4 and FW-EGNA's search space is continuous.

For the genetic approach, the same parameter selection performed in Chapter 6 is selected. The probability of crossover is set to 1.0 and the mutation probability to $1/n$. Fitness-proportional selection is used to select individuals for crossover. In order to avoid any bias in the comparison, the remaining FW-GA-o parameters are the same as the EDA approaches: the population size is set to 1,000 and the new population is formed from the best members of both the old population and its offspring.

Because of the non-deterministic nature of FW-EBNA, FW-EGNA and FW-GA-o, 5 replications of 2-fold cross-validation (5x2cv) are applied to assess the predictive generalization accuracy of all the FW algorithms being compared.

Table 7.2. Accuracy percentages of the Nearest Neighbor algorithm using the 5 FW methods shown and without FW. The standard deviation of the estimated percentage is also reported.

<i>Domain</i>	<i>no-FW</i>	<i>DIET-10</i>	<i>FW-GA-o</i>	<i>FW-EBNA</i>	<i>IB4</i>	<i>FW-EGNA</i>
<i>LED24</i>	$47.37 \pm 3.36\dagger$	$63.84 \pm 2.42^*$	68.64 ± 1.30	69.03 ± 1.54	66.70 ± 1.80	$61.55 \pm 1.90\dagger$
<i>Waveform-21</i>	76.20 ± 1.48	76.66 ± 1.62	76.71 ± 1.57	76.87 ± 1.04	77.96 ± 1.62	76.90 ± 1.48
<i>3-Weights</i>	$77.19 \pm 3.36\dagger$	$81.91 \pm 1.98^*$	82.88 ± 1.66	85.99 ± 1.57	$80.32 \pm 4.46\dagger$	82.00 ± 2.72
<i>C-Weights</i>	$81.01 \pm 1.14\dagger$	83.55 ± 1.56	83.93 ± 1.24	83.55 ± 1.56	$81.98 \pm 1.84\dagger$	84.33 ± 1.31
<i>Glass</i>	$64.85 \pm 2.15\dagger$	71.34 ± 4.89	71.32 ± 2.97	71.12 ± 5.01	$61.13 \pm 5.55^*$	70.09 ± 2.83
<i>CRX</i>	$81.56 \pm 1.92^*$	$82.12 \pm 2.01^*$	83.14 ± 1.81	83.74 ± 1.94	85.48 ± 0.92	$82.17 \pm 2.14^*$
<i>Vehicle</i>	67.33 ± 2.11	68.71 ± 1.48	69.86 ± 1.42	69.43 ± 2.11	$64.65 \pm 2.32^*$	69.58 ± 2.33
<i>Contraceptive</i>	$43.61 \pm 0.97\dagger$	47.54 ± 2.99	48.10 ± 2.50	48.32 ± 2.34	$45.66 \pm 2.66^*$	$44.95 \pm 2.10\dagger$
Avg. artif.	70.44	76.49	78.04	78.86	76.74	76.19
Avg. real	64.33	67.42	68.10	68.15	64.23	66.69

We extend the comparison by running the Nearest Neighbor algorithm with homogeneous weights (no-FW). Table 7.2 shows the accuracy results for these algorithms.

A deeper analysis of the accuracy results is carried out by using statistical tests. The 5x2cv F (Alpaydin, 1999) test is performed to determine the significance degree of accuracy differences among the proposed algorithms. In Table 7.2, the symbol ‘ \dagger ’ denotes a statistically significant difference from the best FW algorithm in the domain at the 0.05 confidence level; ‘ $*$ ’, significance at the 0.1 level. These symbols have the same meaning in all the tables in this chapter. Experiments are executed on a SGI-Origin 200 computer.

Using different FW techniques, statistically significant accuracy improvements are achieved relative to the no-FW approach in all datasets except *Waveform-21* and *Vehicle*. Although accuracy differences between FW algorithms are not statistically significant for most of the datasets, FW-EBNA has the best average accuracy among the compared methods in the artificial and real datasets. However, on average, the accuracies of IB4, DIET-10 and FW-EGNA in the artificial domains and IB4 in the real datasets are notably poorer than the results obtained by FW-EBNA.

It is not an easy task to show significant accuracy differences among classification algorithms in real datasets. Kohavi and John (1997) argue that real datasets are already preprocessed to include only relevant features, and this makes the appearance of significant accuracy differences among compared classification techniques unlikely. In natural and artificial domains, FW-GA-o and FW-EBNA achieve similar average values, and they do not show statistically significant accuracy differences in any domain. The other population based algorithm, FW-EGNA, has an inferior average behavior in both natural and artificial domains with respect to FW-GA-o and FW-EBNA. In this way, FW-EGNA shows statistically significant differences in *LED24* and *Contraceptive* domains with respect to the best algorithm in these tasks, FW-EBNA.

Note the irregular behaviour of IB4: while it shows the best accuracies in *Waveform-21* and *CRX*, it has significant differences relative to the best algorithm in all the other datasets except *LED24*. On the other hand, the comparable behaviour of DIET-10 relative to the population-based algorithms in many of the datasets must be noted.

FW-EBNA has the best estimated accuracy in the artificial domains with discrete weights, *LED24* and *3-Weights*. On the other hand, in the artificial domains with continuous true weights, while FW-EGNA shows the best accuracy in *C-Weights*, its accuracy is surpassed by IB4 in *Waveform-21*.

Table 7.3. Mean stop-generation for FW-GA-o, FW-EBNA and FW-EGNA. The standard deviation of the mean is also reported. The initial generation is considered to be the zero generation.

<i>Domain</i>	<i>FW-GA-o</i>	<i>FW-EBNA</i>	<i>FW-EGNA</i>
<i>LED24</i>	$6.50 \pm 0.54^\dagger$	4.50 ± 0.47	8.66 ± 0.51
<i>Waveform-21</i>	2.50 ± 1.37	2.33 ± 1.03	2.16 ± 1.83
<i>3-Weights</i>	2.50 ± 1.22	2.66 ± 0.81	3.50 ± 1.04
<i>C-Weights</i>	5.16 ± 1.16	3.33 ± 1.50	3.16 ± 1.16
<i>Glass</i>	1.00 ± 0.89	1.00 ± 0.63	1.66 ± 1.50
<i>CRX</i>	1.33 ± 1.75	1.33 ± 0.51	1.83 ± 1.83
<i>Vehicle</i>	$3.16 \pm 0.75^*$	1.16 ± 0.40	1.33 ± 0.81
<i>Contraceptive</i>	$2.66 \pm 0.81^*$	1.83 ± 0.98	2.00 ± 0.63

It must be remembered that among the FW algorithms being compared, only FW-EGNA and IB4 perform the search in a continuous weight space.

In spite of the similar accuracy levels obtained by FW-GA-o and FW-EBNA in all the datasets, it must be noted that in several domains, there are significant differences between both algorithms in the number of generations needed to obtain these similar accuracies. Table 7.3 shows the generations in which population-based FW-GA-o and FW-EBNA stop with the stopping criterion shown earlier. Statistically significant differences between these algorithms are marked. Table 7.3 also shows FW-EGNA's mean-stop generation, but since the nature of their search spaces is different, comparisons between FW-EGNA and the population-based discrete FW algorithms (FW-GA-o and FW-EBNA) should be made with caution.

Although FW-GA-o and FW-EBNA do not have statistically different accuracies in the *LED24*, *Vehicle* and *Contraceptive* domains, FW-EBNA needs statistically fewer generations than FW-GA-o to obtain the percentages shown. With these datasets, it seems that FW-EBNA, by using Bayesian networks, is able to capture the underlying structure of the problem faster than FW-GA-o. This superiority of EDA approaches using Bayesian networks relative to GAs is also noted in the Chapter 6, when dealing with the FSS problem.

When the wrapper approach is used to calculate the evaluation function value of each found weight set, fast discovery of similar or better accuracies becomes a critical task. As the Nearest Neighbor algorithm needs several CPU seconds to estimate the LOOCE of the proposed set of weights, a faster discovery of these similar fitness solutions is highly desirable. By thus, avoiding the simulation of several generations of solutions, a large amount of CPU time is saved.

In order to understand the advantages of this fast discovery of similar fitness solutions, CPU times for the induction of the probabilistic models must be studied: the EDA approaches have the computational overhead of the calculation of probabilistic models in each generation. Table 7.4 shows, for each domain, the average CPU time to induce the Bayesian network structure in each generation and the average CPU time needed to estimate the predictive accuracy of a single feature weight set. Although FW-EGNA is not included in this comparison between FW-GA-o and FW-EBNA, its average CPU time for the induction of the Gaussian network structure in each generation is also shown in Table 7.4.

From the results of Table 7.4, we can conclude that since the CPU times for the induction of Bayesian networks in each EDA generation are insignificant, their CPU time saving relative to the GA approach shown in Table 7.3 is maintained.

Table 7.4. Average CPU times (in seconds) for the induction of different probabilistic models (standard deviations are nearly zero) in each generation of the EDA search. The last column shows the average CPU time to estimate the predictive accuracy of a feature weight set.

<i>Domain</i>	<i>FW-EGNA</i>	<i>FW-EBNA</i>	<i>1-NN acc. estim.</i>
<i>LED24</i>	39.4	5.5	29.8
<i>Waveform-21</i>	21.3	5.0	35.0
<i>3-Weights</i>	2.2	3.6	22.7
<i>C-Weights</i>	1.5	3.4	21.4
<i>Glass</i>	1.0	2.2	3.3
<i>CRX</i>	5.2	4.2	34.9
<i>Vehicle</i>	10.3	4.4	60.4
<i>Contraceptive</i>	1.0	2.2	20.7

Let us now consider the accuracy results obtained by FW-EGNA. In the scheme used of 5 replications of 2-fold cross-validation (5x2cv), for most of the datasets, the feature weight selected by FW-EGNA usually has the best accuracy estimation of the FW algorithms compared over the training fold. However, when this weight set is tested on the novel instances that form the second fold (instances that are unseen during the training process), a notable decay in the percentage accuracy is noted, and its accuracy levels are similar or inferior to these of the other FW algorithms. Although this overfitting risk also exists for the rest of the algorithms, the largest accuracy differences between the training fold and the test fold of the 5x2cv scheme appear in the FW-EGNA algorithm. It seems that allowing a continuous set of feature weights does not result in better accuracy levels.

These findings agree with those of Kohavi et al. (1997a). The authors find that the extra power given by an increased set of weights does not further reduce the bias of the error and usually increases its variance, resulting in an overfitting problem: to avoid this, they recommend the use of a set of 2 or 3 different weights. As we note in the first section of this Chapter, the Nearest Neighbor classifier is traditionally applied over datasets with few instances because of its high computational cost. In this way, when we design a wrapper algorithm that works with the Nearest Neighbor classifier, we must bear in mind that this overfitting risk seems to be inherent to its execution over this kind of datasets.

As the feature weights of the *C-Weights* and *Waveform-21* domains are continuous, it is interesting to study the differing behaviour of FW-EGNA in these two domains. In *C-Weights*, a non-noisy domain where the range of the feature weights is continuous, FW-EGNA has a slight advantage in its final accuracy relative to the other algorithms. On the other hand, in *Waveform-21*, a domain with an important noise degree (Breiman et al., 1984), FW-EGNA's accuracy significantly decays and the other continuous FW algorithm, IB4, has the most accurate result. Although FW-EGNA has the best estimation percentages for the training folds (in the 5x2cv estimation scheme) of *Waveform-21* and *C-Weights*, its accuracy for new instances in the test fold significantly decays in the *Waveform-21* domain. As *C-Weights* is not a noisy domain, the existence of a significant noise degree is the reason for the appearance of this overfitting problem in *Waveform-21*. We have also seen this overfitted behaviour with the real datasets, domains where we can assume that noise exists.

In our case, unless we have a specifically designed and non-noisy domain such as *C-Weights* where the use of a continuous weight set gives a slight advantage, the naive assumption that using more weights with a wrapper scheme as in FW-EGNA will reduce the classification error seems false.

7.6 Summary and future work

The application of the EDA approach to the FW problem for the Nearest Neighbor algorithm is studied in this chapter. Two powerful probabilistic models, Bayesian and Gaussian networks, are applied to factorize the probability distribution of weight set solutions: Bayesian networks are used with a set of 3 possible discrete weights and Gaussian networks are used with a continuous range of weights. Both new methods, FW-EBNA (Feature Weighting by Estimation of Bayesian Network Algorithm) and FW-EGNA (Feature Weighting by Estimation of Gaussian Network Algorithm), use the wrapper scheme for the evaluation of proposed weight set solutions, calculating the LOOCE value of each proposed weight, and using it to guide the search. A comparison is performed in a set of natural and artificial domains with two sequential and one genetic inspired algorithm.

With the application of the FW techniques, notable accuracy improvements are achieved in the major part of datasets. While FW-EBNA shows the best average accuracy results for both natural and artificial domains, the impact of overfitting on noisy datasets notably reduces the accuracy of FW-EGNA. We therefore confirm the findings of Kohavi et al. (1997a), who find that the extra power given by an increased set of weights does not further reduce the bias of the error and usually increases the variance and the overfitting risk. Thus, an advice for the use of a set of 2 or 3 different weights seems to be appropriate for this context of work. Unless we have a specifically designed and non-noisy domain with continuous feature weights, the naive assumption that using more weights with a wrapper scheme will reduce the classification error seems to be false.

As in the case of FSS EDA approaches, it is noted that the genetic approach needs more generations than FW-EBNA to discover similar fitness solutions. In order to save CPU time, when the wrapper approach is used to calculate the evaluation function value of each found weight set, the fast achievement of similar or better accuracies becomes a critical task.

As future work, we envision the development of evolutionary algorithms that tackle both the FW and the prototype selection tasks within the EDA approach. Another interesting research avenue is the design of a different stopping criteria, in an attempt to avoid the overfitting risk in FW-EGNA.

Chapter 8

Conclusions

In Section 8.1, we briefly summarize the contributions of this work. In Section 8.2 we conclude with some future lines of research, also exposed in the previous chapters of this dissertation.

8.1 Summary of contributions

This dissertation is focused in three topics that arise in Supervised Classification:

- the representation of the joint behaviour of a set of supervised classifiers;
- the Feature Subset Selection task;
- the Feature Weighting for Nearest Neighbor algorithm.

By the use of probabilistic graphical models and the EDA evolutionary search paradigm, we have presented a set of contributions which try to solve these issues. This dissertation has contributions of other types: study of related topics and extensive reviews of the related literature.

These contributions can be summarized as follows:

- review of the Supervised Classification paradigm;
- detailed review of two Probabilistic Graphical Models: Bayesian and Gaussian networks. Special attention is paid to the score+search approach for the induction of these graphical models. The existing literature about Bayesian and Gaussian networks is reviewed;
- detailed presentation of the EDA approach. Existing approaches for discrete and continuous domains are reviewed and details about the related literature are supplied. Special attention is paid to the reasons that motivate the apparition of the EDA paradigm. Its close relations with the GA paradigm are studied;
- application of Bayesian networks to study the relationships among the predictive models induced by a set of supervised classifiers. In order to reflect the relationships among single classifiers or predefined families of classifiers, development of the following three properties, based on the conditional independence property:
 - the hard conditional independence property of a single classifier;

- the conditional independence in a predefined family of classifiers;
- the conditional independence between a pair of predefined families of classifiers. Based on these properties, analysis of the relationships shown by a group of fourteen supervised classifiers in a set of eleven medical datasets;
- detailed review of the FSS approach: its basic components are extensively studied and a large list of literature references in the field is supplied;
- study of the ‘overfitting’ phenomenon in FSS. Its large impact on datasets with few instances is noted;
- development of a novel EDA inspired algorithm, which makes use of the EBNA search procedure, for FSS in small and medium dimensionality domains: FSS-EBNA. In a comparison with two sequential and two genetic inspired algorithms on real and artificial datasets, FSS-EBNA shows the best average accuracy results and performs a competitive dimensionality reduction. GA approaches need more generations than FSS-EBNA to discover similar fitness solutions. In the case of GA with one-point crossover, this difference is enlarged when the interacting variables are not coded together in the individual’s representation;
- development of four novel EDA inspired algorithms for FSS in large dimensionality domains: FSS-PBIL, FSS-BSC, FSS-MIMIC and FSS-TREE. These algorithms make use of PBIL, BSC, MIMIC and TREE search procedures, respectively. In a comparison with two genetic inspired approaches, four algorithms obtain competitive accuracy results and dimensionality reductions. In real datasets, FSS-BSC, FSS-TREE and FSS-MIMIC arrive faster to similar fitness areas than the other algorithms. The following conclusions are extracted for artificial datasets:
 - for a dataset where no interactions appear among the domain features, genetic approaches obtain faster similar fitness solutions than four EDA approaches;
 - for datasets where interactions of order two and order three exist among domain features, FSS-MIMIC and FSS-TREE outperform the rest of the approaches with respect to the number of generations needed to obtain similar fitness solutions;
 - GA with one-point crossover needs more generations to arrive to a specific fitness value when the interacting variables are not coded together in the individual’s representation.
- a study about the time requirements needed by PBIL, BSC, MIMIC and TREE algorithms to factorize the probability distribution of best individuals in large dimensionality domains within the FSS problem;
- development of two novel algorithms, FW-EBNA and FW-EGNA, to solve the FW problem for the Nearest Neighbor classifier. FW-EBNA and FW-EGNA use the EDA inspired EBNA and EGNA procedures, respectively. In a comparison with two sequential and one genetic approach on real and artificial datasets, FW-EBNA shows the best average accuracy results;
- the overfitted behaviour of FW-EGNA, which considers a continuous space of weights in the $[0,1]$ range. In this way, the confirmation of previous literature findings that, unless we have a non-noisy domain with many different weights, the assumption that considering more weights with a wrapper scheme will improve the classification accuracy is false;
- an extensive review of previous works in the FW task;
- the GA approach needs more generations than FW-EBNA to discover similar fitness solutions;

8.2 Future work

This section groups the future lines of research that have been exposed in the previous chapters of this dissertation:

- in a continuation about the joint behaviour of supervised classifiers, the use of unsupervised hierarchical classification to determine clusters or families of algorithms;
- the study of the connection between our conclusions about the joint behaviour of supervised classifiers and the accuracy improvement usually obtained by the combination of classifiers;
- the application of the developed probabilistic relationships in the joint behaviour approach to the genetic network paradigm of the Bioinformatics discipline;
- in order to reduce the CPU times in large dimensionality FSS problems within the EDA approach, the development of other simple probabilistic models and parallel algorithms to induce Bayesian networks;
- the application of EDA inspired FSS processes in DNA microarray datasets;
- the use of other probabilistic models for the FW EDA approach;
- the development of other stopping criteria that reduce the overfitting risk of the FW-EGNA approach.

References

- Aha, D. W. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36:267–287.
- Aha, D. W. (1999). Personal communication.
- Aha, D. W. and Bankert, R. L. (1994). Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the AAAI’94 Workshop on Case-Based Reasoning*, pages 106–112.
- Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.
- Akaike, H. (1974). New look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.
- Alba, E., Santana, R., Ochoa, A., and Lazo, M. (2000). Finding typical testors by using an evolutionary strategy. In *Proceedings of the Fifth Ibero American Symposium on Pattern Recognition*, pages 267–278.
- Almuallin, H. and Dietterich, T. G. (1991). Learning with many irrelevant features. In *Proceedings of AAAI-91*, pages 547–552.
- Alpaydin, E. (1999). Combined 5x2cv f test for comparing Supervised Classification learning algorithms. *Neural Computation*, 11:1885–1892.
- Andersen, S. ., Olesen, K. ., Jensen, F. ., and Jensen, F. (1989). HUGIN: a shell for building Bayesian belief universes for expert systems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1128–1133.
- Auer, P., Holte, R., and Maass, W. (1995). Theory and applications of agnostic PAC-learning with small decision trees. In *Proceedings of the Twelfth International Conference*.
- Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.
- Baluja, S. (1994). Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University.
- Baluja, S. and Caruana, R. (1995). Removing the genetics from standard Genetic Algorithm. In Frieditis, A. and Russell, S., editors, *Proceedings of the International Conference on Machine Learning*, pages 38–46. Morgan Kaufmann.
- Baluja, S. and Davies, S. (1998). Fast probabilistic modeling for combinatorial optimization. In *Fifteenth National Conference on Artificial Intelligence, AAAI-98*, pages 469–476.

- Bell, D. A. and Wang, H. (2000). A formalism for relevance and its application in Feature Subset Selection. *Machine Learning*, 41:175–195.
- Ben-Bassat, M. (1982). Pattern recognition and reduction of dimensionality. In Krishnaiah, P. and Kanal, L., editors, *Handbook of Statistics II*, pages 773–791. North-Holland. Amsterdam.
- Bengoetxea, E., Larrañaga, P., Bloch, I., Perchant, A., and Boeres, C. (2002). Learning and simulation of Bayesian networks applied to inexact graph matching. *Pattern Recognition*, to appear.
- Blanco, R., Larrañaga, P., Inza, I., and Sierra, B. (2001). Selection of highly accurate genes for cancer classification by Estimation of Distribution Algorithms. In *Workshop of Bayesian Models in Medicine, AIME 2001*, pages 29–34.
- Blum, A. L. and Langley, P. (1997). Selection of relevant features and examples in Machine Learning. *Artificial Intelligence*, 97:245–271.
- Boddy, M. and Dean, T. (1994). Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67(2):245–285.
- Bosman, P. A. N. and Thierens, D. (1999). Linkage information processing in distribution estimation algorithms. In Banzhaf, W., Daida, J., Eiben, A. E., Garzón, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, volume 1, pages 60–67. Morgan Kaufmann Publishers. San Francisco, LA.
- Bouckaert, R. R. (1995). *Bayesian Belief Networks: From Construction to Inference*. PhD Thesis, University of Utrecht.
- Bouckaert, R. R., Castillo, E., and Gutiérrez, J. M. (1996). A modified simulation scheme for inference in Bayesian networks. *International Journal of Approximate Reasoning*, 14:55–80.
- Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.
- Buntine, W. (1991). Theory refinement in Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60.
- Buntine, W. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210.
- Cardie, C. (1993). Using decision trees to improve case-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 25–32.
- Cardie, C. and Howe, N. (1997). Improving minority class prediction using case-specific feature weights. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 57–65.
- Caruana, R. and Freitag, D. (1993). Greedy attribute selection. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 28–36.
- Castillo, E., Gutiérrez, J. M., and Hadi, A. S. (1997). *Expert Systems and Probabilistic Network Models*. Springer-Verlag, New York.
- Cestnik, B. (1990). Estimating probabilities: a crucial task in Machine Learning. In *Proceedings of the European Conference on Artificial Intelligence*, pages 147–149.
- Chickering, D. M., Geiger, D., and Heckerman, D. (1994). Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research, Redmond, WA.
- Chickering, D. M., Geiger, D., and Heckerman, D. (1995). Learning Bayesian networks: Search methods and experimental results. In *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128.
- Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467.
- Clark, P. (1998). Personal communication.

- Clark, P. and Boswell, R. (1991). Rule induction with CN2: some recent improvements. In Kodratoff, Y., editor, *Machine Learning - EWSL91*, pages 151–163. Springer Verlag.
- Clark, P. and Nibblet, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3(4):261–283.
- Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference in Machine Learning*, pages 115–123.
- Cooper, G. F. (1990). The computational complexity of probabilistic inference using belief networks. *Artificial Intelligence*, 42:393–405.
- Cooper, G. F. and Herskovits, E. A. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.
- Cost, S. and Salzberg, S. (1993). A weighted Nearest Neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78.
- Cotta, C., Alba, E., Sagarna, R., and Larrañaga, P. (2001). Adjusting weights in artificial neural networks using evolutionary algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 361–377. Kluwer Academic Publishers.
- Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.
- Crecy, R. H., Masand, B. M., Smith, S. J., and Waltz, D. L. (1992). Trading MIPS and memory for knowledge engineering. *Communications of the ACM*, 35:48–64.
- Das, S. (2001). Filters, wrappers and a hybrid: evaluating Feature Selection algorithms. In *Proceedings of the Eighteenth International Conference in Machine Learning*, pages 74–81.
- Dasarathy, B. V. (1991). *Nearest Neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press.
- Dawid, A. P. (1979). Conditional independence in statistical theory. *Journal of the Royal Statistics Society, Series B*, 41:1–31.
- De Bonet, J. S., Isbell, C. L., and Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, Vol. 9:424–430.
- de Campos, L. M. (1998). Automatic learning of graphical models. I: Basic methods. In Gámez, J. A. and Puerta, J. M., editors, *Probabilistic Expert System*, pages 113–140. Ediciones de la Universidad de Castilla-La Mancha. (In spanish).
- de Campos, L. M., Gámez, J. A., Larrañaga, P., Moral, S., and Romero, T. (2001). Partial abductive inference in Bayesian networks: an empirical comparison between GAs and EDAs. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 323–342. Kluwer Academic Publishers.
- DeGroot, M. (1970). *Optimal Statistical Decisions*. McGraw-Hill, New York.
- Dempster, A. P. (1972). Covariance selection. *Biometrika*, 32:95–108.
- Dietterich, T. G. (1997). Machine Learning research: four current directions. *Artificial Intelligence Magazine*, 18(4):97–136.
- Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised learning algorithms. *Neural Computation*, 10(7):1895–1924.
- Dietterich, T. G. and Shavlik, J. W. (1990). *Readings in Machine Learning*. Morgan Kaufmann Publishers.
- Doak, J. (1992). An evaluation of Feature Selection methods and their application to computer security. Technical Report CSE-92-18, University of California at Davis.
- Dutton, D. M. and Conroy, G. V. (1996). A review of machine learning. *The Knowledge Engineering Review*, 12(4):341–367.

- Eberhardt, M., Kossebau, F. W. H., and König, A. (2001). Automatic Feature Subset by Genetic Algorithms. In *Proceedings of the Fifth International Conference in Artificial Neural Networks and Genetics Algorithms*, pages 256–259.
- Etzeberria, R. and Larrañaga, P. (1999). Global optimization with Bayesian networks. In *II Symposium on Artificial Intelligence. CIMA99. Special Session on Distributions and Evolutionary Optimization*, pages 332–339.
- Ferri, F. J., Pudial, P., Hatef, M., and Kittler, J. (1993). Feature subset search using genetic algorithms. In *Proceedings of the IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, pages 23/1–23/7.
- Ferri, F. J., Pudil, P., Hatef, M., and Kittler, J. (1994). Comparative study of techniques for large scale feature selection. In Gelsema, E. and Kanal, L., editors, *Multiple Paradigms, Comparative Studies and Hybrid Systems*, pages 403–413. North Holland.
- Fisher, R. (1936). The use of multiple measurements in Taxonomic problems. *Annals of Eugenics*, 7:179–188.
- Friedman, N. and Yakhini, Z. (1996). On the sample complexity of learning Bayesian networks. In *Proceedings of the Twelveth Conference on Uncertainty in Artificial Intelligence*, pages 274–282.
- Gallagher, M. (2000). *Multi-layer perceptron error surfaces: visualization, structure and modelling*. PhD Thesis, Department of Computer Science and Electrical Engineering, University of Queensland.
- Geiger, D. and Heckerman, D. (1994). Learning Gaussian networks. Technical report, Microsoft Advanced Technology Division, Microsoft Corporation, Seattle, Washington.
- Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley.
- Goldberg, D., Korb, B., and Deb, K. (1989). Messy Genetic Algorithms: motivation, analysis, and first results. *Complex Systems*, 3(5):493–530.
- Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caliguri, M., Bloomfield, C., and Lander, E. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537.
- González, C., Lozano, J. A., and Larrañaga, P. (2000). Analyzing the PBIL algorithms by means of discrete dynamical systems. *Complex Systems*, 12(4):465–479.
- González, C., Lozano, J. A., and Larrañaga, P. (2001). The convergence behavior of PBIL algorithm: a preliminar approach. In Kůrková, V., Steel, N. C., Neruda, R., and Kárný, M., editors, *International Conference on Artificial Neural Networks and Genetic Algorithms. ICANNGA-2001*, pages 228–231. Springer.
- González, C., Lozano, J. A., and Larrañaga, P. (2002). Mathematical modelling of the UMDA_c algorithm with selection by tournament applied to linear functions. In *Proceedings of Primer Congreso Ibero Americano de Algoritmos Evolutivos y Bioinspirados, AEB'02*. Accepted (in spanish).
- Grefenstette, J. J. (1986). Optimization of control parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 1:122–128.
- Guerra-Salcedo, C. and Whitley, D. (1998). Genetic search for Feature Subset Selection. In *Proceedings of the Third Annual Genetic Programming Conference*, pages 504–509.
- Güvenir, H. A. and Akkus, A. (1997). Weighted K Nearest Neighbor classification on feature projections. In *Proceedings of the Twelfth International Symposium on Computer and Information Sciences*, pages 44–51.
- Hall, M. A. (1999). *Correlation-based Feature Selection for Machine Learning*. PhD Thesis, Waikato University.

- Hall, M. A. and Holmes, G. (2000). Benchmarking attribute selection techniques for Data Mining. Technical Report Working Paper 00/10, Department of Computer Science. University of Waikato, Waikato, New Zealand.
- Harik, G. (1999). Linkage learning in via probabilistic modeling in the ECGA. Technical Report 99010, IlliGAL Technical Report.
- Harik, G., Lobo, F. G., and Golberg, D. E. (1998). The compact Genetic Algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 523–528.
- Harik, G. R. and Goldberg, D. (1996). Learning linkage. Technical Report IlliGAL Report 96006, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.
- Heckerman, D. (1995). A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Advanced Technology Division, Microsoft Corporation, Seattle, Washington.
- Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.
- Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In Lemmer, J. F. and Kanal, L. N., editors, *Uncertainty in Artificial Intelligence*, volume 2, pages 149–163. North-Holland, Amsterdam.
- Herskovits, E. A. and Cooper, G. F. (1990). Kutató: An entropy-driven system for construction of probabilistic expert systems from database. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 54–62.
- Höhfeld, M. and Rudolph, G. (1997). Towards a theory of population-based incremental learning. In *Proceedings of the 4th International Conference on Evolutionary Computation*, pages 1–5. IEEE Press.
- Holte, R. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91.
- Hosmer, D. and Lemeshow, S. (1989). *Applied Logistic Regression*. Wiley Series in Probability and Mathematical Statistics.
- Howe, N. and Cardie, C. (1997). Examining locally varying weights for Nearest Neighbor algorithms. In *Lecture Notes in Artificial Intelligence. Case-Based Reasoning Research and Development. Second International Conference on Case-Based Reasoning*, pages 455–466.
- Inza, I. (1999). Feature Weighting for Nearest Neighbor algorithm by Bayesian networks based combinatorial optimization. In *Proceedings of the Student Sessions ACAI'99: Machine Learning and Applications*, pages 33–35.
- Inza, I., Larrañaga, P., Etxeberria, R., and Sierra, B. (2000). Feature Subset Selection by Bayesian network-based optimization. *Artificial Intelligence*, 123(1-2):157–184.
- Inza, I., Larrañaga, P., and Sierra, B. (2001a). Feature Subset Selection by Bayesian networks: a comparison with genetic and sequential algorithms. *International Journal of Approximate Reasoning*, 27(2):143–164.
- Inza, I., Larrañaga, P., and Sierra, B. (2001b). Feature Subset Selection by Estimation of Distribution Algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 269–294. Kluwer Academic Publishers.
- Inza, I., Larrañaga, P., and Sierra, B. (2001c). Feature Weighting in K-NN by means of Estimation of Distribution Algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 295–312. Kluwer Academic Publishers.

- Inza, I., Larrañaga, P., and Sierra, B. (2002a). Estimation of Distribution Algorithms for Feature Subset Selection in large dimensionality domains. In Abbass, H. A., Sarker, R. A., and Newton, C. S., editors, *Data Mining: a Heuristic Approach*. IDEA Group Publishing.
- Inza, I., Larrañaga, P., Sierra, B., Etxeberria, R., Lozano, J. A., and Peña, J. M. (1999). Representing the joint behaviour of Supervised Classification learning algorithms by Bayesian networks. *Pattern Recognition Letters*, 20:1201–1209.
- Inza, I., Merino, M., Larrañaga, P., Quiroga, J., Sierra, B., and Giralá, M. (2001d). Feature Subset Selection by Genetic Algorithms and Estimation of Distribution Algorithms. A case study in the survival of cirrhotic patients treated with TIPS. *Artificial Intelligence in Medicine*, 23(2):187–205.
- Inza, I., Sierra, B., Blanco, R., and Larrañaga, P. (2002b). Gene selection by sequential search wrapper approaches in microarray cancer class prediction. Submitted.
- Jain, A. and Zongker, D. (1997). Feature Selection: evaluation, application, and small sample performance. *IEEE Transactions and Pattern Analysis and Machine Intelligence*, 19(2):153–158.
- Jain, A. K. and Chandrasekaran, R. (1982). Dimensionality and sample size considerations in Pattern Recognition practice. In Krishnaiah, P. R. and Kanai, L. N., editors, *Handbook of Statistics*, volume 2, pages 835–855. North-Holland.
- Jain, A. K., Duin, R. W., and Mao, J. (2000). Statistical Pattern Recognition. A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37.
- Jensen, C. S., Kong, A., and Kjærulff, U. (1993). Blocking Gibbs sampling in very large probabilistic expert systems. Technical Report R 13-2031, Department of Mathematics and Computer Science, University of Aalborg, Denmark.
- Jensen, F. (2001). *Bayesian networks and decision graphs*. Springer-Verlag.
- Jensen, F. V., Olesen, K. G., and Andersen, S. K. (1990). An Algebra of Bayesian belief universes for knowledge based systems. *Networks*, 28(5):637–659.
- John, G., Kohavi, R., and Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129.
- Kelly, J. D. and Davis, L. (1991). A hybrid Genetic Algorithm for classification. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 645–650.
- Kira, K. and Rendell, L. A. (1992). A practical approach to Feature Selection. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 249–256.
- Kittler, J. (1978). Feature set search algorithms. In Chen, C., editor, *Pattern Recognition and Signal Processing*, pages 41–60. Sithoff and Noordhoff.
- Kohavi, R. (1994). Feature Subset Selection as a search with probabilistic estimates. In *Proceedings of the AAAI Fall Symposium on Relevance*, pages 122–126.
- Kohavi, R. (1995a). Bottom-up induction of oblivious, read-one decision graphs: strengths and limitations. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 613–618.
- Kohavi, R. (1995b). Feature Subset Selection using the wrapper method: overfitting and dynamic search space topology. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 192–197.
- Kohavi, R. (1995c). The power of decision tables. In Lavrac, N. and Wrobel, S., editors, *Lecture Notes in Artificial Intelligence*, volume 914, pages 174–189. Springer Verlag.
- Kohavi, R. (1995d). *Wrappers for performance enhancement and oblivious decision graphs*. PhD Thesis, Stanford University.
- Kohavi, R. (1996). Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*.

- Kohavi, R. (1999). Personal communication.
- Kohavi, R. and John, G. (1997). Wrappers for Feature Subset Selection. *Artificial Intelligence*, 97(1-2):273–324.
- Kohavi, R., Langley, P., and Yun, Y. (1997a). The utility of Feature Weighting in Nearest-Neighbor algorithms. In *European Conference on Machine Learning*, page poster.
- Kohavi, R., Sommerfield, D., and Dougherty, J. (1997b). Data mining using MLC++, a Machine Learning library in C++. *International Journal of Artificial Intelligence Tools*, 6:537–566.
- Kohavi, R. and Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference in Machine Learning*.
- Koller, D. and Sahami, M. (1996). Toward optimal Feature Selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292.
- Krause, P. J. (1998). Learning probabilistic networks. *Knowledge Engineering Review*, 13:321–351.
- Kudo, M. and Sklansky, J. (2000). Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33:25–41.
- Kullback, S. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86.
- Kuncheva, L. (1993). Genetic Algorithms for Feature Selection for parallel classifiers. *Information Processing Letters*, 46:163–168.
- Kushmerick, N. (1999). Learning to remove Internet advertisements. In *Proceedings of the AGENTS-99 Conference*, pages 175–181.
- Lachenbruch, P. A. (1967). An almost unbiased method for obtaining confidence intervals for the probability misclassification in discriminant analysis. *Biometrics*, 23:639–645.
- Lachenbruch, P. A. and Mickey, R. (1968). Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–11.
- Lam, W. and Bacchus, F. (1994). Learning Bayesian belief networks. An approach based on the MDL principle. *Computational Intelligence*, 10(4):269–293.
- Langley, P. and Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406.
- Larrañaga, P. (2001a). An introduction to Probabilistic Graphical Models. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 27–56. Kluwer Academic Publishers.
- Larrañaga, P. (2001b). A review on Estimation of Distribution Algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 57–100. Kluwer Academic Publishers.
- Larrañaga, P., Etxeberria, R., Lozano, J., and Peña, J. (2000a). Optimization in continuous domains by learning and simulation of Gaussian networks. In *Proceedings of the Workshop in Optimization by Building and using Probabilistic Models*, pages 201–204.
- Larrañaga, P., Etxeberria, R., Lozano, J. A., and Peña, J. M. (1999). Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report KZZA-IK-4-99, Department of Computer Science and Artificial Intelligence, University of the Basque Country.
- Larrañaga, P., Etxeberria, R., Lozano, J. A., and Peña, J. M. (2000b). Combinatorial optimization by learning and simulation of Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 343–352.
- Larrañaga, P., Etxeberria, R., Lozano, J. A., and Peña, J. M. (2000). Optimization in continuous domains by learning and simulation of Gaussian networks. In Wu, A. S., editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, pages 201–204.

- Larrañaga, P. and Lozano, J. A. (2001). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Press.
- Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R. H., and Kuijpers, C. M. H. (1996). Structure learning of Bayesian networks by Genetic Algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926.
- Last, M., Kandel, A., and Maimon, O. (2001). Information-theoretic algorithm for Feature Selection. *Pattern Recognition Letters*, 22(6-7):799–811.
- Lauritzen, S. L. (1996). *Graphical Models*. Oxford University Press.
- Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50(2):157–224.
- Li, L., Pedersen, L. G., Darden, T. A., and Weinberg, C. (2002). Computational analysis of leukemia microarray expression data using the GA/KNN method. In Lin, S. M. and Johnson, K. F., editors, *Methods of Microarray Data Analysis. First Conference on Critical Assessment of Microarray Data Analysis, CAMDA2000*, pages 81–96.
- Liu, H. and Motoda, H. (1998). *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers.
- Liu, H. and Setiono, R. (1996). Feature Selection and classification - a probabilistic wrapper approach. In *Proceedings of the Ninth International Conference on Industrial and Engineering Applications of AI and ES*, pages 419–424.
- Liu, H. and Setiono, R. (1998). Incremental Feature Selection. *Applied Intelligence*, 9(3):217–230.
- Lowe, D. (1995). Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 7:72–85.
- Lozano, J. and Mendiburu, A. (2001). EDAs applied to the job scheduling problem. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 231–242. Kluwer Academic Publishers.
- Lozano, J. A. (1998). *Genetic Algorithms applied to Unsupervised Classification*. PhD Thesis, Department of Computer Science and Artificial Intelligence, University of the Basque Country (in spanish).
- Lozano, J. A., Sagarna, R., and Larrañaga, P. (2001). Parallel Estimation of Distribution Algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 129–146. Kluwer Academic Publishers.
- Madigan, D. and Raftery, A. (1994). Model selection and accounting for model uncertainty in graphical models using Occams window. *Journal of the American Statistical Association*, 89:1535–1546.
- Mahnig, T. and Mühlenbein, H. (2000). Mathematical analysis of optimization methods using search distributions. In Wu, A. S., editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, pages 205–208.
- Maxwell, B. and Anderson, S. (1999). Training hidden Markov models using population-based learning. In *Genetic and Evolutionary Computation Conference, GECCO-99*.
- Michalski, R., Bratko, I., and Kubat, M. (1998). *Machine Learning, and Data Mining: Methods and Applications*. Wiley.
- Michalski, R. S. (2000). Learnable Evolution Model: evolutionary processes guided by Machine Learning. *Machine Learning*, 38(1-2):9–40.
- Michie, D., Spiegelhalter, D., and Taylor, C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

- Miller, A. J. (1990). *Subset Selection in Regression*. Chapman and Hall.
- Mitchell, T. (1982). Generalization as a search. *Artificial Intelligence*, 18:203–226.
- Mitchell, T. (1997). *Machine Learning*. McGraw Hill.
- Mladenić, M. (1998). Feature Subset Selection in Text Learning. In *Proceedings of the Tenth European Conference on Machine Learning*, pages 95–100.
- Moore, A. W. and Lee, M. S. (1994). Efficient algorithms for minimizing cross validation error. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 190–198.
- Mühlenbein, H. and Paaß, G. (1996). From recombination of genes to the estimation of distributions. In *Lecture Notes in Computer Science: Parallel Solving from Nature IV*, volume 1411, pages 178–187. Springer Verlag.
- Mühlenbein, H. (1998). The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303–346.
- Mühlenbein, H. and Mahnig, T. (2000). Evolutionary algorithms: from recombination to search distributions. *Theoretical Aspects of Evolutionary Computing. Natural Computing*, pages 137–176.
- Mühlenbein, H., Mahnig, T., and Ochoa, A. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5:215–247.
- Murphy, P. (1995). *UCI Repository of Machine Learning databases*. University of California, Department of Information and Computer Science.
- Murthy, S. K., Kasif, S., and Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32.
- Myers, J. W., Laskey, K. B., and Levitt, T. (1999). Learning Bayesian networks from incomplete data with stochastic search algorithms. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 476–485.
- Narendra, P. and Fukunaga, K. (1977). A branch and bound algorithm for Feature Subset Selection. *IEEE Transactions on Computer*, C-26(9):917–922.
- Neapolitan, E. (1990). *Probabilistic Reasoning in Expert Systems*. John Wiley and Sons, New York.
- Ng, A. Y. (1997). Preventing ‘overfitting’ of cross-validation data. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 245–253.
- Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, Palo Alto, CA.
- Pelikan, M. and Goldberg, D. E. (2000). Hierarchical problem solving and the Bayesian optimization algorithm. In Whitley, D., Goldberg, D. E., Cantú-Paz, E., Spector, L., Parmee, I., and Beyer, H.-G., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 267–274. Morgan Kaufmann.
- Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1998). Linkage problem, distribution estimation, and Bayesian networks. Technical Report IlliGAL Report 98013, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.
- Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, volume 1, pages 525–532. Morgan Kaufmann Publishers, San Francisco, CA. Orlando, FL.
- Pelikan, M. and Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. *Advances in Soft Computing-Engineering Design and Manufacturing*, pages 521–535.

- Peña, J. M. (2001). *On Unsupervised Learning of Bayesian Networks and Conditional Gaussian Networks*. PhD Thesis, Department of Computer Science and Artificial Intelligence, University of the Basque Country.
- Peña, J. M., Lozano, J. A., Larrañaga, P., and Inza, I. (2001). Dimensionality reduction in unsupervised learning of conditional Gaussian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):590–603.
- Perner, P. (2001). Improving the accuracy of decision tree induction by feature pre-selection. *Journal of Applied Artificial Intelligence*, 15(8):747–760.
- Pudil, P., Novovicova, J., and Kittler, J. (1994). Floating search methods in Feature Selection. *Pattern Recognition Letters*, 15(1):1119–1125.
- Punch, W. F., Goodman, E. D., Pei, M., Chia-Shun, L., Hovland, P., and Enbody, R. (1993). Further research on Feature Selection and classification using Genetic Algorithms. In *Proceedings of the International Conference on Genetic Algorithms*, pages 557–564.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.
- Quinlan, J. R. (1989). Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.
- Rana, S., Whitley, L. D., and Cogswell, R. (1996). Searching in the presence of noise. In *Parallel Problem Solving in Nature 4*, pages 198–207.
- Ripley, B. D. (1987). *Stochastic Simulation*. John Wiley and Sons.
- Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, pages 465–471.
- Rivera, J. and Santana, R. (1999). Improving the discovery component of classifier systems by the application of Estimation of Distribution Algorithms. In *Proceedings of the Student Sessions ACAI'99: Machine Learning and Applications*, pages 43–44.
- Robles, V., de Miguel, P., and Larrañaga, P. (2001). Solving the travelling salesman problem with Estimation of Distribution Algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 211–230. Kluwer Academic Publishers.
- Roure, J., Sangüesa, R., and Larrañaga, P. (2001). Partitional clustering by means of Estimation of Distribution Algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 343–360. Kluwer Academic Publishers.
- Rozsypal, A. and Kubat, M. (2001). Using the Genetic Algorithm to reduce the size of a Nearest-Neighbor classifier and to select relevant attributes. In *Proceedings of the Eighteenth International Conference in Machine Learning*, pages 449–456.
- Rudlof, S. and Köppen, M. (1996). Stochastic hill climbing by vectors of normal distributions. In *Proceedings of the First Online Workshop on Soft Computing (WSC1)*. Nagoya, Japan.
- Russell, S. J. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall.
- Salmerón, A., Cano, A., and Moral, S. (2000). Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413.
- Salzberg, S. (1991). A nearest hyperrectangle learning method. *Machine Learning*, 6:251–276.
- Sangüesa, R. and Cortés, U. (1998). Learning causal networks from data: a survey and a new algorithm for recovering possibilistic causal networks. *AI Communications*, 10:31–61.
- Sangüesa, R., Cortés, U., and Gisolfi, A. (1998). A parallel algorithm for building possibilistic causal networks. *International Journal of Approximate Reasoning*, 18(3-4):251–270.

- Santana, R. and Ochoa, A. (1999). Dealing with constraints with Estimation of Distribution Algorithms: the univariate case. In *Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAFA 99*, pages 378–384. La Habana.
- Santana, R., Ochoa, A., Soto, M., Pereira, F. B., Machado, P., Costa, E., and Cardoso, A. (2000). Probabilistic evolution and the busy beaver problem. In Whitley, D., Goldberg, D., Cantú-Paz, E., Spector, L., Parmee, I., and Beyer, H.-G., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 380–380. Morgan Kaufmann.
- Scherf, M. and Brauer, W. (1997). Feature Selection by means of a Feature Weighting approach. Technical Report FKI-221-97, Forschungsberichte Künstliche Intelligenz, Institut für Informatik, Technische Universität München, München, Germany.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 7(2):461–464.
- Schwarz, J. and Ocenasek, J. L. (1999). Experimental study: hypergraph partitioning based on the simple and advanced algorithms BMMA and BOA. In *Proceedings of the Fifth International Conference on Soft Computing*, pages 124–130.
- Sebag, M. and Ducoulombier, A. (1998). Extending population-based incremental learning to continuous search spaces. In *Parallel Problem Solving from Nature - PPSN V*, pages 418–427. Springer-Verlag, Berlin.
- Servet, I., Trave-Massuyes, L., and Stern, D. (1997). Telephone network traffic overloading diagnosis and evolutionary techniques. In *Proceedings of the Third European Conference on Artificial Evolution, (AE'97)*, pages 137–144.
- Shachter, R. and Kenley, C. (1989). Gaussian influence diagrams. *Management Science*, 35:527–550.
- Shafer, G. R. (1996). *Probabilistic Expert Systems*. Society for Industrial and Applied Mathematics.
- Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.
- Siedelecky, W. and Sklansky, J. (1988). On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2:197–220.
- Sierra, B. (2000). *Methodological advances in Supervised Classification*. PhD Thesis, Department of Computer Science and Artificial Intelligence, University of the Basque Country (in spanish).
- Sierra, B., Jiménez, E., Inza, I., Larrañaga, P., and Muruzábal, J. (2001a). Rule induction using Estimation of Distribution Algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 313–322. Kluwer Academic Publishers.
- Sierra, B., Lazkano, E., Inza, I., Merino, M., Larrañaga, P., and Quiroga, J. (2001b). Prototype Selection and Feature Subset Selection by Estimation of Distribution Algorithms. A case Study in the survival of cirrhotic patients treated with TIPS. In *Proceedings of the Eighth Artificial Intelligence in Medicine in Europe*, pages 20–29.
- Skalak, D. (1994). Prototype and Feature Selection by sampling and random hill climbing algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 293–301.
- Smith, P. W. F. and Whittaker, J. (1998). Edge exclusion tests for graphical Gaussian models. In *Learning in Graphical Models*, pages 555–574. Kluwer Academic Publishers, Dordrecht, The Netherlands.
- Speed, T. P. and Kiiveri, H. (1986). Gaussian Markov distributions over finite graphs. *Annals of Statistics*, 14:138–150.
- Spirites, P., Glymour, C., and Scheines, R. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computing Reviews*, 9:62–72.

- Spirites, P., Glymour, C., and Scheines, R. (1993). *Causation, Prediction, and Search*. Lecture Notes in Statistics 81, Springer-Verlag.
- Stanfill, C. and Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228.
- Stearns, S. D. (1976). On selecting features for pattern classifiers. In *Proceeding of the Third International Conference on Pattern Recognition*, pages 71–75.
- Stone, M. (1974). Cross-validation choice and assesment of statistical predictions. *Journal of the Royal Statistic Society*, 36:111–147.
- Syswerda, G. (1993). Simulated crossover in Genetic Algorithms. *Foundations of Genetic Algorithms 2*, pages 239–255.
- Talavera, L. (2000). Dependency-based Feature Selection for clustering symbolic data. *Intelligent Data Analysis*, 4:19–28.
- Thierens, D. and Goldberg, D. E. (1993). Mixing in Genetic Algorithms. In *Proceedings of the Fifth International Conference in Genetic Algorithms*, pages 38–45.
- Urban, J. S. (1994). *Computer intensive statistical methods*. Chapman and Hall.
- Vafaie, H. and Jong, K. (1993). Robust Feature Selection algorithms. In *Proceedings of the Fifth International Conference on Tools with Artificial Intelligence*, pages 356–363.
- Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27:1134–1142.
- van den Bosch, A. and Daelemans, W. (1993). Data-oriented methods for grapheme-to-phoneme conversion. Technical Report 42, Tilburg University, Institute for Language Technology and Artificial Intelligence, Tilburg, The Netherlands.
- van Kemenade, C. H. M. (1998). Building block filtering and mixing. Technical Report SEN-R9837, Centrum voor Wiskunde en Informatica from the Neetherlands.
- Wettschereck, D., Aha, D. W., and Mohri, T. (1997). A review and empirical evaluation of Feature Weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314.
- Wettschereck, D. and Dietterich, D. (1995). An experimental comparison of the Nearest-Neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19:1–25.
- Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. John Wiley and Sons.
- Wilson, R. and Martinez, T. R. (1996). Instance-based learning with genetically derived attribute weights. In *Proceedings of the International Conference on Artificial Intelligence, Expert Systems and Neural Networks*, pages 11–14.
- Wong, M. L., Lam, W., and Leung, K. S. (1999). Using evolutionary computation and minimum description length principle for Data Mining of probabilistic knowledge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(2):174–178.
- Xiang, Y. and Chu, T. (1999). Parallel learning of belief networks in large and difficult domains. *Data Mining and Knowledge Discovery*, 3(3):315–338.
- Xing, E. P., Jordan, M. I., and Karp, R. M. (2001). Feature Selection for high-dimensional genomic microarray data. In *Proceedings of the Eighteenth International Conference in Machine Learning, ICML2001*, pages 601–608.
- Yang, J. and Honavar, V. (1998). Feature Subset Selection using a Genetic Algorithm. *IEEE Intelligent Systems*, 13(2):44–49.
- Yang, Y. and Pedersen, J. O. (1997). A comparative study on Feature Selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420.

Index

- AIC, 24
- Abductive inference, 41
- Accuracy estimation, 14, 51, 68, 87
- Algorithm B, 23, 44, 67, 86
- Ancestral ordering, 27, 31
- BDe, 27, 30, 44
- BGe, 30, 86
- BIC, 24, 44, 48, 67, 86
- BMDA, 42
- BOA, 44
- BSC, 38, 76, 96
- Bayesian network, 2, 20, 44, 52, 67, 86, 95
 - simulation, 27
- Bayesian score, 24, 30
- Best-first, 64, 85, 89
- Bias, 7
- Bioinformatics, 38, 59, 97
- Branch and bound, 63
- C4.5, 8, 52
- CGA, 39
- CN2, 10, 52
- COMIT, 42
- Clustering
 - partitional, 38
- Conditional (in)dependence, 2, 18, 22, 30, 55, 95
- Covariance matrix, 29
- Crossover, 33, 72, 89
- DIET, 89
- Data Mining, 61, 65
- Deceptive problems, 34
- Density function, 18
- Dependencies
 - bivariate, 39, 47
 - multivariate, 43, 47
 - without, 37, 46
- Dependency graph, 43
- Directed Acyclic Graph, 18
- Discriminant Analysis, 6
- EBNA, 44, 66, 86, 96
- EDAs, 2, 34, 66, 75, 85–86, 95
- EGNA, 48, 86, 96
- EMNA, 48
- EcGA, 44
- Edge exclusion tests, 29
- Embedded Feature Selection, 65
- Empirical Learning, 1
- F test, 16, 72, 90
- FDA, 44
- Feature
 - Subset Selection, 2, 38–39, 41–42, 45, 62, 95–96
 - Weighting, 2, 45, 49, 84, 95–96
- Filter, 65, 86
- Five times two-fold cross-validation, 16, 68, 77, 89
- GAs, 2, 33, 48, 53, 64, 72, 77, 84–85, 89, 95
- Gaussian distribution
 - multivariate, 28
- Gaussian network, 2, 28, 48, 86, 95
 - simulation, 31
- Genetic network, 59, 97
- Gradient descent, 84
- Graph matching, 45
- HOODG, 8, 52
- Hebbian rule, 46
- Hill-climbing, 71, 84, 89
- Holdout, 15, 52
- IB1, 9, 52
- IB4, 9, 52, 89
- ID3, 8, 52
- Jobshop Scheduling Problem, 42, 49
- K-fold cross-validation, 15, 65, 68–69, 85, 89
- K2, 25, 45, 53
- Knapsack problem, 42
- Kullback-Leibler, 27, 40–41
- LEM, 35
- LOOCE, 15, 84–85, 87, 91
- Las Vegas Algorithm, 64
- Likelihood ratio statistic, 29, 54
- Linear-regression model, 28
- Log marginal likelihood, 25
- Logistic Regression, 7
- MDL, 44
- MGA, 34
- MIMIC, 40, 76, 96
- MIMIC_C^G, 47
- MLC++, 53, 72
- MWST, 41

- Machine Learning, 1, 7, 51, 61, 83, 95
- Maximized log likelihood, 23
- Maximum likelihood estimate, 23
- Model averaging
 - Bayesian, 24
 - selective, 25
- Model selection, 25
- Monte Carlo sampling, 85
- Moralize, 19
- Multivariate normal, 28, 48
- Mutation, 33, 72, 89
- NB, 9, 52, 68–69, 76
- NBTree, 9, 52
- NP-hard, 21, 63, 84
- Nearest Neighbor, 2, 9, 52, 83, 95–96
- OC1, 8, 52
- Occam's razor, 6
- OneR, 8, 52, 86
- Overfitting, 2, 53, 68, 85, 87, 96
- PBIL, 38, 76, 96
- PBIL_c, 47
- PC, 45
- PEBLS, 9, 52
- PLS, 27, 31, 37, 46
- Parametric learning, 22
- Pattern Recognition, 62, 83
- Pearson's χ^2 statistic, 43
- Penalized maximum likelihood, 23, 30
- Precision matrix, 29
- Probabilistic Graphical Model, 2, 18, 52, 66, 86, 95
- Probability distribution, 17, 34
- Probability mass, 17
- Random subsampling, 15
- Random variable
 - n -dimensional, 17
- Resubstitution estimate, 15
- Ripper, 10, 52
- Rule induction, 42, 45
- SBE, 64, 71
- SFS, 64, 71
- SHCLVND, 46
- Score+search, 2, 23, 29, 44, 48, 53, 67, 86, 95
- Sequential Floating, 64
- Shannon entropy, 41
- Simulated Annealing, 64
- Statistics, 6, 62
- Stopping criteria, 66, 68, 87, 97
- Structure learning, 22, 40, 43
- Supervised Classification, 1, 5, 51, 61, 83, 95
- T test, 15, 70, 87
- T2, 8, 52
- TREE, 41, 76, 96
- TSP, 41–42, 45
- Table-Majority, 9, 52
- Text-Learning, 62
- UCI Repository, 10, 52
- UMDA, 37, 44
- UMDA_c, 46
- UMDA_c^G, 46
- Unsupervised Classification, 1, 59, 62, 97
- Wrapper, 64, 68, 85, 87, 96