Universidad del País Vasco
Euskal Herriko Unibertsitatea

Intelligent Systems Group

informatika fakultatea
facultad de informática

Konputazio Zientziak eta Adimen Artifizialaren Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Positive Unlabelled Learning with Applications in Computational Biology

by

Borja Calvo

Supervised by Jose A. Lozano and Pedro Larrañaga

Dissertation submitted to the Department of Computer Science and Artificial Intelligenceof the University of the Basque Country as partial fulfilment of the requirements for the PhD degree in Computer Science

Donostia - San Sebastián, 30^{th} of September, 2008

Dedicated to those that are gone
and to those that are to come ...

Izan ziren eta
Izango direnentzat

Acknowledgement

Writing the acknowledgements of a work like this is a very gratifying experience because it has shown me how many people have been there to help and support me.

The very first people I owe the greatest thanks to my advisers, Pedro Larrañaga and Jose Antonio Lozano without whose wise guidance I am certainly sure I would not be here writing these lines. I really feel very lucky to have been under the suppervision of such good people, as I know that finding good advisers is not easy at all. They have always had time and a smile when I needed their help.

I am also in great debt to all the people in the Intelligent Systems Group with whom I have shared these four years: Alex Mendiburu, Aritz Pérez, Carlos Echegoyen, Dinora Morales, Endika Bengoetxea, Guzmán Santafé, Iñaki Inza, Jose Antonio Fernandes, Jose Luis Flores, Josu Galdiano, Juan Diego Rodríguez, Ramón Sagarna, Roberto Santana, Rosa Blanco, Rubén Armañanzas and Tesesa Miquélez (and Joseba Esteban, our most frequent visitor :P). I cannot fail to mention here two people that, though not official members of the group, should be considered as part of it: Johnny Kennedy and Elena Bidondo. There are no words to describe how thankful I am to all these people that have contributed with their good mood to turn the daily work into a really enjoyable experience. They have much of the merit of my being here.

During these four years I have had the fortune to meet and collaborate with many people. Among these people I am specially thankful to Nuria López-Bigas and Simon J. Furney from the Biomedical Genomics group from the

Research Unit on Biomedical Informatics in Barcelona, not only because all the applications presented in this dissertation are the result of this collaboration, but also because their interest in the identification of disease genes was the motivation for our starting with the positive unlabelled learning problem.

I have a special thanks to Martin Vingron and all the people in his group for being my hosts me during my stay at the Max Plank Institute for Molecular Genetics in Berlin.

Little (and not so little) decisions determine our path through life. It would be impossible to remember all the key events that have led me to this point, but there are some of them that I cannot forget. In particular, I am specially thankful to Mikel Santos, not only for the years we shared in Leioa, but specially for convincing me to start reading computer science.

If there is anybody that I should be thankful to that is my family. I am absolutely sure that I would not be writting these lines without the constant encouragement and support of my mother, my father and my brother. To them I owe everything.

Last but not least, the most special thanks are for my wife, Arantzazu, the love of my life. I would need several books like this one to properly thank her for her support, not only in this project but in all aspects of my life. Mpmz.

Summary

With the increasing amount of stored information, the use of data mining techniques to extract useful knowledge from them has become a key issue in many domains. Classifier induction algorithms are useful tools for this purpose as they allow us to summarise the information contained in the datasets into classification functions that can be used to make predictions on new data.

One of the applications of classifier induction algorithms is the information retrieval, this can be defined as the recovery of objects of a particular class (those that we are interested in, normally known as 'positive') from a big set of unlabelled objects (i.e., we do not know whether they are positive or not). The classical approaches to this problem require the use of positive examples (objects of the kind we want to recover) and negative examples (examples different to the kind of objects we want to retrieve), but negative examples are not always available. Therefore, for the last ten years people have been developing algorithms that are able to learn binary classifiers in absence of negative examples.

The positive unlabelled learning problem is the topic of this dissertation. The contributions presented in this work cover model induction, model averaging and feature subset selection algorithms and the evaluation of classifiers in absence of negative examples. In the applied part of the work presented in this dissertation some of the proposed algorithms are used to solve two computational biology problems, the identification of genes associated with hereditary diseases and with cancer.

Resumen

Con el aumento de la cantidad de información almacenada, el uso de técnicas de minería de datos (data mining) se han convertido en una pieza clave en muchos campos. Los algoritmos de inducción de clasificadores son herramientas muy útiles ya que permiten condensar la información contenida en las bases de datos en clasificadores que pueden luego ser usados para realizar predicciones sobre nuevos datos.

Una de las aplicaciones de los algoritmos de inducción de clasificadores es la recuperación de información, que puede ser definida como la recuperación de los objetos de un tipo determinado (aquellos en los cuales estamos interesados, normalmente llamados 'positivos') de grandes conjuntos de objetos no etiquetados (es decir, objetos que no sabemos a que clase pertenecen). Las aproximaciones clásicas implican tener ejemplos positivos (ejemplos del tipo de objetos que queremos recuperar) y ejemplos negativos (ejemplos de objetos diferentes a los que queremos recuperar), pero no siempre hay disponibles ejemplos negativos. Por este motivo, durante los últimos años se han venido desarrollando algoritmos que permitan aprender clasificadores binarios en ausencia de ejemplos negativos.

El tema de esta tésis es el aprendizaje a partir de ejemplos positivos y no etiquetados. Las contribuciones de esta tésis abarcan la inducción de modelos de clasificación, el promediado de clasificadores, la selección de variables y la evaluación de clasificadores. En la parte aplicada, algunos de los algoritmos propuestos son utilizados para resolver dos problemas del area de la biología, la identificación de genes asociados a enfermedad y genes involucrados en cancer.

Laburpena

Datu baseetan bildutako informazio kantitatearen gorakadarekin, datu meatzaritza (data mining) teknikak arlo askotan informazio baliagarria lortzeko oso garrantzitsuak bilakatu dira. Klasifikatzaileak ikasten dituzten algoritmoak oso tresna erabilgarriak dira, datu baseetan dagoen informazioa la-burtzen dutelako, gero datu berriekin iragarpenak egin ahal izateko.

Aurretik aipatutako algoritmoen aplikaziotariko bat, informazioa berreskuratzea da, hau da, mota bereziko objektuak (normalean "positiboak" deritzotenak) etiketatu gabeko objektu multzo handietan identifikatzea. Algoritmo klasikoak bai objektu positiboak, baita negatiboak ere (hau da, positiboak ez direnak) behar dituzte klasifikatzailea ikasteko. Hala ere, zenbait egoeretan adibide negatiboak ez daude eskura. Hori dela eta, azken hamar urte hauetan adibide negatiborik gabe klasifikatzaile binarioak ikasteko gai diren algoritmoak garatu dira.

Tesi honen gaia klasifikatzaileen ikasketa da, adibide positibo eta etiketatu gabeko adibideak erabiliz. Tesiaren ekarpenen artean klasifikatzaile ikasketa, konbinaketa eta ebaluaketa eta aldagai aukeraketa aurki daitezke. Aplikazioetan, proposatutako algoritmoak biologiako bi arazoetan erabili dira: herentziazko eritasun eta kantzerrarekin lotuta dauden geneen identifikazioa.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Introduction

1

Introduction

We live in a world where information is everywhere and, due to the great advances in technology, the storage of this information in big databases is more and more frequent in many and such diverse domains as biology, economy or information technology. But most of these large databases are of very reduced utility if they are not 'mined' in search of knowledge.

The machine learning discipline provides the data miner with the tools required for the data mining process (see Figure 1.1). The first step in the knowledge extraction workflow is the preprocessing of the raw data gathered that has to be selected and formated before running the different machine learning algorithms.

Machine learning algorithms can be applied to the formated data to extract some of the underlying knowledge. Among the machine learning tools used in the knowledge extraction step we can find the classification model induction algorithms that, given a dataset of examples, are able to learn predictive models that can be used to classify new unseen instances. In other words, they are able to model the past experience stored in databases.

Depending on the examples available, we can find different classification problems. Supervised classification term refers to the process of learning classification models from labelled examples, i.e., we have a set of examples and we know the class of all of them. When we do not know the class of the examples we have an unsupervised classification problem, also known as clustering, where the main goal is to identify the cluster structure of the examples. Indeed, there is not an explicit class variable, we create it to indicate which instances belong to which cluster.

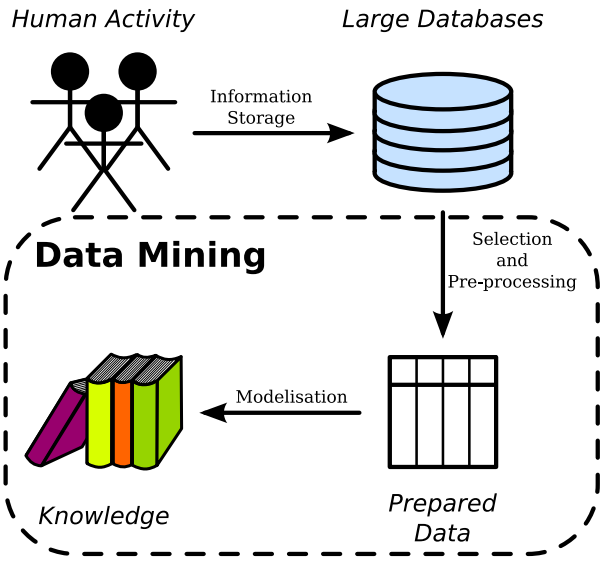


Fig. 1.1. Knowledge extraction workflow.

Sometimes very few labelled examples are available. In such a situation, the information provided by the unlabelled instances can be exploited to improve the model learning process. The term semi-supervised classification is applied to these problems where both labelled and unlabelled examples are used in the learning.

A particular situation where machine learning algorithms are useful is the information retrieval framework. Let us suppose that we have a large database of objects (books, genes, people, etc.) and we have a list of these objects that are of particular interest to us (books of a particular topic, genes with a given function, people interested in our product, etc.). Our goal is, given the list of interesting objects, to retrieve from the database those objects that are similar to the ones in our list. This can be seen as a binary classification problem where the objects should be classified as interesting (or positive) and not-interesting (or negative). From the supervised point of view we would need labelled examples from both classes (positive and negative). Our list of interesting objects constitutes a set of positive examples, but we lack negative examples.

We could hand-label some of the objects in the database as non-interesting, but this task can be tedious, expensive or even impossible in some cases.

Moreover, we need to get a set of negative examples that represents all the possible negative instances, which can be an even harder task.

So the question is, is it possible to learn binary classification problems without negative examples? As we will see later in this dissertation the answer to this question is yes, provided that we have unlabelled instances. Learning from positive and unlabelled instances is the topic of this dissertation. It has been named in the literature as partially supervised classification (Liu et al., 2002), positive example based learning (Yu et al., 2002) or positive unlabelled learning (Denis et al., 2002). We will use the latter term as it is the one that best reflects the problem. Throughout the following chapters we will see how this problem has been tackled so far and we will present our contributions in different aspects such as model induction, feature subset selection or classifier evaluation.

1.1 Contributions of the dissertation

The contributions of this dissertation are both in the methodology and the applications of positive unlabelled learning algorithms. The methodological contributions comprise classifier induction algorithms, feature subset selection and classifier evaluation.

All the algorithms proposed in this dissertation have been empirically tested. Due to the nature of the problem, the model validation in real-life positive unlabelled learning problems is difficult, as the classical performance measures rely on the confusion matrix and half of this matrix cannot be estimated without negative examples (for more details see Section 2.2). This is one of the reasons why we have tested the algorithms in datasets where the absence of negative examples has been simulated. We have created these simulated positive unlabelled learning problems starting from real-life databases obtained from the UCI repository (Blake and Merz, 1998) and from datasets sampled from Bayesian network models.

In the applied part of this dissertation some of these methodological developments have been used to solve two computational biology problems.

1.1.1 Methodological contributions

The methodological developments presented in this dissertation can be divided into three groups: classifier induction algorithms, feature subset selection and classifier evaluation.

1.1.1.1 Classifier induction algorithms

The contributions in the induction of classifiers from positive and unlabelled instances can be divided into two groups. On one hand, we have developed an algorithm based on a sampling/model averaging process that, given a set of positive and unlabelled examples, outputs a classification for the unlabelled instances. On the other hand, we have extended and improved the concept behind the positive naive Bayes algorithm (PNB) (Denis et al., 2002) in several ways, proposing five new positive Bayesian network classifiers (PBC)[1] .

The first extension proposed is the positive TAN (PTAN), which is the adaptation of the tree augmented naive Bayes model induction algorithm (TAN) (Friedman et al., 1997) to the positive unlabelled learning framework. Both in the PNB and the PTAN algorithms the a priori probability of the positive class ($P(C = 1)$) has to be introduced as a parameter. This is a big problem as normally this probability is unknown and it cannot be estimated from positive and unlabelled examples. We have proposed a Bayesian approach where the uncertainty about this parameter is modelled by means of a Beta distribution. This allows us to consider all the possible values the parameter can take rather than setting it at a particular value. As a final improvement we have approached the problem of setting the a priori probability of the positive class from a wrapper point of view. This last improvement is linked with the contributions in the evaluation of classifier in absence of negative examples that, as we will see later, is a non-trivial question.

1.1.1.2 Classifier evaluation

As mentioned in the previous section, the wrapper approach to the a priori probability of the positive class implies that we need a way to evaluate clas-

[1] This is the term we propose for the algorithms that learn Bayesian network models from positive and unlabelled instances. Although the name positive unlabelled Bayesian network classifiers would be more accurate we use PBC to be coherent with the name of the first PBC proposed, the PNB

sifiers in positive unlabelled learning problems. The main problem with the evaluation is that the lack of negative examples makes it impossible to obtain any of the classical performance measures. We have proposed a positive unlabelled estimator of the F measure and a new metric, the pseudo F measure, that can be used in the wrapper selection of the a priori probability of the positive class.

1.1.1.3 Feature subset selection

Besides the contributions in the classifier induction from positive and unlabelled examples we have also tackled the feature subset selection problem in the absence of negative examples. To the best of our knowledge, this is the first time the dimensionality reduction is explicitly considered in the positive unlabelled learning framework. In this dissertation, we propose an adaptation of the correlation based feature selection algorithm (CFS) (Hall and Smith, 1997) to the positive unlabelled learning framework.

1.1.2 Applications

Regarding the applications, we have run our algorithms in two computational biology problems: The identification of disease genes and the prediction of cancer genes. Since the sequencing of the human genome, a lot of information about our genes is available. This information can be (and is being) used for different purposes. One of the applications of this information is the prediction of which genes may be involved in genetic diseases and which not. The goal of having such a prediction is to serve as a guide for the researchers that are trying to identify the mechanisms behind a particular disease. Many times these researchers face the situation of having reduced the candidate list of genes to a few hundred. Wet-lab checking all of these genes is a very expensive, time-consuming task. A tool that ranks these genes according to the probability of being related with a genetic disease would increase the efficiency of this search, saving lots of time and resources.

Getting a set of positive examples for the identification of disease genes is more or less easy as there are databases with genes that are known to be disease related, but producing a dataset of negative examples is not possible as we cannot ensure that a given gene is not related with any genetic disease. Thus, the disease gene identification is a typical positive unlabelled learning

problem. The variables used in the identification of disease genes are properties such as the gene length or the conservation score obtained from the sequence.

The other problem where our algorithms have been applied is cancer gene prediction. The problem is very similar to the previous one in the sense that some genes are known to be cancer related but we cannot obtain a set of negative examples. The variables are also different because using only sequence properties is not enough to identify the cancer genes and, thus, other kinds of features, such as regulatory properties of protein-protein interactions, have to be introduced.

Previous works have suggested that the features used in the prediction differ in dominant and recessive genes so we have tackled the identification of dominant and recessive disease/cancer genes separately.

1.1.3 Overview of the dissertation

The dissertation is organised in four parts:

- **Part I** - Introduction to the classification problems and the Bayesian network classifiers.
- **Part II** - Methodological contributions, including the proposed positive unlabelled learning classifiers, evaluation measures and feature subset selection algorithms.
- **Part III** - Applications of the proposed algorithms in computational biology problems (identification of disease and cancer genes).
- **Part IV** - Conclusions and future work.

2

Classification problems

Learning classification functions from datasets of examples is one of the main areas in the machine learning discipline with applications in such different fields as medical diagnosis, image processing, customer management or text mining. In this chapter we will try to summarise the different situations one can face when dealing with a classification problem, placing the topic of this dissertation (the positive unlabelled learning) in the general framework of classification problems. In addition to the induction of classifiers, the feature subset selection problem and the classifier evaluation will be introduced.

2.1 A taxonomy of classification problems

Before starting with the taxonomy of classification problems we need to introduce some of notation that will be used throughout the dissertation and define what a classification problem is. The starting point in any classification problem is a dataset $\mathcal{D}$ of examples (also referred to as cases or instances). The instances are characterised by a vector of n random variables $\boldsymbol{X} = (X_1, \ldots, X_n)$ (also known as attributes or features) that can be either discrete or continuous[1] . A discrete random variable X_i can take r_i possible values. Random variables are always represented as capital letters and their values as lower case letters; vectors are represented in bold font.

[1] In this disseration we are focusing on the probabilistic approach of the classification problems where the attributes of the instances are represented as random variables

In the most general framework, the instances can be classified according to different criteria. These criteria are represented by a class vector of m class variables $\boldsymbol{C} = (C_1, \ldots, C_m)$. All the class variables C_i are discrete (the prediction of continuous variables is known as regression rather than classification) and can take r_{c_i} values. The value of both predicting and class variables can be missing. We will represent the missing values with a question mark. $\mathcal{D}$ denotes a dataset of N examples $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, \boldsymbol{c}^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, \boldsymbol{c}^{(N)})\}$. In order to simplify the notation, and when the meaning is clear from the context, a random variable taking a particular values will be represented by the value alone. For instance, instead of writing $P(X_i = x_i | C = c)$ we will write $P(x_i|c)$. When we need to specify that a variable X_i takes its j-th value, the notation $X_i = x_{ij}$, abbreviated as x_{ij}, will be used.

A classification problem can be defined as the induction, from a dataset $\mathcal{D}$, of a classification function ψ that, given the attribute vector of an instance, returns a class vector $\boldsymbol{c}$.

$$\begin{aligned} \psi : \quad \Omega_{\boldsymbol{X}} &\longrightarrow \Omega_{\boldsymbol{C}} \\ \boldsymbol{x} &\longrightarrow \boldsymbol{c} \end{aligned}$$

The classification problems are typically taxonomised according to the nature of the examples in $\mathcal{D}$. The first division of the classification problems can be carried out according to the number of class variables to predict. Most of the work in classification has been done considering only one class variable to predict (i.e. $m = 1$), but it is not infrequent to find classification problems where there are more than a class variable. If these variables are independent, then we can consider each prediction as an independent problem, but in some situations there may be some relationship between the class variables that can be exploited to improve the learning process. If we tackle the prediction of each variable independently, we may be losing very valuable information. A good example of such a situation would be the prediction of the prognosis of a disease. We may define the prognosis of a disease as a list of possible consequences that may appear in different degrees. Each of these consequences can be represented by a class variable that we want to predict. We could predict each consequence independently but it is very likely that the different aspects of the disease are related and, thus, we could use this relationship to predict the prognosis as a whole more accurately. Very few works have focused

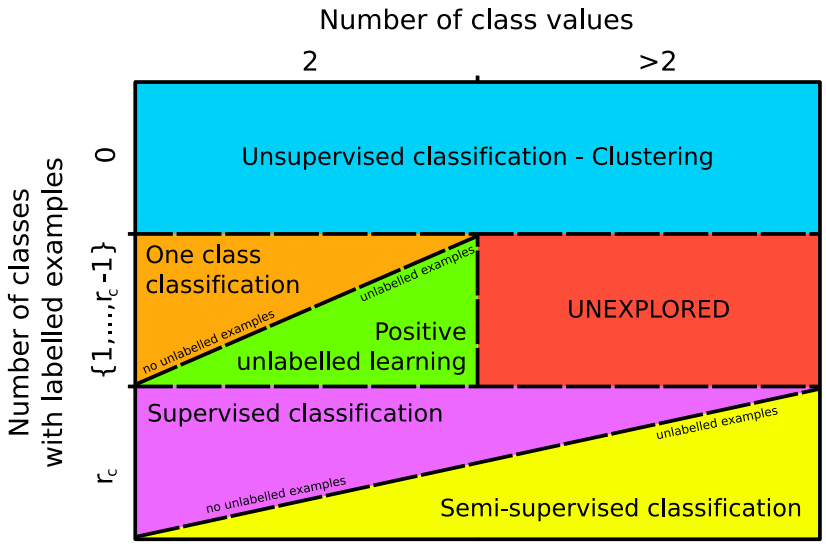


Fig. 2.1. Scheme of the classification problems when there is only one class variable to predict (m=1). The horizontal divisions of the picture represent the number of class values for which we have labelled examples in $\mathcal{D}$ (0 when all examples are unlabelled, r_c when we have examples from all the classes and $\{1, \cdots, r_c - 1\}$ when only examples from some classes are available). The vertical division indicates the number of possible class values (2 for binary classification problems and > 2 for the rest). The diagonal divisions separate the problems where unlabelled examples are available from those where only labelled examples are used.

on the simultaneous prediction of more than one class variable (van der Gaag and de Waal, 2006; de Waal and van Der Gaag, 2007; Rodríguez and Lozano, 2008).

If we focus on problems where $m = 1$, we can further taxonomise them based on the sort of examples we have. From now on we will suppose that we have only one class variable C that can take r_c values. There are two kinds of examples: labelled and unlabelled instances. A labelled instance is an example where the value of the class variable is known; when the class value is missing we have an unlabelled instance. Depending on the sort of examples we have, the following classification problems can be defined:

- *Supervised classification* - Classification problems where $\mathcal{D}$ contains examples from all the classes and all the instances are labelled.

- *Semi-supervised classification* - Classification problems where $\mathcal{D}$ contains labelled examples from all the classes and unlabelled examples.
- *Unsupervised classification* - Classification problems where all the instances are unlabelled.
- *Positive unlabelled learning* - Classification problems where the class takes only two values (positive and negative) and $\mathcal{D}$ contains only positive and unlabelled examples.
- *One class classification* - Classification problems where the class takes only two values (positive and negative) and $\mathcal{D}$ contains only positive examples.

Figure 2.1 shows a summary of the unidimensional (m=1) classification problems.

2.1.1 Supervised classification

The supervised classification concept (Duda et al., 2001; Bishop, 2006) refers to the induction of classifiers starting from a set of labelled examples. It can be seen as an attempt to emulate human learning from experience. Any expert in a particular topic has a past experience from where his/her brain has created a model of the problem. For instance, a computer technician whose job is to fix hardware problems in the PCs of an office, after checking and fixing hundreds of computers, he/she creates a model that helps him/her to identify the problem in new broken PCs just by looking at the 'symptoms'. Supervised algorithms try to imitate the human brain and, starting from the 'experience' collected in datasets, create a model of the topic under study.

Several paradigms have been proposed to the induction of classifiers from labelled databases. One of the most obvious approach is the comparison. If we have a database with labelled examples and we need to predict the label of a new instance, we can search the database for the most similar example and assign its label to the new instance. This is the concept behind the *nearest neighbour rule* (Fix and Hodges, 1951). The *k-nearest neighbour* algorithm is the extension of this idea where, instead of looking at the closest example, we consider the label of the k closest examples.

Another very intuitive type of model is *classification tree* (Breiman et al., 1984; Quinlan, 1993), where the new instances are passed through a tree of questions (involving the predicting variables) until they reach a leaf that contains the label we have to assign to the new instance.

The probabilistic classifiers are those that base the prediction on the estimation of the posterior probability of the class (i.e., the probability distribution of the class variable given that we know the value of the predicting variables). One of these algorithms is the *logistic regression* (Kleinbaum et al., 1982) that models the a posteriori probability of the class by means of the logistic function of a linear combination of the predicting variables:

$$P(C = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-z}}$$

where

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

The *Bayesian network classifiers* (Duda and Hart, 1973) model the joint probability $P(\boldsymbol{X}, C)$ by means of a Bayesian network and then, using the Bayes rule, estimate for every new instance $\boldsymbol{x}$ the a posteriori probabilities $P(c|\boldsymbol{x})$. In order to reduce the number of parameters to estimate, some assumptions about the conditional dependencies between variables are made. In the naive Bayes model (Minsky, 1961), the most simple Bayesian network classifier, all the predicting variables are supposed to be conditionally independent given the class. More complex Bayesian network classifiers, such as the tree augmented naive Bayes models (Friedman et al., 1997), allow dependencies between variables with some restrictions. These dependencies are represented by a directed acyclic graph (DAG) that is used to obtain the factorisation of the conditional probability $P(\boldsymbol{x}|c)$. As most of the contributions of this dissertation are related with the Bayesian network classifiers, they will be explained in more detail in Chapter 3.

The *neural network classifiers* are based on artificial neural networks (McCulloch and Pitts, 1943) which are an attempt to model the human brain. The basic units in neural networks are the nodes or neurons that are organised in layers. The most simple neural network, known as perceptron (Rosenblatt, 1962) consists of a single neuron that separates the two classes by a linear discrimination function (given the appropriate activation threshold). More complex networks are obtained by connecting perceptrons in what is known as multilayer perceptron.

The *support vector machines* (Vapnik, 1995) are classifiers based on the transformation of the feature space into a (normally much) higher dimensional space where the different classes can be separated by simple hyperplanes.

2.1.2 Semi-supervised classification

In some fields, such as text mining or computer vision, the amount of labelled examples is reduced while huge amounts of unlabelled examples are readily available. In such situations the performance of the supervised algorithms can be very poor due to the low number of labelled instances. The semi-supervised algorithms are based on the idea that the learning process can be improved using the information underlying the unlabelled examples.

Although one can think that semi-supervised classification is somehow between the supervised and the unsupervised classification, conceptually it is much closer to the supervised learning. The unlabelled instances are just an aid to improve the supervised learning process. In unsupervised classification the goal is to group the instances into clusters rather than to learn a classifier from the data.

During the last decades several algorithms have been developed in order to include the unlabelled instances in the learning process. In this section we will only mention a couple of approaches to the problem. More information about the semi-supervised classification problem can be found in Zhu (2005) and Chapelle et al. (2006).

The first approach to the problem is the *co-training* concept (Blum and Mitchell, 1998). In certain problems, such as text classification, different views (i.e. different sets of attributes) of the instances can be used to train a classifier. For instance, if our goal is to classify web pages, we can do it according to the words contained in the web page itself or according to the words that appear in the web pages that contain links to the web page we are considering. These two sets of labels could be used independently to classify the web pages. The co-training concept is based on the idea of building two (or more) different classifiers, each based on a view of the dataset, and then use each classifier to enrich the set of labelled examples used to train the other classifier by classifying the unlabelled instances.

The other approach we will mention in this section is the use of the expectation maximisation (EM) algorithm (Dempster et al., 1977). In the unsupervised version of the EM algorithm the probabilistic labels of all the instances

change dynamically during the process according to the posterior probabilities $P(c|\boldsymbol{x})$. In the semi-supervised framework the labelled instances are added to the process fixing their posterior probabilities so as $P(c|\boldsymbol{x}) = 1$ being c the label associated to $\boldsymbol{x}$ and $P(c'|\boldsymbol{x}) = 0$ for the rest of the labels. At each iteration the posterior probabilities of the unlabelled instances are updated using the model obtained in the previous step but not the posterior probabilities of the labelled instances.

2.1.3 Unsupervised classification

The unsupervised classification is also known as clustering, as its goal is to identify the underlying cluster structure of the data. In the unsupervised classification we have a set of unlabelled instances, but there is not an explicit class variable as in the supervised or semi-supervised classification problem. The class variable (whose value is unknown for all of our examples) can be seen as a hidden variable representing the cluster membership. This poses one of the main problems in clustering: We do not know how many values the class variable should take (i.e., how many clusters should we obtain). There are some approaches to automatically set the number of clusters but this topic is out of the scope of this introduction.

The aim of the unsupervised classification algorithms is to group the instances such that inside each cluster all the instances are very similar (normally in terms of a distance or a correlation measure) and the instances in different clusters are very different. There are several approaches to the clustering problem that can be mainly divided into three categories: Hard clustering, fuzzy clustering and probabilistic clustering. The hard clustering algorithms assume that a given instance can belong to only one cluster. Examples of this type of method are the k-means (Forgy, 1965) and the hierarchical clustering (Jardine and Sibson, 1971) algorithms. The fuzzy clustering is a generalisation where a particular instance can belong to any of the possible clusters with a degree of membership that ranges from 0 to 1. The best known example of fuzzy clustering algorithm is the fuzzy c-means (Bezdek, 1981). In the probabilistic clustering we assume that each cluster has been generated by a probability distribution and thus the algorithms try to model these probability distributions in order to obtain the clustering. It is similar to the fuzzy clustering in the sense that each instance has a degree of membership. The

basic difference with the fuzzy clustering is that the membership is the probability of belonging to that cluster and, thus, the sum of the probabilities of belonging to each cluster has to be 1. In the fuzzy clustering the membership is a weighting factor that is not subject to such restrictions.

2.1.4 Positive unlabelled learning

The problem of learning binary classification functions starting from positive and unlabelled examples has been named in the literature as partially supervised classification (Liu et al., 2003), positive example based learning (Yu et al., 2002) or positive unlabelled learning (Denis et al., 2002). Although this problem may look very similar to the semi-supervised classification (as we have labelled and unlabelled examples), the absence of examples from one of the classes poses some extra challenges. Nevertheless, conceptually the problem is the same as in supervised and semi-supervised classification as there is an explicit class variable that we want to predict. The difference lies in the lack of examples of one of the classes. The difference with the supervised classification is the sort of data from where we want to learn the classifier.

This kind of problem is quite frequent in the information retrieval framework. We have a big set of (unlabelled) instances and a list of examples of instances that we are interested in, and what we want is to recover, from the big unlabelled set, those instances that are similar to the ones in our list. If we want to model this recovery as a semi-supervised problem, we need some negative examples, i.e. examples of instances that we are not interested in, so we should hand-label some uninteresting instances from the unlabelled set. On one hand, this task can be tedious, expensive, time-consuming or even impossible in some situations. On the other hand, even if we are able to detect some negative cases, if we want to learn a good classification function we should produce a set of negative examples that represent all the possible sorts of uninteresting instances, and this can be an even harder task. Figure 2.2 illustrates the problem of getting a representative set of negative examples. The other alternative is to learn classifiers without negative examples.

Most of the classifier induction algorithms for positive unlabelled learning have been developed in the text classification domain, which is a good example of information retrieval framework. For instance, we could be interested in retrieving, from a big database of papers, those related to a set of papers

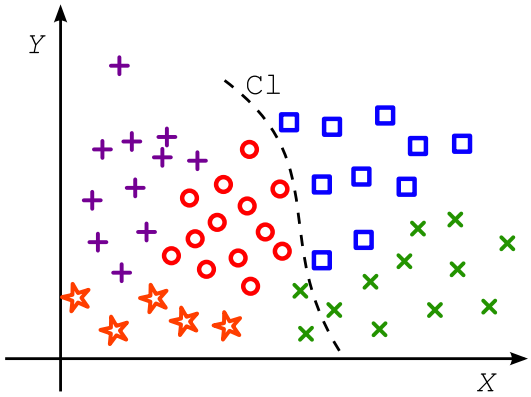


Fig. 2.2. Illustration of the problem of not having a representative set of negative examples. Suppose that we have some examples of red circles and we want to retrieve all the red circles. From a supervised classification point of view we need examples that are not red circles. In order to obtain a good classifier to distinguish red circles from the rest we need examples from all the classes that are not red circles. If we only have examples from a particular class (let us say blue squares) that are not red circles, then we would learn a classifier (represented by the boundary labelled as 'Cl') that distinguishes red circles from blue squares, but fails to distinguish red circles from purple crosses or orange stars.

about a particular topic. The same can be done with web pages or any other type of document.

Not only text classification problems can be modelled in this way. The identification of potential clients from databases based on the information about the current clients is another example of information retrieval problem that could be tackled from the positive unlabelled learning point of view. Computer vision problems are also a good example of the sort of classification problems that could be modelled in the same way. For instance, we may want to identify, from a set of unlabelled satellite images, those corresponding to a particular kind of terrain. In the applied part of this dissertation we will introduce two computation biology problems where we have applied learning algorithms that use only positive and unlabelled examples.

As the algorithms available for supervised and semi-supervised classification cannot be directly used to learn models from positive and unlabelled examples, specific algorithms have been developed to cope with this problem.

Chapter 4 is devoted to the state of the art and there the main algorithms proposed in the field are reviewed.

In addition to the classifier induction problem, the absence of negative cases poses challenging problems in other fields such as the evaluation of classifiers or the feature subset selection (FSS). Regarding the classifier evaluation, most of the performance measures are based on the confusion matrix that summarises the performance of the classifier in four values: true positives, true negatives, false positives and false negatives. Without actual negative examples only true positives and false negatives can be estimated so we cannot obtain the typical performance measures such as the accuracy or the classification error. In the chapter devoted to the wrapper positive Bayesian network classifiers (Chapter 8) we will go into detail regarding the problems of evaluating classifiers in absence of negative examples.

As many FSS algorithms depend on estimations that involve the class variable, the absence of negative examples implies that they cannot be directly applied in the positive unlabelled learning context. One of the chapters in this dissertation (Chapter 9) is dedicated to the FSS problem in positive unlabelled learning problems.

2.1.5 One-class classification

The one-class classification concept (Manevitz and Yousef (2001); Tax (2001); Tax and Duin (2002)) refers to the induction of classifiers when only positive examples are available. The goal is to obtain, given a set of examples (also known as target examples), a classification function that distinguishes between target instances and the rest of possible instances (also known as outliers). Compared to the positive unlabelled learning, in one-class classification problems we do not have unlabelled instances.

The lack of unlabelled instances is not the only difference between one-class classification and positive unlabelled learning. There are also differences in the concept itself. In one-class classification we have a set of target objects, and we want to distinguish from all the rest of possible objects, regardless of how probable these objects are. In positive unlabelled learning the unlabelled instances give us information about the probability distribution of the objects in our problem. For example, suppose that we want to learn a model that classifies any web as conference or non-conference page. From the positive

unlabelled learning point of view, we would focus only on web pages. We would take some conference web sites as positive examples and a set of pages from the internet as unlabelled examples and then we would train a classifier. In one-class classification we would only have a set of conference web pages (positive or target examples) so the classification function that we obtain tries to distinguish between a conference web page and any other element (web page or not).

2.2 Evaluation of classifiers

The goal of a classifier induction algorithm is to build a classification function, ψ, that predicts the class of new, unseen instances with a certain degree of accuracy. In other words, we want to induce good classification functions. Thus, it is important to evaluate the goodness of the model we have learnt. First of all we need to define how the goodness of a classifier can be evaluated. In Section 2.2.1 the most commonly used performance measures are introduced. Section 2.2.2 is dedicated to the different estimation schemes proposed to obtain the peformance measures from the dataset.

2.2.1 Classifier performance measures

There is a good amount of metrics that are used to assess the quality of classification functions. Probably the most commonly used measure is the *classification error*, which is the probability that a classification function incorrectly classifies a new instance. The classification error of a function ψ is:

$$\epsilon_\psi = \sum_{\boldsymbol{x}} P(\psi(\boldsymbol{x}) \neq c) P(\boldsymbol{x})$$

being c the actual class of $\boldsymbol{x}$[2] . The complementary measure of the classification error is the *accuracy*, that can be defined as the probability of correctly classifying a new instance:

[2] In the general framework any instance can belong to any class with a given probability. Throughout this dissertation we are considering a particular case where a given instance only belongs to a class (the actual class) with a probability of 1 and to the rest with a probability of 0

		Predicted P	Predicted N
Actual	P	TP	FN
	N	FP	TN

Table 2.1. Confusion matrix in binary classification problems.

$$Acc_\psi = \sum_{\boldsymbol{x}} P(\psi(\boldsymbol{x}) = c) P(\boldsymbol{x})$$

The classification error and the accuracy are the most widely used performance measures when the cost of making an error is independent of the classes involved, but in many real situations the cost of an error depends on the classes involved. For instance, suppose that we want to obtain a system to help the doctors in a medically assisted reproduction clinic. We need a classifier that, given some attributes extracted from embryo images and some information about the patients, predicts whether the embryo is going to implant (and thus lead to pregnancy) or not. The (emotional) cost of a false positive (predicting that an embryo is going to implant correctly and not getting pregnancy) is not the same as the cost of a false negative (a viable embryo discarded). Under this sort of situation the classification error is not the best option, and the total expected cost is preferred.

In the particular case of binary classification problems, where we have only two classes (positive and negative, normally represented as 1 and 0 respectively), many other performance measures can be defined based on the confusion matrix (Table 2.1). The *sensitivity*, S_n, is the ratio of positive cases that are correctly classified as positive. It is also known as *recall*, r, or *true positive rate*, TPR, and is calculated as:

$$S_n = r = TPR = \frac{TP}{TP + FN} \tag{2.1}$$

The *specificity*, S_p, is the equivalent of the sensitivity but with the negative cases, that is, the ratio of actual negative cases that are correctly classified:

$$S_p = \frac{TN}{TN + FP}$$

The *false positive rate*, FPR, is the ratio of negative cases that have been classified as positive and can be estimated as:

$$FPR = \frac{FP}{TN + FP} = 1 - S_p$$

Another interesting measure is the *precision*, p_r, that can be defined as the probability that an instance classified as positive is actually positive. It can be computed from the confusion matrix as:

$$p_r = \frac{TP}{TP + FP} \tag{2.2}$$

Given a new instance, $\boldsymbol{x}$, a probabilistic classifier estimates the probability of each of the classes given the attribute values of the instance $P(c|\boldsymbol{x})$. The 'winner takes all' consists of assigning the new instance to the class with the highest a posteriori probability[3] :

$$\psi(\boldsymbol{x}) = argmax_c(P(c|\boldsymbol{x}))$$

This means that, in binary classification problems, we will say that the instance $\boldsymbol{x}$ is positive when $P(1|\boldsymbol{x}) > 0.5$. Nevertheless, 0.5 is not the only threshold that we can use. Actually, we can set any number between 0 and 1 as the threshold, resulting in different classification functions (even if we have a single model). The receiver operating characteristic (ROC) curve is a graphical representation that allows us to evaluate the model considering all possible thresholds for the conditional probability. It is the graphical representation of sensitivity vs. 1-specificity. Figure 2.3 shows some examples of ROC curves. The ROC curve allows us to graphically compare different classifiers just looking at the shape of their curves. The area under the ROC curve (AUC) summarises the curve and gives us an idea of the goodness of the model. The closer this value is to 1, the better the model we have.

[3] Though there are other kind of classifiers, in this dissertation we will focus on the probabilistic classifier, i.e., those that base their predictions on the estimation of the posterior probability of the class variable

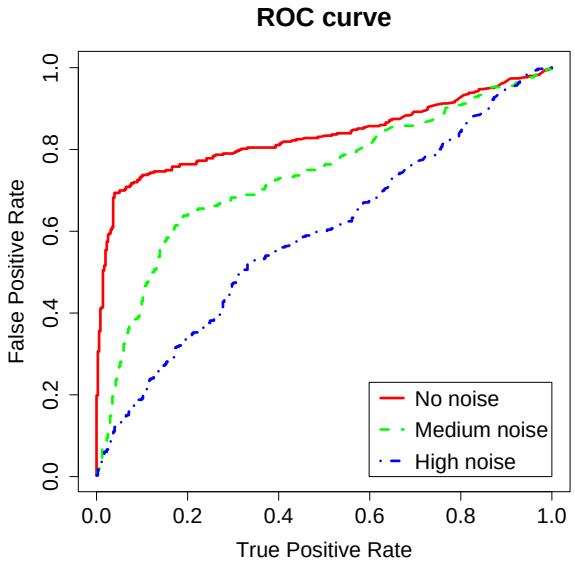


Fig. 2.3. Examples of ROC curves. The curves represent the performance of a naive Bayes classifier in the classification of breast cancer as benign or malign in a dataset obtained from the UCI repository (Asuncion and Newman, 2007). The curve labelled as 'No noise' corresponds to the results obtained in the original dataset, and the other two curves are the results obtained adding different amounts of noise to the dataset.

In the information retrieval context we are especially interested in the positive examples. The goal of a classification function is to recover the positive cases from the set of unlabelled instances. The two aspects of this recovery, the quantity and the quality, are measured by the recall and the precision respectively. The recall gives us an idea of how many of the actual positive cases have been identified as positive and the precision tells us how many of the instances identified as positive are actually positive.

Any of these measures alone is not enough to evaluate a classifier. If we have a classification function that always outputs 'positive' as the predicted class, it would have a recall of 1, even though the classifier is completely useless. On the other hand, a classification function that classifies a single (real) positive case as positive would get a precision of 1. The *F measure* is a metric that combines these two measures and, thus, it gives us a global idea

of the performance of the classification function in the information retrieval task. This measure is the harmonic mean of the precision and the recall, and its most general definition, $F_\alpha(r, p_r)$, is:

$$\frac{1}{F_\alpha(r,p_r)} = \frac{1}{\alpha+1}\left(\frac{\alpha}{r} + \frac{1}{p_r}\right)$$
$$\alpha \in (0, +\infty)$$

The weighting factor α allows us to put more emphasis on the quantity (the recall) or the quality (the precision) of the recovery. When the weighting factor α is 1, we have the most commonly used definition of F measure:

$$\frac{1}{F(r,p_r)} = \frac{1}{2}\left(\frac{1}{r} + \frac{1}{p_r}\right) \tag{2.3}$$
$$F(r,p_r) = \frac{2rp_r}{r+p_r}$$

All the measures and concepts presented up to now in this section are concerned with the evaluation of classifiers when examples from all the classes are available. When we are facing a clustering problem all the instances are unlabelled and, thus, none of these metrics can be computed. The validation of clustering algorithms is out of the scope of this introduction. A good review on validation methods for clustering techniques can be found in Halkidi et al. (2001).

2.2.2 Estimation of the performance measures

In the previous section we have described some of the most important performance measures used in the assessment of the classification function. In this section we will show how these measures can be estimated from the data. Throughout this section we will refer to the estimation of the classification error, but any other performance measure can be estimated using the schemes described here.

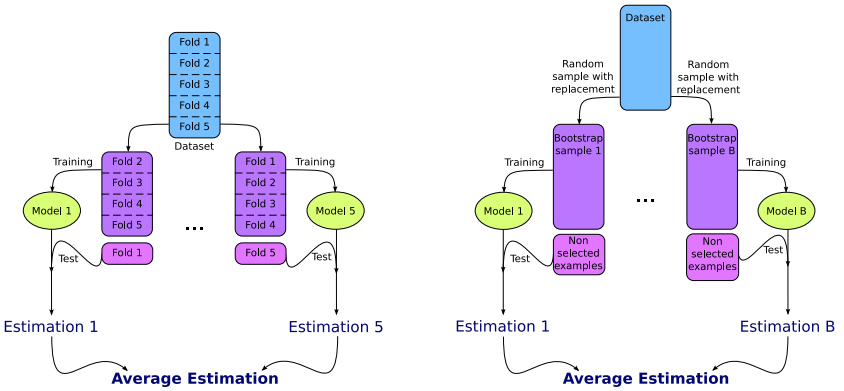
The most simple approach to the estimation consists of learning a classification function using all the examples available and then testing the classifier in the same data. This way of estimating the error is known as resubstitution error. The problem with the resubstitution error is that, in general, a

classifier tends to make less errors when classifying the examples used in its training than classifying new, unseen examples. This is because the classifier is overfitted to the data used in the training and, thus, estimating the error over the same data used in the training provides optimistically good results. In other words, the resubstitution error estimation has a negative bias (it is lower than the actual classification error), and is not a good way to evaluate the generalisation error of the classifier (the error when classifying unseen instances).

The key to obtaining a fair estimation of the generalistion error is to test the model in examples that have not been used in the training. The easiest way to do this is by dividing the dataset into two subsets, one used for training the classifier and the other for testing it. This estimation scheme is known as hold-out error estimation. Using this very simple scheme we can have a fair estimation of the classification error of our model, but the problem is that the model has been trained only on a part of the available examples. Given that in many situations the examples are expensive, 'wasting' some of these examples to test the model may not be acceptable.

To avoid reducing the training set, some other error estimation schemes have been proposed. The common feature in all of these schemes is that several models are trained in part of the dataset and are tested on the rest of the examples and then the estimations of the error are averaged. The key of these procedures is that this averaged error is an estimation of the error of a classifier trained on all the data. One of these estimations schemes is the repeated hold-out, which consists of repeating the hold-out scheme R times and then estimating the error as the averaged error obtained in the R repetitions.

One of the most commonly used schemes is the k fold cross validation (k-cv) (Efron, 1983) (see Figure 2.4-a). It consists of (randomly) dividing the dataset into k folds. Then, a model is trained on all the folds except one and the error is estimated on the fold that has been left out. This is repeated k times, leaving at each step one of the folds, and then the error is estimated as the average of the k errors. The bias and the variance of the estimation depends on the k used. The most commonly used values for k are 5 and 10. A special case of k fold cross validation is the leave-one-out scheme, where at each step a classifier is trained on all the examples except one, which is used to test the model (i.e., it is a k-cv with $k = N$, where N is the number of instances in the dataset). The estimation obtained using a k fold cross



(a) 5 fold cross validation (b) Bootstrap

Fig. 2.4. Schemes of k-cv ($k = 5$) and bootstrap estimation.

validation depends on the partitioning of the dataset (except in the leave-one-out). To overcome this problem a repeated k-cv can be used, consisting of repeating the k-cv r times. For example, a 5 times 5-cv consists of repeating 5 times a 5-cv, having a different partition of the dataset in each repetition. The final estimation is the average of the rk individual estimations.

Another commonly used procedure is the bootstrap error estimation (Efron and Tibshirani, 1993) (see Figure 2.4-b), where B bootstrap samples of the original dataset are obtained (i.e., random samples with replacement of size N). These bootstrap samples are used to train a model that is evaluated on the examples that have not been included in the sample (as the bootstrap sample is carried out with replacement, some examples can appear more than once while others are not included). The final estimation is the average of the estimations obtained with the B samples. The bootstrap estimation of the error tends to have a positive bias (the estimated error is higher than the actual error). A commonly used correction of this bias consists of combining the bootstrap error estimation (ϵ_B) with the resubstitution error estimation (ϵ_r), which has a negative bias, in the so called bootstrap .632 error estimation:

$$\epsilon_{.632} = 0.632\epsilon_B + (1 - 0.632)\epsilon_r$$

2.3 Feature subset selection

In this chapter we have introduced the concept of supervised classification, where classification functions predict the class of a new instance based on the values of its attributes. In principle, one can be tempted to think that the more attributes we have to describe the instance, the better we can predict its class, but this is only partially true. Indeed, not all the features are equally useful for the classification purpose.

There may be features that have nothing to do with the class. For instance, if we are trying to predict the prognosis of a patient, the first name of his/her grandmother will not be of great help. Thus, we can divide the predicting variables into relevant, when they have some predictive power, and irrelevant, when there is no relationship between them and the class.

We may also find two or more variables that give us (more or less) the same information. For instance, the date of birth and the age are two variables directly correlated. When two variables are highly correlated we say that they are redundant.

Non informative and redundant variables can be harmful for some model induction algorithms, they also lead to models that are too complex and increase the computational time required to obtain the classifier. Thus, producing a small set of relevant, non-redundant variables is a very important step in many machine learning applications (Saeys et al., 2007).

Two ways to reduce the dimensionality of the classification problems have been proposed in the literature: feature extraction (Liu and Motoda, 1998) and feature subset selection (FSS)(Guyon and Elisseeff, ????; Liu and Motoda, 2008). The former consists of combining the features to obtain new, better features. The main problem with this approach is that the meaning of the original variables is lost in the newly constructed features. The latter consists of selecting the best subset of features for the classification purpose. The feature extraction methods are out of the scope of this dissertation and, thus, we will focus on the FSS techniques.

There are three main approaches to the FSS problem (Guyon and Elisseeff, ????; Saeys et al., 2007), namely embedded, wrapper and filter methods. Some classifier induction algorithms, such as the C4.5 (Quinlan, 1993), discard some variables in the learning process. This sort of FSS is known in the literature as *embedded* FSS.

The *wrapper* approaches (Kohavi and John, 1997) try to identify the subset of variables that, given a classification paradigm and a dataset, provides the best classification function. The process consists of a search in the feature subset space guided by a performance measure (typically the accuracy, though other measures can be used). Each subset is evaluated by testing the performance of the chosen paradigm in the dataset, using only the variables in the subset at evaluation. The estimation of the performance of the classifiers requires a validation scheme, such as cross validation (Stone, 1974) or bootstrap estimation (Efron and Tibshirani, 1993)[4] . As a result, the evaluation of each subset involves the training and testing of several classification functions, increasing the computational time required for the FSS process. Besides, the search for the best subset is an NP-hard problem (Amaldi and Kann, 1998) and, thus, an exhaustive search quickly becomes computationally infeasible and search heuristics have to be used to obtain a good feature subset in a reasonable computational time (Inza et al., 2000). This is the main drawback of these methods. Another characteristic of the wrapper methods (that can be good or bad, depending on the point of view) is that the subset produced by the algorithm depends on the classification paradigm considered for the search. This means that the selection obtained with a classification paradigm cannot be applied to other classification paradigms, as the solution is tuned for that particular classifier.

The *filter* approaches search for the best subset of variables, independently of the classification paradigm, considering the relationship between the predicting variables and the class and (sometimes) the relationship among the predicting variables. One of the most simple approaches consists of ranking the variables according to their usefulness and then selecting only those variables on the top of the ranking. The usefulness of a variable is measured univariately by means of different metrics. For instance, information theory related metrics (Cover and Thomas, 2006) evaluate the usefulness of the feature by measuring the reduction on the uncertainty of the class variable when the value of the feature at evaluation is known (Ben-Bassat, 1982). Once the features are ranked, a threshold must be set to obtain the final subset.

[4] It is important to point out that this kind of FSS technique, due to their wrapper nature, can lead to overfitting problems that have to be considered in the estimation of the classifier performance

The ranking methods are only concerned with the relevancy of the features considered and, thus, they do not filter out redundant variables. The problem of searching for relevant, non-redundant features can be solved by a multivariate filter method known as the *correlation based filter selection* (CFS)(Hall and Smith, 1997). This method searches for the best feature subset guided by a metric that measures both the correlation between each variable and the class and the correlation among the selected variables. The aim is to obtain a subset of relevant variables (i.e. features strongly correlated with the class) without redundancies (i.e. features with a small correlation between them). Filter methods are much faster than wrapper approaches and they are independent of the classification paradigm. Therefore, once we have a subset of features, this subset can be used in the training of any sort of model.

3

Bayesian Network Classifiers

Classification functions based on Bayesian networks constitute a classification paradigm frequently used in supervised, semi-supervised and even unsupervised classification problems. It ranges from very simple (yet effective) classifiers, such as the naive Bayes (Minsky, 1961), to models where the predicting variables can depend on as many other variables as we want.

Bayesian network classifiers are based on the sound, well-studied theory of probabilistic graphical models (Pearl, 1988; Whittaker, 1991; Lauritzen, 1996; Castillo et al., 1997), and in particular on Bayesian networks (Neapolitan, 2003; Korb and Nicholson, 2004; Jensen and Nielsen, 2007) whose applications include not only the induction of classification functions but also probabilistic inference (Lauritzen and Spiegelhalter, 1988), optimisation (Mühlenbein and Mahning, 2001; Larrañaga and Lozano, 2002) or bioinformatics (Friedman et al., 2000; Friedman, 2004; Larrañaga et al., 2006).

In the first part of this chapter we will introduce the main concepts of probabilistic graphical models, focusing on the description of Bayesian network models and how they are induced. In the second part, the most important Bayesian network classifiers will be described.

3.1 Some concepts from the graph theory

The theory of probabilistic graphical models (PGMs) is partly based on the graph theory. Therefore, before introducing the PGMs some concepts and definitions have to be presented.

A graph is a pair $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{L})$ where $\boldsymbol{X}$ is a finite set of elements $\{X_1, \ldots, X_n\}$, known as 'nodes', and $\boldsymbol{L}$ is a set of ordered pairs of nodes known as 'edges' ($L_{ij} = (X_i, X_j)$). There are two kinds of edges, directed and undirected edges. We say that an edge L_{ij} is directed when $L_{ij} \in \boldsymbol{L}$ and $L_{ji} \notin \boldsymbol{L}$ and it is represented as $X_i \rightarrow X_j$. We say that an edge L_{ij} is undirected when $L_{ij} \in \boldsymbol{L}$ and $L_{ji} \in \boldsymbol{L}$ and it is represented as $X_i - X_j$. From now on the term 'arc' will be used to refer to directed edges and the term 'edge' will be used only to refer to undirected edges.

A directed graph is defined as a graph $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{L})$ where every $L_{ij} \in \boldsymbol{L}$ is an arc. An undirected graph is defined as a graph $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{L})$ where every $L_{ij} \in \boldsymbol{L}$ is an edge. A directed acyclic graph (DAG) is a directed graph where there are not cycles (a path that starts in a node and ends in the same node).

Definition 1. *Let $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{L})$ be a directed graph and $L_{ij} \in \boldsymbol{L}$ an arc in $\boldsymbol{G}$. We say that X_i is parent of X_j and X_j is child of X_i.*

The d-separation criterion is a key concept to understand the mechanism behind DAG based PGMs. There are several definitions of d-separation, but here we include the one given by Lauritzen et al. (1990) because, in our opinion, it provides the most intuitive interpretation of the concept. This definition is based on the concepts of ancestral set of nodes, moral graph and u-separation criterion in undirected graphs

Definition 2. *Let $\boldsymbol{G}$ be a directed graph and X_i and X_j be two variables of $\boldsymbol{G}$. X_i is an ancestral node of X_j in $\boldsymbol{G}$ if there is a directed path from X_i to X_j in $\boldsymbol{G}$.*

Definition 3. *Let $\boldsymbol{G}$ be a DAG. An ancestral ordering of $\boldsymbol{G}$ is a total ordering of $\boldsymbol{X}$ where $\forall L_{ij} \in \boldsymbol{L}$, X_i appears before X_j in the order.*

Definition 4. *Let $\boldsymbol{G}$ be a graph and $\boldsymbol{Y}$ be a set of variables of $\boldsymbol{G}$. The ancestral set of nodes containing $\boldsymbol{Y}$ in $\boldsymbol{G}$ is the set of nodes formed by $\boldsymbol{Y}$ and all the ancestral nodes of the variables contained in $\boldsymbol{Y}$.*

Definition 5. *Let $\boldsymbol{G}$ be a DAG, the moral graph associated to $\boldsymbol{G}$ is the graph obtained by adding an arc between parents with a common child and then making all arcs in $\boldsymbol{G}$ undirected.*

Definition 6. *Let $\boldsymbol{G}$ be an undirected graph and let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sets of nodes. $\boldsymbol{W}$ u-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in $\boldsymbol{G}$ if every path in $\boldsymbol{G}$ between*

a node belonging to $\boldsymbol{Y}$ *and a node belonging to* $\boldsymbol{Z}$ *contains at least one node belonging to* $\boldsymbol{W}$*.*

The d-separation criterion based on the u-separation is defined as:

Definition 7. *Let* $\boldsymbol{G}$ *be a DAG and let* $\boldsymbol{Y}$*,* $\boldsymbol{Z}$ *and* $\boldsymbol{W}$ *be three disjoint sets of variables.* $\boldsymbol{W}$ *d-separates* $\boldsymbol{Y}$ *and* $\boldsymbol{Z}$ *in* $\boldsymbol{G}$ *if* $\boldsymbol{W}$ *u-separates* $\boldsymbol{Y}$ *and* $\boldsymbol{Z}$ *in the moral graph of the smallest ancestral set of nodes which contains* $\boldsymbol{Y}$*,* $\boldsymbol{Z}$ *and* $\boldsymbol{W}$*.*

Figure 3.1 shows an example of the procedure to check the d-separation between subsets of nodes.

3.2 Probabilistic graphical models

PGMs are mathematical tools that allow us to model joint probability distributions over a set of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$. They are made of two basic components, a structure $\boldsymbol{G}$ (a graph that represents the dependence relationships between the random variables) and a set of parameters $\boldsymbol{\theta}$. Different types of graphs, such as DAGs or undirected graphs, are used to represent the structure of probabilistic graphical models.

DAG based PGMs are based on the concept of conditional independence

Definition 8. *Let* $\boldsymbol{Y}$*,* $\boldsymbol{Z}$ *and* $\boldsymbol{W}$ *be three disjoint sets of random variables.* $\boldsymbol{Y}$ *is conditionally independent of* $\boldsymbol{Z}$ *given* $\boldsymbol{W}$ *if and only if*

$$P(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{w}) = P(\boldsymbol{y}|\boldsymbol{w})$$

for any possible configuration $\boldsymbol{y}$*,* $\boldsymbol{z}$ *and* $\boldsymbol{w}$*.*

The conditional independence is denoted as $CI(\boldsymbol{Y}, \boldsymbol{Z}|\boldsymbol{W})$, and the interpretation is that, given that we know $\boldsymbol{W}$, the knowledge about $\boldsymbol{Z}$ does not give us any extra information about $\boldsymbol{Y}$. In DAG based PGMs, the conditional independence is directly related to the d-separation criterion introduced in the previous section.

Definition 9. *Let* $\boldsymbol{Y}$*,* $\boldsymbol{Z}$ *and* $\boldsymbol{W}$ *be three disjoint sets of random variables.* $\boldsymbol{Y}$ *is conditionally independent of* $\boldsymbol{Z}$ *given* $\boldsymbol{W}$ *in a probabilistic graphical structure,* $\boldsymbol{G}$*, if* $\boldsymbol{W}$ *d-separates* $\boldsymbol{Y}$ *and* $\boldsymbol{Z}$ *in* $\boldsymbol{G}$*.*

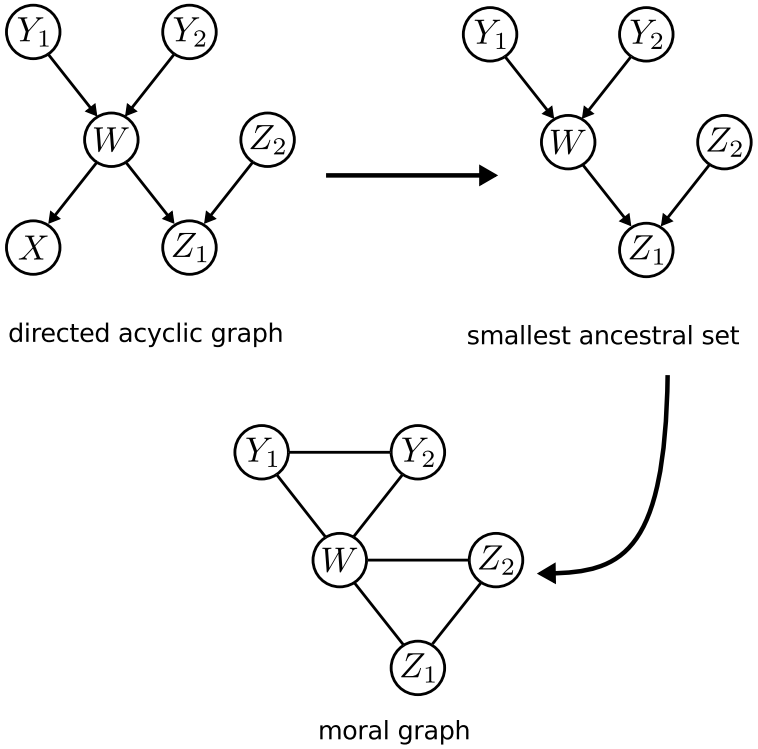


Fig. 3.1. Illustration of the concept of d-separation. We want to check whether W d-separates $\boldsymbol{Y} = \{Y_1, Y_2\}$ and $\boldsymbol{Z} = \{Z_1, Z_2\}$ or not. First we have to obtain the smallest ancestral set of nodes containing $\boldsymbol{Y}$, $\boldsymbol{Z}$ and W, which is the original graph except the node X. Then we moralise the resulting graph and check whether W u-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in the undirected graph or not. As any path between Y_1 or Y_2 and Z_1 or Z_2 contains the node W we can say that W u-separates $\boldsymbol{Z}$ and $\boldsymbol{Y}$ in the moral graph of the smallest ancestral set and, thus, W d-separates $\boldsymbol{Z}$ and $\boldsymbol{Y}$ in the original DAG. We can conclude that, in the original DAG, $\boldsymbol{Z}$ and $\boldsymbol{Y}$ are conditionally independent given W.

Given the chain rule we can factorise the joint probability distribution of $\boldsymbol{X}$ as:

$$P(\boldsymbol{x}) = P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \ldots, x_{i-1})$$

Without any loss of generality we can assume that the structure of a DAG based PGM $\boldsymbol{G}$ obeys an ancestral ordering such that each node X_i takes the i-th place in the ordering. Given this ancestral ordering we have that for every ancestral node X_j of X_i, $j < i$.

Taking into account that the set of parents of a node X_i ($\boldsymbol{Pa_i}$) d-separates X_i from any previous node in the ancestral ordering, X_i is conditionally independent of any X_j, $j < i$ given its parents. Therefore, given the structure $\boldsymbol{G}$, the ancestral ordering and the chain rule, the joint probability distribution codified by a DAG based PGM can be factorised as:

$$P(\boldsymbol{x}) = \prod_{i=1}^{n} P(x_i|\boldsymbol{pa_i})$$

Thus, the structure of a PGM indicates us how the joint probability can be factorised (reducing, as we will see later, the amount of parameters needed to model it). Besides the structure we need a set of parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ in order to define the joint probability distribution. Thus, the previous equation should be rewritten as:

$$P(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{n} P(x_i|\boldsymbol{pa_i}, \boldsymbol{\theta})$$

3.2.1 Bayesian network models

Bayesian network models are PGMs based on DAGs where all the random variables, X_i $i = 1, \cdots, n$, are discrete and can take r_i possible values. A Bayesian network consists of a DAG $\boldsymbol{G}$ and a set of parameters $\boldsymbol{\theta}$. The structure $\boldsymbol{G}$ gives us the factorisation of the joint probability distribution $P(\boldsymbol{x})$ as:

$$P(\boldsymbol{x}) = \prod_{i=1}^{n} P(x_i|\boldsymbol{pa}_i)$$

The set of parameters $\boldsymbol{\theta}$ consists of all the local probability distributions $P(x_{ik}|\boldsymbol{pa}_{ij})$. By θ_{ijk} we will denote the probability that the i-th component of $\boldsymbol{X}$ takes its k-th value and its parents $\boldsymbol{Pa}_i$ their j-th value[1] :

[1] Note that the set of parents can be a multidimensional variable and, thus, it can take $q_i = \prod_{a/X_a \in Pa(x_i)} r_a$ different values

X_i	r_i	$\boldsymbol{Pa}_i$	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

Table 3.1. Description of the variables and the parent set for each variable of the Bayesian network model from Figure 3.2. q_i denotes the number of possible values of $\boldsymbol{Pa}_i$.

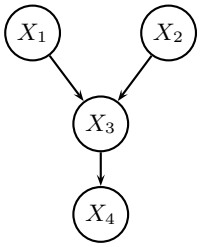
$$\theta_{ijk} = P(x_{ik}|\boldsymbol{pa}_{ij})$$

Figure 3.2 and Table 3.1 show an example of a Bayesian network model. The $\boldsymbol{X}$ vector has four components, three that can take two values and one that can take three values. Thus, the total number of possible values for $\boldsymbol{X}$ is 24. This means that, in order to define the joint probability distribution, we need 23 parameters but, given the factorisation defined by the graph, eleven parameters are enough to model the joint probability.

3.2.1.1 Learning Bayesian network models from data

So far we have seen how Bayesian networks are defined and how they can be used to model joint probability distributions. Now the question is: Where do we get these models from? The first approach to obtain Bayesian network models is to create them based on the knowledge of experts in the particular topic. For very simple models this may be feasible but, as the size of the model grows, the creation of the model gets more and more difficult as the probabilities that have to be set by the expert(s) get more complicated to understand and estimate.

A better approach consists of inducing the models from datasets of examples. There are several good reviews of methods to learn Bayesian networks, for instance Heckerman (1995), Heckerman et al. (1995) and Neapolitan (2003) provide in-depth tutorials on learning Bayesian networks, Buntine (1996) presents a literature survey and Jordan (1998) collects several introductory surveys as well as papers discussing advances on learning Bayesian networks.



Local probabilities:

$$
\begin{array}{ll}
\boldsymbol{\theta}_1 = (\theta_{1-1}, \theta_{1-2}) & P(x_{11}), p(x_{12}) \\
\boldsymbol{\theta}_2 = (\theta_{2-1}, \theta_{2-2}, \theta_{2-3}) & P(x_{21}), p(x_{22}), p(x_{23}) \\
\boldsymbol{\theta}_3 = (\theta_{311}, \theta_{321}, \theta_{331}, & P(x_{31}|(x_1,x_2)_1), P(x_{31}|(x_1,x_2)_2), P(x_{31}|(x_1,x_2)_3), \\
\quad \theta_{341}, \theta_{351}, \theta_{361}, & P(x_{31}|(x_1,x_2)_4), P(x_{31}|(x_1,x_2)_5), P(x_{31}|(x_1,x_2)_6), \\
\quad \theta_{312}, \theta_{322}, \theta_{332}, & P(x_{32}|(x_1,x_2)_1), P(x_{32}|(x_1,x_2)_2), P(x_{32}|(x_1,x_2)_3), \\
\quad \theta_{342}, \theta_{352}, \theta_{362}) & P(x_{32}|(x_1,x_2)_4), P(x_{32}|(x_1,x_2)_5), P(x_{32}|(x_1,x_2)_6), \\
\boldsymbol{\theta}_4 = (\theta_{411}, \theta_{421}, \theta_{412}, \theta_{422}) & P(x_{41}|x_31), P(x_{41}|x_{32}), P(x_{42}|x_{31}), P(x_{42}|x_{32})
\end{array}
$$

Factorisation of the joint probability distribution:

$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2)P(x_3|x_1, x_2)P(x_4|x_3)$$

Fig. 3.2. Structure, local probabilities and factorisation of the joint probability distribution for a Bayesian network with four variables, where X_1, X_3 and X_4 can take two values and X_2 three. The θ_{i-k} parameters correspond to those variables whose set of parents is empty.

The learning process can be divided into two steps: the structural learning, where the structure S of conditional (in)dependencies is inferred, and the parametric learning, where all the local probabilities needed to codify the joint probability distribution are estimated.

A. Parametric learning

In the parametric learning step all the probabilities defined in $\boldsymbol{\theta}$ have to be estimated from the data. From a Bayesian point of view, all the possible sets of parameters should be taken into account. In the particular case of Bayesian network models, under the assumption of parameter independence (Spiegelhalter and Lauritzen, 1990) and supposing Dirichlet priors for all the parameters, the Bayesian approach is equivalent to a single parameter set $\breve{\boldsymbol{\theta}} = \{\breve{\theta}_{ijk}\}$:

$$\breve{\theta}_{ijk} = \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}}$$

where N_{ijk} is the number of data samples in $\mathcal{D}$ where X_i takes its k-th value and $\boldsymbol{Pa}_i$ takes its j-th configuration, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ for all i,j and k, $(\alpha_{ij1}, \ldots, \alpha_{ijr_i})$ are the hyperparameters of the Dirichlet distribution and $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$ for all i, j and k.

Other approaches to the parametric learning focus on selecting a particular set of parameters based on some criterion, being the best known approaches the maximum likelihood (ML) and the maximum a posteriori (MAP) estimations.

The ML approach selects the set of parameters $\hat{\boldsymbol{\theta}}$ that maximises the probability of observing the dataset $\mathcal{D}$:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\mathcal{D}|S, \boldsymbol{\theta})$$

where the likelihood function $P(D|S, \boldsymbol{\theta})$ is defined as

$$P(\mathcal{D}|S, \boldsymbol{\theta}) = \prod_{d=1}^{N} P(\boldsymbol{x}^{(d)}|S, \boldsymbol{\theta})$$

The parameters that maximise the likelihood function are:

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$$

In the MAP approach we look for the set of parameters that maximises the posterior probability of the parameters $P(\boldsymbol{\theta}|S, \mathcal{D})$. Under the assumptions mentioned at the beginning of this section, the MAP estimator calculation is equivalent to the Bayesian approach, though both approaches are essentially different.

A more detailed derivation of the parameters can be consulted in Heckerman (1995), Bernardo and Smith (2000) and MacKay (2003).

B. Structural learning

The structural learning of Bayesian networks is the step where the graph of conditional (in)dependencies is inferred from $\mathcal{D}$. There are two main approaches to the problem: Algorithms based on the detection of the conditional (in)dependencies and algorithms based on a score+search procedure.

(In)dependency detection based algorithms. According to de Campos (1998) this type of algorithm starts taking as input any of these kinds of information:

- A database from where the conditional independencies can be tested running an independence test
- A joint probability distribution where the conditional independencies can be checked
- A list of conditional independencies

Given this input the algorithms try to represent with the DAG a large percentage (or even all if possible) of the relationships. Two examples of learning algorithms based on detecting conditional (in)dependences are, the PC algorithm (Spirtes et al., 1991), which is probably the best known algorithm to learn Bayesian network structures by detecting conditional independences, and the algorithm to learn polytrees presented in Acid and de Campos (1995).

Two good reviews of structure learning algorithms based on detecting conditional (in)dependences are provided by Spirtes et al. (1993) and de Campos (1998).

Score+search algorithms. The score+search algorithms tackle the problem of structure induction as an optimisation problem. In order to search for the best structure we need, first, to define what a good structure is (i.e., we need to define a score that measures the goodness of a structure).

Regarding the score metrics, there are several functions that can be used to measure the goodness of a structure. One of these measures is the marginal likelihood, which is the probability of a dataset $\mathcal{D}$ given a structure S. It is a Bayesian score as it is the averaged probability over all the possible values of the parameters. Under certain assumptions the marginal likelihood can be calculated in a closed form (Cooper and Herskovits, 1992). This metric is known as the K2 score.

Another type of metric used to evaluate the possible structures is the penalised log-likelihood. For a given structure S we can obtain the maximum likelihood estimations of the parameters and, thus, we can compute the log-likelihood of a dataset $\mathcal{D}$ given the structure and the parameters. The problem with this score is that it increases as the S structure gets more complex. This is solved by adding a penalisation term consisting of the product of the network dimension (normally measured in terms of the number of parameters) and a penalisation function. The Bayesian information criterion (BIC,

Schwarz (1978)) is an example of penalised log-likelihood criterion where the penalisation function is $\frac{1}{2}log(N)$.

The information theory (Cover and Thomas, 2006) provides a good framework to define metrics that can be used to evaluate the goodness of a given structure. The minimum description length criterion (MDL) (Rissanen, 1978) states that the best structure to describe a dataset is the one that minimises the encoding length of the data and the model. The final equation is the same as in the BIC metric, although the way to obtain it is completely different. The literature includes several works where scores based on information theory such as the entropy (Herskovits and Cooper, 1990; Geiger, 1992) or mutual information (Chow and Liu, 1968) are used to learn the structure of Bayesian network models.

The search of the best structure is an NP-hard problem and, thus, heuristic search methods are required to obtain good enough structures in a reasonable time. Greedy search (Buntine, 1991; Cooper and Herskovits, 1992), simulated annealing (Chickering et al., 1995), tabu search (Bouckaert, 1995), genetic algorithms (Larrañaga et al., 1996; Etxeberria et al., 1997), estimation of distribution algorithms (Blanco et al., 2003), Markov chain Monte Carlo (Myers et al., 1999), variable neighbourhood search (de Campos and Puerta, 2001) or ant colony optimisation (de Campos et al., 2002) are examples of search methods used to learn the structure of Bayesian networks.

3.3 Bayesian network classifiers

Bayesian network classifiers constitute a family of classification functions based on Bayesian networks. The number of members of this family is very high and an exhaustive review of all of them is out of the scope of this introductory chapter. In this dissertation we will not consider paradigms such as general Bayesian networks or Bayesian multinets and we will focus on classifiers whose structure is specially designed for the classification task.

In a classification problem we have a special variable (the class, C) that we want to predict based on a set of predicting variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$. We start from the assumption (not always fulfilled) that all the predicting variables are able to provide us with information about the class. Therefore, in most of the Bayesian network classifiers we make the initial assumption that all the predicting variables are conditionally dependent on the class (i.e.,

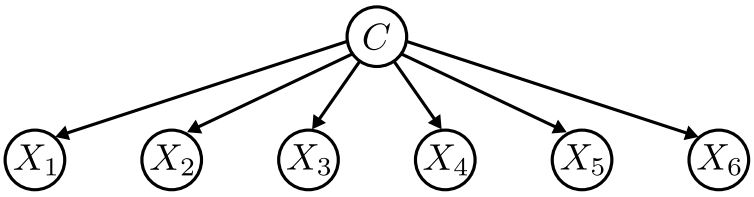


Fig. 3.3. Example of the structure of a naive Bayes model. All the predicting variables (from X_1 to X_6) are conditionally independent given the class variable (C).

the class variable is parent of all the predicting variables). Depending on the restrictions imposed on the rest of the DAG in the model we can have very simple models, like the naive Bayes (Minsky, 1961), or arbitrarily highly complex models such as k-dependence Bayesian classifiers (Sahami, 1996).

In this section we will focus on two models that are of special interest for the development of the dissertation: the naive Bayes and the tree augmented naive Bayes (Friedman et al., 1997). A brief review on other Bayesian network classifiers will be also provided.

3.3.1 Naive Bayes models

The naive Bayes model (also known in the literature as idiot Bayes (Ohmann et al., 1988; Hand and You, 2001), simple Bayesian classifier (Domingos and Pazzani, 1997; John, 1997) or independent Bayes (Todd and Stamper, 1994; Crichton et al., 1998)) is the most simple Bayesian network classifier, as it assumes the conditional independence of all the predicting variables given the class (see Figure 3.3).

Given its simplicity, the learning process is very fast even when we have very high dimensional problems. This is, on one hand, because as the structure of the model is fixed, there is no need for a structural learning step and, on the other hand, because the number of parameters we need to estimate is reduced as the predicting variables depend only on the class variable. For instance, in a binary classification problem with 50 binary predicting variables, without any assumption of conditional independence we would need to estimate $2^{51}-1$ parameters to define the joint probability distribution, while in a naive Bayes model the total number of parameters is reduced to 51. This leads to another

very important question. The more simple the model is, the fewer the number of parameters we have to estimate and, thus, the more sound the estimations from the (limited) data are. Probably this is one of the reason why, regardless the strong assumption of independence, naive Bayes models are competitive with respect to other, more sophisticated classifiers.

Under the hypothesis of conditional independence, the only parent of all the predicting variables is the class. Thus, the parameters we have to estimate from the data are the conditional probabilities, $P(x_{ij}|c)$, and the a priori probabilities of the class, $P(c)$. Typically, maximum likelihood estimators are used to estimate these probabilities. These estimators are normally smoothed using the Laplace correction in order to avoid the presence of null parameters that can lead to problems when the classifier is used to obtain the posterior probability of the class. The smoothed estimators would be[2] :

$$P(x_{ij}|c) = \frac{1 + N_{ijc}}{r_i + N_c} \tag{3.1}$$

$$P(c) = \frac{1 + N_c}{r_c + N} \tag{3.2}$$

where N_{ijc} is the number of instances of the class c where $X_i = x_{ij}$ and N_c is the number of instances labelled as c. Given the assumption of conditional independence and the Bayes rule, we have that:

$$P(c|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|c)P(c)}{P(\boldsymbol{x})} = \frac{\prod_{i=1}^{n} P(x_i|c)P(c)}{P(\boldsymbol{x})}$$

given that (for the prediction) we are interested in obtaining the probability distribution of the class variable given an instance $\boldsymbol{x}$, the denominator is the same for all the class values and, thus, we have that:

$$P(c|\boldsymbol{x}) \propto \prod_{i=1}^{n} P(x_i|c)P(c)$$

For each c we can obtain $\prod_{i=1}^{n} P(x_i|c)P(c)$ and then, taking into account that $\sum_c P(c|\boldsymbol{x}) = 1$ we can normalise all the products to obtain the actual values of the posterior probabilities of the class variable.

[2] The smoothed estimator can also be seen as a MAP or a Bayesian estimator where the a priori probability of all the variables is modelled as a Dirichlet distribution where all the α_{ijk} hyperparameters are 1 (see Section 3.2.1.1)

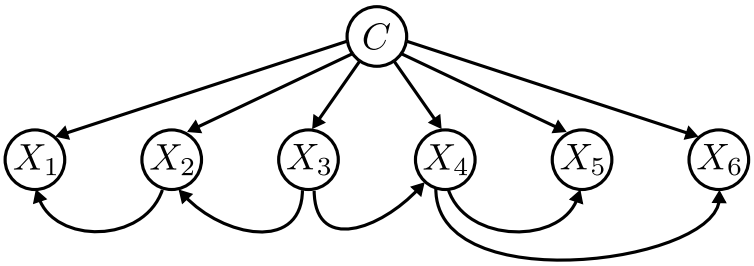


Fig. 3.4. Example of the structure of a TAN model. The dependences between predicting variables are represented as a tree structure.

3.3.2 Tree augmented naive Bayes models

The tree augmented naive Bayes model (TAN) was proposed by Friedman et al. (1997) as an extension of the naive Bayes model that could overcome the assumption of conditional independence. The model keeps the class variable as parent of all the predicting variables but it extends the DAG with conditional dependences between predicting variables represented as a tree structure (see Figure 3.4).

The tree structure of dependences is induced in the structural learning step using the construct-TAN algorithm (see Figure 3.1) which is an adaptation of the Chow-Liu algorithm (Chow and Liu, 1968) that uses the conditional mutual information between predicting variables rather than the mutual information. The conditional mutual information between two variables X_i and X_k given the class C is:

$$I(X_i, X_k|C) = \sum_{x_i} \sum_{x_j} \sum_{c} P(x_i, x_k, c) \frac{P(x_i, x_k|c)}{P(x_i|c)P(x_k|c)}$$

The algorithm starts calculating the conditional mutual information between every pair of predicting variables. Then, the complete undirected graph is created and the edges are weighted with the conditional mutual information between the variables it connects. After that, the Kruskal algorithm is used to obtain the maximum weighted spanning tree as follows:

1. Add the two edges with the maximum weight to the tree
2. Look at the next edge with the highest weight. If the addition of this edge leads to a cycle ignore it. If not, add the edge to the tree

Structural learning of TAN models

1	Compute $I(X_i, X_k\|C)$ between each pair of variables with $i < k$, $i, k = 1, \ldots, n$
2	Build a complete undirected graph in which the vertices are $X_1, \ldots, X_n$
3	Set the weight of the edge connecting X_i and X_k to $I(X_i, X_k\|C)$
4	Use the Kruskal algorithm to build a maximum weighted spanning tree from the previous graph
5	Transform the undirected tree into a directed tree selecting a variable as the root and setting the direction of the edges
6	Construct a TAN model by adding a C vertex and an arc from C to each variable X_i

Alg. 3.1: Pseudo code of the algorithm used in the structural learning of TAN models

3. Repeat step 2 until selecting $n - 1$ edges

Once the tree is constructed, one of the predicting variables is selected as the root and the direction of the edges is set. The class variable is included in the structure and edges from it to the rest of the variables are added. The result is a DAG where the conditional dependences between predicting variables are represented by a tree structure.

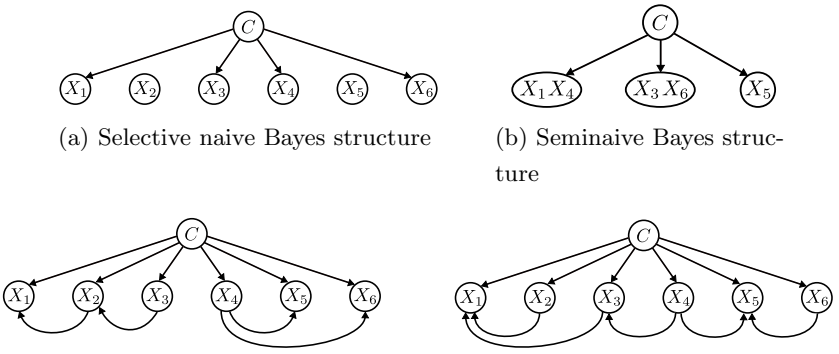
Given the structure of the model, the parameters can be learnt by means of maximum likelihood estimators (modified with the Laplace correction):

$$P(x_{ij}|c, pa_i) = \frac{1 + N_{ijcpa_i}}{r_i + N_{cpa_i}}$$
$$P(c) = \frac{1 + N_c}{r_c + N}$$

where N_{ijcpa_i} represents the number of instances of the class c where $X_i = x_{ij}$ and $Pa_i = pa_i$ and N_{cpa_i} the number of instances labelled as c where $Pa_i = pa_i$.

3.3.3 Other Bayesian network classifiers

In some problems not all the domain variables are useful for the classification purpose, some being even harmful in some situations. The feature subset selection paradigm tackles this problem, trying to identify the optimal subset of variables that leads to the best possible classification function. One of the



(a) Selective naive Bayes structure

(b) Seminaive Bayes structure

(c) Forest augmented network structure

(d) 2 dependence Bayesian structure

Fig. 3.5. Examples of the structure of different Bayesian network classifiers.

improvements of the naive Bayes algorithm, the selective naive Bayes (Kohavi and John, 1997), mixes the variable selection with the naive Bayes algorithm. As a result we have a naive Bayes model where not all the predicting variables are present (see Figure 3.5b).

The naive Bayes is limited by its inability to model dependences between predicting variables. Although it gives very competitive results in many classification problems, there are problems where its performance is poor (Pazzani, 1997), probably due to the independence assumption. This problem is considered in the seminaive Bayes algorithm (Kononenko, 1991; Pazzani, 1997). The idea behind the seminaive Bayes is to consider dependences by creating new variables as the Cartesian product of some predicting variables. Figure 3.5-a shows an example of a seminaive Bayes structure.

In Kononenko (1991) the predicting variables are joined when they meet a condition related with the concept of independence and the reliability in the estimation of the parameters. In Pazzani (1997) the author proposes a greedy wrapper approach to obtain seminaive Bayes structures by removing irrelevant variables and joining correlated variables. Two algorithms are proposed in this work, the forward sequential selection and joining algorithm (FSSJ) and the backward sequential elimination and joining (BSEJ). The search in both cases is guided by the accuracy (estimated with a 10-cv) and during the search two options are considered: joining two variables or add/eliminate (add in the FSSJ and eliminate in the BSEJ) one of the variables. The option that leads

to the highest increase in the accuracy is chosen and the process follows until no improvement is obtained.

The TAN algorithm allows the inclusion of dependences in the structure of the model, but the restrictions are still quite strong. One of the attempts to create more flexible models is the forest augmented network algorithm (FAN) (Lucas, 2004) where the dependences are represented by a 'forest' of tree structures rather than a single tree structure (see Figure 3.5-c).

In both TAN and FAN models the predicting variables are allowed to have up to one parent (besides the class). The k dependence Bayesian model (k-DB) (Sahami, 1996) relaxes this restriction, allowing the predicting variables to have up to k predicting variables as parent. Depending on the k parameter, we can have structures ranging from a simple naive Bayes ($k = 0$) to a complete Bayesian network ($k = n-1$). Figure 3.5-d represents the structure of a k-DB model with $k = 2$.

Part II

Learning from Positive and Unlabelled Examples

4

State of the Art

The positive unlabelled learning topic is relatively recent, most of the works in the field being less than ten years old. The development of algorithms able to learn binary classification functions from positive and unlabelled examples arises as the answer to the need of inducing classifiers in problems, such as text classification, where negative examples can be difficult to get while unlabelled examples are readily available. It is in this field (text classification) where most of the algorithms have been developed and applied.

In this chapter we will review literature related with the learning from positive and unlabelled examples. In the first section we will analyse the attempts to answer the question of whether it is possible to learn models from positive and unlabelled examples or not. Then, in Section 4.2, we will briefly comment on the most representative algorithms proposed to learn classification functions in absence of negative examples. Finally, Section 4.3 is devoted to the applications of these algorithms.

4.1 Theoretical analysis of the positive unlabelled learning problem

There are not many works that tackle the problem of modelling the learning from positive examples. Due to the conclusions derived, Denis (1998) is probably the most significant work for the positive unlabelled learning field from it. This work is based on Valiant's PAC learning model (Valiant, 1984) and the statistical query model (Kearns, 1993).

Before defining the different concepts of learnability some definitions are required. Let $\mathcal{B}_n$ be the set of all Boolean functions from $X_n = \{0,1\}^n$ into $\{0,1\}$, and let $X = \bigcup_{n\geq 1} X_n$ and $\mathcal{B} = \bigcup_{n\geq 1} \mathcal{B}_n$. A class concept $\mathcal{C}$ over X is defined as a subset of $\mathcal{B}$, and $\mathcal{C}_n = \mathcal{C} \cap \mathcal{B}_n$.

An example of a concept $f \in \mathcal{C}_n$ is a pair $(x, f(x))$ where $x \in X_n$. We say that an example is positive when $f(x) = 1$ and negative when $f(x) = 0$; $pos(f)$ and $neg(f)$ represent, respectively, the set of x such that $f(x) = 1$ and the set of x such that $f(x) = 0$. Let μ be a probability distribution defined over X_n and $\mu(f) = \mu(pos(f))$. If $\mu(f) \neq 0$, μ_f denotes the restriction of μ to the set of positive examples of f, and is defined as:

$$\mu_f(x) = \begin{cases} \frac{\mu(x)}{\mu(f)} & \text{if } x \in pos(f) \\ 0 & \text{otherwise} \end{cases}$$

Given a concept $f \in \mathcal{C}$ and a hypothesis $h \in \mathcal{C}$, the error of the hypothesis is measured as:

$$error(h) = \sum_{x \in f \Delta h} \mu(x) = \mu(f \Delta h)$$

where $f \Delta h$ represents the symmetrical difference of the sets $pos(f)$ and $pos(h)$ (see Figure 4.1). That is, the error of a hypothesis is the probability that it disagrees with the concept about an instance x drawn randomly according to μ.

A representation scheme for a concept class $\mathcal{C}$ is a function $R : \mathcal{C} \to 2^{\Sigma^*}$ where Σ is a finite alphabet and such that:

$$\forall f \in \mathcal{C}, R(f) \neq \emptyset$$
$$\forall f, f' \in \mathcal{C}, R(f) \cap R(f') = \emptyset$$

We suppose that R is computable in polynomial-time. The size of a concept f is defined as:

$$size(f) = min\{|c|/c \in R(f)\}$$

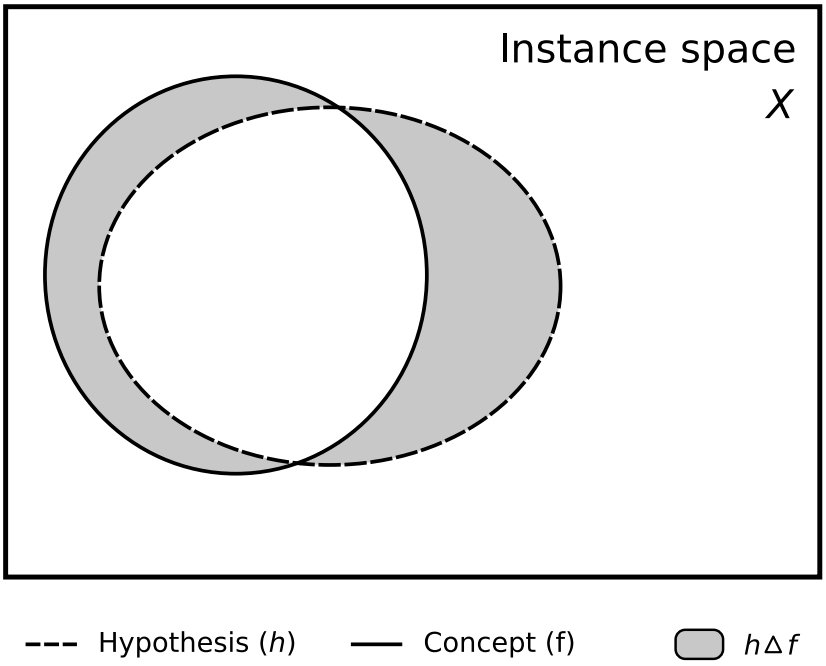


Fig. 4.1. Representation of the instance space, concept and hypothesis in the PAC learning theory. The hypothesis and the concept lines enclose the positive examples ($pos(h)$ and $pos(f)$ respectively).

A statistical query χ over X_n is defined as:

$$\chi : X_n \times \{0,1\} \to \{0,1\}$$

If $f \in \mathcal{B}_n$, $\chi_f(x,y) = 1$ if and only if $y = f(x)$.
We can define the following oracles:

Definition 10. *Let $\mathcal{C}$ be a concept class over X. Let $f \in \mathcal{C}_n$ and μ a distribution over X_n*

- *The oracle $EX(f,\mu)$ is a procedure that returns at each call an example $(x, f(x))$ drawn randomly according to μ*
- *The oracle $UNL(\mu)$ is a procedure that returns at each call an x drawn randomly according to μ*

- *The oracle $STAT(f, \mu)$ is a procedure that, for every statistical query χ and every $\tau \in (0, 1]$, with input (χ, τ) returns an approximation of $\mu(x|\chi(x, f(x)) = 1)$ with an accuracy at least τ*

All the oracles run in the unit time

Now we can define the PAC learnability as:

Definition 11. *Let $\mathcal{C}$ be a concept class over X. We say that $\mathcal{C}$ is **PAC learnable** if there exists a learning algorithm L and a polynomial $p(\cdot, \cdot, \cdot, \cdot)$ such that, for any $f \in \mathcal{C}$, any distribution μ on X and any $0 < \epsilon, \delta < 1$, if L is given access to $EX(f, \mu)$ and to inputs δ and ϵ then, with probability at least $1 - \delta$, L outputs a hypothesis concept $h \in \mathcal{C}$, in time bounded by $p(\frac{1}{\epsilon}, \frac{1}{\delta}, size(f), n)$, such that $\mu(f \Delta h) \leq \epsilon$.*

The learnability from statistical queries is defined as:

Definition 12. *Let $\mathcal{C}$ be a concept class over X. We say that $\mathcal{C}$ is **learnable from statistical queries** if there exists a learning algorithm L and polynomials $p(\cdot, \cdot, \cdot)$, $q(\cdot, \cdot, \cdot)$ and $r(\cdot, \cdot, \cdot)$ such that, for any $f \in \mathcal{C}$, any distribution μ over X and any $0 < \epsilon < 1$, if L is given access to $STAT(f, \mu)$ and to input ϵ, then:*

- *For every query (χ, τ) made by L, the predicate χ can be evaluated in time $q(\frac{1}{n}, n, size(f))$, and $\frac{1}{\tau}$ is bounded by $r(\frac{1}{n}, n, size(f))$*
- *L halts in time bounded by $p(\frac{1}{n}, n, size(f))$*
- *L outputs a hypothesis $h \in \mathcal{C}$ that satisfies $\mu(f \Delta h) \leq \epsilon$.*

In order to apply the PAC learnability concept to the case of positive learning, we have to face the first problem: How should the error of a hypothesis be measured? If it is measured only on the positive examples then a hypothesis that contains all the instances would be always the best option. Thus, it is mandatory to take the negative cases into account but this is also problematic because there can be two distributions μ and μ' with the same restriction on the positive examples of the target but very different on the negatives. Another problem comes from the fact that we cannot differentiate a negative instance from a positive one with probability equal to zero.

The conclusion is that it is not possible to learn a concept only from positive examples, but it is possible as long as we have information about the underlying distribution. This extra information is given by the oracles

$UNL(\mu)$ or $STAT(1,\mu)$. Taking the unlabelled examples into account, the PAC learnability from positive examples is defined as:

Definition 13. *Let $\mathcal{C}$ be a concept class over X. We say that $\mathcal{C}$ is* ***PAC learnable from positive examples*** *if there exists a learning algorithm L and a polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that, for any integer n, any $f \in \mathcal{C}_n$, any distribution μ on X_n and any $0 < \epsilon, \delta < 1$, if L is given access to $EX(f,\mu_f)$, $UNL(\mu)$ and to inputs δ and ϵ then, with probability at least $1-\delta$, L outputs a hypothesis concept $h \in \mathcal{C}_n$, in time bounded by $p(\frac{1}{\epsilon},\frac{1}{\delta},size(f),n)$, such that $\mu(f \Delta h) \leq \epsilon$.*

and the learnability from positive statistical queries is defined as:

Definition 14. *Let $\mathcal{C}$ be a concept class over X. We say that $\mathcal{C}$ is* ***learnable from positive statistical queries*** *if there exists a learning algorithm L and polynomials $p(\cdot,\cdot,\cdot)$, $q(\cdot,\cdot,\cdot)$ and $r(\cdot,\cdot,\cdot)$ such that, for any integer n, any $f \in \mathcal{C}_n$, any distribution μ over X_n and any $0 < \epsilon < 1$, if L is given access to $STAT(f,\mu_f)$, $STAT(1,\mu)$ and to input ϵ, then:*

- *For every query (χ,τ) made by L, the predicate χ can be evaluated in time $q(\frac{1}{n},n,size(f))$, and $\frac{1}{\tau}$ is bounded by $r(\frac{1}{n},n,size(f))$*
- *L halts in time bounded by $p(\frac{1}{n},n,size(f))$*
- *L outputs a hypothesis $h \in \mathcal{C}_n$ that satisfies $\mu(f \Delta h) \leq \epsilon$.*

The conclusion that can be drawn from the work of Denis (1998) is that learning from positive examples is possible with the aid of unlabelled instances.

Denis (1998) also analyses the learnability of two particular class concepts, the k-disjunctive normal forms (k-DNF) and the k-decision lists (k-DL). The conclusion reached is that both concepts are learnable from positive statistical queries.

4.2 Classifier induction algorithms in positive unlabelled learning problems

During the last ten years many algorithms have been developed to tackle the learning of binary classifiers from positive and unlabelled examples. Most of these algorithms come from the text classification domain, though recently

other fields are starting to pay more attention to this kind of problems. In this section we will review some of the most significant works in the induction of binary classifiers without negative examples.

There are two main approaches to the problem of learning from positive and unlabelled instances:

- A set of classifier induction algorithms is based on a two step process. First, a reliable set of negative instances is identified and, then, the known positive examples and this set of identified negative instances are used to build the final classifier.
- The second approach is a more direct application of the conclusion presented in the previous section. In this approach information about the underlying probability distribution (μ in the PAC learning framework) is directly inferred from the set of unlabelled instances and is then used in the learning of the model from the positive examples.

In the first category, the two-step algorithms, we have the spy-EM (Liu et al., 2002), where the set of reliable negative examples is obtained in a first run of the expectation-maximisation algorithm (EM) (Dempster et al., 1977). In this first run of the EM the probabilities of the known positive cases are initialised as $P(C = 1) = 1$, $P(C = 0) = 0$ and those of the unlabelled instances are initialised as $P(C = 1) = 0$, $P(C = 0) = 1$ (i. e., the unlabelled examples are initialised as negative) [1] . While the probabilities of the unlabelled instances change iteratively during the process, the probabilities of the positive cases are kept fixed. The set of reliable negative instances is obtained setting a threshold estimated using the 'spy' cases (positive cases that are introduced in the set of unlabelled instances before the EM algorithm is run). A second run of the EM algorithm is carried out, initialising the positive cases as positive, the reliable negative cases as negative and leaving the rest of the instances without initialisation (these instances are not used in the training of the first model). In this second run the positive cases are kept fixed but the 'negative' instances are allowed to evolve. After the second run of the EM algorithm, instead of taking the last model as the final classifier, the authors propose a

[1] This is a relatively frequent approach in positive unlabelled learning problems. In many situations the number of negative examples in the set of unlabelled instances is much higher than the number of positive examples

criterion to select the best classifiers among all the models obtained during the EM.

In Yu et al. (2002) and Yu (2005) the authors introduce the positive example based Learning framework (PEBL) for the classification of web pages without negative examples, and they propose the mapping-convergence algorithm (M-C). This algorithm takes two steps. The first one, called mapping stage, is based on 1-DNF (which, as Denis demonstrated, is learnable from positive and unlabelled examples (Denis, 1998)) to identify a reliable set of negative examples. It basically consists of identifying a list of 'positive features', i.e., features (words) that are much more frequent in positive cases than in unlabelled instances. This list of features (which is a 1-DNF) is used to filter out, from the set of unlabelled instances, those examples that are likely to be positive. The problem with these negative examples is that they are very far from the decision boundary and, thus, a SVM learnt with these examples would be quite inaccurate. The goal of the second step, called convergence stage, is to refine this boundary based on the marginal property of the SVMs. The experimental evaluation of this classifier shows that it is able to outperform (linear) SVMs trained on positive and negative examples.

In a later work, Yu (2003) presents the support vector mapping convergence (SVMC) algorithm, which is based on the M-C algorithm, and tackles the problem of the increase in the computational time when the size of the set of unlabelled instances increases.

In Yu et al. (????) the authors propose some improvements on the M-C algorithm (Yu et al., 2002). On one hand, in the identification of 'positive features' they not only consider the relative frequency of the terms, but also the frequency in the set of positive cases. Thus, their list of positive features is much smaller than in the M-C algorithm as the less frequent terms are not included in the list. This modification leads to a much bigger set of reliable negatives. On the other hand, in the second step, instead of picking up one of the classifiers in the iterative process as the final one, they propose a weighted voting scheme to average all the classifiers. The weighting factors are related to the capacity of each classifier to predict the positive instances. They compare their algorithm with M-C (Yu et al., 2002) and one-class SVM (Manevitz and Yousef, 2001) obtaining similar results as the M-C algorithm and much better ones than the one-class SVM.

In Duan et al. (2007) the authors propose an algorithm (TRS-SVM) for the classification of web pages. This algorithm bases the identification of the reliable set of negative examples on a relaxation of the rough set theory (Pawlak, 1991) and then, uses the identified negative examples to iteratively build a classifier using SVMs.

Li and Liu (2003) present an algorithm where the identification of the reliable negative instances is based on the Rocchio algorithm (Rocchio, 1971). The reliable negative examples identified, along with the known positive cases, are used in the second step of the algorithm to iteratively train SVM classifiers.

In Liu et al. (2003) the authors compare the combination of different techniques for the identification of reliable negative examples with different classifier induction algorithms. They also proposed two new approaches to both steps. For the identification of negative instances they propose a naive Bayes model trained on positive and unlabelled examples (using the latter as negative examples). For the second step they propose the use of a single SVM, rather than the iterative process used in other works. They also propose a new SVM-based algorithm, the biased-SVM.

Li and Liu (2005) presents a new algorithm, the A-EM (augmented EM) to cope with the positive unlabelled learning problem when the positive examples and the unlabelled instances come from different probability distributions.

There are some works in the positive unlabelled learning field that tackle the problem of having small sets of positive examples which, in certain domains, can be quite a common problem. One of these papers is Li et al. (2007a). In this paper the authors propose the LPLP algorithms (learning from probabilistically labelled positive examples) for the classification of product pages in commercial web sites. This algorithm first identifies a set of representative words and considers this list as a 'representative document'. They get a set of 'likely positive' examples by comparison and then they use these identified examples, along with the known positive examples, to initialise the EM algorithm (Dempster et al., 1977).

Regarding the algorithms that directly infer the information about the underlying probability distribution from the set of unlabelled instances, in Letouzey et al. (2000) the authors propose an adaptation of the C4.5 algorithm (Quinlan, 1993), the POSC4.5, that is able to learn decision tree models from only positive and unlabelled examples.

Another example of this sort of algorithms is the positive naive Bayes (PNB) (Denis et al., 2002), an adaptation of the naive Bayes induction algorithm (Minsky, 1961) to the positive unlabelled learning context, which is explained in detail in Section 7.1. In Denis et al. (2003) the authors tackle the problem of learning from very few positive instances by incorporating the co-training concept (Blum and Mitchell, 1998) to the PNB algorithm.

Zhang and Lee (2005) present a probabilistic approach to the problem where the prediction of a class is based on the probability of being in the set of positive cases and the probability of being in the set of unlabelled instances.

In Lee and Liu (2003) the authors tackle the problem of learning from positive and unlabelled examples as a problem of learning from noisy examples. They label all the unlabelled instances as negative and then use a weighted logistic regression to handle the noise in the set of 'negative' examples. In this work the authors also propose a metric to measure the performance of the classifiers. This metric, which is proportional to the product of the recall and the precision, is used to set the regularisation parameters of the logistic regression.

4.3 Applications of the positive unlabelled learning algorithms

Most of the algorithms cited in the previous section are devised to classify texts (Yu et al., 2002; Li et al., 2007a; Schneider, 2004), and most of them are tested on different document datasets, but there are some works where these algorithms are applied to real-life positive unlabelled learning problems from other fields.

One of these papers is Zhao et al. (2008). In this work the authors propose a positive unlabelled learning procedure to predict the function of genes. With the development of high-throughput biotechnologies, huge amounts of data about genes and proteins have been collected. These big databases can be used to predict the function of novel discovered genes. Although there are genes whose function is known, it is difficult to ensure that a given gene has not some particular function. As a result, positive examples (genes with the function at study) and unlabelled genes are available, but not negative genes. The authors propose a method basically consisting of identifying a set of

reliable negative examples. The goal is achieved using a similar procedure as in Liu et al. (2002) but using SVMs instead of the EM as the base algorithm. Then an iterative step is carried out training a SVM on the known positives and the identified negative examples and then using it to generate a new set of negative instances for the next step in the iteration.

Another application in the computational biology field is the prediction of non-coding RNA genes (ncRNA) (Wang et al., 2006). The authors develop an algorithm (positive sample only learning, PSoL) that labels those instances most distant to the set of positive cases as negative examples and, then, iteratively increments the set of negative examples using SVM classifiers.

In Liu et al. (2005) the authors develop a new SVM-based method for the identification of specific land-cover class on remote sensing images. They test their classifier induction algorithm on satellite images of Victoria Harbor in Hong Kong.

A somehow different application of the positive unlabelled learning algorithms can be found in Li et al. (2007b). The problem they face in this work is the identification of unexpected instances in test sets. When a classifier is induced it learns to distinguish between some predefined classes, but it can be that some of the test or future instances that the classifier is required to classify do not belong to any of these classes. This is quite obvious if we think of classifiers learnt in fast evolving fields such as cancer subtype identification or computer science text classification. The identification of this sort of 'outliers' is very interesting as it can point out new discoveries, such as new subtypes of cancer. The author propose to model the identification of this sort of examples as a positive unlabelled learning problem where the instances from the known classes are the positive examples. The particularity of this problem is that, contrary to what happens in most of the other applications, the amount of positive cases in the set of unlabelled instances is very high (it is not very likely to find an instance that does not belong to any of the originally defined classes). The authors propose a new algorithm based on the generation of negative data and the naive Bayes algorithm (Minsky, 1961) to solve the problem of identifying unexpected instances in test sets.

5

Description of the Datasets

The evaluation of classifiers in positive unlabelled learning problems is not a straightforward issue. The classical performance measures rely on the confusion matrix but, in binary problems where no negative examples are available, only half of the matrix can be computed, and, thus, the classical performance measures can not be estimated from the data. The only exception is the recall as this measure does not depend on the negative instances. However, this measure alone is not enough to evaluate the goodness of a classification function (see Section 2.2).

The experimental evaluation of the algorithms proposed in this dissertation has been carried out in problems where the absence of negative examples has been simulated. The datasets of positive and unlabelled instances are obtained from sets of positive and negative examples. Therefore, we know the class of the unlabelled instances (though it is not used in the positive unlabelled learning algorithms) and we can use this information to obtain any desired performance measure. The reasons why simulated problems have been used to test the algorithms are the following:

- The absence of a proper statistical way to evaluate classifiers. The supervised and semi-supervised approaches to evaluate classification functions are not directly applicable in positive unlabelled learning problems. Part of the contributions of this dissertation are in the field of classifier evaluation in absence of negative examples (see Chapter 8). However, this way of estimating the performance of a classifier suffers from some problems and, thus, we have decided to use simulated problems as a more robust way of evaluating the new proposed algorithms.

- Simulating the absence of negative examples allows us to control the number of positive and negative examples in the set of unlabelled instances. This is very interesting when we want to test the performance of the algorithms in different situations.
- The final reason for using simulated problems is that there are lots of supervised databases available in repositories such as UCI (Blake and Merz, 1998) but no positive unlabelled datasets are so readily available.

In the experimentation throughout the dissertation we have used different types of datasets. Some of them are based on real-life problems where the absence of negative examples has been simulated and some others have been generated from known probability distributions with some specific goals. As some of these datasets have been used in different parts of the dissertation, we have gathered all the descriptions in this chapter. In the following sections we will describe all the datasets that have been used to test the different algorithms and the processes followed to create them. Table 5.1, at the end of this chapter, summarises the datasets used in the experimental part of the dissertation.

5.1 Real-life datasets

In this section we will show how simulated positive unlabelled learning problems can be obtained from completely labelled datasets and we will describe the real-life based datasets used in the experimental part of the dissertation.

5.1.1 Dataset generation process

Positive unlabelled learning problems have two classes: the instances we are interested in (the positive class) and the instances we are not interested in (the negative class). Nevertheless, some of the (supervised) databases available in different repositories have more than just two classes, so the first step in order to build positive unlabelled learning problems from labelled databases is the binarisation of the class. This step consists of selecting one of the classes in the

problem as the positive class[1] . Then, all the instances belonging to this class are labelled as positive and the rest of the instances are labelled as negative.

Once the original labelled database has been binarised, positive unlabelled learning problems are built creating a set of positive examples ($\mathcal{D}_p$) and a set of unlabelled instances ($\mathcal{D}_u$). The set of positive examples can be obtained by randomly drawing, from the labelled database, some positive instances. The set of unlabelled instances is created by mixing (in the desired proportion) positive and negative instances randomly drawn from the labelled database and then removing their labels (we 'remove' the labels because the algorithms do not use them, but we actually keep them in order to be able to obtain the performance measures).

There are several parameters that have to be set in the process of simulating positive unlabelled learning problems. First, we have to select the original database from where the instances will be drawn. We also have to set the size of both the set of positive examples and the set of unlabelled instances. Finally, we can mix positive and negative examples in different proportions in the set of unlabelled instances so we have to select the ratio of positive cases hidden in the set of unlabelled instances. Once these four parameters have been set, we can obtain as many positive unlabelled learning problems as we want. This is very interesting because we can create a list of similar problems and then compare the results using statistical tests to assess the significance of the differences.

In order to simplify the descriptions in the rest of the chapter, we will define a 'scheme' as the set: original database, cardinality of $\mathcal{D}_p$, cardinality of $\mathcal{D}_u$ and ratio of positive cases in $\mathcal{D}_u$. For instance, a scheme would be database A, $|\mathcal{D}_p|$ =100, $|\mathcal{D}_u|$ =1,000, ratio=0.2, meaning that the positive unlabelled learning problems that follow this scheme are obtained from the database A and that the number of known positive examples is 100 and the number of unlabelled instances is 1,000 (200 positive examples and 800 negative examples).

The real-life based datasets have been built based on three supervised databases: ACCDON, Letter Recognition and Nursery.

[1] An alternative would be mixing more than one class to create the set of positive examples. We have used the first approach (a single class as positive) because it is closer to the concept of information retrieval

5.1.2 ACCDON datasets

The ACCDON is a database presented in Castelo and Guigó (2004) for the problem of splice site prediction. It contains sequences of donor and acceptor splice sites labelled as true donor/acceptor sites and false donor/acceptor sites[2] . It is a binary problem, but there are two kinds of negative splice sites, false sites obtained from coding regions and false sites obtained from non-coding regions.

The predicting variables are the nucleotides at each position around the splice site. This means that each predicting variable can take four values (a, c, t and g, the bases that can be found in any DNA sequence). All the splice sites have a two base pair constant part that has been removed, as it does not give any information about the class. The total number of predicting variables (once the constant part has been removed) is 21 for the acceptor sites and seven for the donor sites.

Starting from the database described in Castelo and Guigó (2004) we have created six supervised binary databases:

- Accpetor sites coding - Containing true acceptor sites as positive examples and false acceptor sites obtained from coding regions as negative examples.
- Accpetor sites intron - Containing true acceptor sites as positive examples and false acceptor sites obtained from non-coding regions as negative examples.
- Accpetor sites mixed - Containing true acceptor sites as positive examples and false acceptor sites obtained from both coding and non-coding regions as negative examples.
- Donor sites coding - Containing true donor sites as positive examples and false donor sites obtained from coding regions as negative examples.
- Donor sites intron - Containing true donor sites as positive examples and false donor sites obtained from non-coding regions as negative examples.
- Donor sites mixed - Containing true donor sites as positive examples and false donor sites obtained from both coding and non-coding regions as negative examples.

[2] In the process of protein expression, the messenger RNA is modified, being one of these modifications the elimination of parts of the sequence (the introns). The splice sites are the points in the sequence where the RNA is cut and joined again to eliminate the introns. See Chapter 10 for more information about this process

There are two sets of positive unlabelled learning problems obtained from the ACCDON database, that from now on, will be referred to as ACCDON datasets v.1 and ACCDON datasets v.2.

5.1.2.1 ACCDON datasets v.1

In the first version of datasets the six original labelled databases described above are combined with three cardinalities for $\mathcal{D}_p$ (100, 200 and 300) and three cardinalities/ratio of positive cases for $\mathcal{D}_u$ (2,100/0.048, 2,300/0.13 and 2,500/0.2), resulting in a total of 54 schemes. Using each scheme fifty sets of positive and unlabelled examples were obtained, making a total of 2,700 unlabelled learning problems.

5.1.2.2 ACCDON datasets v.2

In the second version of the datasets three cardinalities for the set of positive examples (100, 500 and 1,000) are combined with six ratios of positive cases in $\mathcal{D}_u$ (0.01, 0.1, 0.2, 0.3, 0.4 and 0.5), yielding a total of 108 schemes. The total number of unlabelled instances in all the schemes is 10,000. Following each scheme 100 datasets were constructed, resulting in a total of 10,800 positive unlabelled learning problems.

5.1.3 Letter Recognition datasets

The Letter Recognition database (Blake and Merz, 1998) contains a set of letters from the English alphabet characterised by some measures obtained from their graphical representation. The sixteen attributes that represent each letter are the following[3] :

- x-box: Horizontal position of box
- y-box: Vertical position of box
- width: Width of box
- high: Height of box
- onpix: Total number on pixels
- x-bar: Mean x of on pixels in box

[3] The predicting variables have been renamed in the positive unlabelled datasets as V_1 to V_{16}

- y-bar: Mean y of on pixels in box
- x2bar: Mean x variance
- y2bar: Mean y variance
- xybar: Mean x y correlation
- x2ybr: Mean of x * x * y
- xy2br: Mean of x * y * y
- x-ege: Mean edge count left to right
- xegvy: Correlation of x-edge with y
- y-ege: Mean edge count bottom to top
- yegvx: Correlation of y-edge with x

All the variables described above are discrete and take values from 0 to 15. The class variable can take 26 values (all the English alphabet letters, from A to Z), and the total number of examples from each class ranges from 735 to 813 instances.

In order to binarise the database we have to pick up one of the letters as the positive class and label the rest of the instances as negative. Three different databases of positive and negative instances were created in this way, considering letters 'D' (805 instances), 'P' (803 instances) and 'U' (813 instances) as the positive class. We have named the three original databases Letter Recognition D, Letter Recognition P and Letter Recognition U.

As in the case of datasets based on the ACCDON database, there are two sets of positive unlabelled problems that we have labelled as Letter Recognition datasets v.1 and Letter Recognition datasets v.2.

5.1.3.1 Letter Recognition datasets v.1

In the first version of datasets the three labelled databases described above are combined with three cardinalities for $\mathcal{D}_p$ (100, 200 and 300) and three cardinalities/ratio of positive cases for $\mathcal{D}_u$ (2,100/0.048, 2,300/0.13 and 2,500/0.2), resulting in a total of 27 schemes. Using each scheme fifty sets of positive and unlabelled examples were obtained, totaling 1,350 unlabelled learning problems.

5.1.3.2 Letter Recognition datasets v.2

In the second version of the datasets three cardinalities for the set of positive examples (100, 200 and 300) are combined with six ratios of positive cases

in $\mathcal{D}_u$ (0.01, 0.1, 0.2, 0.3, 0.4 and 0.5), yielding a total of 54 schemes. The cardinality of $\mathcal{D}_u$ was 5,000 in all the schemes. Following each scheme 100 datasets were constructed, resulting in a total of 5,400 positive unlabelled learning problems.

5.1.4 Nursery datasets

The Nursery database (Blake and Merz, 1998) was obtained from a decision model used in the eighties in Ljubljana (Slovenia) to rank applications for nursery schools. It consists of 12,960 instances represented by the following eight attributes:

- parents: Discrete variable with three possible values (usual, pretentious, great_pret)
- has_nurs: Discrete variable with five possible values (proper, less_proper, improper, critical, very_crit)
- form: Discrete variable with four possible values (complete, completed, incomplete, foster)
- children: Discrete variable with four possible values (1, 2, 3, more)
- housing: Discrete variable with three possible values (convenient, less_conv, critical)
- finance: Discrete variable with two possible values (convenient, inconv)
- social: Discrete variable with three possible values (non-prob, slightly_prob, problematic)
- health: Discrete variable with three possible values (recommended, priority, not_recom)

The instances are divided into five classes: not_recom, recommended, very_recom, priority and spec_prior. The binarisation of this database was done by selecting the spec_prior class (4,044 instances) as positive and labelling the rest as negative. Thus, only one binary labelled dataset was used in the construction of positive unlabelled learning problems from the Nursery database.

Again, two sets of positive unlabelled learning problems (Nursery datasets v.1 and Nursery datasets v.2) were obtained from the Nursery database.

5.1.4.1 Nursery datasets v.1

In the first version of datasets three cardinalities for $\mathcal{D}_p$ (250, 500 and 1,000) and three cardinalities/ratio of positive cases for $\mathcal{D}_u$ (9,016/0.011, 9,960/0.075 and 11,916/0.25) were combined, resulting in a total of nine schemes. Using each scheme fifty sets of positive and unlabelled examples were obtained, making a total of 450 unlabelled learning problems.

5.1.4.2 Nursery datasets v.2

In the second version of the datasets three cardinalities for the set of positive examples (100, 200 and 300) are combined with six ratios of positive cases in $\mathcal{D}_u$ (0.01, 0.1, 0.2, 0.3, 0.4 and 0.5), yielding a total of eighteen schemes. The cardinality of $\mathcal{D}_u$ was 5,000 in all the schemes. Following each scheme 100 datasets were constructed, resulting in a total of 1,800 positive unlabelled learning problems.

5.2 Synthetic datasets

In part of the experimentation it is convenient to have positive unlabelled learning problems with some specific features. For instance, in the evaluation of the algorithms that induce TAN models we are interested in ensuring that there are conditional dependencies between the predicting variables. In the datasets used to test our proposal of adaptation of the CFS algorithm, we want to know which features are relevant, which are not and, also, which attributes are redundant. Finally, in the experimentation regarding the evaluation of clasiffiers in absence of negative examples we need to know the original distribution that generated the data in order to be able to compute the actual value of the performance measures.

We have created three sets of synthetic positive unlabelled learning problems, one for each of the problems mentioned above: TAN based datasets, FSS datasets and classifier evaluation datasets.

5.2.1 Datasets sampled from TAN models

As part of the contribution of this dissertation we have proposed the extension of the concept behind PNB to the induction of more complex networks.

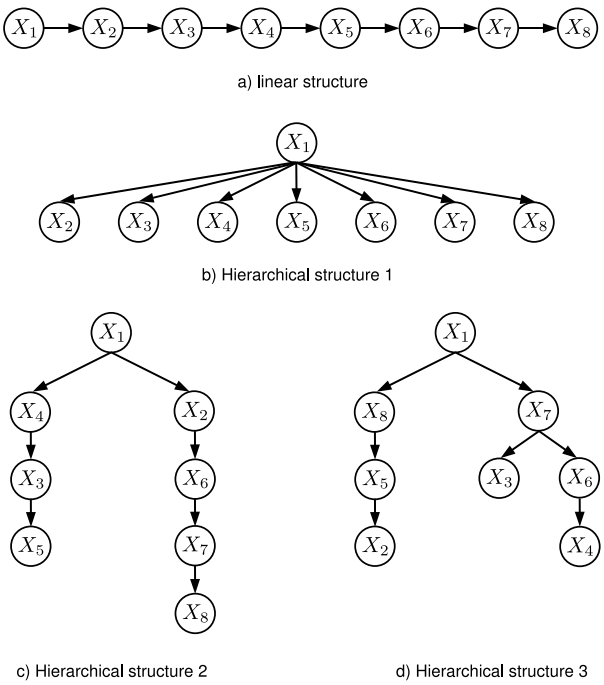


Fig. 5.1. Dependency structures used to build the TAN models from where synthetic datasets used in the evaluation of positive Bayesian network classifiers have been sampled. The class variable has been removed for clarity.

In particular, we have proposed two algorithms that are able to learn TAN models from positive and unlabelled examples. These algorithms learn models that take into account (to a certain degree) the conditional dependencies that may be between predicting variables.

In order to experimentally evaluate these algorithms it would be interesting to have some positive unlabelled learning problems where we know that there are such conditional dependencies. To ensure this point we have obtained sets of problems sampled from TAN models created based on the four topologies shown in Figure 5.1.

In order to increase the variability of the resulting datasets, five different sets of (random) parameters were assigned to each topology. Thus, for each topology we have five TAN models from where the datasets have been sampled.

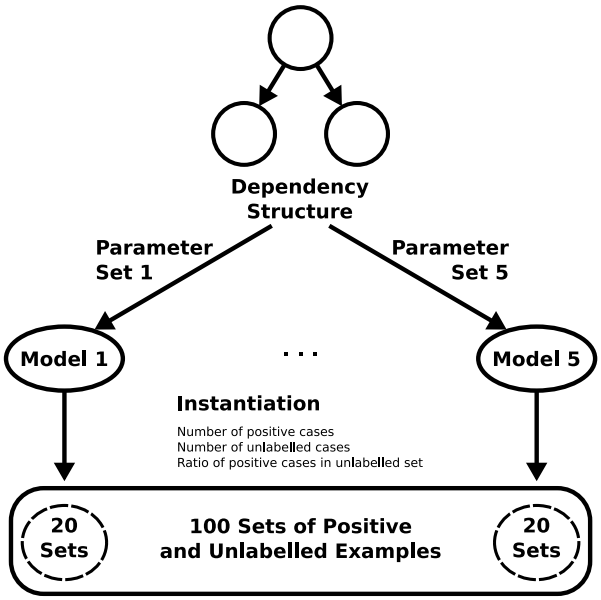
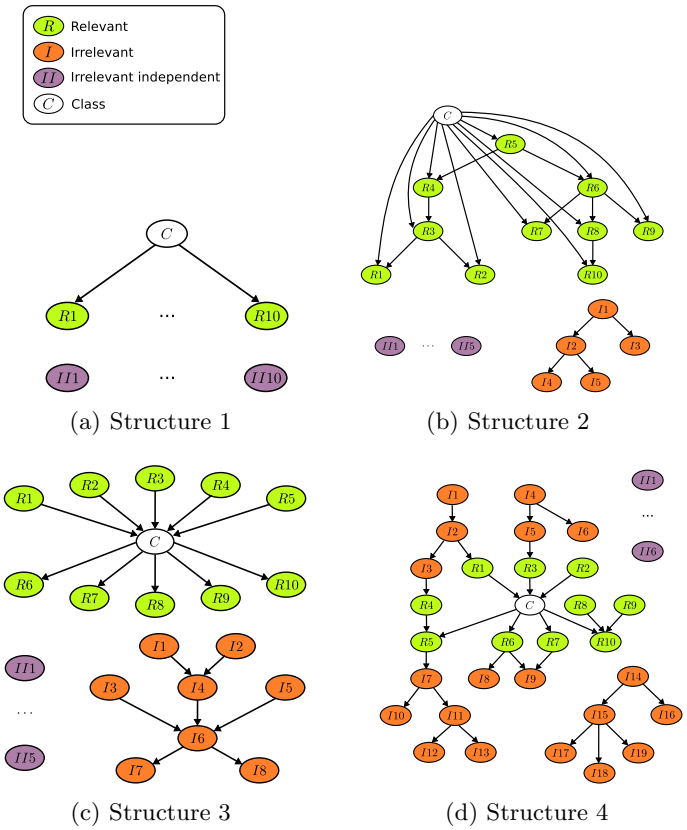


Fig. 5.2. Process followed to obtain the positive unlabelled learning problems based on TAN models.

The models defined for each topology have been sampled using three different cardinalities of $\mathcal{D}_p$ (100, 1,000 and 10,000), three cardinalities of $\mathcal{D}_u$ (1,000, 10,000 and 100,000) and six ratios of positive cases in $\mathcal{D}_u$ (0.01, 0.1, 0.2, 0.3, 0.4 and 0.5). For a given topology, twenty positive unlabelled learning problems were drawn from each of the five models (sets of parameters), resulting in a total of 100 sets of positive and unlabelled cases for every combination of topology, cardinality of $\mathcal{D}_p$, cardinality of $\mathcal{D}_u$ and ratio of positive cases in $\mathcal{D}_u$. Thus, there is a total of 21,600 positive unlabelled learning problems sampled from TAN models. Figure 5.2 shows a scheme of the sampling process.

5.2.2 Feature subset selection datasets

In Chapter 9 the adaptation of the CFS algorithm (Hall and Smith, 1997) to the positive unlabelled learning context is introduced. As the goal of this algorithm is to remove both redundant and irrelevant variables, we need datasets where we know which features are irrelevant and which are redundant.



(a) Structure 1 (b) Structure 2

(c) Structure 3 (d) Structure 4

Fig. 5.3. Bayesian network structures used to build the models from which synthetic data have been generated for the experimental evaluation of positive unlabelled feature subset selection algorithms.

The datasets described in this section have been created with this idea in mind. They have been sampled from Bayesian network models of increasing complexity (see Figure 5.3). Given that we know the structure of conditional dependencies between the variables we can label each feature as relevant (conditionally dependent of the class) or irrelevant (conditionally independent of the class). The redundant variables were added once the datasets were drawn from the probability distributions defined by the model.

All the structures shown in Figure 5.3 have four kinds of variables:

- The class variable, labelled as C

- Relevant variables, labelled as R followed by a number. These are the variables in the Markov blanket of C
- Irrelevant variables, labelled as I followed by a number. These are variables out of the Markov blanket but conditionally dependent on other variables
- Irrelevant independent variables, labelled as II followed by a number. These are variables conditionally independent of any other variable (including C)

The simplest model (see Figure 5.3-a) consists of a naive Bayes model (Minsky, 1961) and some independent variables. The second model (Figure 5.3-b) has a more complex structure, consisting of a tree augmented naive Bayes model (TAN) (Friedman et al., 1997)), five independent variables and a tree structure of five irrelevant features. The third model (Figure 5.3-c) includes a structure containing the relevant variables where the class variable is in the middle, five independent variables and a structure of eight irrelevant variables. The fourth structure (Figure 5.3-d), the most complex, includes a big structure containing the class, relevant variables and irrelevant variables, a small structure of irrelevant variables and five independent variables.

Redundant variables are added to the final datasets, once the models have been sampled, by just copying the relevant variables with a certain degree of noise σ (where σ is the percentage of instances where the variable was incorrectly copied). The redundant variables are represented by the name of the variable copied followed by R and the degree of redundancy (100-σ). For instance, the redundant variable obtained copying $R5$ with a noise degree of 20% is denoted as $R5R80$. Redundant variables of all the relevant variables with noise degrees of 10%, 20% and 30% were added to all the positive unlabelled learning problems.

Each of the models (with the structures shown in Figure 5.3 and random parameters) were sampled to obtain positive unlabelled learning problems. From each model, 100 positive sets of cardinalities 10, 25, 50, 100, 250, 500, 750 and 1,000 were drawn. Unlabelled sets of size 1,000 and 10,000 with ratios of positive cases of 0.01, 0.15, 0.25, 0.35 and 0.75 were also sampled from the probability distributions (ten from each distribution). As a result, for a particular scheme we have 1,000 sets of positive and unlabelled instances (the combination of the 100 positive sets with the ten unlabelled sets). The total number of positive unlabelled learning problems created for the experimental evaluation of the feature subset selection algorithms was 320,000.

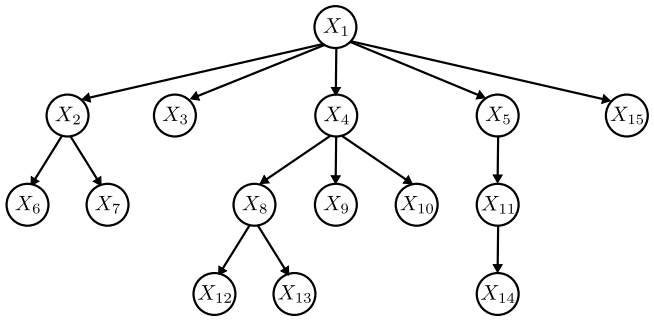


Fig. 5.4. Structure of the TAN models used to generate the original probability distributions from where synthetic datasets used in the experimental evaluation of the performance metric have been sampled.

5.2.3 Classifier evaluation datasets

The evaluation of classifiers in absence of negative examples is tackled in Chapter 8, which is dedicated to the wrapper positive Bayesian network classifiers. In this chapter we show how the F measure can be estimated in positive unlabelled learning problems and propose a metric that can be used to select the value of the p parameter in PNB (see Section 7.1) that maximises the F measure. In order to test this metric, we need to obtain the actual value of the F measure and, thus, we need to know the original probability distribution that generated the data. As we will see later, the equations presented in Chapter 8 assume that a particular instance belongs to a single class (in the most general framework a particular instance belongs to any possible class with different probabilities). Given this simplification, our original (empirical) probability distributions are described by a dataset containing all the possible instances along with their corresponding label.

The databases used in this section to generate the positive unlabelled learning problems contain instances characterised by fifteen binary variables, resulting in a total of 32,678 possible instances. All the instances have to be labelled in order to create the original distribution. The labels were obtained using naive Bayes and TAN models. Six different models were used to label the 32,678 instances, three based on a naive Bayes structure and three based on a TAN structure (Figure 5.4). As a result we get six complete databases from where the positive unlabelled learning problems were sampled (in the same way explained in Section 5.1).

Once the complete datasets are labelled we can estimate the apriori probability of the positive class. The aprioris in the databases labelled by naive Bayes models were 0.19, 0.28 and 0.45 and the aprioris in the databases labelled by TAN models were 0.15, 0.30 and 0.50. Thus, our original databases are the complete set of instances labelled by naive Bayes models with $p \in \{0.19, 0.28, 0.45\}$ and by TAN models with $p \in \{0.15, 0.30, 0.50\}$.

Starting from each original empirical distribution, sets of positive and unlabelled learning problems were obtained with cardinalities of the positive set of 0.1%, 0.5%, 1%, 5% and 10% of the total number of instances (i. e., 33, 164, 328, 1,638 and 3,277 respectively). In all the cases the amount of unlabelled instances was set to 10% of the total number of instances and the ratios of positive cases in $\mathcal{D}_u$ were not forced (the unlabelled instances were obtained by randomly sampling the original distributions, without considering their class). Using each scheme, 100 different positive unlabelled learning problems were obtained, resulting in a total of 3,000 sets of positive and unlabelled instances.

			0.3; 0.4; 0.5		
Letter Recognition v.1	16	100; 200; 300	0.048; 0.13; 0.2	50	1,350
Letter Recognition v.2	16	100; 200; 300	0.01;0.1;0.2 0.3; 0.4; 0.5	100	5,400
Nursery v.1	8	250; 500; 1,000	0.011; 0.075; 0.25	50	450
Nursery v.2	8	100; 200; 300	0.01;0.1;0.2 0.3; 0.4; 0.5	100	1,800
TAN based datasets	8	100; 1,000; 10,000	0.01; 0.1; 0.2 0.3; 0.4; 0.5	100	21,600
FSS datasets	40; 50 53; 64	10; 25; 50; 100; 250 500; 750; 1,000	0.01; 0.15; 0.25 0.35; 0.75	1,000	320,000
C. evaluation datasets	15	33; 164; 328 1,638; 3,277	0.19; 0.28; 0.45 0.15; 0.30; 0.50	100	3,000

Table 5.1. Summary of the positive unlabelled learning datasets. The first column indicates the set of problems. The second column represents the cardinalities of the set of positive cases $\mathcal{D}_p$ used in the problem sampling. The third column shows the list of ratios of positive cases hidden in $\mathcal{D}_u$. The fourth column indicates the number of positive unlabelled learning problems sampled using each scheme. The final column represents the total number of problems in the set. Globally, 367,100 sets of positive and unlabelled examples have been used in the experimental part of the dissertation.

6

The Divergence Convergence Division Algorithm

One of the main applications of positive unlabelled learning algorithms is the recovery of positive examples from big datasets of unlabelled instances. In a typical situation, we have a database of objects that belong to different types (for example, papers related to different topics), and we have a sample of objects from a particular kind that are interesting for us (papers about, for instance, positive unlabelled learning). The goal of positive unlabelled learning algorithms is to obtain a classifier that, given the attributes of an object, predicts whether it belongs to the class from which we have a sample or not. In the example of the papers, we want to learn a classifier that, given a paper from the database, predicts whether it is a positive unlabelled learning paper or not.

In a scenario such as this it is quite reasonable to assume that, in the database of unlabelled objects, the number of objects belonging to the class we are interested in is very low compared to the number of those that do not belong to that class. This can be clearly seen in the example of papers. The number of positive unlabelled learning papers in the database is very low compared to the number of papers on all the other topics. Thus, we can assume that the ratio of positive cases in the set of unlabelled instances is lower than 0.5. This assumption is the starting point of the divergence convergence division algorithm (DCDiv).

6.1 DCDiv algorithm

Let $\mathcal{D}$ be a dataset of positive and unlabelled examples. Let $\mathcal{D}_p$ denote the set of positive examples and $\mathcal{D}_u$ the set of unlabelled examples with $\mathcal{D} = \mathcal{D}_p \cup \mathcal{D}_u$. Suppose we draw v random samples with replacement from $\mathcal{D}_u$, $\mathcal{N} = \{(\boldsymbol{x}^{(1)}, ?), \ldots, (\boldsymbol{x}^{(v)}, ?)\}$. Given the assumption that the number of positive cases in the set of unlabelled instances is much lower than the number of negative examples, this random sample $\mathcal{N}$ can be considered as a noisy set of negative instances. Indeed, the amount of noise in this set (the positive cases that we may have picked up from the set of unlabelled instances) is directly proportional to the ratio of positive cases in the set of unlabelled instances.

If we define the noise ν in a set of negative instances as the probability of randomly selecting an instance that is not negative, the noise in $\mathcal{N}$ is the a priori probability of the positive class in the empirical probability distribution defined by the set of unlabelled instances:

$$\nu(\mathcal{N}) = P_{\mathcal{D}_u}(C = 1)$$

thus, the higher the ratio of positive cases in $\mathcal{D}_u$ the more noise we have in the random sample $\mathcal{N}$.

If we induce a classification model M using as training set $\mathcal{D}_p \cup \mathcal{N}$, where all the examples in $\mathcal{N}$ have been assigned to the negative class, we could use this model to estimate, for every instance $\boldsymbol{x} \in \mathcal{D}_u$, the probabilities $P_M(C = 1|\boldsymbol{x})$ and $P_M(C = 0|\boldsymbol{x})$. These estimations will be affected by the noise we have introduced in the learning process ($\nu(\mathcal{N})$). The former probabilities can be used to identify whether $\boldsymbol{x}$ is a positive or a negative example. If $P_M(C = 1|\boldsymbol{x}) > P_M(C = 0|\boldsymbol{x})$ then we say that $\boldsymbol{x}$ is a positive example. If not, we say that it is a negative example. The same decision can be made looking at the odds ratio:

$$F(\boldsymbol{x}) = \frac{P_M(C = 0|\boldsymbol{x})}{P_M(C = 1|\boldsymbol{x})} \tag{6.1}$$

If this ratio is greater than one we say that $\boldsymbol{x}$ is negative while if it is lower or equal than one we say that it is positive.

This way of identifying which cases in the set of unlabelled instances are positive and which are negative is quite unstable, as it depends very much on

the noise in the sample $\mathcal{N}$. One way to cope with the instability of classification functions is the model averaging. A good example of how the model averaging can increase the stability of a classification function is the random forest algorithm (Breiman, 2001). This algorithm is based on the bagging process (Breiman, 1996) which basically consists of combining classifiers (of the same type) trained on bootstrap replicates (Efron, 1983) of the training set. When the bagging concept is applied to decision trees we have the random forest algorithm.

Suppose that we draw m samples $\{\mathcal{N}_1, \ldots, \mathcal{N}_m\}$ and, using each of these samples as a noisy set of negative instances, we build m classification models $\{M_1, \ldots, M_m\}$. Each of these models can be used to compute the odds ratio (Equation (6.1)). In order to combine the predictions of each model we can take advantage of the fact that this ratio is greater than one for the negative cases and lower than one for the positive cases. If we compute the product of these odds ratios, we have:

$$\pi_m(\boldsymbol{x}) = \prod_{s=1}^{m} F_s(\boldsymbol{x}) = \prod_{s=1}^{m} \frac{P_{M_s}(C=0|\boldsymbol{x})}{P_{M_s}(C=1|\boldsymbol{x})} \tag{6.2}$$

When $\boldsymbol{x}$ is a negative example, as m tends to infinity the product should diverge to infinity but, when $\boldsymbol{x}$ is positive, as m tends to infinity the product should converge to 0.

DCDiv exploits this differential behaviour to separate the positive and the negative cases based on the product of odds ratios. In order to distinguish between positive and negative examples, we have to set a threshold t. Those instances with a product value higher than the threshold are considered as negative and those with a product value lower or equal to the threshold are considered as positive.

In order to set the threshold, the concept of 'spy' cases is used. This concept was introduced by Liu et al. (2002) and consists of selecting a set of known positive cases (the spy cases, $\mathcal{S}$) and introducing them into the set of unlabelled instances. Thus, after the inclusion of spy cases in $\mathcal{D}_u$, new sets of positive and unlabelled examples are obtained ($\mathcal{D}_p^{'} = \mathcal{D}_p \backslash \mathcal{S}$ and $\mathcal{D}_u^{'} = \mathcal{D}_u \cup \mathcal{S}$). Figure 6.1 shows a scheme of this process.

The idea is that these spy cases will behave in the same way the positive cases hidden in $\mathcal{D}_u$ do, and, thus, we can use them to estimate the recall

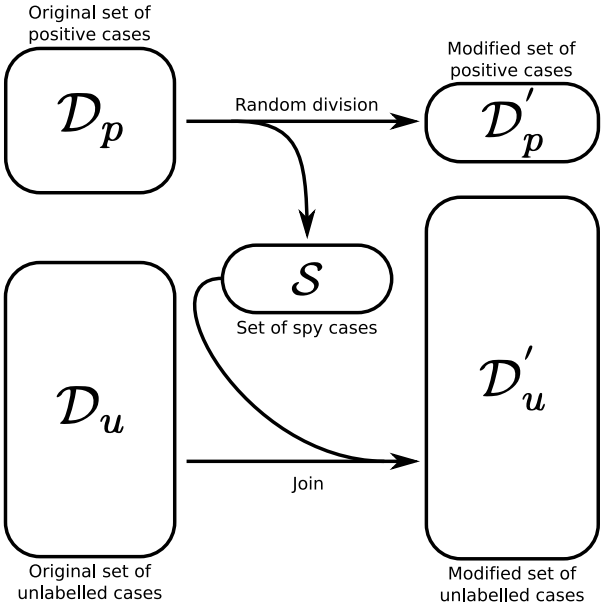


Fig. 6.1. Scheme of the process of insertion of spy cases in the set of unlabelled instances.

we get setting the threshold at a given value. Given a predefined recall (as a parameter of the algorithm) we set the threshold in the smallest value such that we get the desired recall[1] .

The algorithm works in an iterative way. There is an initialisation step where the spy cases are introduced into the set of unlabelled instances. Then, at each step of the iteration, a random sample $\mathcal{N}_s$ is drawn from $\mathcal{D}'_u$ and is used as a set of negative examples to learn a model. Using this model, the odds ratio of each instance in $\mathcal{D}'_u \setminus \mathcal{N}_s$ is computed and added to its product[2] . At each iteration the threshold is set at the minimum value that ensures the recall set in the parameters of the algorithm.

[1] We want to classify only the positive instances as positive and, thus, for a given recall, we have to classify the minimum number of instances as positive, and we obtain this using the smallest threshold that ensures the desired recall

[2] To avoid overfitting problems, the odds ratio of the instances used in the training of the model at the current iteration is not included in the product. In other words, if $\boldsymbol{x} \in \mathcal{N}_s$ then $F_s(\boldsymbol{x}) = 1$

The threshold is set based on the products of the odds ratios of the spy cases. Therefore, as it is the product of a positive case, we expect it to converge to 0. The DCDiv algorithm uses this feature as the stop criterion. The algorithm halts when the threshold is 0^3 . As it can happen that the threshold does not converge to 0, a maximum number of iterations has to be set, halting the algorithm whether when the threshold converges to 0 or when the maximum number of iterations is reached. Figure 6.1 shows the pseudo code of the DCDiv algorithm.

As we have said, DCDiv is a model averaging process where bootstrap samples from the set of unlabelled instances are obtained and the predictions of the models are combined by means of Equation (6.2). If we take the logarithm of this equation we have that:

$$ln(\pi_m(\boldsymbol{x})) = \sum_{s=1}^{m}[ln(P_{M_s}(C=0|\boldsymbol{x})) - ln(P_{M_s}(C=1|\boldsymbol{x}))]$$

This is the sum of the logit function of $P(C = 0|\boldsymbol{x})$, used in logistic regression. This Equation can be rewritten as:

$$ln(\pi_m(\boldsymbol{x})) = \sum_{s=1}^{m} w_s(\boldsymbol{x})$$
$$w_s(\boldsymbol{x}) = ln(P_{M_s}(C=0|\boldsymbol{x})) - ln(P_{M_s}(C=1|\boldsymbol{x}))$$

We can see this expression as a weighted voting scheme where the weights depend on the logarithm of the posterior probabilities. This resembles bagging predictors (Breiman, 1996), but with some differences. In DCDiv, bootstrap samples are only drawn from the set of unlabelled instances, and the voting is weighted by a function that depends on the logarithm of the posterior probabilities, rather than bagging's simple voting. The main difference with bagging is that, in DCDiv, the threshold is set dynamically at each step, instead of using a prefixed threshold.

6.1.1 Setting the parameters of the DCDiv algorithm

As can be seen in Figure 6.1, there are four parameters that have to be set. The ratio of positive cases that are going to be used as spy cases will depend on

[3] Obviously, the threshold cannot actually reach the mathematical 0. Reaching the 0 is interpreted as reaching a certain value that we consider small enough

DCDiv algorithm

input:
  $\mathcal{D}_u$:= set of unlabelled examples
  $\mathcal{D}_p$:= set of positive examples
  r := recall
  σ := ratio of cases in $\mathcal{D}_p$ used as spies
  $Limit$:= maximum number of iterations allowed
  $Zero$:= threshold for the convergence. Values lower than $Zero$ are considered to be 0

Initialisation
  Randomly divide $\mathcal{D}_p$ into $\mathcal{D}_p^{'}$ and $\mathcal{S}$, such that the number of cases in $\mathcal{S}$ is $\sigma \cdot |\mathcal{D}_p|$
  Create $\mathcal{D}_u^{'}$ by joining $\mathcal{S}$ and $\mathcal{D}_u$
  for each case $\boldsymbol{x}$ in $\mathcal{D}_u$
    $\pi_0(\boldsymbol{x}) := 1$
  rof
  $s := 1$

Iterations
  do
    Select randomly with replacement $|\mathcal{D}_p|$ cases from $\mathcal{D}_u^{'}$ to construct $\mathcal{N}_s$
    Build a model M_s using $\mathcal{D}_p^{'}$ and $\mathcal{N}_s$ as positive and negative examples respectively
    for each case $\boldsymbol{x} \in \mathcal{D}_u^{'} \setminus \mathcal{N}_s$
      $\pi_s(\boldsymbol{x}) = \pi_{s-1}(\boldsymbol{x}) \cdot F_s(\boldsymbol{x})$
    rof
    Set t at the lowest value such that, for r% of cases in $\mathcal{S}$, $\pi_s(\boldsymbol{x}) < t$
    $s = s + 1$
  while $t > Zero$ **and** $s \leq Limit$

Labelling
  for each $\boldsymbol{x}$ in $\mathcal{D}_u$
    if $\pi_s(\boldsymbol{x}) > t$
    then
      $\boldsymbol{x}$ is labelled as negative
    else
      $\boldsymbol{x}$ is labelled as positive
    fi
  rof

Alg. 6.1: Pseudo code of the DCDiv algorithm.

the amount of positive cases available. The more spy cases, the more confident we can be about how the threshold is set (the more confident we are on the estimation of the recall), but the less positive cases that we will have in the learning of each classifier.

The recall is probably the most difficult parameter to set. We want to have the highest possible recall but, if the recall is set at a very high value, this can lead to the non-convergence of the threshold. For instance, if we set the recall at one, this means that the threshold will be set, at each step s, at the highest value of $\pi_s(\boldsymbol{x})$ with $\boldsymbol{x} \in \mathcal{S}$. If any of the spy cases is very close to the boundary between the positive and the negative cases, or if any of the spy cases is actually negative (we can have some noise in the set of known positive cases), then its product can diverge to infinity and, thus, so will the threshold. The criterion we have used to set the recall is to use the highest value that allows the threshold to converge to 0.

The factors that influence in the time the threshold needs to converge are the nature of the problem, the base classifier and the recall used to set the threshold. There is a direct relation between the threshold and the recall. If we increase the recall, the threshold that we will have at each step will also increase. Thus, setting the recall in high values leads to slower convergences of the threshold. Actually, using the criterion proposed to set the recall will lead to the slowest convergence of the threshold. If the estimations of the posterior probabilities provided by the base classifier are very similar for the positive and the negative class, then the odds ratio will be close to one and the convergence of the threshold will be slow. The time the algorithm needs to finish can be controlled by the zero (the higher the zero the faster the algorithm will stop). The maximum number of iterations sets an upper bound to the running time of the algorithm.

The final parameter we have to set is the base classifier. Due to the nature of the algorithm, the posterior probabilities of the positive and negative class have to be computed. Thus, the only requirement for the classification paradigm used as base classifier is that it allows to estimate these probabilities.

6.2 Experimental evaluation of the DCDiv algorithm

We have tested the DCDiv in datasets where the absence of negative examples has been simulated. For this purpose we have used ACCDON datasets v.1,

Letter Recognition - F measure

			LRd		LRp		LRu	
	$N_{\mathcal{D}_p}$	rat.	PNB	DCDiv	PNB	DCDiv	PNB	DCDiv
$p = 0.25$	100	0.05	36.80	**53.79**	43.42	**68.19**	45.93	**69.71**
	100	0.13	66.07	64.62	74.01	70.65	72.88	74.40
	100	0.20	**73.93**	66.30	**81.67**	70.56	**78.99**	73.71
	200	0.05	37.05	**67.09**	44.98	**73.01**	47.96	**74.83**
	200	0.13	66.67	**69.35**	74.86	73.14	75.10	76.66
	200	0.20	**76.70**	71.82	**83.47**	73.45	**81.62**	75.33
	300	0.05	36.49	**70.50**	46.11	**75.73**	49.23	**77.79**
	300	0.13	66.27	**73.02**	75.92	75.04	76.60	78.01
	300	0.20	76.68	74.60	**84.04**	76.22	**82.92**	76.61
Actual p	100	0.05	**58.26**	53.79	**72.18**	68.19	**74.45**	69.71
	100	0.13	**69.68**	64.62	**80.14**	70.65	**78.39**	74.40
	100	0.20	**74.08**	66.30	**82.53**	70.56	**79.54**	73.71
	200	0.05	62.48	**67.09**	73.32	73.01	76.59	74.83
	200	0.13	**72.71**	69.35	**81.61**	73.14	**80.78**	76.66
	200	0.20	**77.80**	71.82	**84.54**	73.45	**82.41**	75.33
	300	0.05	61.64	**70.50**	73.07	**75.73**	78.31	77.79
	300	0.13	72.85	73.02	**82.25**	75.04	**82.12**	78.01
	300	0.20	**77.96**	74.60	**85.21**	76.22	**83.61**	76.61

Table 6.1. Extract of the results of the comparison between PNB and DCDiv. The results correspond to the datasets obtained from the Letter Recognition database; LRd, LRp and LRu stand for Letter Recognition D, P and U respectively. The table shows the comparison of the F measure obtained with DCDiv and PNB setting the p parameter at 0.25 (rows labelled as '$p = 0.25$') and at the actual value (rows labelled as 'Actual p'). The column labelled as '$N_{\mathcal{D}_p}$' indicates the cardinality of $\mathcal{D}_p$ and the column labelled as 'rat.' denotes the ratio of positive cases in $\mathcal{D}_u$. All the values shown in this table are the average of the results obtained in the fifty datasets built according to each scheme (dataset, cardinality of $\mathcal{D}_p$ and ratio of positive cases in $\mathcal{D}_u$). The bold font indicates the winner in the comparisons where significant differences at a significance level of 1% were found.

Letter Recognition datasets v.1 and Nursery datasets v.1 described in Chapter 5.

The results obtained by the DCDiv algorithm have been compared with the results obtained by PNB setting the p parameter at 0.25, 0.5 and the

actual value (the description of this algorithm can be found in Section 7.1)[4] . The statistical significance of the differences between the results obtained by the two algorithms has been assessed running Wilcoxon paired tests on the results obtained in each of the fifty problems created using each scheme.

Each algorithm has been applied to the sets of positive and unlabelled examples described in Section 5.1. The output of the DCDiv algorithms is a classification of the examples in the set of unlabelled instances. In the case of the PNB algorithm, the model learnt has been used to obtain the classification of the instances in the set of unlabelled instances. Once we have the set of unlabelled instances classified by each algorithm we can obtain the confusion matrix (as we actually know the labels of the 'unlabelled' instances). Using the confusion matrix the F measure and the accuracy were obtained (see Section 2.2 for the definitions of the performance measures).

This way of estimating the performance measures may look non-honest as we are using the unlabelled examples in the training and the evaluation of the classifiers, but it is important to take into account that during the training the information about the class of these instances is not used at all.

TAN models (Friedman et al., 1997) have been used as base classifier for the DCDiv algorithm. The ratio of positive examples used as spy cases has been set at 0.1 and the number of iterations limited to 1,000. The threshold was considered 0 when it reached the computational 0 (10^{-324} in our case).

Tables 6.1, 6.2 and 6.3 show the results obtained in the comparison of the two algorithms in terms of the F measure. The complete set of results can be consulted in Appendix A.

Table 6.1 contains the results of the comparison of the F measure obtained by DCDiv and PNB in problems sampled from the Letter Recognition database. In most of the problems created using letter D as the positive class, DCDiv outperforms the PNB when the p parameter is set to 0.25. In some datasets it even improves or there are no significant differences with PNB when the actual value of p is used (note that the actual value of p is not known in a real-life problem). In datasets where P or U was used as the positive class, PNB ($p = 0.25$) only outperforms DCDiv when the ratio of positive

[4] The actual value of the a priori probability of the positive class is unknown in real-life based problems, but it can be estimated as the ratio of positive cases in the set of unlabelled instances. Throughout this dissertation, in the context of the experimental evaluation, the term 'actual p' is used to refer to this ratio

Nursery - F measure

		$p = 0.50$		$p = 0.25$		Actual p	
$N_{\mathcal{D}_p}$	rat.	PNB	DCDiv	PNB	DCDiv	PNB	DCDiv
250	0.01	03.94	**55.34**	20.80	**55.34**	00.00	**55.34**
250	0.08	25.45	**71.76**	69.08	**71.76**	20.84	**71.76**
250	0.25	65.85	**73.96**	74.72	73.96	75.16	73.96
500	0.01	04.12	**66.26**	21.37	**66.26**	00.00	**66.26**
500	0.08	26.76	**71.97**	70.36	71.97	21.18	**71.97**
500	0.25	65.33	**72.53**	**75.22**	72.53	**75.68**	72.53
1000	0.01	04.37	**65.62**	21.90	**65.62**	00.00	**65.62**
1000	0.08	27.48	**71.10**	71.13	71.10	20.92	**71.10**
1000	0.25	64.36	**69.94**	**75.88**	69.94	**76.33**	69.94

Table 6.2. Extract of the results obtained in the comparison between PNB and DCDiv. The results correspond to the datasets obtained from the Nursery database. The table shows the comparison of the F measure obtained with DCDiv and PNB setting the p parameter at 0.50 (labelled as '$p = 0.50$'), at 0.25 (labelled as '$p = 0.25$') and at the actual value (rows labelled as 'Actual p'). The column labelled as '$N_{\mathcal{D}_p}$' indicates the cardinality of $\mathcal{D}_p$ and the column labelled as 'rat.' denotes the ratio of positive cases in $\mathcal{D}_u$. All the figures shown in this table are the average of the results obtained in the fifty datasets built according to each scheme (dataset, cardinality of $\mathcal{D}_p$ and ratio of positive cases in $\mathcal{D}_u$). The bold font indicates the winner in the comparisons where significant differences at a significance level of 1% were found.

cases in $\mathcal{D}_u$ is 0.20 (this makes sense as we are setting the parameter in the algorithm very close to its real value). In the comparison with PNB, when p is set at 0.50 DCDiv clearly outperforms PNB in all the datasets (results not shown in the table).

Table 6.2 shows the comparison between the results obtained by DCDiv and PNB in Nursery based datasets. In the first two columns we can see how DCDiv is clearly better than PNB when p is set at 0.50. When p is set at 0.25 PNB only outperforms DCDiv in two of the nine schemes used to create the positive unlabelled learning problems. The last two columns correspond to the comparison when p was set at the actual value. As can be seen in the table, contrary to what is observed in the rest of the datasets, DCDiv outperforms PNB in most of the experiments. This is due to the bad performance of the PNB in these datasets, probably due to setting the p parameter at a very low

value (when it was set at 0.01 none of the unlabelled instances was classified as positive, leading to a F measure of 0).

Table 6.3 shows the result of the comparison between DCDiv and PNB in datasets obtained from the ACCDON database. The results obtained in this comparison are very similar to those obtained in datasets based on Letter Recognition. When p was set at 0.25 PNB only outperforms DCDiv when the actual a priori probability is set at 0.20 (except in three sets where it also wins when the actual probability was 0.13). When the actual value is used PNB performs better than DCDiv in most of the datasets (in some datasets there are no significant differences). As in the rest of the experiments, DCDiv is significantly better than PNB when p was set at 0.50 (results not shown in the table).

6.3 Conclusions

In this chapter we have presented a new model averaging algorithm (DCDiv) that can be used in positive unlabelled learning problems to obtain a classification of the unlabelled instances. We have compared the performance of DCDiv and PNB, one of the state of the art algorithms in the positive unlabelled learning context. We have carried out the comparison in simulated problems based on data of very different nature. From the comparison we can conclude that DCDiv is a competitive algorithm in terms of performance. It clearly improves PNB when p is set at 0.50 and, in many of the problems, when it is set at 0.25. Even when the true value of p is used DCDiv outperforms PNB or no significant differences are found in some datasets. This is important because the actual value of p is generally unknown in real-life problems and, thus, it is not possible to use it in the PNB algorithm.

If we look at the results we can see that DCDiv has a good performance (compared to the PNB algorithm) specially when the actual value of p is low. This has to do with the assumption that we made at the beginning of this chapter. We are supposing that the number of positive cases in $\mathcal{D}_u$ is much lower than the number of negative examples. The lower the noise (which is exactly, as we have defined it, the ratio of positive cases in $\mathcal{D}_u$), the better the individual predictions will be and, thus, the better the algorithm will separate the positive and negative instances. Conversely, if we set the p parameter in the PNB algorithm at a very low value the final model will

classify very few instances as positive. This can be clearly seen in Table 6.2 in those problems where the actual value of p is 0.01. The F measure obtained with the classifier trained with $p = 0.01$ is 0 because the classifier classifies all the instances as negative and, thus, the recall is 0. Thus, if we know that, in a particular problem, the ratio of positive cases to recover from the set of unlabelled instances is very low, the DCDiv algorithm is a good option because the noise is very low, while PNB will probably perform poorly because it will recover very few positive examples.

Nevertheless, DCDiv has some problems. The main issue is that it is a very time-consuming algorithm, as it has to build a model in each iteration. In any of the presented experiments where the algorithm reached the maximum number of iterations, 1,000 TAN models were learnt, compared to a single naive Bayes model in PNB. The other main drawback of the DCDiv algorithm is that, at the end, we have a classification of the unlabelled instances, and not a classification function. An interesting line for the future would be how to modify the algorithm to obtain one single classifier as output.

ACCDON - F measure

		$N_{\mathcal{D}_p}$	rat.	Coding PNB	Coding DCDiv	Intron PNB	Intron DCDiv	Mixed PNB	Mixed DCDiv
Acceptor Sites	$p = 0.25$	100	0.05	43.30	**66.81**	38.18	**51.47**	41.02	**56.82**
		100	0.13	74.70	74.77	66.07	64.97	69.90	67.91
		100	0.20	**83.95**	75.43	**75.19**	65.37	**78.98**	70.89
		200	0.05	44.65	**66.44**	38.53	**53.95**	41.53	**58.31**
		200	0.13	75.69	76.21	67.03	66.91	70.69	69.52
		200	0.20	**84.93**	73.89	**76.54**	71.21	**79.86**	70.42
		300	0.05	44.36	**68.81**	38.29	**54.55**	41.58	**59.25**
		300	0.13	**75.59**	71.98	66.87	66.05	**70.88**	68.02
		300	0.20	**85.26**	77.58	**77.12**	69.31	**80.32**	69.97
	Actual p	100	0.05	**71.77**	66.81	**54.67**	51.47	**59.90**	56.82
		100	0.13	**80.87**	74.77	**69.31**	64.97	**73.93**	67.91
		100	0.20	**84.75**	75.43	**75.46**	65.37	**79.29**	70.89
		200	0.05	**72.23**	66.44	**56.37**	53.95	**60.70**	58.31
		200	0.13	**82.09**	76.21	**71.36**	66.91	**75.78**	69.52
		200	0.20	**85.67**	73.89	**77.02**	71.21	**80.47**	70.42
		300	0.05	**73.26**	68.81	**56.43**	54.55	**61.62**	59.25
		300	0.13	**81.94**	71.98	**71.45**	66.05	**75.88**	68.02
		300	0.20	**86.13**	77.58	**77.64**	69.31	**80.92**	69.97
Donor Sites	$p = 0.25$	100	0.05	39.23	**67.61**	51.33	**72.54**	45.21	**72.40**
		100	0.13	75.87	74.10	79.37	77.66	73.36	74.63
		100	0.20	**86.49**	75.18	**88.91**	77.08	**87.41**	74.05
		200	0.05	39.30	**67.43**	51.96	**75.58**	45.42	**71.84**
		200	0.13	76.48	75.57	79.85	78.27	73.36	**78.28**
		200	0.20	**86.96**	72.91	**88.83**	80.02	**88.22**	77.67
		300	0.05	39.73	**70.04**	52.10	**76.23**	45.74	**73.30**
		300	0.13	**76.47**	72.73	80.18	80.22	73.42	**78.00**
		300	0.20	**87.45**	74.26	**88.65**	80.90	**88.57**	78.53
	Actual p	100	0.05	67.86	67.61	**78.53**	72.54	73.01	72.40
		100	0.13	**81.49**	74.10	**87.32**	77.66	**83.78**	74.63
		100	0.20	**85.95**	75.18	**90.65**	77.08	**87.63**	74.05
		200	0.05	69.71	67.43	**79.00**	75.58	73.48	71.84
		200	0.13	**81.66**	75.57	**88.17**	78.27	**84.15**	78.28
		200	0.20	**86.39**	72.91	**90.85**	80.02	**88.51**	77.67
		300	0.05	69.27	70.04	**79.07**	76.23	73.99	73.30
		300	0.13	**81.84**	72.73	**88.13**	80.22	**84.40**	78.00
		300	0.20	**86.65**	74.26	**91.10**	80.90	**88.49**	78.53

Table 6.3. Extract of the results obtained in the comparison between PNB and DCDiv. The results correspond to the datasets obtained from the ACCDON database. The table shows the comparison of the F measure obtained with DCDiv and PNB setting the p parameter at 0.25 (rows labelled as '$p = 0.25$') and at the actual value (rows labelled as 'Actual p'). The column labelled as '$N_{\mathcal{D}_p}$' indicates the cardinality of $\mathcal{D}_p$ and the column labelled as 'rat.' denotes the ratio of positive cases in $\mathcal{D}_u$. All the figures shown in this table are the average of the results obtained in the fifty datasets built according to each scheme (dataset, cardinality of $\mathcal{D}_p$ and ratio of positive cases in $\mathcal{D}_u$). The bold font indicates the winner in the comparisons where significant differences at a significance level of 1% were found.

7

Positive Bayesian Network Classifiers

In previous chapters we have seen what the Bayesian network classifiers are, and that the most simple of this sort of classifier, the naive Bayes (Minsky, 1961), has already been adapted to the positive unlabelled learning context (Denis et al., 2002). What the authors propose in this work is a modification of the classical naive Bayes induction algorithm (see Section 3.3.1) that infers the model parameters from a set of positive and unlabelled examples. The basic idea presented in this paper can be used to modify other Bayesian network classifier induction algorithms to learn models without negative examples.

This chapter is devoted to the positive Bayesian network classifiers (PBC). In the first part of this chapter we will see the concept introduced in Denis et al. (2002) and how it can be extended to the induction of more complex Bayesian network classifiers. As an example, the adaptation of the tree augmented naive Bayes model induction algorithm (Friedman et al., 1997) will be presented. The second part is devoted to the modellisation of the uncertainty about the a priori probability of the positive class (as we will see later, setting this parameter is the key issue in the PBCs). The last part of the chapter shows the results obtained in the experimental evaluation of the proposed algorithms.

7.1 Positive naive Bayes

The positive naive Bayes algorithm (PNB) (Denis et al., 2002)) is able to learn naive Bayes models (Minsky, 1961) from a training dataset $\mathcal{D}$ of positive and unlabelled examples ($\mathcal{D} = \mathcal{D}_p \cup \mathcal{D}_u$). As we saw in Section 3.3.1, in a naive

Bayes model we assume that all the predicting variables are conditionally independent given the class variable. Given this assumption, the parameters we need to estimate are the conditional probabilities $P(x_{ij}|c)$ and the a priori probabilities of the class variable $P(c)$. When the class variable C takes only two values (0 and 1, negative and positive), the conditional probabilities can be divided into two groups, $P(x_{ij}|1)$ and $P(x_{ij}|0)$. In the classical supervised induction algorithm, the first group of parameters is estimated by maximum likelihood estimators from the set of positive examples, while the parameters in the second group are estimated from the set of negative examples. The a priori probability of the positive class ($P(1)$, that from now on we will represent as p for simplicity) is estimated from the whole dataset $\mathcal{D}$.

In the positive unlabelled learning framework we have no negative examples and, thus, the parameters related to the negative class cannot be estimated from $\mathcal{D}$. Given that[1] :

$$P(x_{ij}) = P(x_{ij}|1)p + P(x_{ij}|0)(1-p)$$

we have that $P(x_{ij}|0)$ can be estimated as:

$$P(x_{ij}|0) = \frac{P(x_{ij}) - P(x_{ij}|1)p}{1-p} \tag{7.1}$$

where $P(x_{ij})$ can be estimated from the unlabelled instances as it does not depend on the class[2] . The p parameter (the a priori probability of the positive class) cannot be estimated without negative examples and, thus, it has to be introduced by the user. Indeed, this is the main drawback of the PNB, as if we had the exact value of this a priori probability, the estimation of the parameters would be exactly the same as in the supervised algorithm.

$P(x_{ij})$ can be estimated from the unlabelled instances as:

$$P(x_{ij}) = \frac{N_{ij\mathcal{D}_u}}{N_{\mathcal{D}_u}}$$

[1] As in positive unlabelled learning problems there are only two classes, from now on we will assume that the class C is a binary random variable. 0 and 1 will represent the negative and the positive class respectively

[2] We assume that the unlabelled instances are a random sample of the probability distribution that originated the data where the labels of the instances are lost

where $N_{\mathcal{D}_u}$ is the number of unlabelled instances and $N_{ij\mathcal{D}_u}$ the number of unlabelled instances where $X_i = x_{ij}$. Replacing this probability in Equation (7.1), we have that:

$$P(x_{ij}|0) = \frac{N_{ij\mathcal{D}_u} - P(x_{ij}|1)pN_{\mathcal{D}_u}}{(1-p)N_{\mathcal{D}_u}}$$

The problem with this estimator is that it can be negative because, although $P(x_{ij}) \geq P(x_{ij}|1)p$, the bias in the estimation of these probabilities can lead to situations where the estimation of $P(x_{ij}|1)p$ is greater than the estimation of $P(x_{ij})$, yielding a negative estimation of $P(x_{ij}|0)$. To solve this problem, Denis et al. (2002) propose to replace the negative estimations by zero and then normalise the probabilities so that for each predicting variable X_i, $\sum_{j=1}^{r_i} P(x_{ij}|0) = 1$. We can define the normalisation factor Z_i for the predicting variable X_i as:

$$Z_i = \sum_{j=1}^{r_i} max\left(0; \frac{N_{ij\mathcal{D}_u} - P(x_{ij}|1)pN_{\mathcal{D}_u}}{(1-p)N_{\mathcal{D}_u}}\right)$$

After the normalisation, the estimator of the probabilities related with the negative class would be:

$$P(x_{ij}|0) = \frac{max(0; R_i(j))\frac{1}{Z_i}}{(1-p)N_{\mathcal{D}_u}}$$
$$R_i(j) = N_{ij\mathcal{D}_u} - P(x_{ij}|1)pN_{D_u}$$

Finally, if we take the Laplace correction into account, we have that:

$$P(x_{ij}|0) = \frac{1 + max(0; R_i(j))\frac{1}{Z_i}}{r_i + (1-p)N_{\mathcal{D}_u}} \tag{7.2}$$

verifying that $\sum_{j=1}^{r_i} P(x_{ij}|0) = 1$.

To summarise, the PNB algorithm estimates the parameters of a naive Bayes model as follows:

- $P(x_{ij}|1)$ is estimated from the positive examples
- $P(x_{ij}|0)$ is estimated using Equation (7.2).
- p is a parameter of the algorithm.

As we have pointed out, setting the p parameter is the main drawback of this algorithm as, in general, we do not know the a priori probability of the positive class (though some information can be available).

7.2 Positive tree augmented naive Bayes

In the previous section we have seen how the parameters of a two class naive Bayes model can be estimated from a set of positive and unlabelled examples. In this section, as part of the contributions of this dissertation, we will see how this idea can be extended to the induction of more complex Bayesian network classifiers. As an example, the extension to the TAN models (Friedman et al., 1997), that we have named positive tree augmented naive Bayes algorithm (PTAN), will be presented.

Unlike the naive Bayes model where the structure is fixed, the induction of TAN models implies a structural learning step where the tree structure of conditional dependencies between predicting variables is inferred (see Section 3.3.2). Thus, in order to learn TAN models from positive and unlabelled examples, we need to adapt not only the parametric learning as in the case of the PNB algorithm, but also the structural learning of the model.

The generalisation of the parametric learning introduced in this section can be applied to any Bayesian network classifier, provided that the class variable is parent of all the predicting variables. Regarding the structural learning, any other algorithm based on mutual information, such as the k-DB algorithm (Sahami, 1996), could also be adapted to the positive unlabelled learning context using the concepts introduced in this section.

7.2.1 Structural learning

In the original TAN model induction algorithm (Friedman et al., 1997), the structural learning step takes as input the dataset $\mathcal{D}$ and outputs a tree structure that represents the conditional dependencies between pairs of predicting variables. As we saw in Section 3.3.2, the induction of the tree structure is based on an adaptation of the Chow-Liu algorithm (Chow and Liu, 1968) proposed by Friedman et al. (1997). This algorithm is based on the conditional mutual information computed for each pair of predicting variables given the

class. The conditional mutual information between two variables X_i, X_k given the class C, $I(X_i, X_k|C)$, can be computed as:

$$\sum_{j=1}^{r_i}\sum_{l=1}^{r_k}\sum_{c=0}^{1} P(x_{ij}, x_{kl}, c)\frac{P(x_{ij}, x_{kl}|c)}{P(x_{ij}|c)P(x_{kl}|c)} =$$
$$\sum_{j=1}^{r_i}\sum_{l=1}^{r_k} P(x_{ij}, x_{kl}, 1)\frac{P(x_{ij}, x_{kl}|1)}{P(x_{ij}|1)P(x_{kl}|1)} +$$
$$\sum_{j=1}^{r_i}\sum_{l=1}^{r_k} P(x_{ij}, x_{kl}, 0)\frac{P(x_{ij}, x_{kl}|0)}{P(x_{ij}|0)P(x_{kl}|0)}$$

As we saw in the previous section, the probabilities conditioned to the positive class can be obtained by maximum likelihood estimators from the set of positive cases $\mathcal{D}_p$, but neither the probabilities related with the negative class nor $P(x_{ij}, x_{kl}, 1)$ can be estimated from $\mathcal{D}$. On one hand, we have that $P(x_{ij}, x_{kl}, 1)$ can be expressed as a function of the p parameter as:

$$P(x_{ij}, x_{kl}, 1) = P(x_{ij}, x_{kl}|1)p$$

On the other hand, we have that:

$$P(x_{ij}, x_{kl}) = P(x_{ij}, x_{kl}, 0) + P(x_{ij}, x_{kl}, 1)$$
$$P(x_{ij}, x_{kl}, 0) = P(x_{ij}, x_{kl}) - P(x_{ij}, x_{kl}, 1)$$

Regarding the probabilities conditioned to the negative class, $P(x_{ij}|0)$ and $P(x_{kl}|0)$ can be estimated by means of Equation (7.2). Equation (7.1) can be generalised as:

$$P(x_{ij}, x_{kl}|0) = \frac{P(x_{ij}, x_{kl}) - P(x_{ij}, x_{kl}|1)p}{1-p}$$
$$P(x_{ij}, x_{kl}) = \frac{N_{ijkl\mathcal{D}_u}}{N_{\mathcal{D}_u}}$$

where $N_{ijkl\mathcal{D}_u}$ is the number of unlabelled instances where $X_i = x_{ij}$ and $X_k = x_{kl}$. The estimator of $P(x_{ij}, x_{kl}|0)$ can be negative so we proceed as in the previous section replacing the negative estimations by zero and then

normalising. After the normalisation and applying the Laplace correction we have that $P(x_{ij}, x_{kl}|0)$ can be estimated as:

$$
\begin{aligned}
P(x_{ij}, x_{kl}|0) &= \frac{1 + max(0; R_{ik}(jl))\frac{1}{Z_{ik}}}{r_i r_k + (1-p)N_{\mathcal{D}_u}} \\
R_{ik}(jl) &= N_{ijkl\mathcal{D}_u} - P(x_{ij}, x_{kl}|1)pN_{\mathcal{D}_u} \\
Z_{ik} &= \sum_{j=1}^{r_i}\sum_{l=1}^{r_k}\frac{max(0; R_{ik}(jl))}{(1-p)N_{\mathcal{D}_u}}
\end{aligned}
\tag{7.3}
$$

With the estimators proposed in this section and the estimator defined in Equation (7.2) we can compute the conditional mutual information between pairs of predicting variables given the class without negative examples as follows:

$$
\begin{aligned}
&I(X_i, X_k|C) = \sum_{j=1}^{r_i}\sum_{l=1}^{r_k} P(x_{ij}, x_{kl}|1)p\frac{P(x_{ij}, x_{kl}|1)}{P(x_{ij}|1)P(x_{kl}|1)} + \\
&\sum_{j=1}^{r_i}\sum_{l=1}^{r_k}(P(x_{ij}, x_{kl}) - P(x_{ij}, x_{kl}|1)p)\frac{[P(x_{ij}, x_{kl}) - P(x_{ij}, x_{kl}|1)p](1-p)}{[P(x_{ij}) - P(x_{ij}|1)p][P(x_{kl}) - P(x_{kl}|1)p]}
\end{aligned}
$$

Once we have the conditional mutual information between every pair of predicting variables we can apply Friedman et al.'s algorithm to obtain the structure of the TAN model.

7.2.2 Parametric learning

Once the structure of the model has been induced, the model has to be completed with the parameters. As we saw in Chapter 3, the parameters of a Bayesian network classifier are the a priori probabilities of the class variable ($P(c)$) and the conditional probabilities $P(x_{ij}|\boldsymbol{pa_i})\ \forall i, j, \boldsymbol{pa_i}$, i. e., the probability that $X_i = x_{ij}$ given the variable parents $\boldsymbol{Pa}(X_i) = \boldsymbol{pa_i}$. In a Bayesian network classifier the class variable is parent of all the predicting variables, so we can decompose the set of parents as:

$$
\begin{aligned}
\boldsymbol{Pa}(X_i) &= \{C, \boldsymbol{Pa}^*(X_i)\} \\
\boldsymbol{Pa}^*(X_i) &= \boldsymbol{Pa}(X_i) \setminus C
\end{aligned}
$$

Given the decomposition the parameters of a Bayesian network classifier can be expressed as $P(x_{ij}|c, \boldsymbol{pa_i^*})$ and, thus, we can divide them into two groups, $P(x_{ij}|1, \boldsymbol{pa_i^*})$ and $P(x_{ij}|0, \boldsymbol{pa_i^*})$. The former can be estimated from the positive examples, but the latter cannot be estimated from $\mathcal{D}$ because we lack negative examples. Following the same steps shown in Section 7.1 we can see that:

$$\begin{aligned} P(x_{ij}|0, \boldsymbol{pa_i^*}) &= \frac{P(x_{ij}|\boldsymbol{pa_i^*}) - P(x_{ij}|1, \boldsymbol{pa_i^*})p}{1-p} \\ &= \frac{N_{ij\boldsymbol{pa_i^*}\mathcal{D}_u} - P(x_{ij}|1, \boldsymbol{pa_i^*})pN_{\boldsymbol{pa_i^*}\mathcal{D}_u}}{(1-p)N_{\boldsymbol{pa_i^*}\mathcal{D}_u}} \end{aligned}$$

where $N_{ij\boldsymbol{pa_i^*}\mathcal{D}_u}$ is the number of unlabelled instances where $X_i = x_{ij}$ and $\boldsymbol{Pa}^*(X_i) = \boldsymbol{pa_i^*}$. As in the case of the PNB, the negative estimations are replaced by zero, the probabilities are normalised so as to sum 1 and the Laplace correction is applied. Therefore, the estimator of the parameters conditioned to the negative class for the general case of a Bayesian network classifier can be expressed as:

$$\begin{aligned} P(x_{ij}|0, \boldsymbol{pa_i^*}) &= \frac{1 + max(0; R_i^*(j))\frac{1}{Z_i^*}}{r_i + (1-p)N_{\boldsymbol{pa_i^*}\mathcal{D}_u}} \\ R_i^*(j) &= N_{ij\boldsymbol{pa_i^*}\mathcal{D}_u} - P(x_{ij}|1, \boldsymbol{pa_i^*})pN_{\boldsymbol{pa_i^*}\mathcal{D}_u} \\ Z_i^* &= \sum_{j=1}^{r_i} \frac{max(0; R_i^*(j))}{(1-p)N_{\boldsymbol{pa_i^*}\mathcal{D}_u}} \end{aligned} \tag{7.4}$$

This generalisation of the parametric learning can be used to estimate the parameters of any Bayesian network classifier. Combining the parametric learning shown in this section with the structural learning presented in Section 7.2.1 we can induce TAN models from positive and unlabelled examples. We have named this algorithm positive tree augmented naive Bayes (PTAN).

7.3 Bayesian approach to the uncertainty in the p parameter

In the first part of this chapter we have seen how some binary Bayesian network classifiers can be induced without negative examples. The parameters

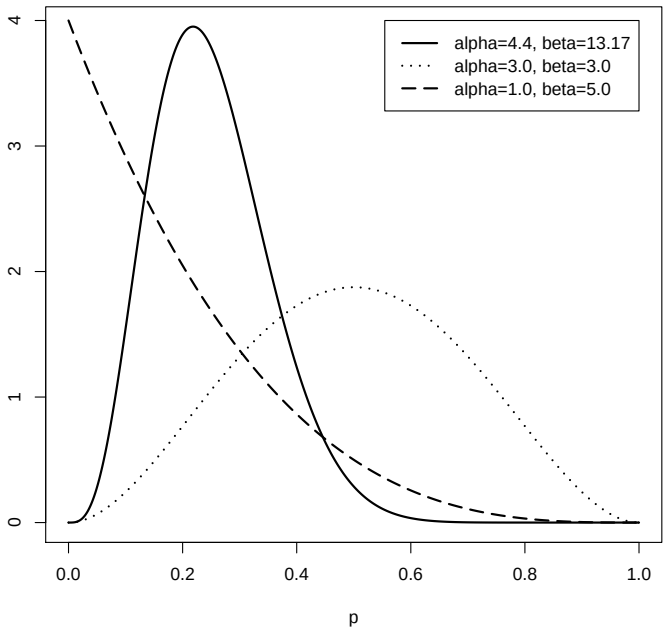


Fig. 7.1. Examples of different Beta distribution shapes.

related with the negative class can be estimated using the unlabelled instances and the parameters related with the positive class, but all the proposed equations depend on the p parameter. Setting this parameter is the main drawback of these algorithms, as this probability cannot be obtained without negative examples and is generally unknown. Nevertheless, in some situations, some information about the a priori probability of the positive class may be available.

In this section we will present a Bayesian approach to integrate the information we have about the a priori probability of the positive class in the model induction process. In this approach the uncertainty about the p parameter is modelled by means of a Beta distribution (see Figure 7.1) and then the estimators defined in Sections 7.1 and 7.2 are averaged over all the possible values of p weighted according to their probability.

Given that C follows a Bernouilli distribution with parameter p, a sample of size N of the class variable follows a Binomial distribution with parameters N, p. Therefore, we have modelled the a priori probability of p as a Beta distribution as this probability distribution is the conjugate to the Binomial distribution (Bernardo and Smith, 2000).

As an example of the averaging process, we will show how the estimator defined in Equation (7.2) can be averaged over all the possible values of the p parameter. The density function of the Beta distribution is defined as follows:

$$
\begin{aligned}
&Beta(p;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \\
&\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt \\
&p \in [0,1]; \alpha,\beta > 0
\end{aligned}
$$

The estimator in Equation (7.2) depends on p, so we can integrate it over all the possible values of p as:

$$
\begin{aligned}
P(x_{ij}|0) &= \int_0^1 \frac{P(x_{ij}) - P(x_{ij}|1)p}{1-p} P(p)dp \\
&= \int_0^1 \frac{P(x_{ij}) - P(x_{ij}|1)\cdot p}{1-p} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}dp
\end{aligned}
$$

we can separate the integral into two terms:

$$
\begin{aligned}
P(x_{ij}|0) = &\int_0^1 \frac{P(x_{ij})\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-2}dp \\
&- \int_0^1 \frac{P(x_{ij}|1)\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha}(1-p)^{\beta-2}dp
\end{aligned}
$$

and thus, we have that:

$$
\begin{aligned}
P(x_{ij}|0) = &\int_0^1 \frac{P(x_{ij})\Gamma(\alpha+\beta)\Gamma(\beta-1)}{\Gamma(\alpha+\beta-1)\Gamma(\beta)} Beta(p;\alpha,\beta-1)dp \\
&- \int_0^1 \frac{P(x_{ij}|1)\Gamma(\beta-1)\Gamma(\alpha+1)}{\Gamma(\beta)\Gamma(\alpha)} Beta(p;\alpha+1,\beta-1)dp
\end{aligned}
$$

Taking the factors that do not depend on p out of the integrals, we have the integral over all the possible values of a Beta distribution with parameters $(\alpha, \beta-1)$ in the first term and $(\alpha+1, \beta-1)$ in the second one, both of which are equal to one. Thus, we have that:

$$P(x_{ij}|0) = \frac{P(x_{ij})\Gamma(\alpha+\beta)\frac{\Gamma(\beta)}{\beta-1}}{\frac{\Gamma(\alpha+\beta)}{\alpha+\beta-1}\Gamma(\beta)} - \frac{P(x_{ij}|1)\frac{\Gamma(\beta)}{\beta-1}\alpha\Gamma(\alpha)}{\Gamma(\beta)\Gamma(\alpha)}$$

$$= \frac{P(x_{ij})(\alpha+\beta-1)}{\beta-1} - \frac{P(x_{ij}|1)\alpha}{\beta-1}$$

Therefore, we can estimate the probabilities related to the negative class as:

$$P(x_{ij}|0) = \frac{\alpha(P(x_{ij}) - P(x_{ij}|1)) + (\beta-1)P(x_{ij})}{\beta-1} \tag{7.5}$$

The problem again is that this estimator can be negative. We can solve it by replacing the negative estimations by a small number, $\frac{1}{r_i}$, and then normalise so that the probabilities sum 1. We have named averaged PNB (APNB) to the naive Bayes model induction algorithm that estimates the parameters related with the negative class from Equation (7.5).

In order to apply this Bayesian approach to the TAN model induction we need to obtain the averaged version of all the estimators proposed in Section 7.2. Following the steps presented in this section we can conclude that:

$$P(x_{ij}, x_{kl}|0) = \frac{\alpha(P(x_{ij}, x_{kl}) - P(x_{ij}, x_{kl}|1)) + (\beta-1)P(x_{ij}, x_{kl})}{\beta-1} \tag{7.6}$$

$$P(x_{ij}|0, \boldsymbol{pa_i^*}) = \frac{\alpha(P(x_{ij}) - P(x_{ij}|1, \boldsymbol{pa_i^*})) + (\beta-1)P(x_{ij})}{\beta-1} \tag{7.7}$$

As in the previous estimator, negative estimations are replaced by a small number ($\frac{1}{r_i r_j}$ in Equation (7.6) and $\frac{1}{r_i}$ in Equation (7.7)) and then normalised. The last estimator to average is $P(x_{ij}, x_{kl}, 1)$, which we saw that can be expressed as $P(x_{ij}, x_{kl}|1)p$. Given that $P(x_{ij}, x_{kl}|1)$ does not depend on p, the averaged estimator can be obtained by just replacing p by its expected value:

$$P(x_{ij}, x_{kl}, 1) = P(x_{ij}, x_{kl}|1)\frac{\alpha}{\alpha+\beta}$$

Given these averaged estimators we can induce, by means of the algorithm of Friedman et al. (1997), a TAN model from only positive and unlabelled examples. We have named this algorithm averaged PTAN (APTAN).

7.3.1 Generalisation of the averaged estimators

At the beginning of this section we have seen how the uncertainty about the p parameter can be modelled by means of a Beta distribution. This gives us more flexibility to introduce the information we may have about the a priori probability compared to just setting p at a given value. This flexibility can be increased by generalising the modellisation of the uncertainty about p to a mixture of Beta distributions (see Figure 7.2).

The definition of a probability distribution made of a mixture of b Beta distributions is as follows:

$$f(p; \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\omega}) = \sum_{i=1}^{b} \omega_i \frac{\Gamma(\alpha_i+\beta_i)}{\Gamma(\alpha_i)\Gamma(\beta_i)} p^{\alpha_i-1}(1-p)^{\beta_i-1}$$

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$$

$$\sum_{i=1}^{b} \omega_i = 1$$

$$\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_b\}, \boldsymbol{\beta} = \{\beta_1, \ldots, \beta_b\}$$

$$p \in [0,1]; \forall i, \alpha_i, \beta_i, \omega_i > 0$$

The generalisation of the averaged estimator defined in Equation 7.1 would be:

$$\begin{aligned} P(x_{ij}|0) &= \int_0^1 P(x_{ij}|0) \sum_{s=1}^{b} \omega_s \frac{\Gamma(\alpha_s+\beta_s)}{\Gamma(\alpha_s)\Gamma(\beta_s)} p^{\alpha_s-1}(1-p)^{\beta_s-1}dp \\ &= \sum_{s=1}^{b} \omega_s \frac{\alpha_s(P(x_{ij}) - P(x_{ij}|1)) + (\beta_S - 1)P(x_{ij})}{\beta_s - 1} \end{aligned}$$

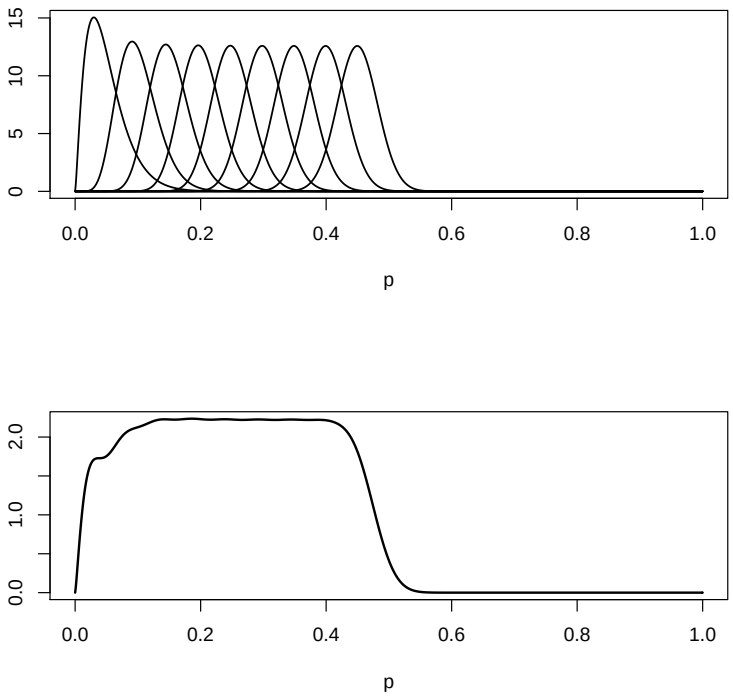


Fig. 7.2. Example of a mixture of Beta distributions. The upper part shows the components of the mixture and the lower part the desnisty function of the mixture.

and the rest of the estimators would be:

$$P(x_{ij}, x_{kl}|0) = \sum_{s=1}^{b} \omega_s \frac{\alpha_s(P(x_{ij}, x_{kl}) - P(x_{ij}, x_{kl}|1)) + (\beta_s - 1)P(x_{ij}, x_{kl})}{\beta_s - 1}$$

$$P(x_{ij}|0, \boldsymbol{pa_i^*}) = \sum_{s=1}^{b} \omega_s \frac{\alpha_s(P(x_{ij}) - P(x_{ij}|1, \boldsymbol{pa_i^*})) + (\beta_s - 1)P(x_{ij})}{\beta_s - 1}$$

$$P(x_{ij}, x_{kl}, 1) = P(x_{ij}, x_{kl}|1) \sum_{s=1}^{b} \omega_s \frac{\alpha_s}{\alpha_s + \beta_s}$$

In a reallife positive unlabelled learning problem we need some knowledge about the a priori probability of the positive class. This knowledge has to be provided by the experts in that particular topic. Modelling the uncertainty

about the p parameter by means of a probability distribution provides us with a graphical tool that can be used by the experts to represent their knowledge about this probability (they can draw a density function that represents what they know about p). Depending on the shape of the density function represented by the experts, modelling it with a single Beta distribution can be difficult. It is in such a situation that we can take advantage of the flexibility provided by a mixture of Beta distributions. For instance, if the available knowledge about p is that it has to be less than 0.5, but it can take any value below 0.5 with equal probability (a uniform distribution between 0 and 0.5), using a single Beta distribution we cannot properly model the uncertainty about p, but using a mixture of Beta distributions we can approximate it as shown in Figure 7.2.

7.4 Experimental evaluation of the PBC

The performance of the four induction algorithms described in this chapter (PNB and our proposals, PTAN, APNB and APTAN) have been compared in positive unlabelled learning problems obtained from real-life datasets (see Section 5.1, version 2) and synthetic datasets sampled from TAN models (see Section 5.2.1).

The induction algorithms were applied to each positive unlabelled learning problem and then the accuracy and the F measure were computed given that we actually know the label of the 'unlabelled' instances. In both PNB and PTAN, p was set at 0.25, while in APNB and APTAN the parameters of the Beta distribution were set at $\alpha = 4.4$ and $\beta = 13.17$ (resulting in an expected p of 0.25 and a variance of 0.01, see Figure 7.1). In many positive unlabelled learning problems the amount of examples to recover from the set of unlabelled instances (the positive cases) is smaller than the amount of instances in that set that we are not interested in (negative cases). Thus, the a priori probability of the positive class, p, should be lower than 0.5. Without further information (i.e., assuming that any value under 0.5 is equally probable) the most reasonable value for this parameter is 0.25 (the expected value in a uniform distribution over [0,0.5]). This is the reason why p was set at 0.25 in PNB and PTAN algorithms and α and β were set at 4.4 and 13.17 respectively (see Figure 7.1) in APNB and APTAN algorithms.

For each cardinality of $\mathcal{D}_p$ and each ratio of positive cases in $\mathcal{D}_u$ we have 100 different sets of positive and unlabelled examples. The significance of the differences between the performance of the algorithms was assessed by means of Wilcoxon paired tests, setting the significance level at 1%.

ACCDON - F measure

		AccCod		AccInt		AccMix		DonCod		DonInt		DonMix	
$N_{\mathcal{D}_p}$	rat.	PNB	APNB	PNB	APNB	PNB	APNB	PNB	APNB	PNB	APNB	PNB	APNB
100	0.01	11.49	**11.90**	**10.42**	10.01	**11.57**	10.99	08.80	**09.38**	**10.15**	08.97	**10.17**	08.86
100	0.10	**67.21**	66.26	**58.85**	58.04	**62.70**	61.71	**66.79**	64.36	**64.16**	62.55	**65.79**	63.94
100	0.20	**83.86**	83.46	**75.33**	75.16	**79.05**	78.83	**84.32**	84.07	**80.98**	80.85	**82.61**	82.42
100	0.30	86.94	**87.18**	79.81	**80.09**	82.83	**83.08**	85.74	**86.24**	**83.31**	83.80	84.41	**84.93**
100	0.40	84.38	**84.97**	78.17	**78.77**	80.86	**81.45**	81.21	**82.11**	79.22	**80.03**	79.91	**80.76**
100	0.50	78.55	**79.35**	72.61	**73.46**	74.93	**75.80**	71.66	**72.85**	70.12	**71.22**	70.65	**71.88**
200	0.01	11.39	**12.26**	**10.67**	10.26	**11.70**	11.12	08.02	**09.45**	**10.16**	08.80	**10.19**	08.86
200	0.10	**68.59**	66.91	**59.88**	59.01	**63.69**	62.66	**67.77**	65.25	**64.68**	62.95	**66.39**	64.45
200	0.20	**85.22**	84.86	**77.10**	76.85	**80.65**	80.38	**85.37**	85.15	**82.40**	82.26	**83.80**	83.62
200	0.30	88.88	**89.02**	82.47	**82.64**	85.41	**85.58**	87.08	**87.60**	84.52	**84.99**	85.55	**86.07**
200	0.40	87.49	**87.95**	82.09	**82.56**	84.47	**84.97**	82.83	**83.66**	81.07	**81.78**	81.77	**82.59**
200	0.50	82.64	**83.36**	77.73	**78.46**	79.66	**80.42**	74.33	**75.49**	73.25	**74.36**	73.13	**74.38**
300	0.01	11.37	**12.32**	**10.73**	10.30	**11.78**	11.19	07.97	**09.42**	**10.15**	08.82	**10.17**	08.84
300	0.10	**68.77**	67.12	**59.93**	59.12	**63.73**	62.68	**67.80**	65.22	**64.75**	62.96	**66.69**	64.64
300	0.20	**85.34**	84.96	**77.53**	77.27	**80.85**	80.54	**85.59**	85.32	**82.57**	82.43	**84.09**	83.89
300	0.30	89.05	**89.21**	82.85	**83.02**	85.63	**85.81**	87.16	87.73	84.59	**85.14**	85.63	**86.22**
300	0.40	87.79	**88.26**	82.52	**82.98**	85.08	**85.55**	83.05	83.86	81.36	**82.10**	81.88	**82.64**
300	0.50	83.13	**83.85**	78.41	**79.16**	80.53	**81.26**	74.80	**75.98**	73.77	**74.81**	73.90	**75.09**

Table 7.1. Extract of the results obtained in the comparison between the PNB and the APNB algorithms. The results correspond to the F measure obtained on datasets based on the ACCDON database. The column labelled as '$N_{\mathcal{D}_p}$' indicates the cardinality of $\mathcal{D}_p$ and the column labelled as 'rat'. denotes the ratio of positive cases in $\mathcal{D}_u$. All the values shown in this table are the average of the results obtained in the 100 datasets built according to each scheme. Bold font indicates the winner when differences were significant at a level of 1%.

7.4.1 PNB vs APNB

If we have a look at Equations (7.1) and (7.5) we can see that both are quite similar. Actually, if we divide the numerator and the denominator in Equation (7.5) by $\alpha + \beta - 1$, we have:

$$P(x_{ij}|0) = \frac{[(\alpha+\beta-1)P(x_{ij}) - \alpha P(x_{ij}|1)]\frac{1}{\alpha+\beta-1}}{(\beta-1)\frac{1}{\alpha+\beta-1}} = \frac{P(x_{ij}) - \frac{\alpha}{\alpha+\beta-1}P(x_{ij}|1)}{1-\left(\frac{\alpha}{\alpha+\beta-1}\right)}$$

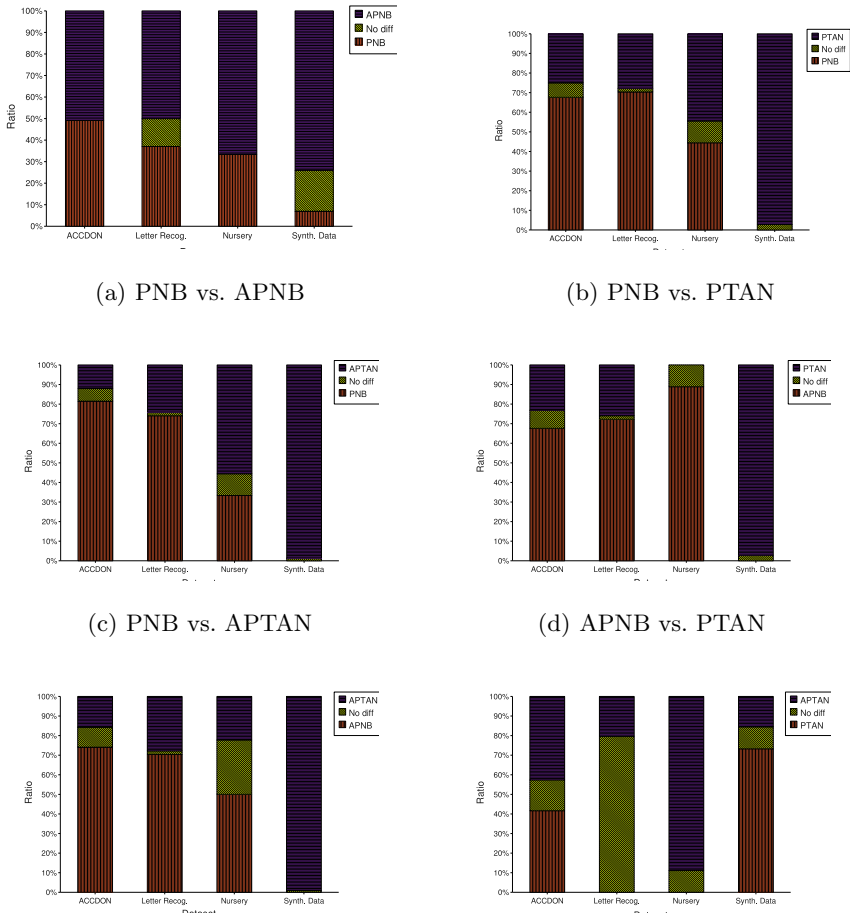
which is exactly Equation (7.1) when $p = \frac{\alpha}{\alpha+\beta-1}$. Thus, we can conclude that the APNB is equivalent to a PNB with a p slightly greater than its expected value according to the Beta distribution (the expected value of a $Beta(\alpha, \beta)$ is $\frac{\alpha}{\alpha+\beta}$).

In the case of our experimentation, where α was set to 4.4 and β at 13.17, the APNB is equivalent to a PNB with p set at 0.266. Given that p was set in the PNB algorithm to 0.25 we should not expect big differences in the results obtained by both algorithms. This can be clearly seen in Table 7.1, where the results of the comparison between PNB and APNB on datasets based on the ACCDON database are shown. Even if the differences are very small, in most of the experiments they were significant at a significance level of 1%. Similar results were obtained on the rest of the datasets used in the experimentation.

Given that the APNB is equivalent to a PNB where p is set at a value greater than the mean value of the Beta distribution, we should expect the APNB to outperform the PNB algorithm in those problems where the actual p is greater that 0.25 (and thus closer to the 0.266 value of the APNB) and vice versa. This is empirically confirmed in some of the datasets (those based on ACCDON and Letter Recognition databases), where PNB improves APNB for datasets where $p \leq 0.2$, while APNB outperforms PNB when $p \geq 0.3$ (except for some datasets where $p = 0.01$).

Having a look at the F measure (Figure 7.3) obtained in datasets based on the Nursery database (detailed results in Appendix B), we can see something similar (when p is small PNB gives better results), but the change is not at 0.25 but at a lower value (APNB improves PNB in datasets where $p \geq 0.2$). On the other hand, in datasets obtained from TAN models, PNB outperforms APNB only in a few datasets where p is set at 0.01. This is reflected in Figure 7.3-a, where we can see that for ACCDON and Letter Recognition PNB beats APNB in approximately half the datasets, while APNB outperforms PNB in most of the datasets sampled from TAN models.

Regarding the accuracy, the results are very similar to those obtained for the F measure, except for the datasets sampled from TAN models, where the



(a) PNB vs. APNB

(b) PNB vs. PTAN

(c) PNB vs. APTAN

(d) APNB vs. PTAN

(e) APNB vs. APTAN

(f) PTAN vs. APTAN

Fig. 7.3. Comparison between positive Bayesian network classifier induction algorithms. The charts show, for the datasets obtained from each original database (ACCDON, Letter Recognition, Nursery and Synthetic Data), the ratio of the problems where each algorithm beats the other (in terms of the F measure) and the ratio of datasets where no significant differences were found at a significance level of 1%. It is important to point out that the amount of problems obtained from each database is not the same (108 for ACCDON, 72 for Letter Recognition, 18 for Nursery and 216 for Synthetic Data).

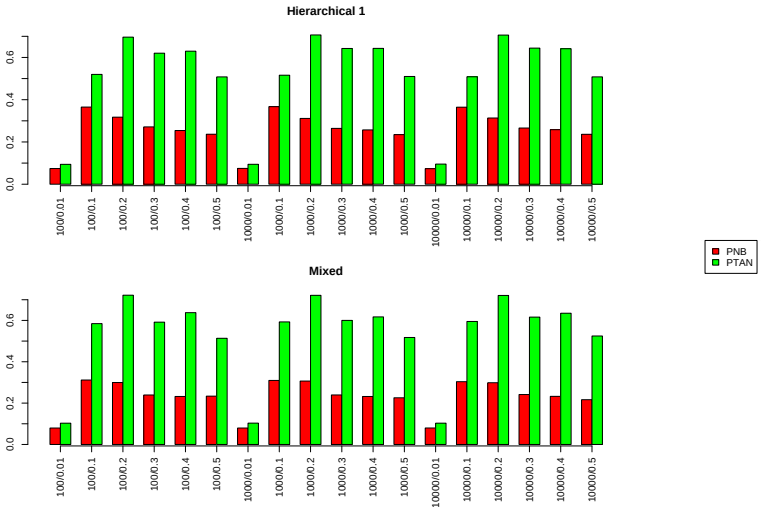


Fig. 7.4. Comparison between the results obtained by the PNB and the PTAN algorithms in some of the datasets sampled from TAN models. The results displayed correspond to the different datasets sampled from the 'hierarchical 1' and the 'mixed' structures with size of the unlabelled examples of 10,000.

situation is inverted (PNB outperforms APNB in most of the datasets). This can be explained by the fact that PNB classifies more instances as negative and, given that most of the cases are negative, it improves APNB in accuracy, but not in the F measure (note that this measure gives an idea of the recovery of positive cases). The complete set of results (including the comparisons of the accuracies) can be consulted in Appendix B.

7.4.2 (A)PNB vs (A)PTAN

The goal of extending the PNB algorithm to the induction of more complex networks is the need for algorithms that allow us to learn, from positive and unlabelled examples only, models that are able to capture the dependencies between variables. Thus, the comparison of these two types of algorithms should be made in problems where we are sure that there are conditional dependencies between predicting variables. The datasets described in Section 5.2.1 have been sampled from TAN models to ensure this point and, thus, the results we are interested in this section are those obtained on the synthetic datasets. Table 7.2 and Figure 7.4 show an extract of these results.

Synthetic datasets - F measure

		Lined		Hierarchical 1		Hierarchical 2		Mixed	
$N_{\mathcal{D}_p}$	rat.	APNB	APTAN	APNB	APTAN	APNB	APTAN	APNB	APTAN
100	0.01	0.1046	**0.1194**	0.0743	**0.1182**	0.0985	0.1061	0.0866	**0.1213**
100	0.1	0.3234	**0.3876**	0.3623	**0.4439**	0.3169	**0.3723**	0.3018	**0.4397**
100	0.2	0.3147	**0.4159**	0.3221	**0.4863**	0.3275	**0.4074**	0.3009	**0.4609**
100	0.3	0.2497	**0.3730**	0.2784	**0.4962**	0.2592	**0.4029**	0.2528	**0.4059**
100	0.4	0.2366	**0.3786**	0.2667	**0.4825**	0.2399	**0.3430**	0.2447	**0.4198**
100	0.5	0.2336	**0.3421**	0.2498	**0.3929**	0.2390	**0.3313**	0.2450	**0.3749**
1,000	0.01	0.1051	**0.1151**	0.0750	**0.1155**	0.0950	**0.1066**	0.0879	**0.1169**
1,000	0.1	0.3290	**0.3895**	0.3592	**0.4416**	0.3155	**0.3732**	0.3081	**0.4275**
1,000	0.2	0.3074	**0.4175**	0.3130	**0.4781**	0.3260	**0.4045**	0.3100	**0.4494**
1,000	0.3	0.2499	**0.3766**	0.2743	**0.4923**	0.2643	**0.4014**	0.2473	**0.4048**
1,000	0.4	0.2292	**0.3854**	0.2699	**0.4839**	0.2284	**0.3423**	0.2497	**0.4178**
1,000	0.5	0.2330	**0.3439**	0.2490	**0.3961**	0.2391	**0.3305**	0.2436	**0.3738**
10,000	0.01	0.1072	**0.1179**	0.0747	**0.1163**	0.0988	**0.1059**	0.0875	**0.1183**
10,000	0.1	0.3281	**0.3840**	0.3590	**0.4410**	0.3102	**0.3738**	0.3042	**0.4279**
10,000	0.2	0.3012	**0.4201**	0.3156	**0.4843**	0.3226	**0.4083**	0.3006	**0.4572**
10,000	0.3	0.2460	**0.3794**	0.2723	**0.4933**	0.2649	**0.3980**	0.2532	**0.4034**
10,000	0.4	0.2301	**0.3838**	0.2705	**0.4771**	0.2238	**0.3443**	0.2438	**0.4132**
10,000	0.5	0.2373	**0.3438**	0.2537	**0.3942**	0.2441	**0.3296**	0.2298	**0.3753**

Table 7.2. Extract of the results obtained in the comparison between the APNB and APTAN algorithms. The results correspond to the F measure obtained on datasets sampled from TAN models. The column labelled as '$N_{\mathcal{D}_p}$' indicates the cardinality of $\mathcal{D}_p$ and the column labelled as 'rat.' the ratio of positive cases in $\mathcal{D}_u$. In all the rows the number of unlabelled examples was 1,000. All the results shown in this table are the average of the results obtained in the 100 datasets built according to each scheme. The bold font indicates the winner in the comparisons where differences at a significance level of 1% were found.

From the results obtained in the comparison of the classifier induction algorithms in datasets sampled from TAN models, we can conclude that both PTAN and APTAN clearly outperform PNB and APNB in problems where there is an underlying tree structure of dependencies between predicting variables.

Figures 7.3-b to 7.3-e show the summary of the results obtained in the comparisons. We can clearly see in these four charts that our extensions to

the induction of TAN models outperform the naive Bayes induction algorithms in most of the synthetic datasets (last column in each chart).

If we have a look at the results obtained in the real-life datasets, (A)PNB give better results in some datasets and (A)PTAN in others.

In a real situation it is difficult to know which kind of model (naive Bayes or TAN) gives better results, but TAN models give us additional information regarding the relationship between variables while it keeps the number of parameters at check. If we know that it is very likely that there are (strong) dependencies between predicting variables in our problem, then TAN induction algorithms will provide us with more realistic models than naive Bayes induction algorithms.

7.4.3 PTAN vs APTAN

In Section 7.4.1 we have seen that APNB is equivalent to PNB when p is set at a value slightly higher than the expected value of the Beta distribution and, as a consequence, the results obtained in certain datasets are very similar and show a characteristic pattern. The comparison between PTAN and APTAN is not that straightforward, as the averaging process includes the estimation of the conditional mutual information and, thus, the learning of the structure of the model. This is reflected in the results (see Figure 7.3-f), where no apparent pattern can be found. PTAN yields better results for certain datasets (those based on acceptor sites and synthetic data), APTAN improves the results obtained by PTAN in other datasets (those based on donor sites and Nursery database), and no significant differences can be found in the rest (those based on Letter Recognition database).

7.5 Conclusions

In this chapter we have seen that the concept behind the PNB algorithm can be successfully extended to more sophisticated Bayesian classifiers. We have shown how to extend it to TAN models, but other induction algorithms could be adapted to the positive unlabelled learning context following a similar procedure. The resulting algorithm (PTAN) has been tested in datasets with an underlying tree structure of conditional dependencies, showing a significant improvement with respect to PNB.

We have also proposed a new Bayesian approach to handle the p parameter that models its uncertainty by means of a Beta distribution. In the empirical comparison between APNB and PNB we have seen that our averaged version outperforms the original algorithm in most of the datasets. In the case of PTAN and APTAN, the comparison is not so clear and, depending on the dataset, PTAN or APTAN yields better results (or no significant differences are observed).

In real-life problems, where the real value of the a priori probability of the positive class is not known, it is not possible to decide whether the normal or the averaged version of the classifiers will build better models. The advantage of the averaged version is that, thanks to the Beta distribution, it provides a more flexible way to incorporate the a priori knowledge about the problem domain (this flexibility can be increased using a mixture of Beta distributions in the modellisation of the uncertainty about p).

In future developments of the work presented in this chapter it would be interesting to extend the family of positive Bayesian network classifiers to k-DB models (Sahami, 1996).

8

Wrapper positive Bayesian network classifiers

In the previous chapter we have shown how the positive Bayesian network classifiers can be learnt from positive and unlabelled instances. The main drawback of these algorithms is that we have to set the a priori probability of the positive class p as it cannot be estimated from data. We have seen that the uncertainty about this parameter can be modelled by means of a Beta probability distribution (or a mixture of them). This provides a more flexible way to introduce any previous knowledge about this parameter but does not solve the problem when no knowledge is available.

In this chapter we propose a wrapper approach to the problem of setting the p parameter in positive Bayesian network classifiers. This wrapper approach requires a way of evaluating or, at least, comparing classifiers in the positive unlabelled learning context. To the best of our knowledge, the only work that proposes a way to compare classifiers in absence of negative instances is Lee and Liu (2003). In this work the authors propose a metric to compare classifiers which is related with the geometric mean of the precision and the recall.

In the following section we will show how the actual F measure can be estimated with positive and unlabelled examples. Unfortunately this estimation depends on the p parameter. We propose a new metric, the pseudo F, which is proportional to the actual F measure and whose estimator does not depend so much on p. Thus, we define the wrapper positive Bayesian network classifiers (wPBC), where the p parameter is set at its optimal value (in terms of the F measure) estimated using either the metric proposed in Lee and Liu

(2003) or the pseudo F. The classifiers trained using each of these two metrics are compared in datasets based on real-life problems (see Section 5.1).

8.1 F measure in the positive unlabelled learning context

One of the main problems in the positive unlabelled learning context is the evaluation of the classifiers. The absence of negative examples makes it impossible to estimate almost any of the classical performance measures (indeed, the only one we can estimate is the recall, as it is only concerned about the positive examples). All the performance measures, such as the accuracy, the classification error or the F measure, are estimated using the information contained in the confusion matrix. In binary classification problems this matrix summarises the performance of the classifier into four values: True positives (TP, the number of positive cases correctly classified), true negatives (TN, the number of negative cases correctly classified), false positive (FP, the number of misclassified negative cases) and false negative (FN, the number of misclassified positive cases). If no negative examples are available only the TP and the FN can be computed and, thus, only the recall can be estimated (for the definition of the performance measures see Section 2.2).

From a probabilistic point of view the recall is the probability that a positive case is correctly classified as positive by the classification function and the precision is the probability that an instance that has been classified as positive is actually positive. Thus, we have that:

$$r = P(\psi(\boldsymbol{x}) = 1|C = 1)$$
$$p_r = P(C = 1|\psi(\boldsymbol{x}) = 1)$$

If we take the Bayes rule into account, we have that:

$$p_r = P(C = 1|\psi(\boldsymbol{x}) = 1) = \frac{P(\psi(\boldsymbol{x}) = 1|C = 1)p}{P(\psi(\boldsymbol{x}) = 1)} \tag{8.1}$$

where $P(\psi(\boldsymbol{x}) = 1)$ is the probability that the classifier classifies an instance as positive. As we have the classification function this probability can be

obtained by classifying all the possible instances and then obtaining the ratio of instances classified as positive[1] .

In Section 2.2 we saw that the F measure is defined as:

$$F(r, p_r) = \frac{2rp_r}{r + p_r} \tag{8.2}$$

Mixing Equations (8.2) and (8.1) we have that:

$$F = \frac{2r\frac{rp}{P(\psi(\boldsymbol{x})=1)}}{r\left(1 + \frac{p}{P(\psi(\boldsymbol{x})=1)}\right)} = \frac{2rp}{P(\psi(\boldsymbol{x}) = 1) + p} \tag{8.3}$$

When only positive and unlabelled examples are available, the recall can be estimated with a k-cv scheme by simply dividing the set of known positive examples into k folds and then building a classifier with the unlabelled cases and all the folds except one. The fold left out is then classified with the learnt model and the recall is estimated as the ratio of cases that have been classified as positive. Figure 8.1 shows the scheme of the k-cv estimation of the recall from positive and unlabelled examples.

As we have the classification function ψ, $P(\psi(\boldsymbol{x}) = 1)$ can be computed (or at least estimated). The problem is the p parameter. If we knew the a priori probability of the positive class, then we could estimate the F measure in positive unlabelled learning problems.

The problem with the estimator shown in Equation (8.3) is that it depends on the p parameter. This dependence is mainly contributed by the p in the numerator that makes the estimation proportional to the a priori probability of the positive class. To partially solve the problem of the strong dependence on the p parameter, we propose a new metric, the pseudo F (F_{ps}), that is proportional to the actual F measure, and is defined as follows:

$$F_{ps} = \frac{r}{P(\psi(\boldsymbol{x}) = 1) + p} \tag{8.4}$$

This metric is proportional to the actual F measure and, thus, it behaves in the same way as the F measure, but its estimation is less dependant on the

[1] The feasibility of this calculation will depend on the number of possible instances. When the exact value cannot be obtained other approaches should be used

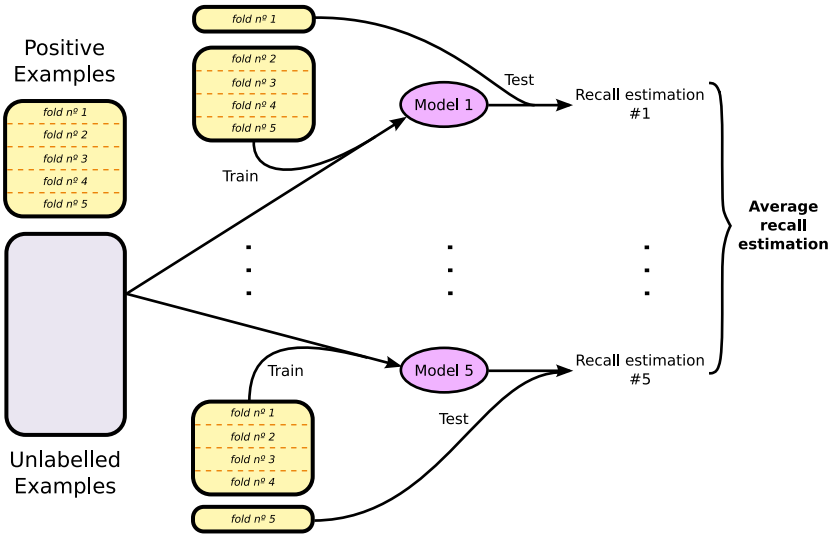


Fig. 8.1. Scheme of k-cv estimation of the recall in positive unlabelled learning problems.

p parameter than the estimation of the actual F measure. Therefore, we can use this metric to look for the p value that maximises the actual F measure (from now on we will refer to this p as the optimal p).

Lee and Liu (2003) propose a different metric based on the geometric mean of the precision and the recall. This metric, that we will call the Lee metric (L_m) is defined as:

$$L_m = \frac{r \cdot p_r}{p}$$

L_m is proportional to the square of the geometric mean of the precision and the recall. As it is, like the F measure, the mean of the precision and the recall, it is supposed to behave in the same way as the F measure in the sense that both increase and decrease with the precision and the recall. From Equation (8.1) we have that:

$$\frac{P(C=1|\psi(\boldsymbol{x})=1)}{p} = \frac{P(\psi(\boldsymbol{x})=1|C=1)}{P(\psi(\boldsymbol{x})=1)}$$
$$\frac{p_r}{p} = \frac{r}{P(\psi(\boldsymbol{x})=1)}$$

and thus:

$$L_m = \frac{r \cdot p_r}{p} = \frac{r^2}{P(\psi(\boldsymbol{x})=1)} \tag{8.5}$$

The Lee metric is independent of the p parameter and, thus, it can be estimated from positive and unlabelled examples. We can use this metric to search for the p parameter that maximises the F measure, but the problem is that what we are actually estimating is the p value that maximises the geometric mean rather than the harmonic mean.

8.2 Wrapper positive Bayesian network classifiers

In the previous section we have seen that the Lee metric and our proposal, the pseudo F measure, can be used to identify the p value that maximises the average of the precision and the recall. Therefore, these metrics can be used in a wrapper version of the positive Bayesian network classifiers (wPBC) that, given a set of positive and unlabelled examples, build a classifier that optimises the recovery of the positive cases (in terms of the F measure).

As there is only one parameter (p) to set in a wrapper way there is no need for a search heuristic. Figure 8.1 shows the pseudo code of the wrapper version of the PTAN (wPTAN) using the pseudo F measure as guiding metric. The algorithm for the wPNB is similar, replacing the PTAN algorithm by the PNB. Both algorithms can be used replacing the pseudo F measure by the Lee metric.

8.3 Experimental evaluation

We have tested the pseudo F and the Lee metric on synthetic datasets obtained from known probability distributions. This allows us to compare these metrics

wPBC algorithm

input:
  ι := number of intervals
  κ := number of rounds
  p_{min} := minimum value of p
  p_{max} := maximum value of p

for $r = 1$ **to** κ
  $p_1 = p_{min}$ and $p_\iota = p_{max}$
  For all $i = 2, \cdots, \iota - 1$, $p_i = p_{i-1} + \frac{p_\iota - p_1}{\iota}$
  For all $i = 1, \cdots, \iota$ build a model M_i using the PTAN algorithm and p_i as the p parameter
  For all M_i, $i = 1, \cdots, \iota$ estimate the recall r_i and $P(\psi_{M_i}(\boldsymbol{x}) = 1)$
  For all M_i, $i = 1, \cdots, \iota$ estimate the pseudo F measure as:

    $F_{ps}^i = \frac{r_i}{P(\psi_{M_i}(\boldsymbol{x})=1)+p}$

  $opt = argmax_i\{F_{ps}^i\}$
  if $opt \neq 1$ **then**
    $p_1 = p_{opt-1}$
  if $opt \neq \iota$ **then**
    $p_\iota = p_{opt+1}$
rof

output:
  M_{opt} is the wPTAN classifier

Alg. 8.1: Pseudo code of the wPTAN algorithm using pseudo F as the guiding metric.

with the actual value of the F measure and compare their evolution as we change the p parameter in PNB and PTAN.

The wrapper positive Bayesian network classifiers have been tested on real-life based datasets, comparing their performance with that of PNB and PTAN when p was set at 0.25 and at the actual value.

8.3.1 Comparison of the pseudo F and the Lee metric in synthetic datasets

The synthetic datasets used to test the performance metrics are those described in Section 5.2.3. These datasets have been obtained from empirical

distributions defined by all the possible instances along with their label. This label was set according to different naive Bayes and TAN models. As we have the whole set of possible instances labelled, given a classification function we can obtain the exact value of the F measure.

We run the PNB and PTAN algorithms on these datasets setting the p parameter at 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 0.95. For each learnt classifier the recall was estimated using a 5 times 5-cv scheme. The classifiers were used to obtain the prediction for all the possible instances. From this classification $P(\psi(\boldsymbol{x}) = 1)$ was obtained as the ratio of instances classified as positive. Given that we have the actual label of all these instances, the confusion matrix was obtained and the actual value of the F measure was estimated.

The pseudo F, the Lee metric and the square root of the Lee metric (which is proportional to the geometric mean of the precision and the recall) obtained with the classifiers learnt with different values of the p parameter have been plotted and compared with the actual value of the F measure. Figure 8.2 shows some of the results obtained. The rest of the results can be consulted in Appendix C.

In this figure we can see that the F measure shows a maximum at a p close to the actual a priori probability of the positive class[2] . Indeed, in some datasets the actual and the optimal p are the same (Figure 8.2-a and 8.2-d). If we have a look at the results obtained (Appendix C) we can see that the optimal p is always greater or equal to the actual p.

As can be seen in the figure, the shapes of the actual F measure and the pseudo F measure, are very similar in all the plots. Indeed, not only have they a maximum at approximately the same point, but they also have similar slopes in the increasing and decreasing part of the plot. Regarding the Lee metric, the first part of the curve (p from 0 to the optimal p) is very similar to the actual F measure, but in the decreasing part (p from the optimal value to 1) it has a slope smaller than that of the F measure. When the F measure decreases quickly with p (after the optimal value) the Lee metric is able to identify the optimal p, but when the decreasing rate of the actual F measure is smaller, the Lee metric does not show a maximum in the optimal p. Nevertheless, if not by means of a maximum, the Lee metric could be also used to identify

[2] Note that the actual p is not necessarily the one that maximises the F measure

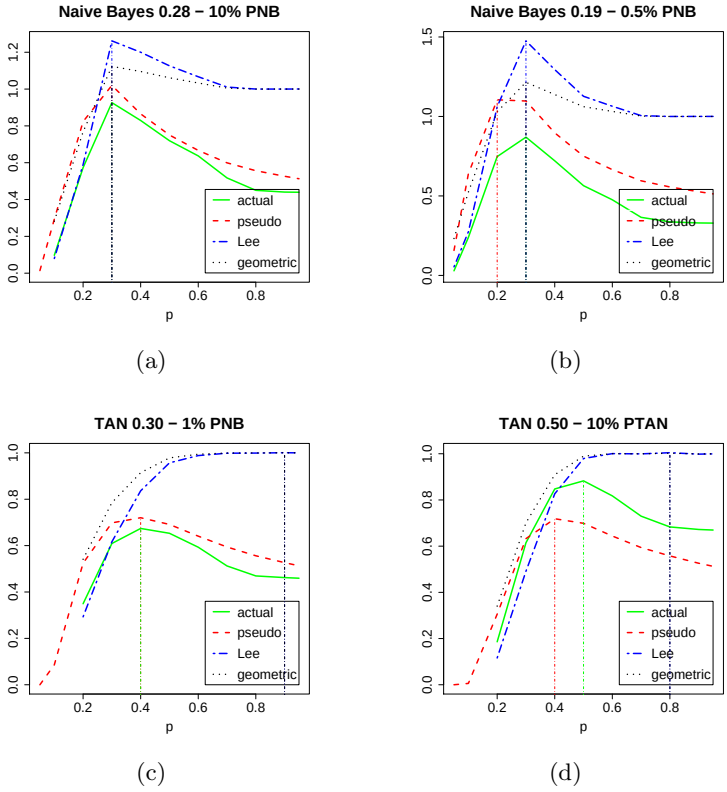


Fig. 8.2. Extract of the results obtained in the comparison between the pseudo F (labelled as 'pseudo'), the Lee metric (labelled as 'Lee'), its square root (labelled as 'geometric') and the actual F measure (labelled as 'actual') in synthetic datasets. The plots represent the metrics vs. the p parameter used in the training of the classifiers.

the optimal p looking at the elbow of the curve. Similar results were obtained for the rest of the datasets (see Appendix C).

The pseudo F and the Lee metric allow us to estimate the optimal p. Though this optimal value is not necessarily the actual a priori probability of the positive class, it seems to be close. If we take this optimal p as an estimation of the actual a priori probability we can obtain an estimation of the actual F measure using Equation (8.3).

We have compared the actual F measure obtained by the classifiers trained with p set at the optimal value (estimated using the pseudo F measure) with

the estimated F measure (i.e., $2p$ times the pseudo F measure of the classifier). The results of the comparison can be seen in Table 8.1. There are several questions to point out in this table. First of all, we can see that most of the estimations have a negative bias[3] . This makes sense if we take into account that these estimations are based on the recall, which has been estimated using a repeated k-cv scheme, which is known to have a negative bias. Nevertheless, not all the estimations have a negative bias. Interestingly, the estimators with a positive bias are those where the optimal p was set at 0.7 (and 0.6 in one of the cases).

When estimating the F measure we have two contributions to the error. On one hand, we have the bias in the estimation of the recall, that should lead to estimations with a negative bias of the F measure. On the other hand, we have the p. We are setting the p at the estimated optimal value, but, even if we perfectly identify this optimal value, it is not necessarily the actual a priori probability of the positive class. The bias in the estimation of p also contributes to the error in the estimation of the F measure. Looking at the results obtained in synthetic datasets (see Appendix C) we can see that the optimal p is always greater or equal to the actual p. Therefore, the optimal p can be considered as an estimator of the actual p with a positive bias. This positive bias in the estimation of p compensates the negative bias in the estimation of the recall. The final error in the estimation of the F measure is a trade-off between these two components, which explains why we have some estimations with negative biases and some estimations with positive biases (and in between some estimations with almost no bias).

In order to check the influence of the bias in the estimation of the recall, we can look at those problems where the optimal p and the actual p are the same. If we calculate the bias in these problems we can see that it ranges between 25% and 30% in most of the cases. For the rest of the cases it is, in general, lower than that, suggesting the contribution of the p in the final bias.

8.3.2 Performance of the wPBC in real-life based datasets

The wPBCs introduced in Section 8.2 have been tested in datasets based on real-life problems. The datasets used for this experimental evaluation are

[3] We will consider the bias as the estimated value minus the actual value. Therefore, a positive bias means that the estimated value is lower than the actual value

those described in sections 5.1.2.2, 5.1.3.2 and 5.1.4.2. The results obtained by the wPBC (using the pseudo F and the Lee metric to estimate the optimal p) have been compared with the results obtained when the probability was set at 0.25 and the actual value. In a real-life positive unlabelled learning problem we do not know the class of the unlabelled instances and, thus, the only way to have an estimation of the F measure is using Equation (8.3) but, given that the datasets used in this experimentation have been obtained from completely labelled databases we actually know which 'unlabelled' cases are positive and which are negative. Thus, we can estimate the F measure directly from the set of unlabelled instances (as we have done in Chapters 6 and 7). We have used these estimations in the evaluation of the algorithms because they do not depend on the p used in the training.

Figure 8.3 and Table 8.2 show some of the results obtained in the experimental evaluation of the wPBCs. The complete set of results can be consulted in Appendix D. Figure 8.3 represents the value of the pseudo F and the Lee metric as a function of the p used in the search for the optimal p. The first two plots are very similar to those obtained in synthetic datasets, showing a maximum close to the actual value of the a priori probability of the positive class, but this behaviour cannot be seen in the lower two plots. In most of the plots (see Appendix D) a maximum can be identified. As in the results obtained in synthetic datasets, the estimation of the optimal p using the pseudo F tends to be lower than the estimation using the Lee metric. Indeed, this can explain curves such as those shown in Figures 8.3-c and 8.3-d, where it seems that the maximum of the pseudo F is so close to the 0 that only the descending part of the curve can be seen. Similarly, in some plots only the ascending part of the Lee metric curve is observed. This suggests that the estimation of the optimal p using the pseudo F measure has a negative bias, while using the Lee metric it has a positive bias. This can also be observed in some of the results obtained in synthetic datasets (see Appendix C).

Table 8.2 shows the comparison between the F measure obtained by the wPBCs (using the pseudo F and the Lee metric) with the results obtained setting p at 0.25 and the actual value. In a situation where we do not know the actual value of the a priori probability of the positive class but we know that the number of negative examples is greater than the number of positive examples, 0.25 is the most reasonable value if we regard any value lower than 0.5 equally likely. It is important to notice that, due to the experimental

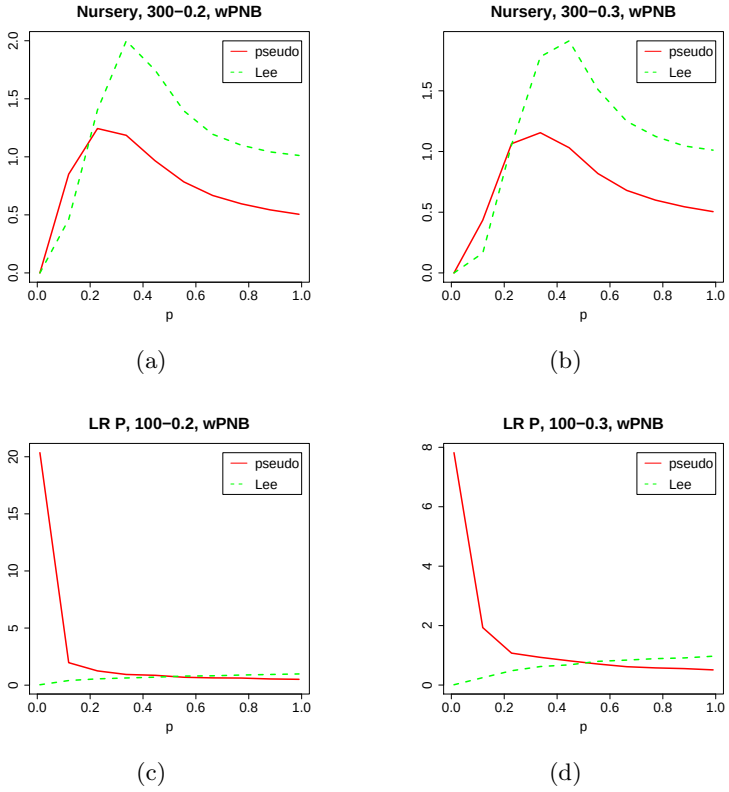


Fig. 8.3. Extract of the results obtained in the comparison between the pseudo F (labelled as 'pseudo') and the Lee metric (labelled as 'Lee') in real-life datasets. The plots represent the metrics vs. the p parameter in the search for the optimal p in the wPNB algorithm.

design, in some of the positive unlabelled learning problems the actual p is very close to 0.25 and, thus, setting p at this value yields good (sometimes the best) results. Regarding the use of the actual probability in the training process, one can suppose that this is the best possible value and, indeed, it is the one that gives the best overall results. Nevertheless, the actual p is not always the one that leads to the best F measure as we have seen in Section 8.3.1. In addition, the actual p is generally unknown in real positive unlabelled learning problems and, thus, it cannot be used in the training of the classifier.

The significance of the differences was assessed running Wilcoxon paired tests (setting the significance level at 1%) on the results obtained in the 100

sets of positive and unlabelled examples sampled using each scheme (see Section 5.1.2). In Table 8.2 we can see that, though in most of the datasets the best results are obtained when p is set at the actual value, in other datasets the wrapper algorithms provide the highest F measure or no significant differences can be found with the best classifier. In particular, the wPTAN outperforms the PTAN in many datasets even when p is set at its actual value, specially in datasets sampled from the Letter Recognition database. Having a look at the plots obtained for wPTAN in these problems we can see that in most of the cases only the ascending part of the curve can be seen. Nevertheless, the wPTAN obtains the highest F measure. This suggests that in these particular problems the optimal p for the PTAN algorithm is much higher than the actual p and, thus, setting p at 0.95 (the biggest value used in the search of the optimal p) yields better results than setting it at its actual value.

8.4 Conclusions

In this chapter we have shown how the F measure can be estimated when no negative examples are available. The problem is that this estimation depends on the p parameter. Regarding this parameter, we have proposed a new metric based on the F measure, the pseudo F measure, that can be used to estimate its optimal value in terms of the F measure. We have tested this metric, along with the metric proposed by Lee and Liu (2003), in synthetic datasets. The results show that the pseudo F has a behaviour similar to that of the actual F measure, being both maximum at approximately the same p. The Lee metric shows a similar behaviour though its tendency to decrease after the optimal p is not as clear as in the pseudo F. The results also suggest that the estimation of the optimal p using the pseudo F tends to have a negative bias (it always gives values lower or equal to the optimal) while the estimation using the Lee metric tends to have a positive bias.

In principle one may think that setting the p parameter at the actual a priori probability of the positive class should lead to the best classification function, but this is not necessarily true. Nevertheless, we have seen in the experimental evaluation that the optimal p (the one that maximises the F measure) is very close to the actual p. Interestingly, the optimal p seems to be always higher than the actual p and, thus, can be considered as an estimator of the actual p with positive bias.

The estimation of the F measure using positive and unlabelled examples is based on the estimation of the recall which, using a cross validation scheme, has a negative bias, and on the estimation of the p by means of the optimal p, which has a positive bias. The result is that the estimation of the F measure can have a positive or a negative bias, depending on the individual contribution of the recall and the p. On one hand, this is a drawback because we cannot be sure when the estimation is optimistic or pessimistic, but, on the other, hand it is good because the two contributions are compensated, resulting in a smaller bias in the estimation of the F measure.

We have proposed the wrapper positive Bayesian network classifiers where p is set at its optimal value (in terms of the F measure) estimated using either the pseudo F or the Lee metric. We have tested these wrapper approaches in real-life based datasets. The results show that in many problems the wrapper Bayesian network classifiers (in particular the wPTAN) obtain results comparable to those obtained when p was set at its actual value. Indeed in some datasets they clearly improve the classifiers where p was set at the actual value. This is specially clear in datasets sampled from the Letter Recognition database where, in some problems, the optimal p seems to be significantly higher that the actual p.

We have seen that the estimation of the optimal p using the pseudo F measure has a negative bias while the estimation of using the Lee metric seems to have a positive bias. Bearing this in mind, it would be advisable setting p at a value between the two estimations.

The study of the behaviour of the pseudo F and the Lee metric and the estimation of the actual F measure has been conducted estimating the recall with a 5 times 5-cv scheme. We have seen that the contribution of the bias in the recall to the error in the estimation of the F measure is quite big. In future works it would be interesting to analyse the bias and the variance of the estimation of the recall using other schemes (such as bootstrap) and its influence in the estimation of the F measure.

We have proposed a wrapper approach to set the p parameter in PNB and PTAN algorithms. In future developments it would be interesting to explore a wrapper approach to set the α and β parameters in APNB and APTAN. This is a more complex problem as there are two parameters to estimate and thus we have to search for the maximum in the surface defined by the pseudo F or the Lee metric in the plane (α, β).

In the next chapter we will present a filter approach to the FSS problem in positive unlabelled learning problems. In a future work it would be interesting to study the wrapper approaches to the FSS based on the metrics introduced in this chapter.

Estimated vs. actual F measure

			PNB			PTAN		
		$\lvert\mathcal{D}_p\rvert$	Actual	Est.	p_{opt}	Actual	Est.	p_{opt}
naive Bayes	0.19	33	66.71	42.78	0.20	56.74	36.59	0.20
		164	74.74	44.19	0.20	73.39	43.91	0.20
		328	75.39	45.39	0.20	76.96	46.57	0.20
		1638	75.93	45.66	0.20	79.09	47.37	0.20
		3277	76.12	45.78	0.20	79.26	47.69	0.20
	0.28	33	75.36	50.98	0.30	69.26	56.54	0.40
		164	86.14	57.59	0.30	81.63	54.67	0.30
		328	88.91	58.36	0.30	86.41	57.39	0.30
		1638	92.33	60.82	0.30	92.80	61.75	0.30
		3277	92.62	61.20	0.30	93.83	62.57	0.30
	0.45	33	66.25	77.28	0.70	62.51	80.86	0.70
		164	70.91	78.97	0.70	68.17	69.10	0.60
		328	77.88	70.03	0.60	71.36	70.53	0.60
		1638	79.33	72.12	0.60	78.44	62.79	0.50
		3277	79.38	72.17	0.60	79.37	64.29	0.50
TAN	0.15	33	53.20	48.13	0.30	54.00	34.74	0.20
		164	53.93	36.38	0.20	74.55	47.49	0.20
		328	54.91	37.35	0.20	79.73	51.31	0.20
		1638	55.78	37.53	0.20	83.40	54.42	0.20
		3277	55.78	37.61	0.20	84.14	54.81	0.20
	0.30	33	60.13	65.22	0.50	56.50	39.88	0.30
		164	66.44	57.00	0.40	74.60	50.14	0.30
		328	67.40	57.62	0.40	79.01	51.84	0.30
		1638	68.22	57.82	0.40	83.10	54.87	0.30
		3277	68.36	58.28	0.40	83.47	55.17	0.30
	0.50	33	71.78	71.08	0.60	69.31	72.56	0.60
		164	75.41	63.15	0.50	74.27	52.34	0.40
		328	76.90	63.42	0.50	79.06	53.87	0.40
		1638	78.12	63.97	0.50	83.90	56.97	0.40
		3277	78.29	63.72	0.50	84.76	57.49	0.40

Table 8.1. Comparison between the actual and the estimated F measure in synthetic datasets. The results shown in this table correspond to the classifiers obtained setting the p at the optimal value estimated using the pseudo F measure. The column labelled as $|\mathcal{D}_p|$ represents the size of the set of positive examples and the columns labelled as 'Actual', 'Est.' and 'p_{opt}' represent the actual F measure, the estimated F measure and the estimated optimal p.

Nursery F measure

		wPNB		PNB			wPTAN		PTAN		
$\lvert\mathcal{D}_p\rvert$	rat.	ps.	Lee	0.25	actual	Wlx.	ps.	Lee	0.25	actual	Wlx.
100	0.01	32.17	3.10	17.40	0.00		19.48	2.97	15.31	15.76	
100	0.10	63.27	26.03	70.66	31.94		63.39	24.30	68.41	51.00	
100	0.20	68.83	43.30	74.14	62.96		69.10	40.49	76.87	70.64	
100	0.30	75.15	54.37	67.92	77.20	c	72.95	51.41	71.56	78.69	
100	0.40	77.30	62.58	56.57	83.09		75.86	60.67	57.87	81.71	
100	0.50	78.42	71.00	43.09	86.33		75.87	68.59	46.69	82.32	
200	0.01	31.10	16.47	18.93	0.00		23.07	18.54	16.09	15.86	b,d
200	0.10	59.40	57.91	72.95	32.22	e	65.85	61.08	69.09	55.82	
200	0.20	67.52	71.51	76.81	65.07	e	62.06	75.52	81.62	77.04	d
200	0.30	73.67	78.32	69.87	80.14	d	77.21	76.63	78.13	83.90	a,b,e
200	0.40	76.57	83.55	58.35	86.32		75.88	82.48	65.09	86.86	
200	0.50	79.22	83.81	42.15	88.99		72.41	82.97	49.45	87.92	
300	0.01	33.22	19.06	19.47	0.00	b	26.57	20.02	16.07	16.85	d
300	0.10	69.89	60.30	74.19	31.47	a	68.11	64.92	69.27	57.74	a
300	0.20	74.78	77.23	77.86	65.76	b,e	76.92	78.54	82.64	79.65	a,d,e
300	0.30	79.38	82.58	70.38	81.17	c,e	77.76	82.58	80.91	85.66	
300	0.40	82.44	85.82	58.21	87.58		82.01	85.35	67.97	88.54	e
300	0.50	82.87	86.71	41.97	90.04		82.54	88.22	50.56	89.04	d

Letter Recognition P

		wPNB		PNB			wPTAN		PTAN		
$\lvert\mathcal{D}_p\rvert$	rat.	ps.	Lee	0.25	actual	Wlx.	ps.	Lee	0.25	actual	Wlx.
100	0.01	42.59	2.28	12.89	53.27		28.90	3.62	35.89	33.99	c
100	0.10	69.54	20.73	67.59	78.52		35.35	29.38	60.10	37.08	c
100	0.20	59.33	36.72	82.48	83.23		37.05	47.60	45.27	37.18	b,c
100	0.30	44.98	49.91	84.27	85.13	e	45.43	58.76	26.61	34.03	
100	0.40	51.08	60.85	80.75	85.79	e	58.61	66.14	15.40	32.99	
100	0.50	60.47	70.02	72.57	85.69	a,e	68.78	69.46	8.88	34.55	e
200	0.01	52.10	38.96	13.67	54.26		54.08	22.32	29.90	54.96	c
200	0.10	74.79	74.16	68.02	80.60	e	60.20	45.05	75.64	67.27	
200	0.20	64.90	80.90	83.90	84.85	b	71.25	56.03	72.18	66.91	a
200	0.30	54.06	84.22	87.35	87.51		72.96	64.44	59.81	65.87	d
200	0.40	57.71	81.57	86.52	88.57		75.87	70.40	44.10	64.57	
200	0.50	67.68	83.65	82.63	89.69		75.04	75.17	31.21	63.19	e
300	0.01	53.94	53.94	14.25	53.94	c,d,e	61.88	56.56	27.80	63.56	c,e
300	0.10	76.89	77.09	69.42	81.49	e	52.57	54.85	77.73	75.82	e
300	0.20	66.69	71.01	84.60	85.78		61.77	62.83	79.31	76.44	e
300	0.30	44.49	73.40	87.78	87.99		72.93	70.81	72.72	76.20	a,c
300	0.40	48.64	83.79	87.73	89.42		80.11	74.59	63.96	76.83	
300	0.50	68.41	83.58	85.19	90.39	b	81.89	77.90	50.62	75.89	

Table 8.2. Extract of the comparison between the F measure obtained by wPNB and wPTAN using the pseudo F (labelled as 'ps.') and the Lee metric (labelled as 'Lee') to estimate the optimal p) and by PNB and PTAN when p was set at 0.25 (labelled as '0.25') and the actual value (labelled as 'actual'). The first column indicates the cardinality of $\mathcal{D}_p$ and the second the ratio of positive cases in $\mathcal{D}_u$. The significance of the differences was assessed running Wilcoxon paired tests on the results obtained in the 100 problems sampled using each scheme. The column labelled as 'Wlx.' indicates the comparisons where no differences were found at significance level of 1% ('a' for the wrapper based on the pseudo F vs. $p = 0.25$; 'b' for wrapper based on Lee vs. $p = 0.25$; 'c' for wrapper based on pseudo vs. p set at the actual value; 'd' for wrapper based on Lee vs. p set at the actual value and 'e' for the two wrapper approaches).

9

Feature Subset Selection from Positive and Unlabelled Examples

In this chapter we will focus on the problem of feature subset selection (FSS) in positive unlabelled learning problems. To the best of our knowledge this problem has never been explicitly treated before[1] .

The contributions presented in this chapter are related with the filter approach to the FSS. In particular, we will introduce an adaptation of the correlation based filter approach (CFS) presented in Hall and Smith (1997). This is a multivariate filter method that searches for a small subset of relevant, non-redundant variables.

The chapter is organised as follows. In Section 9.1 the CFS algorithm is described. Then, in Section 9.2 the adaptation of this algorithm to the positive unlabelled learning context is introduced. The results of the comparison between the proposed adaptations and the original CFS algorithm in simulated problems is shown in Section 9.3. Finally, in Section 9.4 some conclusions are provided.

9.1 The CFS algorithm

In Hall and Smith (1997) the authors propose an algorithm for filtering out irrelevant and redundant variables. This algorithm is based on a metric that, given a candidate subset of predicting variables, measures the goodness of

[1] Some algorithms, such as the decision tree induction algorithm (Quinlan, 1993), that have already been adapted to the positive unlabelled learning context (Letouzey et al., 2000) do not include all the predicting variables in the final model

the set. This goodness is measured in terms of the relevancy and the non-redundancy of the variables. In a subset, relevancy is measured as the correlation between each variable in the set and the class variable while redundancy is measured as the correlation among each pair of predicting variables.

The metric proposed by Hall and Smith (1997) to evaluate a given set S is defined as follows:

$$G_S = \frac{k\overline{u_{ci}}}{\sqrt{k + k(k-1)\overline{u_{ii'}}}}$$

where k is the number of variables in S, $\overline{u_{ci}}$ is the average correlation between the features in S and the class, and $\overline{u_{ii'}}$ is the average correlation among the features included in S.

The correlation between each pair of variables (attributes and/or the class variable) is measured in terms of the uncertainty coefficient $U(X_k|X_i)$ (Press, 1988), which is a measure proposed in the information theory framework and is based on the mutual information $I(X_k; X_i)$ and the entropy $H(X_k)$. When X_i and X_k are discrete variables, it is defined as:

$$U(X_k|X_i) = \frac{I(X_k; X_i)}{H(X_k)} = \frac{H(X_k) - H(X_k|X_i)}{H(X_k)}$$

$$H(X_k) = -\sum_{l=1}^{r_k} P(x_{kl}) log(P(x_{kl}))$$

$$H(X_k|X_i) = -\sum_{j=1}^{r_i} P(x_{ij}) \sum_{l=1}^{r_k} P(x_{kl}|x_{ij}) log(P(x_{kl}|x_{ij}))$$

Looking at the G_S metric we can see that the higher the mean correlation of the features with the class (i.e., the average relevancy of the features) the higher the metric and the higher the correlation among the variables (i.e., the higher the redundancy) the lower the metric. Thus, given a set of variables, adding redundant variables will decrease the score while adding relevant features will increase it.

Given a set of variables and this metric, the search for the subset of features that maximises the metric can be cast as an optimisation problem. The search in the feature subset space is an NP-hard problem and, thus, as the number

of features increases the exhaustive search quickly becomes computationally infeasible. This problem is overcome using search heuristics that are able to find good solutions in a reasonable computational time. In the original work, Hall and Smith (1997) use greedy hill climbing search procedures (forward selection and backward elimination), but any other search heuristic could be used to solve the optimisation problem. The pseudo code of the forward selection applied to the selection of features can be found in Figure 9.1. The backward elimination algorithm would be the same but initialising the solution with all the features and then eliminating at each step the one that leads to the highest increase in the score until no improvement is reached.

CFS algorithm with greedy hillclimbing forward selection strategy

```
1  Initialisation
2     Set the initial solution as S_0 = X_i where X_i is the feature with the highest
      correlation with the class variable (measured in terms of the uncertainty
      coefficient)
3     i := 0
4  Search
5     Obtain all the candidate solutions adding to S_i each of the features not in
      S_i (one feature in each candidate solution)
6     For each candidate solution L_j calculate G_{L_j}
7     if the highest G_{L_j} is greater than G_{S_i}
8     then
9        S_{i+1} = argmax_{L_j} G_{L_j}
10       i = i + 1
11       Keep on searching
12    else
13       return S_i as the best subset
14    fi
```

Alg. 9.1: Pseudo code of the greedy hill climbing forward selection in the CFS algorithm.

9.2 CFS in absence of negative examples

In the supervised classification framework we have a dataset of instances containing examples from all the classes. CFS uses these examples to obtain, for

a candidate feature subset, the correlation between every pair of features and between each feature and the class variable.

In the positive unlabelled learning context only positive and unlabelled examples are available. If we try to estimate the correlations required by the CFS from our set of positive and unlabelled examples, we can see that the correlation among features can be estimated from the unlabelled cases (as it is independent of the class variable), but the correlation between each feature and the class cannot be estimated from the data because we lack negative examples.

The problem comes when we try to obtain $U(X_i|C)$:

$$U(X_i|C) = \frac{I(X_i;C)}{H(X_i)} = \frac{H(X_i) - H(X_i|C)}{H(X_i)}$$

In particular, the problem arises when trying to estimate the conditional entropy $H(X_i|C)$, as the entropy of the predicting variable X_i does not depend on the class variable and, thus, it can be obtained from the set of unlabelled instances.

We can overcome this problem by decomposing the estimator of the conditional entropy into two terms, one containing the probabilities related to the positive class and the other the probabilities related to the negative class:

$$\begin{aligned} H(X_i|C) = &- p \sum_{j=1}^{r_i} P(x_{ij}|1) log P(x_{ij}|1) - \\ &(1-p) \sum_{j=1}^{r_i} P(x_{ij}|0) log P(x_{ij}|0) \end{aligned}$$

where $P(x_{ij}|1)$ can be estimated from the positive examples. As it happens in the learning of Bayesian network classifiers from positive and unlabelled examples (see Chapter 7), neither p nor $P(x_{ij}|0)$ can be obtained directly from the datasets, but we can express $P(x_{ij}|0)$ as:

$$P(x_{ij}|0) = \frac{P(x_{ij}) - P(x_{ij}|1)p}{1-p}$$

Thus, we can estimate $P(x_{ij}|0)$ based on p as:

$$\frac{N_{ij\mathcal{D}_u} - P(x_{ij}|1)pN_{\mathcal{D}_u}}{(1-p)N_{\mathcal{D}_u}} \tag{9.1}$$

where $N_{ij\mathcal{D}_u}$ is the amount of unlabelled instances where $X_i = x_{ij}$ and $N_{\mathcal{D}_u}$ the cardinality of the set of unlabelled examples. The problem with this estimator is that it can be negative. As in the case of PBCs, the negative estimations are replaced by 0, and then all the probabilities are normalised so as $\forall i$, $\sum_{j=1}^{r_i} P(x_{ij}|0) = 1$. After the normalisation and taking the Laplace correction into account, $P(x_{ij}|0)$ can be estimated as:

$$\begin{aligned} P(x_{ij}|0) &= \frac{1 + max(0; R_i(j))\frac{1}{Z_i}}{r_i + (1-p)N_{\mathcal{D}_u}} \\ R_i(j) &= N_{ij\mathcal{D}_u} - P(x_{ij}|1)pN_{\mathcal{D}_u} \\ Z_i &= \sum_{j=1}^{r_i} max\left(0; \frac{R_i(j)}{(1-p)N_{\mathcal{D}_u}}\right) \end{aligned}$$

As p cannot be estimated from only positive and unlabelled examples, it remains as a parameter of the algorithm.

If we take this equation into account, and once the value of p is set, we can estimate the entropy of any predicting variable conditioned to the class variable and, thus, obtain the G_S score for a given set of features from a dataset without negative examples. We have named the CFS algorithm where the G_S score is computed in this way positive unlabelled CFS (puCFS).

9.2.1 Averaging the probability estimations

As we have seen in the previous section, the estimation of G_S depends on the a priori probability of the positive class (p). As this probability cannot be estimated from the data, the user must set it.

As we did in the averaged positive Bayesian network classifiers we can model the uncertainty about this parameter by means of a Beta distribution (see Figure 7.1) and then, average the conditional probability $P(x_{ij}|0)$ for all the possible values of p. The result of this integral is:

$$P(x_{ij}|0) = \frac{\alpha(P(x_{ij}) - P(x_{ij}|1)) + (\beta - 1)P(x_{ij})}{\beta - 1}$$

The negative estimations are replaced by 0 and the probabilities are normalised so that they sum 1. In the induction of Bayesian classifiers, if we replace negative estimation by 0 we can have null parameters that can cause problems in the prediction because a product is involved (the factorisation of the joint probability distribution). Due to this reason, in the averaged positive Bayesian network classifiers the negative probabilities are substituted by a small value ($\frac{1}{r_i}$). However, as the estimation of the conditional entropy is a sum, we can replace the negative estimation by 0 without problems. We have named this algorithm averaged positive unlabelled CFS (apuCFS).

9.3 Experimental evaluation

As we have said in the introduction of this chapter, to the best of our knowledge, no FSS algorithms have been proposed in the positive unlabelled learning context. In the experimental evaluation of our algorithms we have compared them with the original CFS algorithm. We can do this because we are working with simulated problems where we actually know the label of the instances in the set of unlabelled instances. Therefore, the results of the CFS shown in this section are obtained applying the algorithm to the 'unlabelled' datasets, that are actually supervised datasets. It should be noticed that we are comparing our positive unlabelled learning algorithms with a supervised algorithm. The original CFS has access to much more information than any of the adaptations to the positive unlabelled learning context. In addition, the size of the sets of unlabelled instances is relatively big, meaning that the CFS is operating in a very favourable scenario. Bearing this in mind, the results obtained by the CFS should be regarded as the goal of puCFS and apuCFS rather than values to compare with the results of these two algorithms.

In the experimental evaluation we have used two types of datasets. On one hand, we have tested our proposals in synthetic datasets specially designed to evaluate the feature subset selection (see Section 5.2.2). These datasets have been sampled from models where we know which variables are relevant, which are not and which are redundant. Thus, we know beforehand which variables should be selected by the algorithms.

The performance of the algorithms has also been tested in real-life based datasets. In particular, ACCDON datasets v.2, Letter Recognition datasets v.2 and Nursery datasets v.2 have been used (see Section 5.1).

Regarding the parameters in our approaches, three different values for the p parameter were used in puCFS: 0.10, 0.25 and 0.5. For apuCFS, the α and β parameters were set at 4.4 and 13.17 respectively (resulting in a mean value of p of 0.25 and a standard deviation of 0.1). The search procedure used to obtain the subset of features is a greedy hill climbing forward selection in all the algorithms (including the original CFS described in 9.1).

9.3.1 Results on synthetic datasets

The synthetic datasets have been sampled from known probability distributions and, thus, we expect the algorithms to select only the relevant variables (those labelled as R and a number) and filter out both irrelevant (I and II variables) and redundant variables (variables ending as R and the redundancy degree). Figure 9.1 shows the results obtained by the three algorithms (CFS, puCFS with p set at 0.25 and apuCFS) in synthetic datasets. Each subfigure shows the ratio of datasets where each algorithm selected each of the variables given the size of the set of known positive examples in (a)puCFS and the ratio of positive cases in $\mathcal{D}_u$ in CFS[2] . The results of the selection for the (a)puCFS algorithms shown in this figure are the average of the results obtained for all the ratios of positive cases in the set of unlabelled instances (from 0.01 to 0.75).

Looking at Figure 9.1 we can see that the CFS (first chart in each subfigure) filters out irrelevant and redundant variables fairly well in all the datasets. It only has some problems with the irrelevant features (specially those that are also independent) in datasets sampled from model 2 where the ratio of positive cases is low. On the other hand we can see that, in some datasets, CFS does not select all the relevant variables. In datasets based on model 3, for instance, only half of the features are clearly selected and the other half are included in the final solution in between 50% and 75% of the experiments. This is even worse in datasets based on models 2 and 4, where only four or five variables are clearly selected as relevant.

This is explained by the parameters of the models used to generate the data. The categorisation of the variables in relevant or irrelevant was done based on the structure of the models, but the parameters play a very important role. Actually the structure can tell us, regardless of the parameters,

[2] With (a)puCFS we mean both puCFS and apuCFS

which variables are independent, but it cannot ensure that two variables are correlated. Thus, until the parameters are set, we cannot know whether a variable is redundant or not.

In order to check the relevancy of the features labelled as relevant, we have computed the information gain ratio of the class variable given each of these variables. The info gain ratio of two random variables X and Y is a measure of the reduction in the uncertainty about X given that we know the value of Y. In the case of the class variable and a predicting variable X_i, it is calculated as:

$$GainR(C, X_i) = \frac{I(C; X_i)}{H(X_i)} = \frac{H(C) - H(C|X_i)}{H(X_i)}$$

Taking into account that $I(C; X_i) = H(C) - H(C|X_i) = H(X_i) - H(X_i|C)$ (Cover and Thomas, 2006), we can see that the information gain ratio and the uncertainty coefficient introduced in Section 9.1 are equivalent. Thus, this is the same metric used by the CFS to measure the correlation of the variables with the class variable. Table 9.1 shows the info gain ratio between each of the relevant variables in model 4 and the class variable. We can see that variables $R4$ and $R6$ are very poorly correlated with the class while variables $R8$ and $R9$ are not correlated at all. This explains why these four variables are almost not selected by the algorithms in the different runs.

If we have a look now at the results obtained with puCFS and apuCFS, we can see that, regarding the selection of relevant variables, they follow the same pattern as the original CFS except in datasets based on model 2. In general, those variables that are selected by the CFS in most of the datasets are also selected in most of the datasets by both puCFS and apuCFS. It is important to point out here that, due to how the datasets were created (see Section 5.2.2), the results of the CFS algorithm shown in Figure 9.1 are the average of twenty datasets (given a ratio of positive cases we have twenty sets of unlabelled instances, ten of size 10,000 and ten of size 100,000) while the results of (a)puCFS are the average of 10,000 (given a size of the positive cases we have 100 sets of positive instances that are combined with 100 sets of unlabelled instances, twenty for each of the five ratios of positive cases). The detailed results obtained for each ratio of positive cases in $\mathcal{D}_u$ can be consulted in Appendix E.

Feature	Info Gain Ratio
$R1$	$1.1 \cdot 10{-}3$
$R2$	$3.4 \cdot 10{-}2$
$R3$	$5.6 \cdot 10{-}3$
$R4$	$3.1 \cdot 10{-}7$
$R5$	$2.7 \cdot 10{-}2$
$R6$	$2.3 \cdot 10{-}6$
$R7$	$6.2 \cdot 10{-}2$
$R8$	0.0
$R9$	0.0
$R10$	$1.5 \cdot 10{-}3$

Table 9.1. Information gain ratio calculated for the relevant variables in model 4. The values shown in this table have been obtained from the distribution defined by model 4 and not estimated from the data. As we can see in the table, variables $R8$ and $R9$ have a null information gain ratio and variables $R4$ and $R6$ have a very low information gain ratio, meaning that they are poorly correlated or not correlated at all with the class variable. This is due to the fact that, although they are graphically related with the class, when the parameters are (randomly) set, these variables become not (or very poorly) correlated with the class variable.

Regarding the redundant variables, both puCFS and apuCFS filter them out correctly when the number of positive examples is high. When it is low some redundant variables are selected in some datasets (but always less than in half the datasets).

The main problem with the positive unlabelled versions of the CFS algorithm is the selection of irrelevant variables. In particular, these two algorithms tend to select variables that are completely independent (II-labelled features, that neither depend on the class nor on any other variable). This is specially clear when very few positive examples are available. If we look at the top rows of each chart, that corresponding to sets of positive cases of 1,000 examples, we can see that the ratio of datasets where irrelevant independent variables are included is normally lower than 0.5.

The inclusion of independent variables can be easily explained if we consider how the G_S metric is calculated. In the numerator of the score we have the average correlation with the class variable. It is in this term where the differences between the original CFS and the positive unlabelled CFS are. The denominator, the average correlation among the variables in S, is calculated exactly the same way in the original and the positive unlabelled version of

the CFS. Due to this reason, in (a)puCFS the error we make estimating the average correlation with the class is higher than the error we make estimating the mean correlation between predicting variables (because we only lack info about the class and not the other variables). Because of this higher error rate, it can be that the inclusion of an irrelevant variable does not decrease too much (or even increase) the mean correlation with the class while it reduces the mean correlation among predicting variables, leading to an increase in the G_S score. As we increase the number of positive cases available the error in the estimation of the correlation with the class diminishes, reducing the probability of adding irrelevant independent variables.

If we look at the results obtained by puCFS (with p set at 0.25) and apuCFS, we can see that the results obtained by both algorithms are very similar. This makes sense if we take into account the result presented in Section 7.4.1, because the apuCFS is equivalent to a puCFS where p is set at a value slightly greater than its expected value. In our particular case, it is equivalent to a puCFS where $p = 0.266$. This little change in the paremeter does not affect too much the final result of the feature subset selection (the effect of changing the p parameter in puCFS can be seen in Appendix E).

9.3.2 Results on real-life datasets

The algorithms have also been compared on positive unlabelled learning problems based on real-life databases where the absence of negative examples has been simulated (ACCDON datasets v.2, Letter Recognition datasets v.2 and Nursery datasets v.2; see Section 5.1 for more details)

In these real-life datasets we do not know which variables are relevant and which are redundant. Thus, we have compared the features selected by puCFS and apuCFS with those selected by the original CFS. Figure 9.2 shows these results. The figure consists of six subfigures. The upper three correspond to the results obtained by the puCFS algorithm in datasets from ACCDON, Letter Recognition and Nursery respectively; the lower three to the results obtained by the apuCFS algorithm. Each subfigure is made out of bars, each bar representing a particular dataset (of a given size of positive and unlabelled instances and a given ratio of positive cases in the set of unlabelled instances), and it shows the ratio of variables selected by (a)puCFS but not the CFS, by CFS but not (a)puCFS and by both. The bars (datasets) are ordered so

as the ones with the biggest ratio of common features are placed in the lower part of the chart and the ones with the lowest ratio of common features are displayed in the top of the chart.

In Figure 9.2 we can see that in many datasets from ACCDON and Nursery both CFS and our proposal agree in most of the variables. In some datasets in ACCDON and Nursery, and in most of the datasets in Letter Recognition our adaptation selects more variables than the original CFS algorithm. In the previous section we have already seen this behaviour, as our proposal tends to select more variables, especially the independent ones.

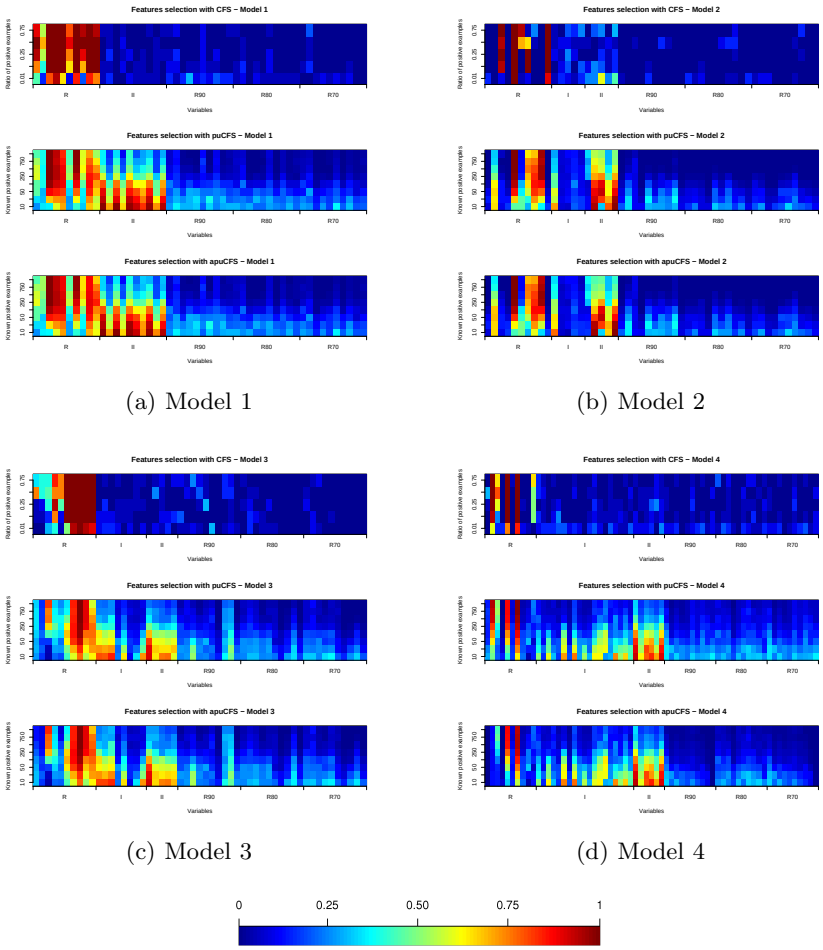
Comparing the upper and lower part in the figure, we can see that the results obtained by both puCFS and apuCFS are almost identical, confirming what we have seen in the previous section.

9.4 Conclusions and future work

In this chapter we have shown how the concept of FSS can be introduced in the context of learning from positive and unlabelled examples. In particular, we have adapted the CFS to the positive unlabelled learning framework and we have tested these adaptations on both synthetic datasets and real-life based problems.

From the results obtained in these comparisons we can conclude that the proposed algorithms can be used to filter out the redundant variables. The irrelevant variables can also be filtered out provided that we have enough positive instances to obtain a good estimation of the correlation between the feature and the class variable. When the number of positive examples available is reduced, it would be advisable to pre-filter irrelevant variables before applying these algorithms to prevent its inclusion due to a reduction in the mean correlation among variables in the final subset.

From the results obtained in the comparisons, we can also conclude that there are no big differences between puCFS and apuCFS.



(a) Model 1 (b) Model 2

(c) Model 3 (d) Model 4

Fig. 9.1. Results obtained in the comparison between CFS, puCFS ($p = 0.25$) and apuCFS in synthetic datasets. The rows in the charts corresponding to the CFS algorithms represent the ratio of positive cases in the dataset used in the experiment. In the charts corresponding to puCFS and apuCFS each row represents the number of known positive examples, being the charts the average of the results obtained for the different ratios of positive cases in $\mathcal{D}_u$. Each column in the charts represents a feature. The features are labelled as R (relevant), I (irrelevant), II (irrelevant and independent) and $R90$, $R80$, $R70$ (redundant with degrees of redundancy of 90%, 80% and 70% respectively). The colour of each square represents the ratio of datasets where the algorithm selected the variable as part of the final subset. All the results are the average of the ratios obtained in all the datasets with the particular cardinality of $\mathcal{D}_p$ or ratio of positive cases in $\mathcal{D}_u$, shown in the Y axis.

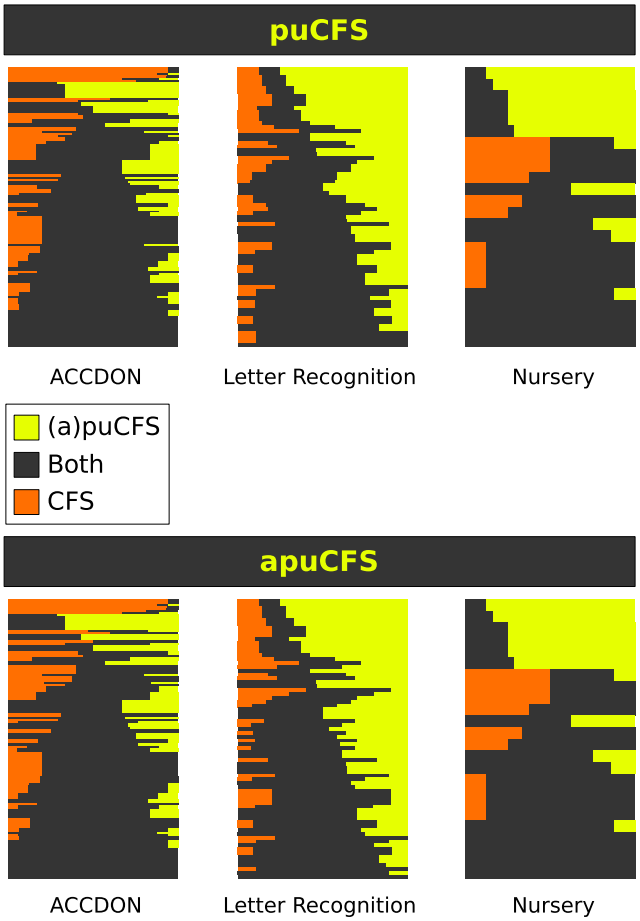


Fig. 9.2. Comparison between CFS and (a)puCFS in real-life data based problems. Each figure shows, for each dataset (each line in the chart) the ratio of variables selected only by (a)puCFS, only by CFS and by both algorithms. The first three figures correspond to the results obtained by the puCFS and the last three those obtained by the apuCFS. The datasets in each chart are ordered according to the ratio of common features.

Part III

Applications

10

Disease Gene Prediction

During the last years the amount of data concerning the molecular basis of life has grown exponentially. There is a huge amount of information about DNA, RNA and proteins contained in databases such as Ensembl or Swissprot. This information can be mined in order to extract useful knowledge to help us understand some of the complex processes that make life possible and the molecular origins of some diseases.

The use of computational techniques on these datasets can be of great help for molecular biologists in their research. In this chapter we will show how positive unlabelled learning algorithms can be used to help molecular biologists in their search for the genetic origins of hereditary diseases.

10.1 Biological background

The desoxyribonicleic acid (DNA) is the biomolecule that stores the information in the living beings. Whenever a cell is divided, its DNA is duplicated and a copy is passed to each daughter cell. The DNA contains the information to create and control the life machinery, but the molecules that constitute this machinery are (mainly) proteins. Proteins have many and very different functions such as catalysation of biochemical reactions, signal transduction or structural function.

DNA is a molecule made out of a long sequence of basic units, the nucleotides. There are, in a DNA molecule, four types of nucleotides: Adenine (A), thymine (T), cytosin (C) and guanine (G). Therefore, a DNA molecule can be seen as a sequence of these four nucleotides, a word in the {A,T,C,G}

alphabet. On the other hand, proteins are made out of different basic units, the aminoacids (twenty in most proteins) and, thus, can be seen as a sequence of these aminoacids.

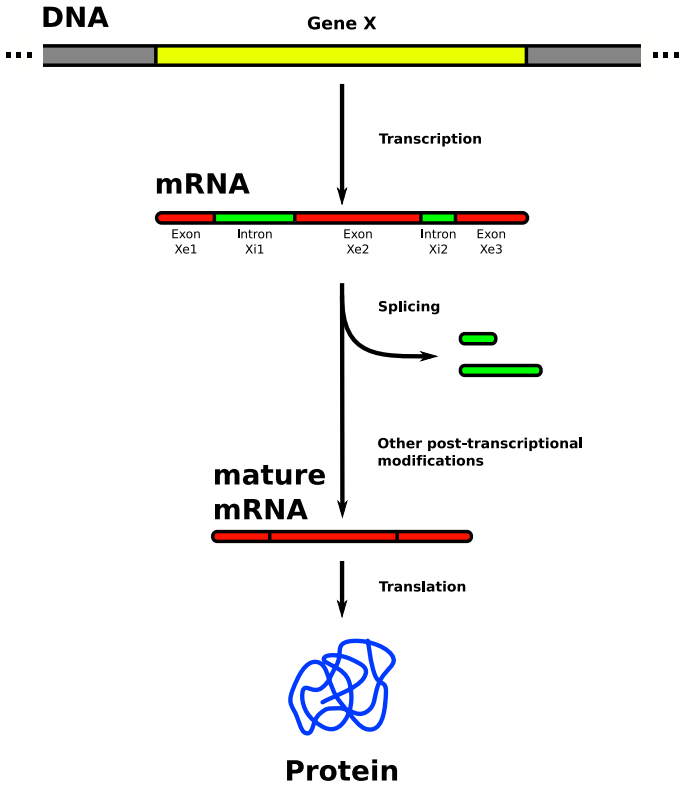


Fig. 10.1. Simplified scheme of the protein synthesis.

The process through which the information contained in the DNA is used to produce a protein can be divided into two steps. In a first step an intermediary molecule, the mRNA, is obtained by copying part of the information stored in the DNA (that corresponding to a coding gene). This molecule is subject to some modifications before it is translated into a protein. One of these modifications is the 'splicing', where parts of the sequence (the introns) are removed (see Figure 10.1). After all the modifications have been performed, we have the mature mRNA which is translated into a protein sequence. The

information in the mRNA is codified, like in the DNA, using four nucleotides (though not exactly the same as in the DNA).

As we have said, proteins are words in the alphabet of aminoacids. In order to codify twenty elements, the nucleotides are grouped in codons. Each codon is made out of three nucleotides and, thus, there are 64 possible codons to codify the twenty aminoacids (actually there are less than 64 because some codons are used to signal the beginning and the end of the translation). Due to this imbalance, most of the aminoacids are codified by more than one codon.

The function of a protein is very closely related to its 3D structure, and this structure depends on the sequence of the protein[1] . Any change in the sequence of a protein can lead to changes in its 3D structure, and these modifications of its structure can affect its function.

The information contained in the DNA not only refers to the sequence of the proteins but also to the control of the protein synthesis process. There are sequences (in non-coding regions of the DNA), such as the transcription factor binding sites, that are involved in the control of when a protein is produced and at which level it is expressed. In the introns removed during the splicing there are also sequences involved in the control of the splicing process.

A point mutation is a change in one of the nucleotides of the DNA sequence. It can be due to errors when copying the information (there are mechanisms to correct these errors, but sometimes they do not work correctly) or due to external factors such as chemical mutagens or ionising radiation.

The consequences of a point mutation depends on where it occurs. If the mutation is in a coding region (a fragment of DNA that codifies part of the sequence of some protein), it can lead to a change in the protein sequence or not (there are some aminoacids that are codified by different codons and, thus, it is possible to change one nucleotide in a codon and obtain a new codon that codifies for the same amino acid). This modification can be due to a change in the amino acid codified by the mutated codon or, in some cases, due to an early end of the translation (if the mutation leads to a termination codon).

The mutation can also be in a non-coding region. Many of these mutations do not have any effect, but it can occur that the mutation is localised in a sequence involved in the control of the synthesis of a protein. For instance,

[1] There are other important factors, such as post-translational modifications, that change the structure of the protein, modifying or activating/deactivating its function, but these modifications are out of the scope of this introduction

if the sequences responsible for the control of the splicing are mutated they can lead to an incorrect removal of the introns. These not removed introns will then be translated as part of the protein sequence. Changes in control sequences can also alter the expression levels of a protein.

There are many pathologies that are known to have a (more or less complex) genetic origin. Huge amounts of time and money are spent trying to identify (in the wet-lab) and understand the origin of these diseases. A very common approach is the association studies where the mutated genes are associated with particular regions of the genome. In many cases, the researchers looking for the genes liable for the disease reduce the candidate genes to a few hundred (those in the region associated with the disease). Checking each of these genes in the lab is a very time-consuming task. This is the reason why different computational approaches have been proposed to accelerate this process. The idea is to provide a tool that, given a set of genes, tells us which of these genes are more likely to be involved in a hereditary disease. When the candidate genes have to be checked in lab one by one, this tool can be of use as a guide.

10.2 Previous work

Different approaches have been proposed to tackle the computational identification of disease genes. There are some approaches that focus on the prioritisation of candidate genes for a particular disease based on information about their function, expression, protein-protein interactions or MEDLINE data (Köhler et al., 2008; Tiffin et al., 2005; van Driel et al., 2005; Silva et al., 2004; Turner et al., 2003; van Driel et al., 2003; Pérez-Iratxeta et al., 2002). Other approaches try to identify disease genes based on sequence properties (Adie et al., 2005; López-Bigas and Ouzounis, 2004). In this chapter we will focus on the latter approach. Recent studies have shown that the sequence properties of disease genes depend on whether they are dominant or recessive genes (Furney and del Mar Albá, Maria and López-Bigas, Núria, ????; López-Bigas et al., 2006). Therefore, we have tackled the identification of dominant and recessive genes separately.

The genome wide identification of disease genes is an intrinsic positive unlabelled learning problem as no negative examples are available. There are databases, such as OMIM (online Mendelian inheritance in man) (Hamosh

et al., 2000), LocusLink (Pruitt et al., 2000) or the Human Gene Mutation Database (Krawczak et al., 2000) where genes that have already been identified as disease related are collected. Therefore, getting a set of positive examples is relatively easy. Regarding the rest of the genes, it is not possible to ensure that any of these genes is not involved in any possible disease and, thus, they should be considered as unlabelled rather that negative.

In this chapter we will use the DCDiv algorithm in the identification of disease genes. In Section 10.3 the datasets used in the prediction will be presented. Section 10.4 contains the results obtained by the DCDiv algorithm and Section 10.5 provides some conclusions.

10.3 Datasets

The instances (genes) are characterised by some features extracted from their sequence that have been proved to be related with the likelihood of being disease related. The variables used in the prediction were: The conservation score (CS) (López-Bigas and Ouzounis, 2004) in several eukaryotic genomes (*Pan troglodytes, Mus musculus, Rattus norvegicus, Gallus gallus, Fugu rubripes, Danio rerio, Drosophila melanogaster, Anopheles gambiae, Caenorhabditis elegans* and *Caenorhabditis briggsae*), the conservation of the paralogues, the protein length, the gene length, the number of introns and the number of low complexity regions. All the sequence properties were computed for the longest protein sequence of each gene in the human genome (Birney et al., 2004).

The CS is a measure of the mutation rate to which the protein has been subject during the evolution process that is independent of the length of the protein (López-Bigas and Ouzounis, 2004). It is based on the BlastP score (the score obtained using the BLAST algorithm (Altschul et al., 1990) against a protein sequence database) and consists of the BlastP score of the closest homologue in the genome divided by the BlastP score of the protein against itself. Therefore, the CS can range between 0 and 1. A previous work (López-Bigas and Ouzounis, 2004) reported a significantly higher amount of conserved proteins (in terms of CS) among the genes already identified as disease related compared to the rest of the genes. This suggests that the genes involved in genetic diseases tend to be more conserved than the average. This is probably because if a change in the sequence of a gene leads to a disease phenotype, it is less likely that these changes will go through the evolution process than if

the mutation has no effect on the phenotype. Thus, the degree of conservation of a gene (measured by means of the CS) provides us with information about how likely a mutation in this gene leads to a disease.

Given a gene, there can be other genes (paralogues) in the same genome with a very similar sequence. This concept is interesting from the point of view of genetic diseases because a very conserved paralogue can possibly provide the function of the mutated gene. Thus, mutations in genes with highly conserved paralogues are less likely to produce a disease phenotype. In López-Bigas and Ouzounis (2004) the authors analyse the presence of conserved paralogues in disease genes. They measure the paralogy as the CS of the closest protein in the human genome (excluding the protein being analysed), and their results show that disease genes have less conserved paralogues than the rest of the genes. Therefore, this feature can also be used in the identification of disease genes.

In López-Bigas et al. (2005) the authors analyse the coding length and the number of intron in those genes that have already been identified as disease related. Their study suggests that many disease causing mutations could be related with changes in splicing patterns of the gene, either by affecting a splice site or any of the regulatory sequences involved in the control of the splicing process. Given these results, the number of intron seems to contain some information about the likelihood of a gene to be disease related (the more introns the more complex the splicing is and, thus, the more regulatory sequences the gene will have). The results obtained in this work also show significant differences in the distribution of gene and protein length between disease genes and the rest of the genes.

A low complexity region in a protein is a fragment of a sequence with a non-random distribution of a restricted number of aminoacids. Suppose we have a protein of l aminoacids and this protein has an f-long fragment F of a particular amino acid. Given that there are twenty different aminoacids and assuming that the aminoacids in the different positions are independent of the rest, the probability of observing such a fragment would be:

$$P(F) = \left(\frac{1}{20}\right)^{n} \cdot 20 \cdot (l - n + 1) = \left(\frac{1}{20}\right)^{(n-1)} \cdot (l - n + 1)$$

If we consider a protein of 400 aminoacids ($l = 400$), a region made of a particular aminoacid with a length $n \geq 5$ will be non-random with a signifi-

cance level of 0.5%. This same concept can be extended to subsets of amino acid, such as those that are hydrophobic or acidic. This complexity regions have already been reported to be associated with disease (Karlin et al., 2002).

Given that our algorithms work with discrete data, we have discretised all the variables into ten intervals using equal frequency.

Modelling the prediction of disease genes as a positive unlabelled learning problem means that we have to create a dataset of known positive examples and a dataset of unlabelled instances. Given that we will tackle the prediction of dominant and recessive genes separately, we have obtained a list of dominant disease genes and a list of recessive disease genes.

The list of known disease genes was obtained from the morbid map table in the OMIM database. The genes in this list (1,647) were classified according to the mode of inheritance using text mining techniques on the clinical synopsis section of the OMIM database and manual curation. As a result, 498 genes were labelled as dominant and 662 as recessive (the rest, 487, were labelled either as X-linked, chromosomal rearrangements, association for complex traits or unknown mode of inheritance). The set of unlabelled instances was, in each case, the list of genes not in the set of positive examples (only those genes for which we could obtain the sequence properties, 19,548, were considered).

10.4 Results

The DCDiv algorithm was run on the datasets described in the previous section. The recall was set at 75% and 10% of the positive examples were used as spy cases. The maximum number of iterations was set at 3,000 and the threshold was considered zero when it reached the computational zero (10^{-324}). TAN models (Friedman et al., 1997) were used as base classifier.

Given the dataset of positive and unlabelled examples the DCDiv algorithm provides us with a classification for the unlabelled instances, rather than a probabilistic model. Thus, we do not have a probability that can help us to rank the genes. To overcome this problem the algorithm was run fifty times and, for each gene, the ratio of runs where it was labelled as positive was calculated. As a result, for each gene we have two ratios, one corresponding to the prediction of dominant disease genes and the other to the prediction of recessive disease genes (except for the dominant and recessive genes for which we have only one of the two ratios).

The goal of this work was to create a tool that can be used to, given a set of genes, rank them according to their probability of being involved in a disease process. This tool, created by López-Bigas et al., can be found at http://genome.imim.es/∼nlopez/DGPv2/. Through this web the predictions can be queried according to different criteria, such as gene ID, chromosomic region, etc.

We can also look at the results more generally, considering the number and the spatial distribution of the genes predicted as causative of genetic disease. In order to obtain an average classification, we need to set a threshold for the previously defined ratios. The higher the threshold, the more restrictive the classification will be and, thus, the fewer identified genes we will have.

If we consider that a gene is disease related when it has been classified as so in, at least, half the repetitions then the threshold has to be set at 0.5. This is equivalent to combining the fifty classifications using a simple voting scheme. Setting the threshold at 0.5, 6,558 genes were identified as associated with dominant diseases (34.5% of the unlabelled genes) and 4,582 as associated with recessive genes (24.2% of the unlabelled genes). 1,742 (18.5%) of the predicted genes have been labelled as causing both dominant and recessive diseases.

The most restrictive threshold would be 1, which means that we will only consider a gene as disease related when it has been identified in all the repetitions. Considering this threshold, only 2,415 genes (12.7%) would be identified as dominant disease-associated and 1,880 (9.9%) as recessive disease-associated. 288 (7.19%) of the predicted genes would be labelled as causing both dominant and recessive diseases.

These results are consistent with the starting hypothesis of the DCDiv, i.e., that the number of disease genes in the set of unlabelled instances is lower than the number of non-disease genes (see Section 6.1). The ratio of unlabelled genes that are classified as disease-associated can be seen as an estimation of the a priori probability of the positive class. As a way to measure the robustness of the results, we have applied the PNB algorithm using these estimations as input, and we have compared the results obtained with the two algorithms. Depending on the dataset and the threshold considered, the two algorithms assign the same class to between 72% and 86% of the instances (see Table 10.1).

Comparison of DCDiv & PNB predictions

	Dominant		Recessive	
	0.5	1	0.5	1
DCDiv	34.67%	12.77%	24.22%	9.94%
PNB	47.69%	25.62%	32.56%	15.71%
Common	73.11%	80.57%	77.83%	86.08%

Table 10.1. Results of the comparison between the predictions made by DCDiv and PNB. The columns labelled as '0.5' correspond to the results obtained when a gene is considered as a disease gene when it has been labelled as positive in at least half the repetitions and the columns labelled as '1' correspond to the results obtained when only the genes labelled as positive in all the repetitions are considered as diseases-related. In all the cases the p parameter in the PNB algorithm was set at the ratio of disease genes identified by the DCDiv algorithm. The rows labelled as 'DCDiv' and 'PNB' show the percentage of unlabelled instances identified as positive by each algorithm. The row labelled as 'Common' shows the percentage of unlabelled instances where both PNB and DCDiv agree in the classification (either labelling them as disease or non-disease).

Focusing on the known disease genes (the positive cases), we can see that the number of genes related to dominant diseases is lower than the number of those associated with recessive diseases (498 and 662, respectively). This relation is inverted in the predicted disease genes, where the number of dominant genes predicted is greater than the number of recessive genes.

Regarding the genes predicted as both dominant and recessive, 5.15% of the known disease genes were labelled as dominant and recessive. Our results show similar ratios in the predictions (specially in the most restrictive classification, with 7.19% of the genes predicted as dominant and recessive).

In addition to the gene classification, we have also represented the spatial distribution of the genes predicted to be associated with disease, both for recessive and dominant mutations. The classification on which this spatial distribution is based is the one obtained by simple voting (i.e., setting the threshold at 0.5). For each chromosomic region, we have obtained the ratio

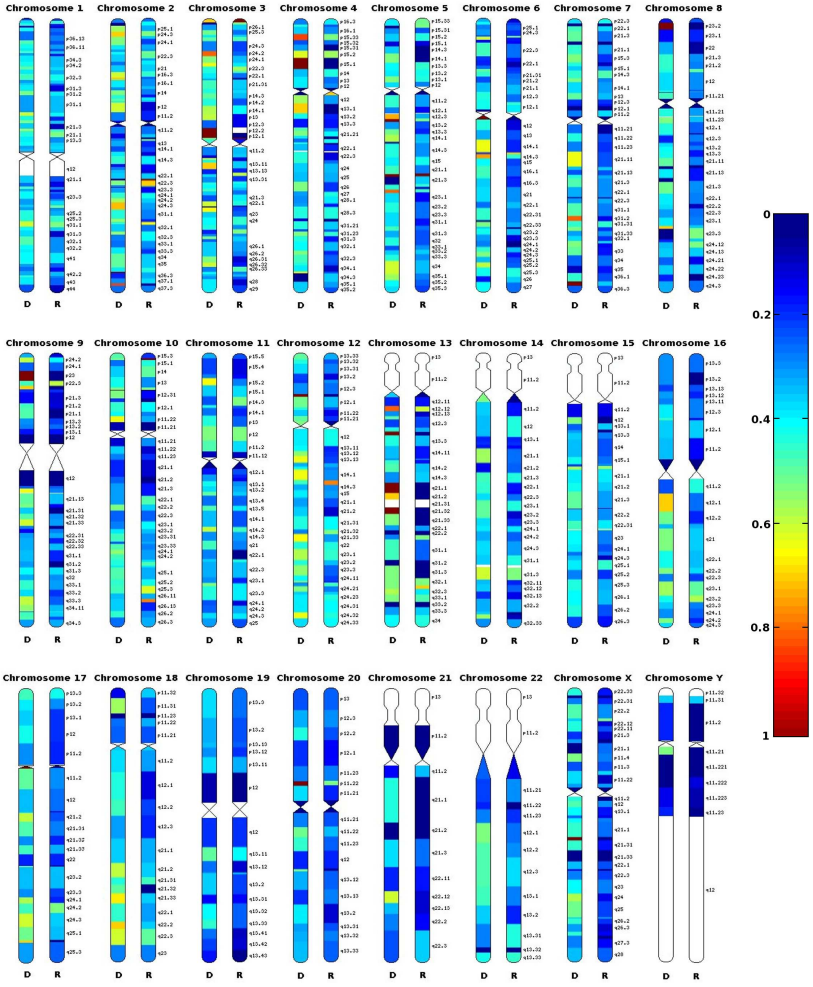


Fig. 10.2. Representation of the fraction of genes that are predicted as related to dominant (D) and recessive (R) disease by chromosome regions. The regions where there are no genes for which we have the sequence properties have been represented in white colour. One should take into account that different regions have different numbers of known genes, and this information is not represented in the figure.

of genes in that region (considering only those genes about which we have all the predicting variables) that have been predicted as disease genes. The

comparison of the spatial distribution of dominant and recessive genes can be seen in Figure 10.2.

10.5 Conclusions

In this chapter we have seen that the prediction of disease genes from sequence properties can be modeled as a positive unlabelled learning problem. In these predictions, recent results suggesting that dominant and recessive disease genes show different sequence properties have been taken into account, predicting dominant and recessive disease genes separately.

In the prediction only Mendelian disease genes have been considered. This is a first approach to the problem. Taking into account other multigenic diseases in the prediction poses a more complex problem that will be considered in future developments of this work.

11

Cancer Gene Prediction

Cancer is one of the main threats in modern society, and it is being intensively studied by the research community[1] . Cancer has been associated with mutations in certain genes that lead to an uncontrolled growth of the cells. It is a particular case of genetic disease and, thus, a genome wide search for genes that, when mutated, lead to some type of cancer can be tackled in a similar way as we did the identification of disease genes.

11.1 Biological background and previous work

Cancer is a very particular kind of genetic disease. Actually, it is not just a single disease but a group of diseases with some common features. The main difference with other genetic diseases is that the origin of the latter is exclusively in mutations localised in germinal cells that, thus, are transferred to the offspring while the origin of cancer is mainly associated to mutations in somatic cells.

The common features of all the diseases categorised as cancer is an uncontrolled growth of some abnormal cells that invade and compete with the surrounding natural cells. Eventually, these cells can acquire the capacity of moving and invading other tissues. This is known as metastasis and leads to the proliferation of new, secondary tumors.

The transformation of a normal somatic cell into an abnormal cancer cell is a multi-step process (Vogelstein and Kinzler, 1993), consisting of the accumu-

[1] This is reflected in the literature; The journal with the highest impact factor, 63.342, is *CA: A cancer journal for clinicians*

lation of mutation in some particular genes[2] . In a normal cell the homeostasis and the cellular division are tightly controlled processes where many signals are involved. Changes (mutations) in the genes responsible for these processes can lead to alterations in the equilibrium of the cell that eventually, transforms the cell into an abnormal cell growing out of control.

The mutations leading to cancer only take place in certain genes. Classically these genes are divided into two groups: proto-oncogenes and tumor suppressor genes. The former are proliferative agents, i.e., they stimulate the cellular proliferation. When these genes are mutated or their expression is altered they lead to an uncontrolled growth of the cells[3] . They are dominant because the alteration of one allele is enough to become oncogenic. The tumor suppressor genes are the opposite, anti-proliferative agents that inhibit the cellular growth. When these genes are mutated they lose their function as suppressors and the cells start to proliferate out of control. If only one of the alleles is mutated, the other copy of the gene keeps its function and the cellular growth is controlled. Thus, this kind of gene is normally phenotypically recessive, as both alleles have to be mutated to induce the uncontrolled growth of the cells.

The classical technique to identify cancer genes is the positional cloning (Futreal et al., 2004), but many other molecular techniques, such as fluorescence *in situ* hybridisation (FISH) or comparative genome hybridisation (CGH), are used in the identification of genes related with the cancer process (Baak et al., 2005). cDNA microarrays have also been used to simultaneously analyse the expression of thousands of genes, showing that the expression profiles are clinically relevant (van 't Veer, ????; Dave et al., 2004) and can be used to identify subtypes of cancer that respond differently to the available therapies (Ramaswamy and Golub, 2002). The mutational profiling and large-scale exon resequencing of tumors have also been used to identify genes

[2] Mutations are not the only mechanism that can turn a normal cell into a cancer cell. The recent field of epigenetics is concerned about non-mutational chemical modifications of the DNA that control the expression of proteins. These non-mutational modifications are also behind the underlying mechanisms that can lead to cancer

[3] The expression of some of these genes is sometimes inhibited by non-mutational modifications of the DNA, such as methylation. The loss of this methylation leads to the expression of these genes and, thus, to cancer onset

involved in the cancer process (Benvenuti et al., 2004; Greenman et al., 2007; Wood et al., 2007).

Recently, more sophisticated methods that combine data obtained from different techniques are being used in cancer research (Liu et al., 2006). Some of these works combine data from different microarray experiments (Tomlins et al., 2006), expression and copy number change data (Carter et al., 2006) and expression of mRNAs and microRNAs (Lu et al., 2005).

All this experimentation provides lots of candidate cancer genes. Futreal et al. (2004) published an initial census of cancer genes containing 291 genes, that was further increased to 350 in 2006. Only genes that fulfilled a number of criteria were included in this census, leaving out genes for which there was only evidence of differential expression or aberrant promoter DNA methylation. Given this increasing number of candidate cancer genes it would be interesting, as in the case of disease genes, to rank these candidate genes according to their probability of being related with cancer. Such a tool would be then used by the cancer research community to optimise their work identifying the genes involved in the different types of cancer.

As we have seen in the previous section, proto-oncogenes and tumor suppressor genes have different mechanisms to promote the oncogenesis. Furney et al. (2007) showed that these two groups of cancer genes have different sequence and regulatory properties. In this chapter we will use these properties to build a classification function that, given an unlabelled gene, estimates the probability of being related with cancer.

11.2 Datasets

The list of cancer genes was obtained from the cancer gene census (Futreal et al., 2004) and contained a total of 338 genes. The instances in this list were divided into two subsets, dominant cancer genes (the proto-oncogenes) and recessive cancer genes (tumor suppressor genes) according to the information in the cancer gene census. Thus, two sets of positive cases were defined, cancer dominant (CD, 272 genes) and cancer recessive (CR, 66 genes). The sequences from where the predicting variables were computed were obtained from the NCBI LocusLink database (Pruitt and Maglott, 2001) and the Ensembl database (Hubbard et al., 2002). All the Ensembl genes not included in the list of cancer genes were regarded as unlabelled.

For validation purpose other sets were created. A set of colon cancer genes (CC, 113 genes) and a set of breast cancer genes (BC, 117 genes) were obtained from (Wood et al., 2007). The original work identified 140 genes in each set, but the genes in the list that were used as known positive examples were removed from the original sets. A set of genes not identified as associated with breast cancer or colon cancer (nCB) was also obtained from Wood et al. (2007). Finally, a set of dominant disease genes (DD) and recessive disease genes (DR) were obtained from the OMIM database (Hamosh et al., 2000) as explained in the previous chapter.

11.2.1 Predicting variables

The properties used as predicting variables in the identification of cancer genes can be divided into five sets: Protein conservation (PC), gene structure (GS), protein domains (PD), protein interactions (PI) and regulatory data (RD).

The protein conservation was computed using the conservation score described in the previous chapter. The genomes used to compute the conservation score were *Homo sapiens, Mus musculus, Rattus norvegicus, Canis familiaris, Bos taurus, Monodelphis domestica, Gallus gallus, Xenopus tropicalis, Fugu rubripes, Tetraodon nigroviridis, Danio rerio, Ciona intestinalis, Anopheles gambiae, Apis mellifera, Drosophila melanogaster, Caenorhabditis elegans* and *Saccharomyces cerevisiae*. The conservation score corresponding to the *Homo sapiens* genome is the measure of paralogy explained in the previous chapter.

The gene structure properties were obtained from the Ensembl database (Hubbard et al., 2002) and consist of the length of the coding sequence, the gene length, the total length and number of the exons and the total length of the introns.

The protein domain data consist of Interpro domains (Mulder et al., 2006) present in 30 or more human proteins (a total number of 244 domains). Only the domains over-represented in the list of cancer genes were included. The over-representation was measured obtaining 10,000 random sets of proteins and calculating the z-score for each domain. Only domains with a z-score higher than ± 2.5 were included in the set of predicting variables.

The set of protein interaction features consisted of the total number of interactions per protein and the number of interaction with genes in the list

known cancer genes (DC and RC). The data about protein interactions used to compute these variables were obtained form Jonsson and Bates (2006).

The regulatory data used was promoter conservation, number of promoter CpG islands, 3' UTR length and number of putative microRNA targets. All this information was obtained from a previous study (Furney et al., 2007) based on human-mouse-rat-dog orthologues alignments.

Some sets of variables, such as protein conservation or gene structure contain some redundant variables (for instance, the CS in rat and in mouse). To address this problem, the puCFS algorithm (see Chapter 9) was used to remove the redundant and/or irrelevant variables from these sets. The subset of variables selected by the algorithm was CS in *Mus musculus* and in *Saccharomyces cerevisiae*, the gene length and the total exon length for dominant cancer genes and the coding sequence length, number of exons, CS in *Homo sapiens* (i.e., conservation of paralogues) and the total exon length for recessive cancer genes.

In order to reduce the dimensionality of the set of protein domain variables the puCFS was applied to obtain a subset of relevant, non-redundant variables (nine domains in dominant cancer gene prediction and ten in recessive cancer gene prediction).

For each problem six sets of positive and unlabelled instances were defined, combining the sets of properties described above. Table 11.1 shows a summary of these datasets (note that not all the information is available for all the genes and, thus, the total number of genes in each set is different).

11.3 Results

The four positive Bayesian network classifiers defined in Chapter 7 were applied to the datasets described in the previous section. There are some studies that have suggested that about a 10% of the human genes could be involved in cancer. Given this estimation, the p parameter in PNB and PTAN was set at 0.1 and the α and β parameters in both APNB and APTAN was set at 1.7 and 15.3 respectively, resulting in an average p of 0.1. Given that the results obtained by all of the classifiers were very similar, further analyses were carried only on the results obtained by the APNB. We selected a naive Bayes model because it is more simple than a TAN model. As the number of

Summary of datasets

Dataset	Var. CD	Var. CR	CD	CR	Total
PC.GS	4	4	272	66	22,125
PC.GS.PD	13	14	259	62	16,300
PC.GS.PI	6	6	266	66	14,857
PC.GS.RD	8	8	238	58	13,928
PC.GS.PD.PI.RD	19	20	226	54	11,560

Table 11.1. Summary of the datasets used in the prediction of cancer genes. The columns labelled as 'Var. CD' and 'Var. CR' indicate the number of variables in the sets used for the prediction of dominant and recessive cancer genes respectively. The column labelled as 'CD' indicates the number of known dominant cancer genes, the column labelled as 'CR' indicates the number of known recessive cancer genes and the column labelled as 'Total' indicates the total number of genes (including the positive and the unlabelled) in each set.

parameters to estimate in a naive Bayes model is lower than in TAN models we can be more confident with the estimations.

Due to the absence of negative examples, we cannot estimate performance measures such as the accuracy or the classification error from the data. As a way of assessing the results, the mean probability assigned by the classifier to genes in different sets was checked. These sets are the set of unlabelled genes (UNL) the known cancer genes (CD and CR), the known disease genes (DD and DR), the genes identified as related with colon cancer (CC), the genes identified as related with breast cancer (BC) and the genes that were not identified as related with colon cancer or breast cancer (non-CB).

When estimating the probabilities assigned by the classifier to the list of known cancer genes, we have to bear in mind the problem of overfitting. It is the same problem we have when estimating the performance measures in supervised classification (see Section 2.2.2). In order to obtain a 'fair' estimation of the probabilities assigned by the classifier to the positive examples, we have used a cross validation scheme. The cross validation is used only to estimate the probabilities of the positive cases because, given that the algorithms are not provided with information about the class of the unlabelled

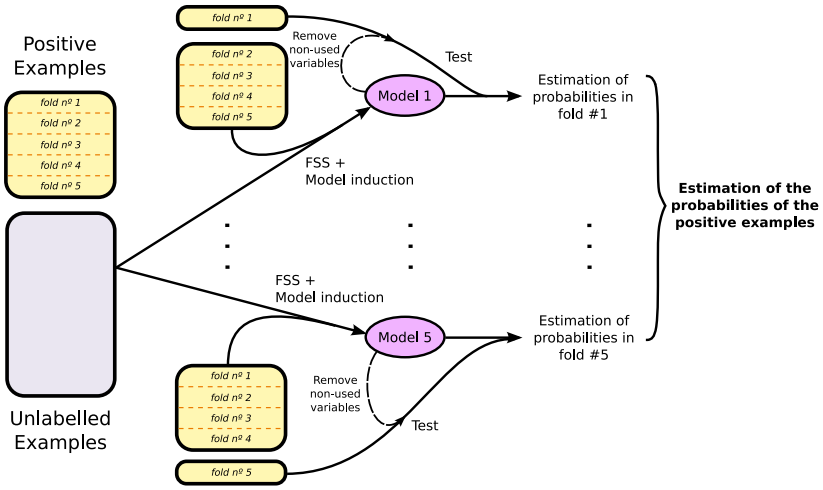


Fig. 11.1. Scheme of a 5-cv process for the estimation of the probabilities assigned by a classifier to the positive examples. The FSS step is included in the cross validation process to avoid overfitting due to the selection of features.

instances, there is no risk of overfitting when estimating their probability of being positive. To avoid the overfitting due to the selection of variables, we have included it in the cross validation scheme (see Figure 11.1).

The probabilities assigned by the classifier to the known dominant cancer genes were estimated using a 10-cv scheme. The probabilities assigned to the known recessive cancer genes were estimated using a 5-cv due to the low number of instances.

The averaged probability assigned by each classifier to the genes in the different sets can be consulted in Table 11.2. As we can see in the table all the classifiers assign an average probability of about 0.5 to the unlabelled instances while the mean probability for the known cancer genes is between 0.71 and 0.83. The set of variables that best discriminate between cancer genes and the rest is the one that contains all the variables, with mean probabilities of 0.75 and 0.83 for dominant and recessive genes respectively. The problem is that we have the value of all the variables for approximately half of the genes. The mean probabilities obtained with the PC-GS dataset (that contains all the genes) is 0.73 and 0.71 for dominant and recessive genes respectively.

Regarding how the classifiers distinguish between cancer and disease genes, we can see that they assign higher probabilities to cancer genes than to disease

Dominant cancer genes

Dataset	UNL	CD	CR	DD	DR	BC	CC	nCB
PC.GS	0.50	0.73	0.71	0.63	0.63	0.78	0.78	0.69
PC.GS.PD	0.50	0.72	0.65	0.57	0.55	0.73	0.73	0.63
PC.GS.PI	0.49	0.73	0.78	0.60	0.55	0.73	0.74	0.63
PC.GS.RD	0.50	0.71	0.58	0.58	0.46	0.69	0.76	0.61
PC.GS.PD.PI.RD	0.49	0.75	0.71	0.59	0.46	0.70	0.74	0.61

Recessive cancer genes

Dataset	UNL	CD	CR	DD	DR	BC	CC	nCB
PC.GS	0.50	0.61	0.71	0.56	0.62	0.75	0.74	0.64
PC.GS.PD	0.50	0.59	0.76	0.53	0.62	0.76	0.74	0.63
PC.GS.PI	0.50	0.67	0.80	0.58	0.59	0.76	0.75	0.64
PC.GS.RD	0.50	0.59	0.73	0.52	0.58	0.75	0.73	0.62
PC.GS.PD.PI.RD	0.50	0.66	0.83	0.55	0.56	0.73	0.71	0.62

Table 11.2. Average probability assigned by APNB (trained with the sets of variables indicated in the 'Dataset' column) to the genes in the different sets. UNL represent the unlabelled instances, CD cancer dominant genes, CR cancer recessive genes, DD disease dominant genes, DR disease recessive genes and BC and CC represent the set of genes associated to breast and colon cancer respectively identified in Wood et al. (2007). nBC represent the genes not identified in Wood et al. (2007) as cancer genes.

genes, though the average probability of disease genes is higher than that of the unlabelled instances. This makes sense as the disease genes share some features with cancer genes. The biggest differences between cancer and disease genes can be found in the classications obtained using all the variables.

As can be seen in Table 11.2 all the classifiers assign also high probabilities to the genes identified as associated to colon and breast cancer (CC and BC). It is important to point out that these genes were not used as positive examples

in the training of the classifiers. Compared to these cancer genes, the genes in the nCB set show a lower average probability. This probability is higher than that obtained for the set of unlabelled examples used in the training (UNL), but this is due to the fact that most of the known cancer genes (CD and CR) are included in this set, increasing the average probability.

As a final evaluation step we have used the learnt classifiers in a comparative oncogenomics study that identified a gene involved in metastasis in melanoma (Kim et al., 2006). In this study the authors reduced the candidate list to the genes in the chromosomic region 6p25-24 and, using expression analysis, the identified Nedd9 as the primary candidate gene. Taking the list of genes in 6p25-24 (a total of 92 genes), Nedd9 is predicted as one of the top candidates by all the classifiers.

11.4 Conclusions

In this chapter we have shown how positive unlabelled learning algorithm can be used to estimate the probability of a gene being associated with cancer. As in the disease gene identification (Chapter 10), the prediction is based on some features related with the sequence of the genes that show significantly different patterns in cancer genes compared with the rest of the genes. Nevertheless, as the identification of cancer genes is more specific than the prediction of disease genes, more variables (such as protein-protein interaction, protein domains and regulatory properties) have been included in the datasets.

In addition to classifier induction algorithms the FSS algorithms for positive unlabelled learning problems presented in Chapter 9 (the puCFS algorithm) was used to remove irrelevant and redundant variables in some sets of properties.

Due to the lack of proper statistical validation procedures the quality of the predictions has been assessed comparing the average probabilities in different groups of genes. We have seen that all the classifiers tend to assign higher probabilities of being positive to the known positive genes compared to the unlabelled genes (to avoid overfitted estimations, a cross validation scheme has been used to estimate the probabilities assigned to the positive cases, including in the process the FSS step). The mean probabilities assigned to the disease genes are also lower than the mean probabilities assigned to the known cancer

genes but higher than the probabilities assigned to the unlabelled genes. This makes sense as cancer genes are a particular case of diseases genes.

We have also tested the estimations in sets of genes identified as related with colon and breast cancer. The classifiers assign to these genes, that were not included as positive examples, probabilities similar to those assigned to the genes used as positive examples. As a final validation step, the estimated probabilities have been used to rank the genes in the 6p25-24 chromosomic region that, in an independent study, was suggested as associated with the metastasis in melanoma. In this study Nedd9 was pointed out as the main candidate to be the gene responsable for the metastasis. This gene was top-ranked by all of our classifiers.

Part IV

Conclusions and Future Work

12

Conclusions and Future Work

The topic of this dissertation is the positive unlabelled learning problem. This is a relatively recent topic in machine learning and, to the best of our knowledge, many interesting issues have not been covered in the literature so far. Indeed almost all the efforts have been focused on developing new algorithms for learning classifiers from positive and unlabelled instances. The main contributions of this dissertation are also concerned about the model induction algorithms but other topics such as FSS or classifier evaluation in absence of negative examples have also been tackled.

Most of the applications of positive unlabelled learning algorithms that can be found in the literature come from the text mining domain. Nevertheless, recent papers explore the use of these kinds of algorithms in other domains such as computational biology or image analysis. In this dissertation we have applied the proposed algorithms in two biomedical problems.

The rest of the chapter is organised as follows. Section 12.1 summarises the contributions of this dissertation in the positive unlabelled learning framework. In Section 12.2 the list of publications obtained during the last four years are provided. Finally, Section 12.3 explores lines of work that this dissertation has left opened.

12.1 Conclusions

The contributions of this dissertation can be divided into two main categories: Methodological contributions and applications.

12.1.1 Methodological contributions

This is the main part of the dissertation. The methodological contributions to the positive unlabelled learning framework include algorithms that cover three main aspects: Classifier induction algorithms, Feature subset selection and classifier evaluation.

12.1.1.1 Classifier induction algorithms

This is the most extensive contribution of this dissertation, with a total of six algorithms. Five of these algorithms are concerned with the problem of learning Bayesian network classifiers from positive and unlabelled examples. We have named this set of algorithms **positive Bayesian network classifiers**, **PBC** (the PNB algorithm developed by Denis et al. (2002) is also included in this set). The list of PBCs proposed in this dissertation is:

- **PTAN** - The positive TAN is the extension of the idea proposed in Denis et al. (2002) to the induction of TAN models (Friedman et al., 1997).
- **APNB and APTAN** - One of the main problems in both PNB and PTAN is setting the a priori probability of the positive class p. Averaged PNB (APNB) and averaged PTAN (APTAN) tackle the problem of the uncertainty about this parameter in a Bayesian way, modelling the a priori probability of p by means of a Beta distribution and then integrating all the estimator in PNB and PTAN for all the possible values of p.
- **wPNB and wPTAN** - The modellisation of the uncertainty about p by means of a Beta distribution provides us with a more flexible way to introduce the previous knowledge about this parameter. However it does not solve the problem when no previous knowledge is available. The wrapper versions of PNB and PTAN try to solve this problem looking for the optimal p in terms of recovery of positive cases. These algorithms are directly linked with the evaluation of classifiers in absence of negative examples, a topic that has been poorly studied in the literature. In Section 12.1.1.3 the contributions of this dissertation in the evaluation of classifiers in absence of negative examples are summarised.

Besides the PBCs, a model averaging algorithm is proposed in this dissertation. The **DCDiv** algorithm takes as input a set of positive and unlabelled instances and, under the assumption that the number of positive cases in the

set of unlabelled instances is much lower than the number of negative cases, it trains and averages several classifiers, providing as output a classification of the unlabelled instances. The main problem with this algorithm compared with the PBCs is that it is computationally expensive. The advantage is that when the starting hypothesis holds (i. e., when there are very few positive examples in the set of unlabelled instances) it provides good results.

12.1.1.2 Feature subset selection

To the best of our knowledge, the dimensionality reduction problem has not been explicitly tackled in the literature. In this dissertation we propose two adaptations of the CFS algorithm (Hall and Smith, 1997) to the positive unlabelled learning framework. The **puCFS** algorithm uses the same concept behind PBCs to estimate the correlation between the features and the class variable, which is the limiting step that makes it impossible to apply the CFS algorithm when no negative examples are available. The **apuCFS** algorithm is the averaged version of puCFS where the estimators are averaged in the same way as in APNB and APTAN. Both algorithms have been tested, showing that they give good results filtering out irrelevant and redundant variables, provided that we have enough positive examples to make a good estimation of the correlation between features and the class variable.

12.1.1.3 Classifier evaluation

The lack of negative examples not only affects the learning of classifiers but also the estimation of performance measures. As far as we know, the only work where the problem of evaluating classifiers in positive unlabelled learning problems is considered is Lee and Liu (2003). In this dissertation we have proposed a positive unlabelled estimator of the F measure. The problem is that this estimator depends on p, which is generally unknown. Besides this estimator we have proposed a new metric, the **pseudo F**, whose behaviour is similar to that of the F measure but does not depend that much on p. The pseudo F measure can be used to identify the optimal p (the p that maximises the F measure). The experimentation in synthetic datasets suggests that this optimal p can be considered as an estimator of the actual p with a positive bias and, thus, it can be used to estimate the actual F. The estimator of the actual F also depends on the estimation of the recall, which has normally a

negative bias. The result is that the biases in the estimations of the recall and p are somehow compensated in the estimation of the F measure.

12.1.2 Applications

Most of the papers that model the information retrieval as a positive and unlabelled learning problems come from the text mining domain. Nevertheless, recent works have shown that many other domains can take advantage of the positive unlabelled learning algorithms.

In this dissertation we have tackled two computational biology problems from the positive unlabelled learning point of view. The first problem is the **genome-wide prediction of disease genes**. The goal of this application is to provide the research community with a tool that, given a set of genes, ranks them according to the probability of being associated to a genetic disease. There are databases from where one can obtain a list of known disease genes, but we do not know whether the rest of the genes are disease related or not. Therefore, the prediction is clearly a positive unlabelled learning problem.

The second problem is the **prediction of cancer genes**. Conceptually it is very similar to the identification of disease genes, as we have some genes that we know are related with cancer but for the rest we cannot say whether they are cancer genes or not. The main difference is the variables used in the prediction. In the disease gene identification the prediction is based on some sequence properties that have been identified as significantly different in known disease genes and the rest. These variables alone are not enough to distinguish between cancer and non-cancer genes and thus new variables such as regulatory properties or protein-protein interactions have to be introduced.

12.2 Publications

The list of publications obtained during the last four years are:

- I. Inza, **B. Calvo**, R. Armañanzas, E. Bengoetxea, P. Larrañaga and J. A. Lozano (to appear in 2009) Machine learning: An indispensable tool in bioinformatics. *Bioinformatics in Clinical OMICs Research* (Rune Mathiessen Ed.), Humana Press.

- **B. Calvo**, J. A. Lozano and P. Larrañaga (2008) Feature subset selection from positive and unlabelled examples. *Pattern Recognition Letters* Submitted.
- S. J. Furney, **B. Calvo**, P. Larrañaga, J. A. Lozano and N. López-Bigas (2008) Prioritization of candidate cancer genes – an aid to oncogenomic studies. *Nucleic Acids Research*, doi:10.1093/nar/gkn482.
- R. Armañanzas, **B. Calvo**, I. Inza, P. Larrañaga, I. Bernales, A. Fullaondo and A. M. Zubiaga (2008) Bayesian classifiers with consensus gene selection: A case study in Systemic Lupus Erythematosus. *Progress in Industrial Mathematics at ECMI 2006*, 560-565.
- **B. Calvo**, P. Larrañaga and J. A. Lozano (2007) Learnign Bayesian classifiers from positive and unlabeled examples. *Pattern Recognition Letters* 28(16); 2375-2384.
- **B. Calvo**, N. López-Bigas, S. J. Furney, P. Larrañaga and J. A. Lozano (2007) A partially supervised classification approach to dominant and recessive human disease gene prediction. *Computer Methods and Programs in Biomedicine* 85(3); 229-237.
- P. Larrañaga, **B. Calvo**, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé, A. Pérez and V. Robles (2006) *Machine learning in Bioinformatics*. Briefings in Bioinformatics 7; 86-112.
- **B. Calvo**, P. Larrañaga and J.A. Lozano (2005) Clasificación binaria en ausencia de ejemplos negativos. *I Workshop sobre Reconocimiento de Formas y Análisis de Imágenes*. Granada, Spain, 3-12.
- R. Armañanzas, **B. Calvo**, I. Inza, P. Larrañaga, I. Bernales, A. Fullaondo and A. M. Zubiaga (2005) Selección de genes asociados a dos enfermedades autoinmunes a partir de microarrays de ADN. *Proceedings of the Sixth Spanish Symposium on Technology Transfer from Artificial Intelligence, TTIA'2005*. Granada, Spain, 63 - 70.
- R. Armañanzas, **B. Calvo**, I. Inza, P. Larrañaga, I. Bernales, A. Fullaondo and A.M. Zubiaga (2005) Clasificadores Bayesianos con selección consensuada de genes en la predicción del lupus eritematoso sistémico. *Minería de Datos: Técnicas y Aplicaciones* (José A. Gámez, Ismael García Varela y José Hernández Orallo, eds.), Ediciones Departamento de Informática de la Universidad de Castilla La Mancha, 107-135.

12.3 Future work

The positive unlabelled learning field is relatively new and there is room for many new developments. The contributions in this dissertation have left different lines of work opened.

Starting from the PNB, in this dissertation we have proposed five new algorithms to learn Bayesian network classifiers from positive and unlabelled examples. A future line of work would be the extension of the family of PBCs. The work presented in this dissertation can be used to extend this family in different ways. Firstly, more complex networks can be added to the family adapting the k-DB algorithm (Sahami, 1996). Secondly, the pseudo F measure proposed in Chapter 8 can be used in wrapper approaches of algorithms such as the seminaive Bayes (Kononenko, 1991; Pazzani, 1997) or selective naive Bayes (Kohavi and John, 1997). Finally, the wrapper PBCs proposed in this dissertation are based on PNB and PTAN. It would also be interesting exploring the wrapper versions of APNB and APTAN. In these algorithms we would need to set in a wrapper way two parameters, α and β. Therefore, we would need to look for a maximum in a 3D representation of the pseudo F measure vs. α and β.

Regarding the DCDiv algorithm, there are a number of questions that can be explored in future works. It would be interesting to analyse the algorithm from a theoretical point of view. The two main drawbacks of the algorithm are the computational time and the lack of a model at the end of the algorithm. Future developments of this algorithm or new model averaging procedures should cope with these two problems.

The FSS has been tackled in this dissertation from a filter point of view. Nevertheless, the pseudo F measure used in the wrapper PBCs could also be used in wrapper feature subset selection algorithms. As we saw in Chapter 9, the stability of the selection is reduced when few positive examples are available. Consensus selection techniques are useful in these situations to increase the robustness of the selection. An interesting line of future work would be the use of consensus selection with (a)puCFS algorithms when few positive examples are available.

In this dissertation we have focused on the applications in computational biology problems, but there are lots of possible applications of these algorithms. Indeed, these algorithms are interesting not only when we lack negative

examples, but also when getting a set of representative negative examples is not possible, a relatively frequent situation in information retrieval problems. Another line of future work could be the identification of real-life problems suitable of being modelled as a positive unlabelled learning problem.

Part V

Appendices

A

Experimental evaluation of the DCDiv algorithm

This appendix contains all the results obtained in the experimental evaluation of the DCDiv. The tables in the following pages show the results (in terms of F measure and accuracy) obtained by DCDiv and PNB when the p parameter was set at 0.50, 0.25 and the actual value (as the actual value is unknown, we use the ratio of positive cases in $\mathcal{D}_u$, which is an estimation of the a priori probability of the positive class). The column labelled as '$N_{\mathcal{D}_p}$' corresponds to the size of the set of positive cases and the column labelled as 'rat.' is the ratio of positive cases in $\mathcal{D}_u$. The values shown are the average of the results obtained in the fifty positive unlabelled learning problems sampled using each scheme. The significance of the differences was assessed by Wilcoxon paired tests. The bold font indicates the winner when the differences were significant (at a significance level of 1%).

UCI datasets - F measure

			$p = 0.50$		$p = 0.25$		Actual p	
	$N_{\mathcal{D}_p}$	rat.	PNB	DCDiv	PNB	DCDiv	PNB	DCDiv
Letter Recognition D	100	0.05	18.56	**53.79**	36.80	**53.79**	**58.26**	53.79
	100	0.13	42.98	**64.62**	66.07	64.62	**69.68**	64.62
	100	0.20	58.42	**66.30**	**73.93**	66.30	**74.08**	66.30
	200	0.05	18.98	**67.09**	37.05	**67.09**	62.48	**67.09**
	200	0.13	44.08	**69.35**	66.67	**69.35**	**72.71**	69.35
	200	0.20	59.64	**71.82**	**76.70**	71.82	**77.80**	71.82
	300	0.05	19.28	**70.50**	36.49	**70.50**	61.64	**70.50**
	300	0.13	45.00	**73.02**	66.27	**73.02**	72.85	73.02
	300	0.20	60.18	**74.60**	76.68	74.60	**77.96**	74.60
Letter Recognition P	100	0.05	21.34	**68.19**	43.42	**68.19**	**72.18**	68.19
	100	0.13	48.26	**70.65**	74.01	70.65	**80.14**	70.65
	100	0.20	62.93	**70.56**	**81.67**	70.56	**82.53**	70.56
	200	0.05	21.30	**73.01**	44.98	**73.01**	73.32	73.01
	200	0.13	48.46	**73.14**	74.86	73.14	**81.61**	73.14
	200	0.20	64.82	**73.45**	**83.47**	73.45	**84.54**	73.45
	300	0.05	21.61	**75.73**	46.11	**75.73**	73.07	**75.73**
	300	0.13	49.46	**75.04**	75.92	75.04	**82.25**	75.04
	300	0.20	65.38	**76.22**	**84.04**	76.22	**85.21**	76.22
Letter Recognition U	100	0.05	17.31	**69.71**	45.93	**69.71**	**74.45**	69.71
	100	0.13	40.60	**74.40**	72.88	74.40	**78.39**	74.40
	100	0.20	57.40	**73.71**	**78.99**	73.71	**79.54**	73.71
	200	0.05	18.24	**74.83**	47.96	**74.83**	76.59	74.83
	200	0.13	42.28	**76.66**	75.10	76.66	**80.78**	76.66
	200	0.20	57.00	**75.33**	**81.62**	75.33	**82.41**	75.33
	300	0.05	18.74	**77.79**	49.23	**77.79**	78.31	77.79
	300	0.13	42.90	**78.01**	76.60	78.01	**82.12**	78.01
	300	0.20	57.98	**76.61**	**82.92**	76.61	**83.61**	76.61
Nursery	250	0.01	3.94	**55.34**	20.80	**55.34**	00.00	**55.34**
	250	0.08	25.45	**71.76**	69.08	**71.76**	20.84	**71.76**
	250	0.25	65.85	**73.96**	74.72	73.96	75.16	73.96
	500	0.01	4.12	**66.26**	21.37	**66.26**	00.00	**66.26**
	500	0.08	26.76	**71.97**	70.36	71.97	21.18	**71.97**
	500	0.25	65.33	**72.53**	**75.22**	72.53	**75.68**	72.53
	1,000	0.01	4.37	**65.62**	21.90	**65.62**	00.00	**65.62**
	1,000	0.08	27.48	**71.10**	71.13	71.10	20.92	**71.10**
	1,000	0.25	64.36	**69.94**	**75.88**	69.94	**76.33**	69.94

Table A.1. Comparison of the F measure obtained by DCDiv and PNB in datasets obtained from UCI databases (Letter Recognition and Nursery).

UCI datasets - Accuracy

			$p = 0.50$		$p = 0.25$		Actual p	
	$N_{\mathcal{D}_p}$	rat.	PNB	DCDiv	PNB	DCDiv	PNB	DCDiv
Letter Recognition D	100	0.05	59.21	**94.95**	85.73	**94.95**	**96.25**	94.95
	100	0.13	66.14	**91.21**	88.77	**91.21**	**92.28**	91.21
	100	0.20	72.42	**88.56**	88.99	88.56	**89.86**	88.56
	200	0.05	59.96	**97.09**	85.10	**97.09**	96.10	**97.09**
	200	0.13	67.37	**93.58**	88.40	**93.58**	92.47	**93.58**
	200	0.20	73.54	**90.85**	89.66	**90.85**	90.79	90.85
	300	0.05	60.89	**97.55**	84.70	**97.55**	95.89	**97.55**
	300	0.13	68.61	**94.27**	88.14	**94.27**	92.29	**94.27**
	300	0.20	74.02	**91.72**	89.41	**91.72**	90.61	**91.72**
Letter Recognition P	100	0.05	67.04	**97.18**	88.88	**97.18**	97.18	97.18
	100	0.13	73.77	**93.79**	92.03	**93.79**	**94.83**	93.79
	100	0.20	78.09	**90.56**	**92.40**	90.56	**93.10**	90.56
	200	0.05	67.01	**97.89**	89.57	**97.89**	97.25	**97.89**
	200	0.13	74.03	**94.45**	92.30	**94.45**	**95.14**	94.45
	200	0.20	79.69	**91.65**	**93.17**	91.65	**93.86**	91.65
	300	0.05	67.59	**98.04**	89.99	**98.04**	97.21	**98.04**
	300	0.13	74.96	**94.81**	92.68	**94.81**	95.29	94.81
	300	0.20	80.20	**92.32**	**93.40**	92.32	**94.11**	92.32
Letter Recognition U	100	0.05	57.18	**97.42**	90.82	**97.42**	97.18	97.42
	100	0.13	64.08	**94.53**	92.09	**94.53**	**94.83**	94.53
	100	0.20	72.56	**91.52**	91.71	91.52	**93.10**	91.52
	200	0.05	59.08	**98.04**	91.29	**98.04**	97.25	98.04
	200	0.13	66.08	**95.08**	92.83	**95.08**	**95.14**	95.08
	200	0.20	71.69	**92.17**	92.80	92.17	**93.86**	92.17
	300	0.05	60.12	**98.27**	91.54	**98.27**	97.21	**98.27**
	300	0.13	66.66	**95.32**	93.35	**95.32**	95.29	95.32
	300	0.20	72.66	**92.47**	**93.31**	92.47	**94.11**	92.47
Nursery	250	0.01	45.58	**98.71**	92.42	**98.71**	98.89	98.71
	250	0.08	54.27	**96.28**	94.18	**96.28**	93.14	**96.28**
	250	0.25	73.96	**89.26**	89.35	89.26	89.48	89.26
	500	0.01	48.15	**99.35**	92.55	**99.35**	98.89	**99.35**
	500	0.08	57.39	**96.57**	94.50	**96.57**	93.16	**96.57**
	500	0.25	73.25	**89.11**	89.57	89.11	89.70	89.11
	1,000	0.01	51.34	**99.42**	92.78	**99.42**	98.89	**99.42**
	1,000	0.08	58.96	**96.54**	94.66	**96.54**	93.15	**96.54**
	1,000	0.25	72.74	**88.56**	**89.93**	88.56	**90.07**	88.56

Table A.2. Comparsion of the accuracy obtained by DCDiv and PNB in datasets obtained from UCI databases (Letter Recognition and Nursery).

ACCDON datasets - F measure

		$N_{\mathcal{D}_p}$	rat.	$p = 0.50$ PNB	$p = 0.50$ DCDiv	$p = 0.25$ PNB	$p = 0.25$ DCDiv	Actual p PNB	Actual p DCDiv
Accpetor Sites	Coding	100	0.05	12.58	**66.81**	43.30	**66.81**	**71.77**	66.81
		100	0.13	34.02	**74.77**	74.70	74.77	**80.87**	74.77
		100	0.20	51.83	**75.43**	**83.95**	75.43	**84.75**	75.43
		200	0.05	12.59	**66.44**	44.65	**66.44**	**72.23**	66.44
		200	0.13	33.60	**76.21**	75.69	76.21	**82.09**	76.21
		200	0.20	52.98	**73.89**	**84.93**	73.89	**85.67**	73.89
		300	0.05	12.57	**68.81**	44.36	**68.81**	**73.26**	68.81
		300	0.13	33.46	**71.98**	**75.59**	71.98	**81.94**	71.98
		300	0.20	53.50	**77.58**	**85.26**	77.58	**86.13**	77.58
	Intron	100	0.05	15.39	**51.47**	38.18	**51.47**	**54.67**	51.47
		100	0.13	40.43	**64.97**	66.07	64.97	**69.31**	64.97
		100	0.20	58.29	**65.37**	**75.19**	65.37	**75.46**	65.37
		200	0.05	15.78	**53.95**	38.53	**53.95**	**56.37**	53.95
		200	0.13	41.85	**66.91**	67.03	66.91	**71.36**	66.91
		200	0.20	59.77	**71.21**	**76.54**	71.21	**77.02**	71.21
		300	0.05	15.75	**54.55**	38.29	**54.55**	**56.43**	54.55
		300	0.13	42.21	**66.05**	66.87	66.05	**71.45**	66.05
		300	0.20	59.89	**69.31**	**77.12**	69.31	**77.64**	69.31
	Mixed	100	0.05	13.73	**56.82**	41.02	**56.82**	**59.90**	56.82
		100	0.13	37.79	**67.91**	69.90	67.91	**73.93**	67.91
		100	0.20	57.52	**70.89**	**78.98**	70.89	**79.29**	70.89
		200	0.05	13.56	**58.31**	41.53	**58.31**	**60.70**	58.31
		200	0.13	39.48	**69.52**	70.69	69.52	**75.78**	69.52
		200	0.20	58.60	**70.42**	**79.86**	70.42	**80.47**	70.42
		300	0.05	13.78	**59.25**	41.58	**59.25**	**61.62**	59.25
		300	0.13	40.01	**68.02**	**70.88**	68.02	**75.88**	68.02
		300	0.20	59.05	**69.97**	**80.32**	69.97	**80.92**	69.97
Donor Sites	Coding	100	0.05	14.85	**67.61**	39.23	**67.61**	67.86	67.61
		100	0.13	36.41	**74.10**	75.87	74.10	**81.49**	74.10
		100	0.20	52.57	**75.18**	**86.49**	75.18	**85.95**	75.18
		200	0.05	15.10	**67.43**	39.30	**67.43**	69.71	67.43
		200	0.13	36.37	**75.57**	76.48	75.57	**81.66**	75.57
		200	0.20	52.29	**72.91**	**86.96**	72.91	**86.39**	72.91
		300	0.05	14.95	**70.04**	39.73	**70.04**	69.27	70.04
		300	0.13	36.13	**72.73**	**76.47**	72.73	**81.84**	72.73
		300	0.20	52.33	**74.26**	**87.45**	74.26	**86.65**	74.26
	Intron	100	0.05	15.52	**72.54**	51.33	**72.54**	**78.53**	72.54
		100	0.13	36.94	**77.66**	79.37	77.66	**87.32**	77.66
		100	0.20	52.49	**77.08**	**88.91**	77.08	**90.65**	77.08
		200	0.05	15.73	**75.58**	51.96	**75.58**	**79.00**	75.58
		200	0.13	36.15	**78.27**	79.85	78.27	**88.17**	78.27
		200	0.20	51.92	**80.02**	**88.83**	80.02	**90.85**	80.02
		300	0.05	15.77	**76.23**	52.10	**76.23**	**79.07**	76.23
		300	0.13	36.50	**80.22**	80.18	80.22	**88.13**	80.22
		300	0.20	52.47	**80.90**	**88.65**	80.90	**91.10**	80.90
	Mixed	100	0.05	15.00	**72.40**	45.21	**72.40**	73.01	72.40
		100	0.13	35.09	**74.63**	73.36	74.63	**83.78**	74.63
		100	0.20	51.55	**74.05**	**87.41**	74.05	**87.63**	74.05
		200	0.05	15.14	**71.84**	45.42	**71.84**	73.48	71.84
		200	0.13	35.15	**78.28**	73.36	**78.28**	**84.15**	78.28
		200	0.20	50.66	**77.67**	**88.22**	77.67	**88.51**	77.67
		300	0.05	15.11	**73.30**	45.74	**73.30**	73.99	73.30
		300	0.13	35.40	**78.00**	73.42	**78.00**	**84.40**	78.00
		300	0.20	49.70	**78.53**	**88.57**	78.53	**88.49**	78.53

Table A.3. Comparison of the F measure obtained by DCDiv and PNB in datasets obtained from ACCDON database.

ACCDON datasets - Accuracy

		$N_{\mathcal{D}_p}$	rat.	$p = 0.50$ PNB	$p = 0.50$ DCDiv	$p = 0.25$ PNB	$p = 0.25$ DCDiv	Actual p PNB	Actual p DCDiv
Accpetor Sites	Coding	100	0.05	33.85	**97.06**	88.09	**97.06**	97.30	97.06
		100	0.13	49.31	**94.29**	91.84	**94.29**	**94.97**	94.29
		100	0.20	62.86	**91.87**	**93.16**	91.87	**93.88**	91.87
		200	0.05	33.83	**97.19**	88.63	**97.19**	97.34	97.19
		200	0.13	48.44	**94.58**	92.17	**94.58**	**95.29**	94.58
		200	0.20	64.50	**91.58**	**93.54**	91.58	**94.20**	91.58
		300	0.05	33.77	**97.14**	88.52	**97.14**	**97.42**	97.14
		300	0.13	48.13	**94.08**	92.13	**94.08**	**95.21**	94.08
		300	0.20	65.25	**92.50**	**93.68**	92.50	**94.37**	92.50
	Intron	100	0.05	47.66	**94.21**	85.79	**94.21**	**95.41**	94.21
		100	0.13	61.77	**91.61**	88.24	**91.61**	91.67	91.61
		100	0.20	71.82	**88.57**	88.94	88.57	**89.83**	88.57
		200	0.05	49.15	**95.02**	85.83	**95.02**	95.48	95.02
		200	0.13	63.95	**92.14**	88.47	**92.14**	92.05	92.14
		200	0.20	73.40	**89.84**	89.37	**89.84**	**90.30**	89.84
		300	0.05	49.14	**95.33**	85.83	**95.33**	95.47	95.33
		300	0.13	64.49	**92.00**	88.39	**92.00**	92.07	92.00
		300	0.20	73.56	**89.58**	89.65	89.58	**90.55**	89.58
	Mixed	100	0.05	40.07	**95.50**	87.13	**95.50**	95.96	95.50
		100	0.13	56.98	**92.32**	89.83	**92.32**	**92.93**	92.32
		100	0.20	70.60	**90.04**	90.70	90.04	**91.43**	90.04
		200	0.05	39.19	**95.91**	87.29	**95.91**	95.98	95.91
		200	0.13	60.00	**92.97**	90.04	**92.97**	**93.37**	92.97
		200	0.20	71.92	**90.11**	**91.06**	90.11	**91.85**	90.11
		300	0.05	40.30	**95.98**	87.29	**95.98**	96.06	95.98
		300	0.13	60.92	**92.92**	90.10	**92.92**	**93.37**	92.92
		300	0.20	72.50	**90.12**	**91.26**	90.12	**92.03**	90.12
Donor Sites	Coding	100	0.05	45.31	**96.96**	85.42	**96.96**	**97.34**	96.96
		100	0.13	54.33	**94.26**	92.03	**94.26**	**95.38**	94.26
		100	0.20	63.84	**91.83**	**94.29**	91.83	**94.49**	91.83
		200	0.05	46.38	**97.23**	85.45	**97.23**	**97.49**	97.23
		200	0.13	54.29	**94.52**	92.26	**94.52**	**95.45**	94.52
		200	0.20	63.32	**91.30**	**94.47**	91.30	**94.64**	91.30
		300	0.05	45.87	**97.22**	85.75	**97.22**	**97.41**	97.22
		300	0.13	53.80	**94.19**	92.24	**94.19**	**95.47**	94.19
		300	0.20	63.52	**91.62**	**94.66**	91.62	**94.73**	91.62
	Intron	100	0.05	48.09	**97.67**	91.08	**97.67**	**98.13**	97.67
		100	0.13	55.26	**95.16**	93.43	**95.16**	**96.79**	95.16
		100	0.20	63.68	**92.62**	**95.26**	92.62	**96.30**	92.62
		200	0.05	49.00	**97.96**	91.29	**97.96**	**98.17**	97.96
		200	0.13	53.87	**95.29**	93.59	**95.29**	**96.99**	95.29
		200	0.20	62.91	**93.34**	**95.20**	93.34	**96.37**	93.34
		300	0.05	49.14	**97.93**	91.32	**97.93**	**98.16**	97.93
		300	0.13	54.61	**95.61**	93.73	**95.61**	**96.98**	95.61
		300	0.20	63.73	**93.61**	**95.11**	93.61	**96.46**	93.61
	Mixed	100	0.05	45.98	**97.54**	88.60	**97.54**	**97.69**	97.54
		100	0.13	51.67	**94.51**	90.83	**94.51**	**95.92**	94.51
		100	0.20	62.29	**91.64**	**94.66**	91.64	**95.12**	91.64
		200	0.05	46.68	**97.48**	88.72	**97.48**	**97.74**	97.48
		200	0.13	51.89	**95.07**	90.78	**95.07**	**96.01**	95.07
		200	0.20	60.96	**92.56**	**95.00**	92.56	**95.45**	92.56
		300	0.05	46.48	**97.61**	88.87	**97.61**	**97.77**	97.61
		300	0.13	52.39	**95.03**	90.83	**95.03**	**96.07**	95.03
		300	0.20	59.49	**92.80**	**95.14**	92.80	**95.45**	92.80

Table A.4. Comparison of the accuracy obtained by DCDiv and PNB in datasets obtained from ACCDON database.

B

Experimental evaluation of the positive Bayesian network classifiers

This appendix contains the complete set of results obtained by PNB, PTAN, APNB and APTAN in both real-life based and synthetic datasets. The results comprise the F measure and the accuracy obtained by each classifier. The significance of the differences between every pair of algorithms has been assessed running Wilcoxon paired tests on the results obtained on the 100 problems sampled using each scheme (see Chapter 5).

The first column in each table represents the number of known positive examples and the second column the ratio of positive cases hidden in $\mathcal{D}_u$. The column labelled as 'Wlx.' shows the results of the Wilcoxon test (with a significance level of 1%). The results of the six tests in each row are codified as follows:

- a - no significant differences between PNB and APNB
- b - no significant differences between PNB and PTAN
- c - no significant differences between PNB and APTAN
- d - no significant differences between APNB and PTAN
- e - no significant differences between APNB and APTAN
- f - no significant differences between PTAN and APTAN

Acceptor Sites - F measure

	$N_{\mathcal{D}_p}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
Coding	100	0.01	11.49	11.90	11.88	11.58	c,d
	100	0.10	67.21	66.26	63.87	63.41	f
	100	0.20	83.86	83.46	76.69	76.91	f
	100	0.30	86.94	87.18	77.15	77.75	-
	100	0.40	84.38	84.97	71.41	72.60	-
	100	0.50	78.55	79.35	62.47	64.20	-
	500	0.01	11.39	12.26	12.43	12.14	d
	500	0.10	68.59	66.91	68.12	66.76	e
	500	0.20	85.22	84.86	84.29	83.32	-
	500	0.30	88.88	89.02	87.79	87.17	-
	500	0.40	87.49	87.95	86.21	85.88	-
	500	0.50	82.64	83.36	81.12	81.17	f
	1,000	0.01	11.37	12.32	12.66	12.29	e
	1,000	0.10	68.77	67.12	69.03	67.31	b,e
	1,000	0.20	85.34	84.96	85.25	83.95	b
	1,000	0.30	89.05	89.21	89.08	88.48	b,d
	1,000	0.40	87.79	88.26	88.28	87.85	c,d
	1,000	0.50	83.13	83.85	84.15	83.69	e
Intron	100	0.01	10.42	10.01	10.90	10.03	e
	100	0.10	58.85	58.04	55.87	54.87	-
	100	0.20	75.33	75.16	67.45	67.24	f
	100	0.30	79.81	80.09	67.70	68.56	-
	100	0.40	78.17	78.77	62.22	63.36	-
	100	0.50	72.61	73.46	54.33	55.94	-
	500	0.01	10.67	10.26	11.99	10.90	-
	500	0.10	59.88	59.01	63.11	60.71	-
	500	0.20	77.10	76.85	78.44	76.92	c,e
	500	0.30	82.47	82.64	82.09	81.49	b
	500	0.40	82.09	82.56	80.43	80.04	-
	500	0.50	77.73	78.46	74.81	74.36	-
	1,000	0.01	10.73	10.30	12.24	11.07	-
	1,000	0.10	59.93	59.12	64.06	61.56	-
	1,000	0.20	77.53	77.27	80.11	78.54	-
	1,000	0.30	82.85	83.02	84.39	83.51	-
	1,000	0.40	82.52	82.98	83.52	82.82	e
	1,000	0.50	78.41	79.16	78.82	78.04	d
Mixed	100	0.01	11.57	10.99	11.76	10.78	-
	100	0.10	62.70	61.71	59.39	58.07	-
	100	0.20	79.05	78.83	71.41	71.23	f
	100	0.30	82.83	83.08	71.06	71.81	-
	100	0.40	80.86	81.45	65.55	66.98	-
	100	0.50	74.93	75.80	57.28	59.20	-
	500	0.01	11.70	11.12	12.61	11.33	-
	500	0.10	63.69	62.66	65.21	63.08	-
	500	0.20	80.65	80.38	80.67	79.45	b
	500	0.30	85.41	85.58	84.52	83.84	-
	500	0.40	84.47	84.97	82.90	82.57	-
	500	0.50	79.66	80.42	77.04	76.84	f
	1,000	0.01	11.78	11.19	12.86	11.45	-
	1,000	0.10	63.73	62.68	65.77	63.75	c
	1,000	0.20	80.85	80.54	81.79	80.64	e
	1,000	0.30	85.63	85.81	86.24	85.49	c
	1,000	0.40	85.08	85.55	85.89	85.11	c
	1,000	0.50	80.53	81.26	81.22	80.49	c,d

Table B.1. Comparison of the F measure obtained by PNB, PTAN, APNB and APTAN in datasets sampled from the database of acceptor sites.

Donor Sites - F measure

	$N_{\mathcal{D}_p}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
Coding	100	0.01	8.80	9.38	9.71	9.58	d,e,f
	100	0.10	66.79	64.36	64.41	63.02	d
	100	0.20	84.32	84.07	78.08	79.27	-
	100	0.30	85.74	86.24	77.12	79.78	-
	100	0.40	81.21	82.11	67.31	72.06	-
	100	0.50	71.66	72.85	52.87	59.07	-
	500	0.01	8.02	9.45	8.28	9.31	e
	500	0.10	67.77	65.25	60.14	62.65	-
	500	0.20	85.37	85.15	80.36	82.31	-
	500	0.30	87.08	87.60	84.74	85.16	f
	500	0.40	82.83	83.66	76.67	80.90	-
	500	0.50	74.33	75.49	64.33	70.91	-
	1,000	0.01	7.97	9.42	7.86	9.28	b,e
	1,000	0.10	67.80	65.22	58.56	61.80	-
	1,000	0.20	85.59	85.32	80.67	82.66	-
	1,000	0.30	87.16	87.73	86.01	86.05	f
	1,000	0.40	83.05	83.86	78.21	81.99	-
	1,000	0.50	74.80	75.98	66.15	72.58	-
Intron	100	0.01	10.15	8.97	10.19	9.55	b
	100	0.10	64.16	62.55	62.52	61.41	d
	100	0.20	80.98	80.85	75.88	77.20	-
	100	0.30	83.31	83.80	74.53	77.77	-
	100	0.40	79.22	80.03	66.48	70.63	-
	100	0.50	70.12	71.22	51.66	57.47	-
	500	0.01	10.16	8.80	6.98	9.71	-
	500	0.10	64.68	62.95	55.15	61.65	-
	500	0.20	82.40	82.26	80.44	80.04	f
	500	0.30	84.52	84.99	82.80	83.14	f
	500	0.40	81.07	81.78	78.93	79.45	f
	500	0.50	73.25	74.36	67.90	70.03	-
	1,000	0.01	10.15	8.82	6.71	9.65	-
	1,000	0.10	64.75	62.96	53.12	61.76	-
	1,000	0.20	82.57	82.43	79.95	80.08	f
	1,000	0.30	84.59	85.14	83.84	83.63	f
	1,000	0.40	81.36	82.10	81.11	80.65	b
	1,000	0.50	73.77	74.81	71.73	71.41	f
Mixed	100	0.01	10.17	8.86	10.46	9.51	-
	100	0.10	65.79	63.94	63.74	62.30	d
	100	0.20	82.61	82.42	77.26	78.36	-
	100	0.30	84.41	84.93	75.47	78.39	-
	100	0.40	79.91	80.76	66.14	70.80	-
	100	0.50	70.65	71.88	52.36	58.53	-
	500	0.01	10.19	8.86	7.20	9.26	-
	500	0.10	66.39	64.45	59.91	61.86	-
	500	0.20	83.80	83.62	80.10	81.46	-
	500	0.30	85.55	86.07	83.37	84.29	-
	500	0.40	81.77	82.59	77.59	79.99	-
	500	0.50	73.13	74.38	64.72	70.34	-
	1,000	0.01	10.17	8.84	6.67	9.16	-
	1,000	0.10	66.69	64.64	56.19	62.29	-
	1,000	0.20	84.09	83.89	80.37	81.10	-
	1,000	0.30	85.63	86.22	84.92	84.72	f
	1,000	0.40	81.88	82.64	80.14	80.67	f
	1,000	0.50	73.90	75.09	68.79	72.12	-

Table B.2. Comparison of the F measure obtained by PNB, PTAN, APNB and APTAN in datasets sampled from the database of donor sites.

Acceptor Sites - Accuracy

	$N_{\mathcal{D}_p}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
Coding	100	0.01	85.07	85.64	86.18	85.57	e
	100	0.10	90.84	90.40	90.10	89.66	d
	100	0.20	93.11	92.85	90.34	90.25	f
	100	0.30	92.44	92.52	87.46	87.62	f
	100	0.40	88.79	89.13	81.08	81.61	-
	100	0.50	82.06	82.60	71.65	72.57	-
	500	0.01	84.87	86.06	86.37	85.95	e
	500	0.10	91.31	90.57	91.18	90.60	b,e
	500	0.20	93.65	93.42	93.24	92.71	-
	500	0.30	93.47	93.51	92.82	92.40	-
	500	0.40	90.81	91.09	89.87	89.59	-
	500	0.50	85.01	85.53	83.75	83.76	f
	1,000	0.01	84.88	86.17	86.62	86.11	e
	1,000	0.10	91.37	90.65	91.48	90.76	b
	1,000	0.20	93.70	93.47	93.64	92.95	b
	1,000	0.30	93.57	93.62	93.52	93.12	b
	1,000	0.40	91.01	91.30	91.24	90.90	c,d
	1,000	0.50	85.38	85.90	86.02	85.64	d
Intron	100	0.01	84.10	83.26	85.93	84.43	-
	100	0.10	87.48	86.95	87.57	86.84	b,e
	100	0.20	88.96	88.74	86.55	86.19	-
	100	0.30	88.19	88.23	82.75	82.94	f
	100	0.40	84.51	84.80	75.94	76.33	-
	100	0.50	77.67	78.17	66.83	67.54	-
	500	0.01	84.25	83.44	86.19	84.61	-
	500	0.10	87.73	87.19	89.28	88.16	-
	500	0.20	89.64	89.39	90.45	89.63	c
	500	0.30	89.53	89.54	89.44	89.01	a,b,d
	500	0.40	86.89	87.13	85.88	85.55	-
	500	0.50	81.13	81.61	79.08	78.69	-
	1,000	0.01	84.27	83.45	86.40	84.70	-
	1,000	0.10	87.75	87.24	89.54	88.41	-
	1,000	0.20	89.80	89.56	91.10	90.26	-
	1,000	0.30	89.73	89.75	90.63	90.06	a
	1,000	0.40	87.15	87.40	87.80	87.28	c,e
	1,000	0.50	81.62	82.11	81.85	81.24	d
Mixed	100	0.01	85.50	84.56	86.65	84.94	-
	100	0.10	89.08	88.52	88.76	87.86	-
	100	0.20	90.77	90.55	88.16	87.80	-
	100	0.30	89.95	90.00	84.38	84.53	f
	100	0.40	86.31	86.62	77.74	78.30	-
	100	0.50	79.35	79.89	68.53	69.45	-
	500	0.01	85.54	84.62	86.80	85.01	-
	500	0.10	89.31	88.76	90.04	89.07	-
	500	0.20	91.37	91.14	91.44	90.77	b
	500	0.30	91.30	91.33	90.80	90.34	-
	500	0.40	88.56	88.85	87.49	87.21	-
	500	0.50	82.62	83.14	80.65	80.47	f
	1,000	0.01	85.62	84.71	87.02	85.12	-
	1,000	0.10	89.33	88.78	90.19	89.27	c
	1,000	0.20	91.44	91.21	91.87	91.24	e
	1,000	0.30	91.41	91.44	91.70	91.21	-
	1,000	0.40	88.97	89.24	89.42	88.86	c
	1,000	0.50	83.26	83.76	83.63	83.06	c,d

Table B.3. Comparison of the accuracy obtained by PNB, PTAN, APNB and APTAN in datasets sampled from the database of acceptor sites.

Donor Sites - Accuracy

	$N_{\mathcal{D}_p}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
Coding	100	0.01	79.22	80.84	81.60	81.39	f
	100	0.10	90.50	89.32	89.72	88.94	-
	100	0.20	93.33	93.12	90.74	91.01	f
	100	0.30	91.90	92.10	87.56	88.73	-
	100	0.40	86.99	87.48	79.17	81.58	-
	100	0.50	77.69	78.41	66.95	70.20	-
	500	0.01	77.20	81.00	77.92	80.66	e
	500	0.10	90.83	89.65	87.13	88.40	-
	500	0.20	93.75	93.55	91.15	92.10	-
	500	0.30	92.61	92.83	91.11	91.32	f
	500	0.40	87.96	88.43	84.05	86.59	-
	500	0.50	79.40	80.13	73.10	77.11	-
	1,000	0.01	77.14	81.04	76.67	80.64	b
	1,000	0.10	90.83	89.64	86.19	87.92	-
	1,000	0.20	93.84	93.63	91.25	92.23	-
	1,000	0.30	92.66	92.90	91.76	91.80	f
	1,000	0.40	88.10	88.56	84.91	87.25	-
	1,000	0.50	79.70	80.45	74.17	78.14	-
Intron	100	0.01	82.55	79.89	82.84	81.39	b
	100	0.10	89.45	88.59	89.14	88.31	e
	100	0.20	91.77	91.58	89.82	90.11	-
	100	0.30	90.47	90.66	86.15	87.60	-
	100	0.40	85.62	86.05	78.43	80.61	-
	100	0.50	76.57	77.21	66.01	69.07	-
	500	0.01	82.54	79.41	73.40	81.57	-
	500	0.10	89.55	88.64	84.13	88.00	-
	500	0.20	92.35	92.16	91.33	91.03	f
	500	0.30	91.12	91.30	90.00	90.14	f
	500	0.40	86.73	87.11	84.96	85.53	-
	500	0.50	78.55	79.23	74.87	76.38	-
	1,000	0.01	82.53	79.51	72.23	81.45	-
	1,000	0.10	89.57	88.63	82.64	88.02	-
	1,000	0.20	92.42	92.23	90.93	90.97	f
	1,000	0.30	91.16	91.38	90.47	90.32	f
	1,000	0.40	86.91	87.31	86.20	86.22	f
	1,000	0.50	78.89	79.54	77.15	77.24	f
Mixed	100	0.01	82.66	79.69	83.28	81.38	-
	100	0.10	90.12	89.20	89.57	88.66	-
	100	0.20	92.55	92.35	90.46	90.69	f
	100	0.30	91.15	91.36	86.75	88.01	-
	100	0.40	86.13	86.59	78.45	80.87	-
	100	0.50	77.00	77.73	66.63	69.85	-
	500	0.01	82.63	79.63	73.94	80.51	-
	500	0.10	90.30	89.35	87.01	88.03	-
	500	0.20	93.03	92.84	91.06	91.72	-
	500	0.30	91.74	91.95	90.39	90.83	-
	500	0.40	87.25	87.71	84.45	85.95	-
	500	0.50	78.56	79.34	73.24	76.67	-
	1,000	0.01	82.58	79.57	71.71	80.24	-
	1,000	0.10	90.41	89.43	84.65	88.21	-
	1,000	0.20	93.15	92.96	91.13	91.45	f
	1,000	0.30	91.79	92.04	91.20	91.02	f
	1,000	0.40	87.31	87.73	85.82	86.37	-
	1,000	0.50	79.05	79.80	75.57	77.78	-

Table B.4. Comparison of the accuracy obtained by PNB, PTAN, APNB and APTAN in datasets sampled from the database of donor sites.

Letter Recognition - F measure

	$N_{\mathcal{D}_p}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
Letter Recognition D	100	0.01	10.70	10.23	26.59	24.70	f
	100	0.10	59.75	58.31	52.77	50.24	f
	100	0.20	74.12	73.94	38.53	38.03	a,f
	100	0.30	75.27	76.42	24.70	23.96	f
	100	0.40	70.60	73.15	14.80	14.03	f
	100	0.50	61.40	65.11	8.45	8.98	f
	200	0.01	10.17	9.83	21.57	19.03	-
	200	0.10	58.87	57.40	67.35	65.15	f
	200	0.20	76.56	76.00	66.45	64.98	f
	200	0.30	81.01	81.27	56.45	54.51	f
	200	0.40	80.35	81.37	43.65	40.80	-
	200	0.50	75.36	77.43	30.96	28.41	f
	300	0.01	10.14	10.01	18.61	16.53	-
	300	0.10	59.16	58.33	69.61	68.14	f
	300	0.20	76.55	76.27	74.90	74.05	b,d,f
	300	0.30	82.08	82.24	70.28	68.40	f
	300	0.40	81.87	82.08	60.90	58.53	-
	300	0.50	80.02	80.32	49.53	45.47	-
Letter Recognition P	100	0.01	12.89	13.25	31.52	28.00	-
	100	0.10	67.59	65.90	63.51	64.07	e,f
	100	0.20	82.48	82.15	50.98	52.00	f
	100	0.30	84.27	84.78	33.69	35.81	f
	100	0.40	80.75	82.18	21.91	22.71	f
	100	0.50	72.57	75.10	13.01	13.53	f
	200	0.01	13.67	14.18	28.69	25.27	-
	200	0.10	68.02	67.00	75.31	74.07	f
	200	0.20	83.90	83.32	73.59	74.17	f
	200	0.30	87.35	87.42	63.60	63.00	a,f
	200	0.40	86.52	87.01	49.89	49.53	f
	200	0.50	82.63	83.76	36.60	34.95	f
	300	0.01	14.25	14.99	25.92	23.01	-
	300	0.10	69.42	69.05	77.10	75.97	a,f
	300	0.20	84.60	84.25	80.67	80.66	f
	300	0.30	87.78	87.85	75.42	74.52	a,f
	300	0.40	87.73	87.96	65.61	65.04	f
	300	0.50	85.19	85.50	53.24	51.14	f
Letter Recognition U	100	0.01	13.46	12.43	27.81	25.73	f
	100	0.10	66.89	65.66	58.87	57.13	f
	100	0.20	79.16	79.08	48.27	46.37	a,f
	100	0.30	80.23	80.85	33.29	32.41	f
	100	0.40	76.48	77.89	21.72	21.33	f
	100	0.50	68.66	71.20	13.84	13.89	f
	200	0.01	13.54	12.95	26.03	22.44	-
	200	0.10	69.56	68.10	71.32	69.81	c,f
	200	0.20	81.48	81.17	70.58	68.99	f
	200	0.30	83.42	83.63	60.67	58.27	f
	200	0.40	82.40	83.04	49.73	46.50	f
	200	0.50	79.04	80.28	36.15	33.99	f
	300	0.01	13.87	13.71	23.12	20.19	a
	300	0.10	70.36	68.85	73.70	72.52	f
	300	0.20	81.87	81.65	77.67	77.09	f
	300	0.30	84.29	84.33	72.42	71.40	a,f
	300	0.40	84.22	84.35	64.61	62.12	-
	300	0.50	81.25	81.50	53.29	50.26	f

Table B.5. Comparison of the F measure obtained by PNB, PTAN, APNB and APTAN in datasets sampled from the database Letter Recognition.

Letter Recognition - Accuracy

	$N_{\mathcal{D}_p}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
Letter Recognition D	100	0.01	84.86	83.96	97.00	96.43	-
	100	0.10	88.55	87.61	92.87	92.40	-
	100	0.20	89.14	88.71	84.33	84.10	f
	100	0.30	86.48	86.75	73.86	73.60	f
	100	0.40	80.61	81.75	62.89	62.48	f
	100	0.50	71.14	73.01	51.84	51.70	f
	200	0.01	83.38	82.68	94.25	93.46	-
	200	0.10	87.57	86.63	93.66	93.16	-
	200	0.20	89.56	89.05	89.26	88.82	b,d,e,f
	200	0.30	88.82	88.76	81.41	80.76	a,f
	200	0.40	85.85	86.34	70.95	69.97	-
	200	0.50	79.54	80.78	58.98	58.10	f
	300	0.01	83.59	83.26	92.53	91.28	-
	300	0.10	87.43	86.90	93.31	92.79	-
	300	0.20	89.41	89.14	91.01	90.69	f
	300	0.30	89.29	89.31	85.71	85.00	a,f
	300	0.40	86.73	86.82	77.16	76.20	-
	300	0.50	82.67	82.87	66.26	64.51	-
Letter Recognition P	100	0.01	88.10	88.49	97.04	96.29	-
	100	0.10	91.55	90.82	93.97	93.77	f
	100	0.20	92.87	92.62	86.47	86.50	f
	100	0.30	91.18	91.36	75.81	76.07	f
	100	0.40	86.60	87.39	64.68	64.74	f
	100	0.50	78.08	79.63	53.18	53.15	f
	200	0.01	88.61	89.08	95.79	94.93	-
	200	0.10	91.63	91.21	95.14	94.71	f
	200	0.20	93.43	93.09	91.17	91.23	f
	200	0.30	92.78	92.75	83.71	83.48	a,f
	200	0.40	90.19	90.46	73.15	72.98	f
	200	0.50	84.98	85.78	61.10	60.47	f
	300	0.01	89.10	89.75	94.97	94.06	-
	300	0.10	92.10	91.97	95.19	94.79	a
	300	0.20	93.68	93.48	93.01	92.90	f
	300	0.30	93.02	93.02	87.83	87.46	a,f
	300	0.40	91.00	91.13	79.36	79.08	f
	300	0.50	86.93	87.16	68.06	67.08	f
Letter Recognition U	100	0.01	89.43	88.26	96.91	96.32	-
	100	0.10	92.14	91.58	93.48	93.01	-
	100	0.20	91.90	91.73	85.91	85.41	f
	100	0.30	89.40	89.56	75.68	75.29	f
	100	0.40	84.28	84.97	64.60	64.33	f
	100	0.50	75.75	77.19	53.41	53.20	f
	200	0.01	88.87	88.25	95.53	94.63	-
	200	0.10	92.75	92.10	94.55	94.14	f
	200	0.20	92.79	92.53	90.43	89.99	f
	200	0.30	90.95	90.96	82.79	82.00	a,f
	200	0.40	87.73	88.05	73.06	71.97	-
	200	0.50	82.49	83.30	60.91	60.13	f
	300	0.01	88.95	88.72	94.50	93.40	a
	300	0.10	92.98	92.40	94.68	94.27	f
	300	0.20	92.96	92.81	92.17	91.95	f
	300	0.30	91.42	91.40	86.71	86.30	a,f
	300	0.40	88.85	88.91	78.94	77.87	-
	300	0.50	84.07	84.24	68.09	66.71	-

Table B.6. Comparison of the accuracy obtained by PNB, PTAN, APNB and APTAN in datasets sampled from the database Letter Recognition.

Nursery - F measure

$N_{\mathcal{D}_p}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	0.01	17.40	15.66	13.57	15.66	e
100	0.10	70.66	70.14	63.28	70.14	-
100	0.20	74.14	75.04	72.17	75.04	c,f
100	0.30	67.92	69.35	66.55	69.35	c
100	0.40	56.57	58.41	56.86	58.41	b,e
100	0.50	43.09	45.05	46.24	45.05	-
200	0.01	18.93	16.98	14.92	16.98	e
200	0.10	72.95	72.29	66.26	72.29	e
200	0.20	76.81	77.74	77.31	77.74	b,d
200	0.30	69.87	71.42	72.85	71.42	f
200	0.40	58.35	60.29	62.06	60.29	-
200	0.50	42.15	44.33	48.31	44.33	-
300	0.01	19.47	17.37	15.34	17.37	-
300	0.10	74.19	73.58	67.88	73.58	e
300	0.20	77.86	78.84	79.28	78.84	d
300	0.30	70.38	72.00	75.78	72.00	-
300	0.40	58.22	60.23	64.32	60.23	-
300	0.50	41.97	44.24	48.61	44.24	-

Table B.7. Comparison of the F measure obtained by PNB, PTAN, APNB and APTAN in datasets sampled from the database Nursery.

Nursery - Accuracy

$N_{\mathcal{D}_p}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	0.01	91.62	90.25	88.73	90.53	e
100	0.10	93.55	93.22	90.70	91.95	-
100	0.20	90.85	91.00	89.55	89.85	f
100	0.30	84.86	85.30	83.77	84.03	f
100	0.40	75.40	76.09	74.94	75.34	c
100	0.50	63.53	64.31	64.27	64.80	d
200	0.01	92.24	90.95	89.38	91.32	e
200	0.10	94.06	93.69	91.25	93.02	-
200	0.20	91.76	91.93	91.16	91.78	c,e
200	0.30	85.72	86.24	86.38	86.67	d,f
200	0.40	76.27	77.03	77.40	78.02	-
200	0.50	63.26	64.13	65.44	66.13	-
300	0.01	92.46	91.18	89.58	91.54	e
300	0.10	94.35	94.02	91.58	93.30	-
300	0.20	92.12	92.33	91.82	92.70	b
300	0.30	85.94	86.49	87.71	88.13	f
300	0.40	76.24	77.03	78.52	79.33	-
300	0.50	63.24	64.15	65.73	66.71	-

Table B.8. Comparison of the accuracy obtained by PNB, PTAN, APNB and APTAN in datasets sampled from the database Nursery.

Lined - F measure

$N_{\mathcal{D}_p}$	$N_{\mathcal{D}_u}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	1,000	0.01	10.55	10.46	15.72	11.94	a
100	1,000	0.10	32.54	32.34	57.36	38.76	a
100	1,000	0.20	30.77	31.47	65.59	41.59	-
100	1,000	0.30	23.89	24.97	53.62	37.30	-
100	1,000	0.40	22.43	23.66	51.38	37.86	-
100	1,000	0.50	22.21	23.36	41.94	34.21	-
100	10,000	0.01	10.57	10.60	11.86	12.15	a,f
100	10,000	0.10	32.81	32.72	55.66	40.44	a
100	10,000	0.20	31.14	31.79	70.44	44.12	-
100	10,000	0.30	23.53	24.54	59.71	40.84	-
100	10,000	0.40	21.98	23.07	56.47	39.54	-
100	10,000	0.50	21.83	22.86	47.29	34.85	-
100	100,000	0.01	10.65	10.70	10.17	12.32	a,b,d
100	100,000	0.10	32.98	32.82	53.94	40.72	a
100	100,000	0.20	31.26	31.91	68.64	46.75	-
100	100,000	0.30	23.61	24.56	59.30	41.39	-
100	100,000	0.40	21.98	23.04	57.15	40.38	-
100	100,000	0.50	21.75	22.73	47.70	35.73	-
1,000	1,000	0.01	10.48	10.51	15.08	11.51	a
1,000	1,000	0.10	32.99	32.90	57.72	38.95	a
1,000	1,000	0.20	29.94	30.74	63.04	41.75	-
1,000	1,000	0.30	23.96	24.99	52.78	37.66	-
1,000	1,000	0.40	21.69	22.92	51.36	38.54	-
1,000	1,000	0.50	22.20	23.30	44.01	34.39	-
1,000	10,000	0.01	10.47	10.58	11.65	11.99	a,f
1,000	10,000	0.10	33.11	32.98	56.59	40.37	a
1,000	10,000	0.20	30.09	30.77	67.89	43.71	-
1,000	10,000	0.30	24.14	25.06	58.42	41.16	-
1,000	10,000	0.40	20.95	22.02	55.47	39.32	-
1,000	10,000	0.50	21.43	22.42	48.68	34.92	-
1,000	100,000	0.01	10.54	10.67	9.96	12.39	a,b,d
1,000	100,000	0.10	33.14	33.01	54.35	40.82	a
1,000	100,000	0.20	30.26	30.92	67.16	45.68	-
1,000	100,000	0.30	24.04	24.98	58.96	41.46	-
1,000	100,000	0.40	20.82	21.90	56.09	39.70	-
1,000	100,000	0.50	21.34	22.31	49.52	36.12	-
10,000	1,000	0.01	10.68	10.72	15.57	11.79	a
10,000	1,000	0.10	32.93	32.81	58.78	38.40	a
10,000	1,000	0.20	29.40	30.12	64.66	42.01	-
10,000	1,000	0.30	23.66	24.60	51.59	37.94	-
10,000	1,000	0.40	21.88	23.01	50.83	38.38	-
10,000	1,000	0.50	22.57	23.73	43.74	34.38	-
10,000	10,000	0.01	10.68	10.72	11.82	12.18	a,f
10,000	10,000	0.10	32.83	32.68	57.31	39.77	a
10,000	10,000	0.20	29.91	30.59	69.18	44.27	-
10,000	10,000	0.30	23.64	24.63	56.62	41.14	-
10,000	10,000	0.40	21.56	22.60	54.85	39.81	-
10,000	10,000	0.50	22.29	23.32	48.81	35.41	-
10,000	100,000	0.01	10.79	10.76	10.12	12.39	a,b,d
10,000	100,000	0.10	32.95	32.78	54.36	40.87	a
10,000	100,000	0.20	29.93	30.61	69.00	46.13	-
10,000	100,000	0.30	23.69	24.68	58.03	41.33	-
10,000	100,000	0.40	21.45	22.50	55.80	40.35	-
10,000	100,000	0.50	22.19	23.19	48.78	36.15	-

Table B.9. Comparison of the F measure obtained by PNB, PTAN, APNB and APTAN in datasets sampled from TAN models where the structure was 'Lined'.

Hierarchical 1 - F measure

$N_{\mathcal{D}_p}$	$N_{\mathcal{D}_u}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	1,000	0.01	7.56	7.43	14.18	11.82	a,f
100	1,000	0.10	36.13	36.23	54.14	44.39	a
100	1,000	0.20	31.48	32.21	67.98	48.63	-
100	1,000	0.30	26.88	27.84	61.71	49.62	-
100	1,000	0.40	25.55	26.67	59.93	48.25	-
100	1,000	0.50	23.92	24.98	48.11	39.29	-
100	10,000	0.01	7.42	7.29	9.41	11.73	-
100	10,000	0.10	36.52	36.53	51.98	49.07	a,f
100	10,000	0.20	31.74	32.38	69.62	55.04	-
100	10,000	0.30	27.14	28.04	62.02	53.64	-
100	10,000	0.40	25.41	26.44	62.97	52.89	-
100	10,000	0.50	23.66	24.68	50.80	42.11	-
100	100,000	0.01	7.44	7.31	8.10	11.89	-
100	100,000	0.10	36.42	36.44	48.36	47.46	a,f
100	100,000	0.20	31.80	32.44	68.07	54.28	-
100	100,000	0.30	26.96	27.84	63.16	50.03	-
100	100,000	0.40	25.45	26.45	62.97	49.60	-
100	100,000	0.50	23.66	24.64	50.52	39.76	-
1,000	1,000	0.01	7.75	7.50	14.21	11.55	-
1,000	1,000	0.10	35.73	35.92	55.32	44.16	a
1,000	1,000	0.20	30.52	31.30	70.08	47.81	-
1,000	1,000	0.30	26.30	27.43	61.62	49.23	-
1,000	1,000	0.40	25.92	26.99	60.19	48.39	-
1,000	1,000	0.50	23.86	24.90	47.77	39.61	-
1,000	10,000	0.01	7.49	7.31	9.43	12.14	-
1,000	10,000	0.10	36.71	36.74	51.60	50.11	a,f
1,000	10,000	0.20	31.14	31.83	70.65	54.17	-
1,000	10,000	0.30	26.43	27.32	64.25	53.68	-
1,000	10,000	0.40	25.71	26.70	64.30	52.91	-
1,000	10,000	0.50	23.47	24.42	51.00	42.12	-
1,000	100,000	0.01	7.48	7.36	8.15	12.01	-
1,000	100,000	0.10	36.77	36.83	47.52	47.48	f
1,000	100,000	0.20	31.17	31.85	68.57	53.04	-
1,000	100,000	0.30	26.33	27.22	64.01	49.75	-
1,000	100,000	0.40	25.67	26.67	63.71	49.58	-
1,000	100,000	0.50	23.35	24.30	50.61	39.53	-
10,000	1,000	0.01	7.62	7.47	14.56	11.63	-
10,000	1,000	0.10	35.72	35.90	53.69	44.10	a
10,000	1,000	0.20	30.77	31.56	68.46	48.43	-
10,000	1,000	0.30	26.20	27.23	60.22	49.33	-
10,000	1,000	0.40	25.92	27.05	61.60	47.71	-
10,000	1,000	0.50	24.21	25.37	47.55	39.42	-
10,000	10,000	0.01	7.35	7.25	9.53	11.90	-
10,000	10,000	0.10	36.46	36.46	50.86	48.65	a,f
10,000	10,000	0.20	31.34	31.96	70.59	54.85	-
10,000	10,000	0.30	26.60	27.50	64.42	53.37	-
10,000	10,000	0.40	25.81	26.83	64.13	52.22	-
10,000	10,000	0.50	23.63	24.64	50.81	42.40	-
10,000	100,000	0.01	7.38	7.28	8.24	11.84	-
10,000	100,000	0.10	36.54	36.46	48.16	45.47	a,f
10,000	100,000	0.20	31.25	31.92	68.14	52.80	-
10,000	100,000	0.30	26.59	27.46	63.57	49.40	-
10,000	100,000	0.40	25.88	26.86	62.97	49.33	-
10,000	100,000	0.50	23.56	24.56	50.51	39.81	-

Table B.10. Comparison of the F measure obtained by PNB, PTAN, APNB and APTAN in datasets sampled from TAN models where the structure was 'Hierarchical 1'.

Hierarchical 2 - F measure

$N_{\mathcal{D}_p}$	$N_{\mathcal{D}_u}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	1,000	0.01	9.91	9.85	13.74	10.61	a,c,e
100	1,000	0.10	31.47	31.69	55.47	37.23	a
100	1,000	0.20	31.99	32.75	63.24	40.74	-
100	1,000	0.30	25.04	25.92	60.30	40.29	-
100	1,000	0.40	22.94	23.99	51.42	34.30	-
100	1,000	0.50	22.89	23.90	40.00	33.13	-
100	10,000	0.01	9.95	9.77	10.93	11.15	f
100	10,000	0.10	32.10	32.16	54.53	39.52	a
100	10,000	0.20	32.20	32.84	68.45	44.65	-
100	10,000	0.30	25.80	26.67	65.48	43.25	-
100	10,000	0.40	22.43	23.40	56.02	37.07	-
100	10,000	0.50	22.40	23.46	43.94	34.29	-
100	100,000	0.01	9.94	9.75	9.89	11.02	b,d
100	100,000	0.10	31.84	31.94	52.43	40.25	-
100	100,000	0.20	32.31	32.94	66.42	44.50	-
100	100,000	0.30	25.72	26.56	65.11	42.13	-
100	100,000	0.40	22.29	23.24	55.49	35.87	-
100	100,000	0.50	22.43	23.46	44.29	33.74	-
1,000	1,000	0.01	9.99	9.50	14.02	10.66	c
1,000	1,000	0.10	31.46	31.55	57.59	37.32	a
1,000	1,000	0.20	31.97	32.60	65.04	40.45	-
1,000	1,000	0.30	25.52	26.43	60.06	40.14	-
1,000	1,000	0.40	21.69	22.84	50.65	34.23	-
1,000	1,000	0.50	22.75	23.91	40.18	33.05	-
1,000	10,000	0.01	10.08	9.83	11.14	11.17	f
1,000	10,000	0.10	31.99	32.11	55.88	39.18	-
1,000	10,000	0.20	31.96	32.58	69.20	44.87	-
1,000	10,000	0.30	25.76	26.63	65.16	43.29	-
1,000	10,000	0.40	21.54	22.53	55.97	36.35	-
1,000	10,000	0.50	22.51	23.53	43.48	34.32	-
1,000	100,000	0.01	10.06	9.81	9.84	11.28	b,d
1,000	100,000	0.10	31.86	31.96	53.14	39.86	-
1,000	100,000	0.20	32.01	32.64	67.82	44.46	-
1,000	100,000	0.30	25.73	26.58	64.49	42.20	-
1,000	100,000	0.40	21.47	22.43	55.69	35.61	-
1,000	100,000	0.50	22.36	23.38	44.16	33.35	-
10,000	1,000	0.01	10.00	9.88	14.25	10.59	a,c,e
10,000	1,000	0.10	30.81	31.02	56.91	37.38	a
10,000	1,000	0.20	31.49	32.26	63.10	40.83	-
10,000	1,000	0.30	25.66	26.49	59.19	39.80	-
10,000	1,000	0.40	21.30	22.38	51.08	34.43	-
10,000	1,000	0.50	23.21	24.41	40.35	32.96	-
10,000	10,000	0.01	10.07	9.76	11.14	11.24	f
10,000	10,000	0.10	31.90	31.99	56.15	39.37	-
10,000	10,000	0.20	31.75	32.42	67.78	44.31	-
10,000	10,000	0.30	25.95	26.78	64.76	42.91	-
10,000	10,000	0.40	21.12	22.06	56.36	36.91	-
10,000	10,000	0.50	22.63	23.74	44.23	34.74	-
10,000	100,000	0.01	10.06	9.76	9.97	11.14	b,d
10,000	100,000	0.10	31.82	31.91	53.66	40.47	-
10,000	100,000	0.20	31.76	32.38	66.97	43.94	-
10,000	100,000	0.30	25.82	26.62	64.53	42.25	-
10,000	100,000	0.40	21.05	22.01	56.01	35.77	-
10,000	100,000	0.50	22.53	23.57	44.49	33.84	-

Table B.11. Comparison of the F measure obtained by PNB, PTAN, APNB and APTAN in datasets sampled from TAN models where the structure was 'Hierarchical 2'.

Mixed - F measure

$N_{\mathcal{D}_p}$	$N_{\mathcal{D}_u}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	1,000	0.01	8.74	8.66	13.88	12.13	a
100	1,000	0.10	30.28	30.18	61.03	43.97	a
100	1,000	0.20	29.53	30.09	68.66	46.09	-
100	1,000	0.30	24.35	25.28	52.70	40.59	-
100	1,000	0.40	23.38	24.47	58.01	41.98	-
100	1,000	0.50	23.38	24.50	46.23	37.49	-
100	10,000	0.01	7.95	8.30	10.29	11.27	-
100	10,000	0.10	31.17	31.36	58.45	44.84	-
100	10,000	0.20	29.96	30.64	72.20	47.98	-
100	10,000	0.30	23.93	24.86	59.16	43.88	-
100	10,000	0.40	23.22	24.22	63.76	43.82	-
100	10,000	0.50	23.37	24.35	51.36	39.54	-
100	100,000	0.01	7.97	8.28	9.10	11.13	-
100	100,000	0.10	31.28	31.41	55.80	46.13	-
100	100,000	0.20	29.89	30.54	69.47	49.05	-
100	100,000	0.30	23.99	24.88	58.98	43.46	-
100	100,000	0.40	23.30	24.28	63.76	44.86	-
100	100,000	0.50	23.22	24.20	51.14	39.21	-
1,000	1,000	0.01	8.60	8.79	13.68	11.69	a
1,000	1,000	0.10	30.77	30.81	61.61	42.75	a
1,000	1,000	0.20	30.25	31.00	68.67	44.94	-
1,000	1,000	0.30	23.64	24.73	53.99	40.48	-
1,000	1,000	0.40	23.93	24.97	56.29	41.78	-
1,000	1,000	0.50	23.22	24.36	46.89	37.38	-
1,000	10,000	0.01	7.95	8.23	10.32	11.19	-
1,000	10,000	0.10	30.97	31.09	59.28	44.37	-
1,000	10,000	0.20	30.67	31.37	72.14	47.92	-
1,000	10,000	0.30	23.95	24.82	60.01	43.70	-
1,000	10,000	0.40	23.21	24.24	61.70	43.87	-
1,000	10,000	0.50	22.58	23.58	51.75	39.30	-
1,000	100,000	0.01	8.02	8.26	9.24	11.11	a
1,000	100,000	0.10	30.92	31.06	55.80	45.74	-
1,000	100,000	0.20	30.68	31.33	70.22	48.88	-
1,000	100,000	0.30	24.09	24.95	59.11	43.61	-
1,000	100,000	0.40	23.16	24.13	61.77	44.64	-
1,000	100,000	0.50	22.57	23.56	51.54	39.21	-
10,000	1,000	0.01	8.74	8.75	14.11	11.83	a
10,000	1,000	0.10	30.28	30.42	61.68	42.79	a
10,000	1,000	0.20	29.24	30.06	68.78	45.72	-
10,000	1,000	0.30	24.25	25.32	55.03	40.34	-
10,000	1,000	0.40	23.24	24.38	58.67	41.32	-
10,000	1,000	0.50	21.79	22.98	47.74	37.53	-
10,000	10,000	0.01	7.97	8.28	10.30	11.15	-
10,000	10,000	0.10	30.37	30.54	59.49	44.22	-
10,000	10,000	0.20	29.87	30.53	72.08	48.01	-
10,000	10,000	0.30	24.15	25.04	61.60	44.14	-
10,000	10,000	0.40	23.27	24.28	63.48	43.80	-
10,000	10,000	0.50	21.66	22.62	52.47	39.46	-
10,000	100,000	0.01	8.01	8.29	9.21	10.99	a
10,000	100,000	0.10	30.59	30.76	56.17	45.89	-
10,000	100,000	0.20	29.95	30.59	69.34	48.54	-
10,000	100,000	0.30	24.09	24.95	60.83	43.93	-
10,000	100,000	0.40	23.25	24.24	63.34	44.30	-
10,000	100,000	0.50	21.58	22.55	52.13	39.22	-

Table B.12. Comparison of the F measure obtained by PNB, PTAN, APNB and APTAN in datasets sampled from TAN models where the structure was 'Mixed'.

Lined - Accuracy

$N_{\mathcal{D}_p}$	$N_{\mathcal{D}_u}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	1,000	0.01	85.71	85.88	90.65	88.91	a
100	1,000	0.10	82.37	81.78	89.82	84.10	-
100	1,000	0.20	76.95	76.62	87.79	78.91	-
100	1,000	0.30	68.75	68.57	78.83	71.13	-
100	1,000	0.40	60.84	60.81	72.27	64.88	a
100	1,000	0.50	53.11	53.24	61.57	56.40	-
100	10,000	0.01	85.15	85.73	86.13	87.95	a d
100	10,000	0.10	82.49	82.00	87.89	83.94	-
100	10,000	0.20	77.26	76.99	88.41	79.35	-
100	10,000	0.30	68.64	68.52	80.18	71.97	-
100	10,000	0.40	60.79	60.82	74.05	65.30	a
100	10,000	0.50	53.12	53.26	63.75	56.48	-
100	100,000	0.01	85.15	85.80	83.70	87.89	a
100	100,000	0.10	82.52	82.05	86.62	83.93	-
100	100,000	0.20	77.32	77.06	87.36	80.16	-
100	100,000	0.30	68.65	68.51	79.73	72.20	-
100	100,000	0.40	60.84	60.87	74.17	65.76	a
100	100,000	0.50	53.07	53.19	63.73	57.01	-
1,000	1,000	0.01	85.51	85.88	90.39	88.80	-
1,000	1,000	0.10	82.51	81.95	89.90	84.14	-
1,000	1,000	0.20	76.94	76.62	86.97	78.91	-
1,000	1,000	0.30	68.65	68.48	78.41	71.26	-
1,000	1,000	0.40	60.66	60.66	72.22	65.19	a
1,000	1,000	0.50	53.13	53.24	62.69	56.42	-
1,000	10,000	0.01	84.84	85.69	85.87	87.89	d
1,000	10,000	0.10	82.74	82.23	88.13	83.88	-
1,000	10,000	0.20	77.15	76.87	87.60	79.26	-
1,000	10,000	0.30	68.77	68.65	79.62	72.13	-
1,000	10,000	0.40	60.68	60.70	73.52	65.27	a
1,000	10,000	0.50	53.03	53.15	64.46	56.54	-
1,000	100,000	0.01	84.82	85.74	83.46	87.91	-
1,000	100,000	0.10	82.73	82.25	86.68	83.99	-
1,000	100,000	0.20	77.19	76.93	86.94	79.80	-
1,000	100,000	0.30	68.76	68.63	79.58	72.15	-
1,000	100,000	0.40	60.71	60.75	73.68	65.44	-
1,000	100,000	0.50	53.02	53.13	64.70	57.16	-
10,000	1,000	0.01	85.70	86.02	90.51	88.97	a
10,000	1,000	0.10	82.53	81.93	90.19	83.87	-
10,000	1,000	0.20	76.67	76.36	87.53	78.84	-
10,000	1,000	0.30	68.86	68.63	77.95	71.41	-
10,000	1,000	0.40	60.59	60.55	72.06	65.14	a
10,000	1,000	0.50	53.22	53.35	62.54	56.47	-
10,000	10,000	0.01	85.04	85.69	85.88	87.92	a d
10,000	10,000	0.10	82.62	82.09	88.32	83.79	-
10,000	10,000	0.20	77.12	76.84	88.00	79.40	-
10,000	10,000	0.30	68.90	68.79	78.88	72.13	-
10,000	10,000	0.40	60.62	60.62	73.20	65.51	a
10,000	10,000	0.50	53.35	53.49	64.61	56.85	-
10,000	100,000	0.01	85.01	85.64	83.57	87.90	a
10,000	100,000	0.10	82.63	82.09	86.70	84.06	-
10,000	100,000	0.20	77.10	76.85	87.45	79.96	-
10,000	100,000	0.30	68.92	68.81	79.23	72.16	-
10,000	100,000	0.40	60.69	60.70	73.46	65.75	a
10,000	100,000	0.50	53.32	53.45	64.36	57.26	-

Table B.13. Comparison of the accuracy obtained by PNB, PTAN, APNB and APTAN in datasets sampled from TAN models where the structure was 'Lined'.

Hierarchical 1 - Accuracy

$N_{\mathcal{D}_p}$	$N_{\mathcal{D}_u}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	1,000	0.01	82.79	82.60	89.21	87.67	-
100	1,000	0.10	83.00	82.55	88.95	85.55	-
100	1,000	0.20	77.22	76.89	88.45	81.10	-
100	1,000	0.30	69.24	69.08	81.77	76.07	-
100	1,000	0.40	61.78	61.82	76.25	69.89	a
100	1,000	0.50	53.49	53.63	64.86	59.20	-
100	10,000	0.01	82.57	82.41	84.03	86.92	-
100	10,000	0.10	82.96	82.55	86.51	86.31	f
100	10,000	0.20	77.41	77.11	88.03	82.77	-
100	10,000	0.30	69.40	69.29	80.95	77.36	-
100	10,000	0.40	62.10	62.13	77.14	71.92	a
100	10,000	0.50	53.40	53.54	65.39	60.35	-
100	100,000	0.01	82.55	82.37	81.30	87.13	-
100	100,000	0.10	82.94	82.56	84.72	85.74	f
100	100,000	0.20	77.45	77.16	87.01	82.43	-
100	100,000	0.30	69.31	69.19	81.17	75.64	-
100	100,000	0.40	62.12	62.15	76.91	70.17	-
100	100,000	0.50	53.47	53.59	65.02	59.01	-
1,000	1,000	0.01	83.07	82.67	89.25	87.78	-
1,000	1,000	0.10	83.06	82.60	89.38	85.66	-
1,000	1,000	0.20	77.20	76.93	89.27	80.92	-
1,000	1,000	0.30	68.84	68.72	81.72	75.93	-
1,000	1,000	0.40	61.83	61.84	76.37	69.93	a
1,000	1,000	0.50	53.62	53.72	64.71	59.49	-
1,000	10,000	0.01	82.82	82.56	84.06	87.18	-
1,000	10,000	0.10	83.21	82.81	86.49	86.70	f
1,000	10,000	0.20	77.45	77.15	88.43	82.44	-
1,000	10,000	0.30	68.99	68.88	82.00	77.43	-
1,000	10,000	0.40	62.12	62.14	77.79	72.03	a
1,000	10,000	0.50	53.51	53.63	65.55	60.32	-
1,000	100,000	0.01	82.75	82.53	81.24	87.20	-
1,000	100,000	0.10	83.23	82.83	84.62	85.85	-
1,000	100,000	0.20	77.49	77.20	87.22	81.98	-
1,000	100,000	0.30	68.97	68.86	81.60	75.55	-
1,000	100,000	0.40	62.12	62.16	77.28	70.21	-
1,000	100,000	0.50	53.50	53.61	65.06	58.92	-
10,000	1,000	0.01	82.90	82.49	89.41	87.70	-
10,000	1,000	0.10	82.93	82.47	88.85	85.55	-
10,000	1,000	0.20	77.07	76.78	88.62	81.03	-
10,000	1,000	0.30	68.91	68.79	81.21	75.90	-
10,000	1,000	0.40	61.90	61.94	77.04	69.53	a
10,000	1,000	0.50	53.61	53.75	64.55	59.19	-
10,000	10,000	0.01	82.53	82.40	84.20	86.98	-
10,000	10,000	0.10	83.01	82.65	86.23	86.12	f
10,000	10,000	0.20	77.35	77.07	88.37	82.68	-
10,000	10,000	0.30	69.17	69.06	82.10	77.28	-
10,000	10,000	0.40	62.12	62.15	77.73	71.57	-
10,000	10,000	0.50	53.43	53.56	65.38	60.55	-
10,000	100,000	0.01	82.52	82.31	81.39	87.09	-
10,000	100,000	0.10	83.04	82.66	84.82	85.06	f
10,000	100,000	0.20	77.33	77.06	87.02	81.82	-
10,000	100,000	0.30	69.14	69.02	81.43	75.38	-
10,000	100,000	0.40	62.19	62.21	76.90	69.99	a
10,000	100,000	0.50	53.46	53.58	64.97	59.07	-

Table B.14. Comparison of the accuracy obtained by PNB, PTAN, APNB and APTAN in datasets sampled from TAN models where the structure was 'Hierarchical 1'.

Hierarchical 2 - Accuracy

$N_{\mathcal{D}_p}$	$N_{\mathcal{D}_u}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	1,000	0.01	85.94	85.82	90.11	88.38	a
100	1,000	0.10	81.88	81.41	89.51	83.54	-
100	1,000	0.20	76.83	76.51	86.83	78.32	-
100	1,000	0.30	69.29	69.08	81.36	72.27	-
100	1,000	0.40	60.62	60.56	72.42	62.96	a
100	1,000	0.50	52.92	53.02	60.54	55.67	-
100	10,000	0.01	85.84	85.86	85.82	87.80	a b d
100	10,000	0.10	82.01	81.54	87.54	83.58	-
100	10,000	0.20	77.18	76.89	87.60	79.32	-
100	10,000	0.30	69.41	69.28	82.64	73.00	-
100	10,000	0.40	60.68	60.67	73.70	64.04	a
100	10,000	0.50	53.11	53.25	61.77	56.10	-
100	100,000	0.01	85.83	85.90	84.03	87.64	a
100	100,000	0.10	81.93	81.48	86.28	83.79	-
100	100,000	0.20	77.22	76.94	86.49	79.12	-
100	100,000	0.30	69.39	69.26	82.19	72.51	-
100	100,000	0.40	60.56	60.54	73.06	63.36	-
100	100,000	0.50	53.09	53.22	61.68	55.70	-
1,000	1,000	0.01	86.34	86.02	90.26	88.68	-
1,000	1,000	0.10	81.88	81.29	89.86	83.65	-
1,000	1,000	0.20	76.90	76.56	87.40	78.23	-
1,000	1,000	0.30	69.25	69.04	81.26	72.21	-
1,000	1,000	0.40	60.36	60.37	71.99	63.03	a
1,000	1,000	0.50	52.95	53.08	60.60	55.51	-
1,000	10,000	0.01	86.08	86.02	86.02	87.71	b d
1,000	10,000	0.10	82.05	81.56	88.01	83.34	-
1,000	10,000	0.20	77.23	76.94	87.86	79.36	-
1,000	10,000	0.30	69.29	69.17	82.47	72.98	-
1,000	10,000	0.40	60.47	60.44	73.68	63.69	-
1,000	10,000	0.50	53.15	53.27	61.55	56.08	-
1,000	100,000	0.01	86.04	86.02	83.93	87.78	a
1,000	100,000	0.10	81.99	81.48	86.51	83.61	-
1,000	100,000	0.20	77.23	76.95	86.97	79.15	-
1,000	100,000	0.30	69.35	69.22	81.92	72.52	-
1,000	100,000	0.40	60.41	60.38	73.19	63.24	-
1,000	100,000	0.50	53.13	53.26	61.65	55.51	-
10,000	1,000	0.01	86.29	86.20	90.29	88.46	a
10,000	1,000	0.10	81.71	81.17	89.76	83.56	-
10,000	1,000	0.20	76.86	76.57	86.76	78.43	-
10,000	1,000	0.30	68.92	68.73	80.91	71.96	-
10,000	1,000	0.40	60.14	60.14	72.26	63.12	a
10,000	1,000	0.50	53.17	53.34	60.74	55.51	-
10,000	10,000	0.01	86.21	86.01	86.04	87.68	a b d
10,000	10,000	0.10	82.04	81.55	88.04	83.52	-
10,000	10,000	0.20	77.11	76.83	87.37	79.13	-
10,000	10,000	0.30	69.22	69.06	82.28	72.78	-
10,000	10,000	0.40	60.44	60.41	73.86	63.99	-
10,000	10,000	0.50	53.15	53.31	61.95	56.31	-
10,000	100,000	0.01	86.14	86.01	84.18	87.69	a
10,000	100,000	0.10	81.99	81.49	86.65	83.87	-
10,000	100,000	0.20	77.10	76.82	86.69	78.99	-
10,000	100,000	0.30	69.17	69.02	81.91	72.47	-
10,000	100,000	0.40	60.40	60.37	73.34	63.33	-
10,000	100,000	0.50	53.13	53.27	61.83	55.76	-

Table B.15. Comparison of the accuracy obtained by PNB, PTAN, APNB and APTAN in datasets sampled from TAN models where the structure was 'Hierarchical 2'.

Mixed - Accuracy

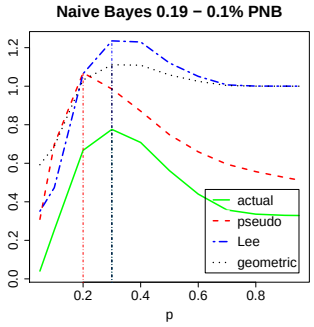
$N_{\mathcal{D}_p}$	$N_{\mathcal{D}_u}$	rat.	PNB	APNB	PTAN	APTAN	Wlx.
100	1,000	0.01	83.74	84.42	89.42	88.69	-
100	1,000	0.10	83.06	82.37	90.94	85.92	-
100	1,000	0.20	76.63	76.24	88.67	80.29	-
100	1,000	0.30	68.66	68.49	78.14	72.35	-
100	1,000	0.40	61.02	61.00	75.26	66.78	a
100	1,000	0.50	53.32	53.46	63.80	58.24	-
100	10,000	0.01	83.16	84.23	84.77	87.62	d
100	10,000	0.10	83.25	82.75	88.64	85.51	-
100	10,000	0.20	76.89	76.61	88.91	80.41	-
100	10,000	0.30	69.00	68.87	79.93	73.33	-
100	10,000	0.40	61.39	61.40	77.49	67.43	a
100	10,000	0.50	53.32	53.43	65.74	59.00	-
100	100,000	0.01	83.16	84.18	82.32	87.40	-
100	100,000	0.10	83.20	82.69	87.19	85.84	-
100	100,000	0.20	76.88	76.58	87.54	80.78	-
100	100,000	0.30	68.88	68.75	79.51	73.09	-
100	100,000	0.40	61.41	61.41	77.24	67.95	a
100	100,000	0.50	53.37	53.48	65.43	58.86	-
1,000	1,000	0.01	83.85	84.50	89.43	88.55	-
1,000	1,000	0.10	82.94	82.29	91.02	85.51	-
1,000	1,000	0.20	76.68	76.33	88.62	79.70	-
1,000	1,000	0.30	68.50	68.31	78.72	72.18	-
1,000	1,000	0.40	61.10	61.09	74.46	66.63	a
1,000	1,000	0.50	53.31	53.42	64.23	58.19	-
1,000	10,000	0.01	83.19	84.20	84.74	87.54	d
1,000	10,000	0.10	82.97	82.43	88.91	85.26	-
1,000	10,000	0.20	77.02	76.74	88.91	80.39	-
1,000	10,000	0.30	68.89	68.73	80.28	73.25	-
1,000	10,000	0.40	61.23	61.23	76.47	67.45	a
1,000	10,000	0.50	53.17	53.28	65.98	58.95	-
1,000	100,000	0.01	83.24	84.18	82.66	87.40	b
1,000	100,000	0.10	82.97	82.42	87.10	85.66	-
1,000	100,000	0.20	77.08	76.79	87.81	80.72	-
1,000	100,000	0.30	68.80	68.66	79.60	73.10	-
1,000	100,000	0.40	61.29	61.28	76.23	67.81	a
1,000	100,000	0.50	53.25	53.37	65.66	58.92	-
10,000	1,000	0.01	83.75	84.31	89.63	88.75	-
10,000	1,000	0.10	82.96	82.35	91.08	85.47	-
10,000	1,000	0.20	76.54	76.21	88.72	80.12	-
10,000	1,000	0.30	68.51	68.37	79.08	72.16	-
10,000	1,000	0.40	61.07	61.08	75.60	66.46	a
10,000	1,000	0.50	52.89	53.02	64.69	58.29	-
10,000	10,000	0.01	83.19	84.18	84.77	87.55	d
10,000	10,000	0.10	82.84	82.36	88.92	85.24	-
10,000	10,000	0.20	76.76	76.45	88.88	80.44	-
10,000	10,000	0.30	68.96	68.81	80.92	73.41	-
10,000	10,000	0.40	61.38	61.39	77.35	67.48	a
10,000	10,000	0.50	52.94	53.03	66.37	59.00	-
10,000	100,000	0.01	83.13	84.09	82.47	87.32	-
10,000	100,000	0.10	82.87	82.39	87.25	85.71	-
10,000	100,000	0.20	76.85	76.54	87.53	80.57	-
10,000	100,000	0.30	68.79	68.66	80.27	73.24	-
10,000	100,000	0.40	61.44	61.44	77.03	67.68	a
10,000	100,000	0.50	52.99	53.09	65.97	58.93	-

Table B.16. Comparison of the accuracy obtained by PNB, PTAN, APNB and APTAN in datasets sampled from TAN models where the structure was 'Mixed'.

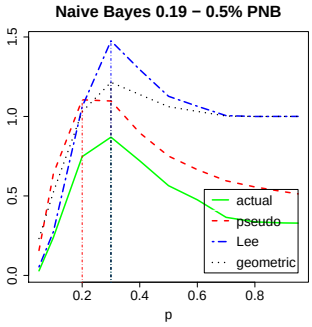
C

Experimental evaluation of pseudo F and Lee metric in synthetic datasets

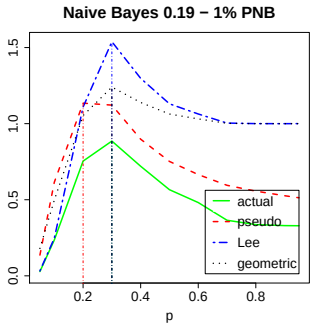
This appendix contains all the results obtained in the comparison between the actual F measure (labelled in the plots as actual), the pseudo F measure (labelled as pseudo), the Lee metric (labelled as Lee) and the square root of the Lee metric (labelled as geometric). Each page shows the results obtained by each classifier (PNB and PTAN) in datasets sampled from each of the six synthetic databases (naive Bayes 0.19, 0.28 and 0.45 and TAN 0.15, 0.30 and 0.45, see Section 5.2.3). The five plots per page correspond to the five different sizes of positive cases used to create the positive unlabelled learning problems (0.1%, 0.5%, 1%, 5% and 10%), and each plot represents the performance measures *vs.* the p used to train the classifier. Each point in the plots is the average result obtained in the 100 datasets built from each original database and with each size of the set of positive cases.



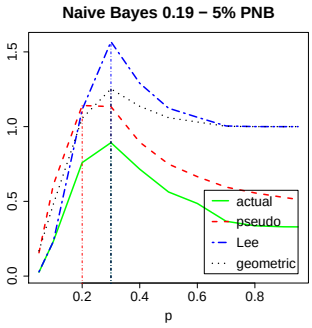
(a) 0.1%



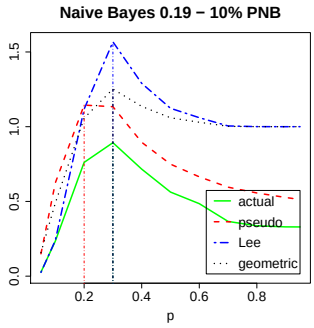
(b) 0.5%



(c) 1%

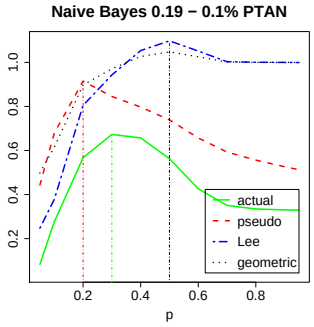


(d) 5%

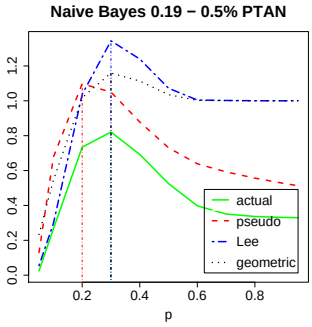


(e) 10%

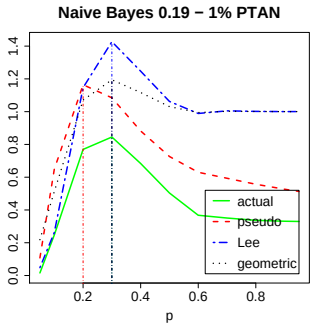
Fig. C.1. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PNB in problems sampled from naive Bayes 0.19



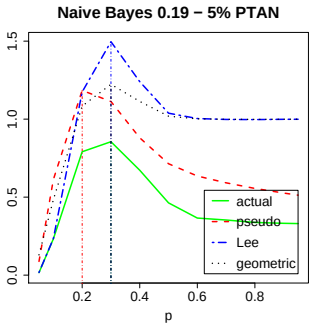
(a) 0.1%



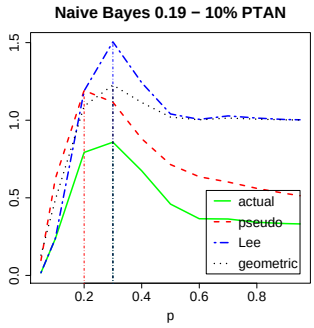
(b) 0.5%



(c) 1%

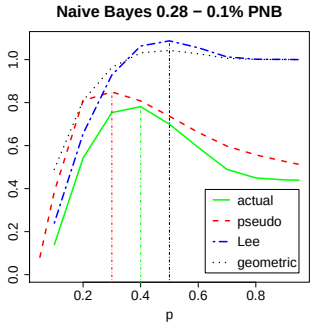


(d) 5%

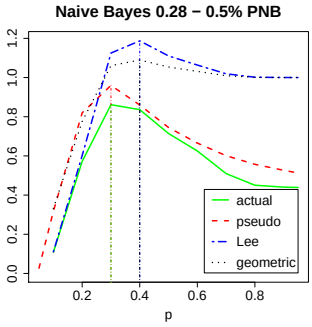


(e) 10%

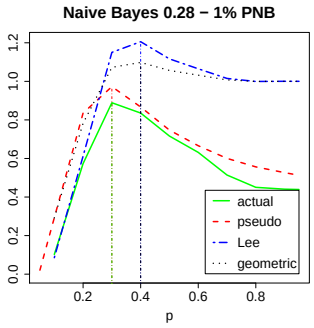
Fig. C.2. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PTAN in problems sampled from naive Bayes 0.19.



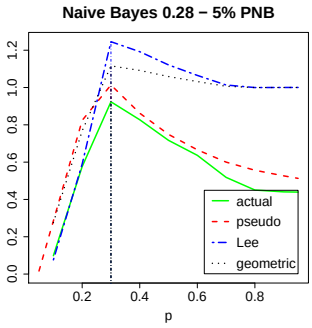
(a) 0.1%



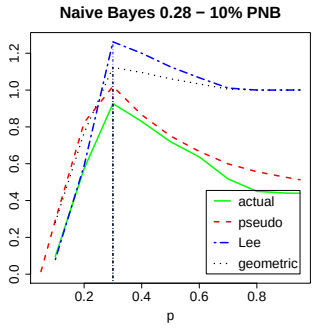
(b) 0.5%



(c) 1%

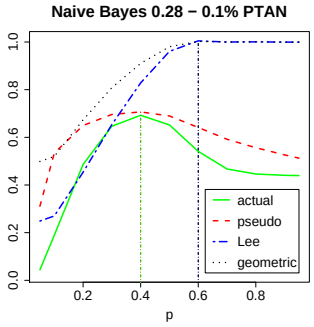


(d) 5%

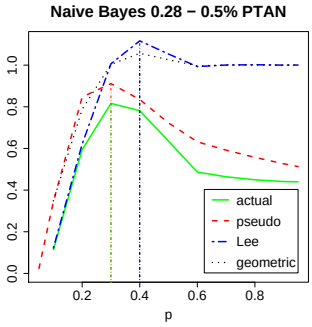


(e) 10%

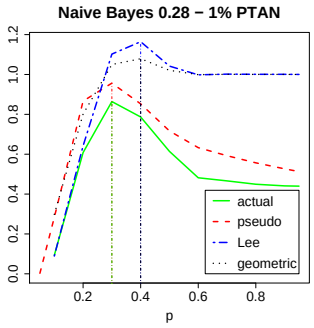
Fig. C.3. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PNB in problems sampled from naive Bayes 0.28.



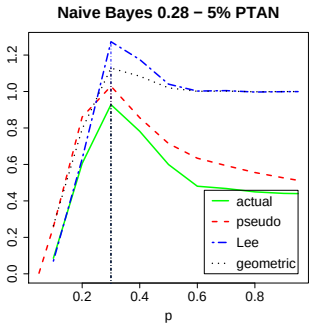
(a) 0.1%



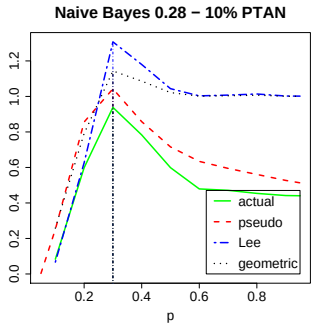
(b) 0.5%



(c) 1%

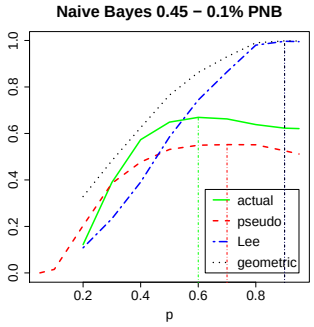


(d) 5%

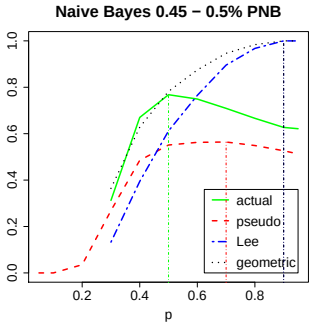


(e) 10%

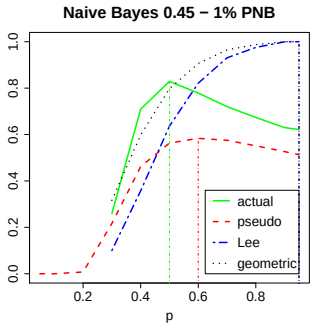
Fig. C.4. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PTAN in problems sampled from naive Bayes 0.28.



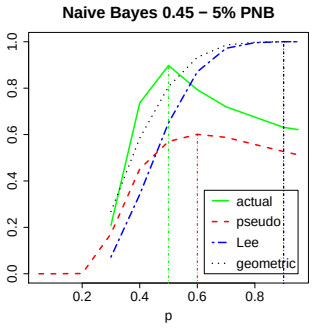
(a) 0.1%



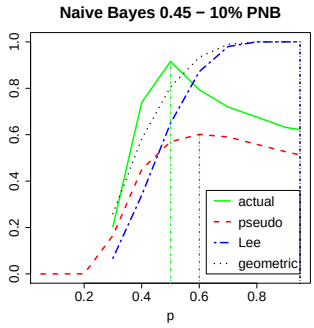
(b) 0.5%



(c) 1%

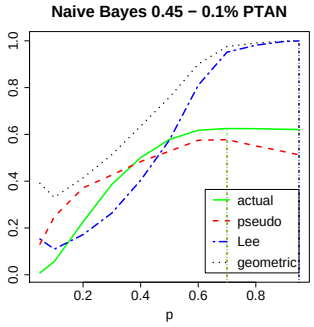


(d) 5%

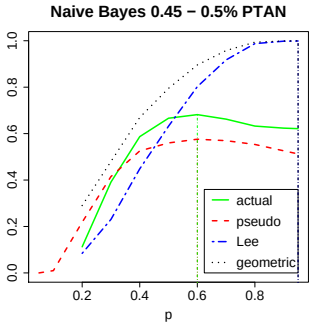


(e) 10%

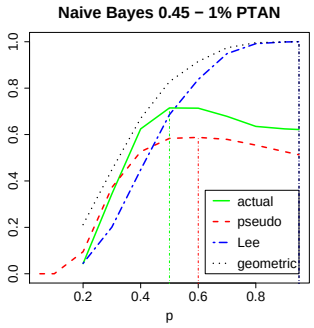
Fig. C.5. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PNB in problems sampled from naive Bayes 0.45.



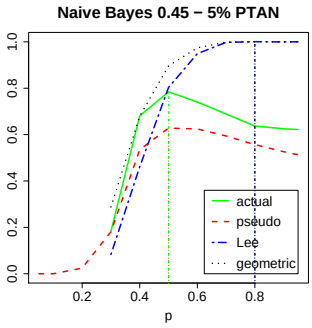
(a) 0.1%



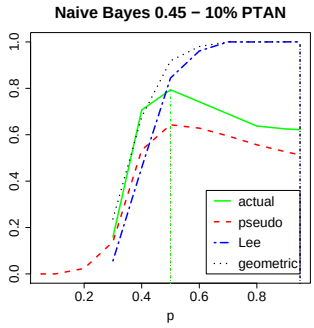
(b) 0.5%



(c) 1%

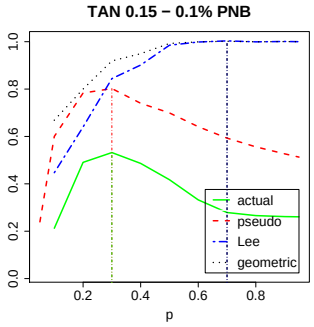


(d) 5%

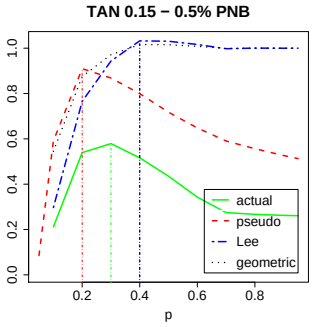


(e) 10%

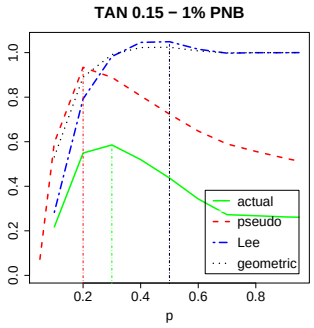
Fig. C.6. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PTAN in problems sampled from naive Bayes 0.45.



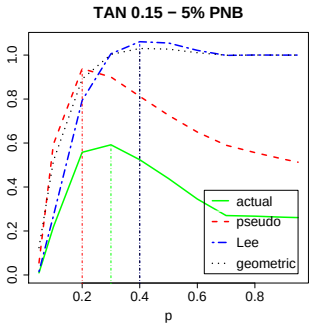
(a) 0.1%



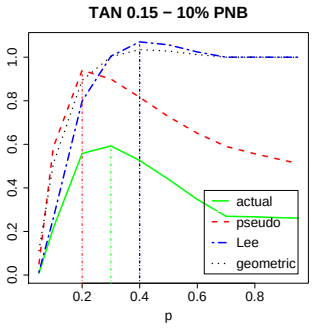
(b) 0.5%



(c) 1%

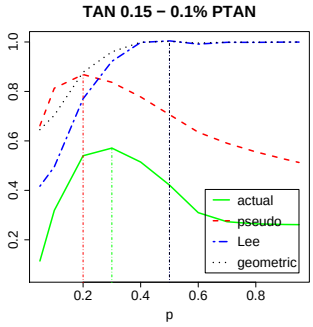


(d) 5%

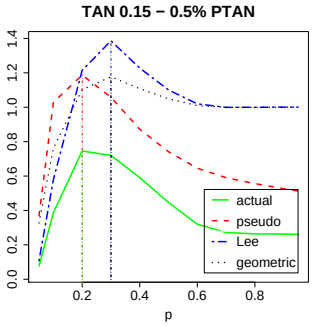


(e) 10%

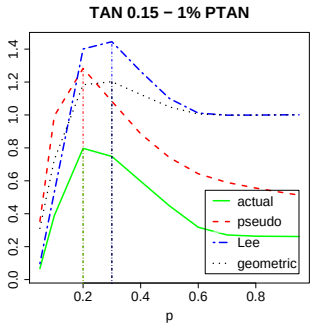
Fig. C.7. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PNB in problems sampled from TAN 0.15.



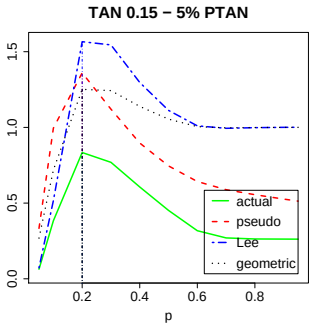
(a) 0.1%



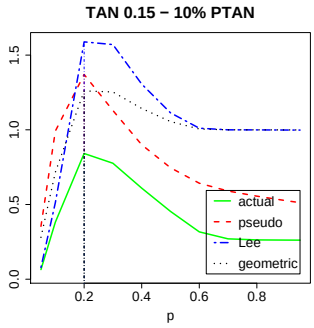
(b) 0.5%



(c) 1%

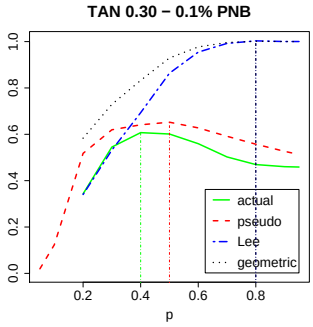


(d) 5%

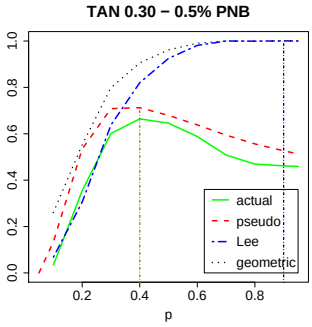


(e) 10%

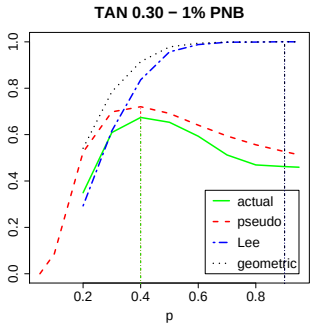
Fig. C.8. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PTAN in problems sampled from TAN 0.15.



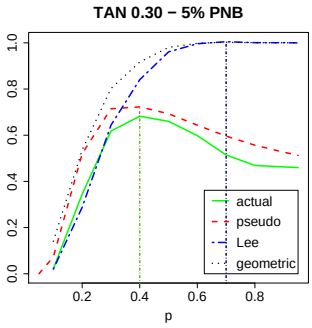
(a) 0.1%



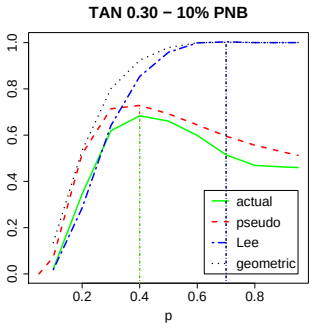
(b) 0.5%



(c) 1%

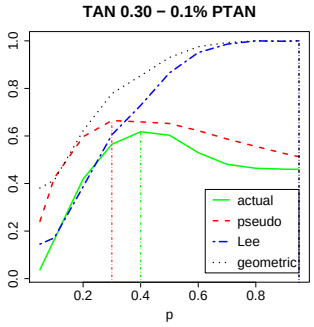


(d) 5%

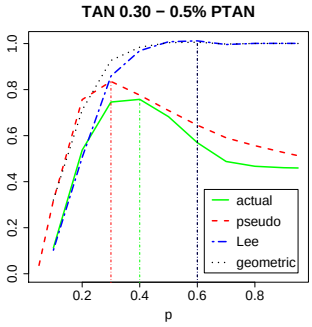


(e) 10%

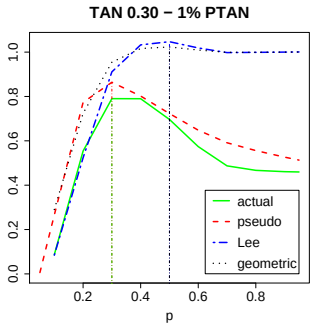
Fig. C.9. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PNB in problems sampled from TAN 0.30.



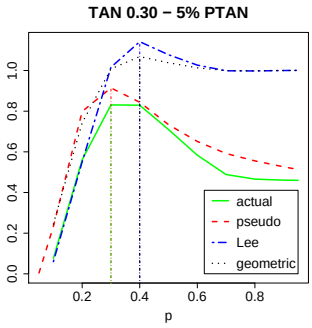
(a) 0.1%



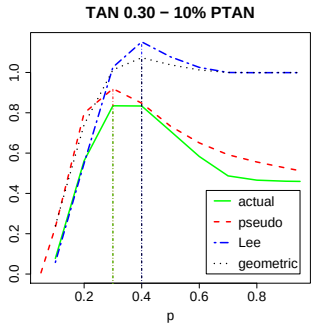
(b) 0.5%



(c) 1%

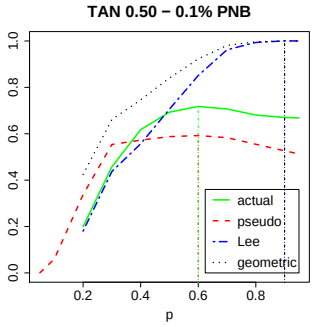


(d) 5%

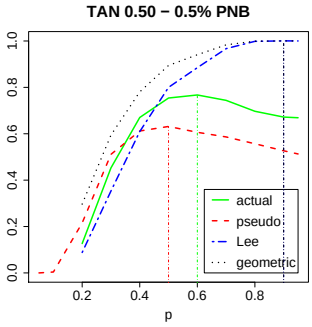


(e) 10%

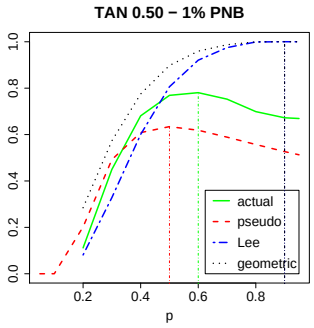
Fig. C.10. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PTAN in problems sampled from TAN 0.30.



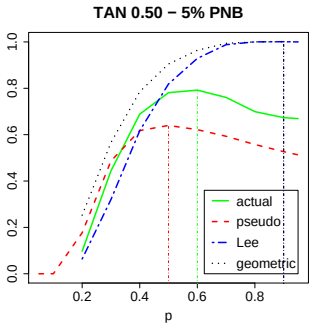
(a) 0.1%



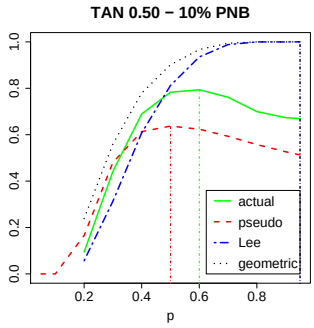
(b) 0.5%



(c) 1%

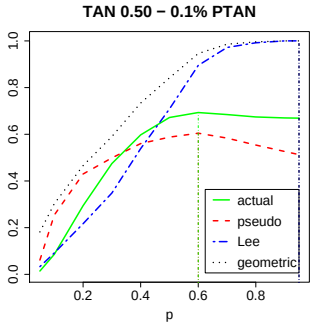


(d) 5%

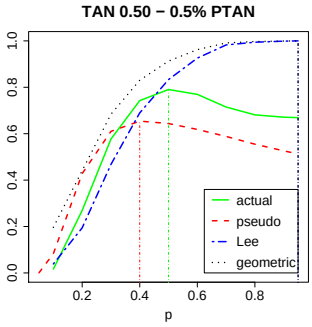


(e) 10%

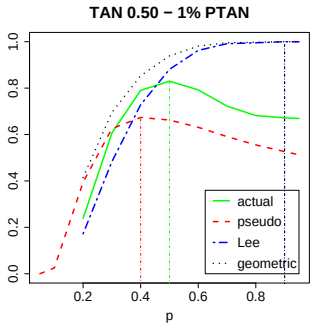
Fig. C.11. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PNB in problems sampled from TAN 0.50.



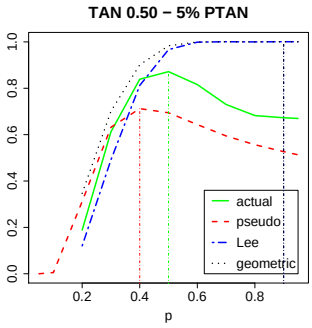
(a) 0.1%



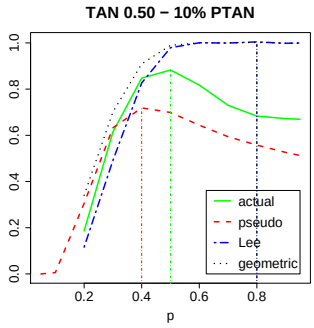
(b) 0.5%



(c) 1%



(d) 5%



(e) 10%

Fig. C.12. Comparison between the actual F measure, the pseudo F measure, the Lee metric and its square root obtained by the PTAN in problems sampled from TAN 0.50.

D

Experimental evaluation of wrapper positive Bayesian network classifiers on real-life datasets

This appendix contains the complete set of results obtained by wrapper positive Bayesian network classifiers in real-life based datasets. The results include the comparison between the F measure obtained by the wrapper approaches and the classifiers trained with a fixed p (0.25 and the actual value) and the graphical representation of the pseudo F and the Lee metric as a function of the p parameter. The significance of the differences between the wrapper approaches and the positive Bayesian classifiers with a fixed p have been assessed running Wilcoxon paired tests on the results obtained on the 100 problems sampled using each scheme (see Chapter 5).

In the tables included in this appendix, the first column represents the number of known positive examples and the second column the ratio of positive cases hidden in $\mathcal{D}_u$. The column labelled as 'Wlx.' shows the results of the Wilcoxon test (with a significance level of 1%). The results of the six tests in each row are codified as follows:

- a - no significant differences between the wrapper approach (based on the pseudo F) and setting the p at 0.25
- b - no significant differences between the wrapper approach (based on the Lee metric) and setting the p at 0.25
- c - no significant differences between the wrapper approach (based on the pseudo F) and setting the p at the actual value
- d - no significant differences between the wrapper approach (based on the Lee metric) and setting the p at the actual value
- e - no significant differences between the two wrapper approaches (based on the pseudo F and on the Lee metric)

The plots in the figures show, for each scheme, the pseudo F and the Lee metric obtained training the positive Bayesian network classifiers with different values of p.

Acceptor Sites F measure

			wPNB		PNB			wPTAN		PTAN		
	$\lvert\mathcal{D}_p\rvert$	rat.	ps.	Lee	0.25	actual	Wlx.	ps.	Lee	0.25	actual	Wlx.
Coding	100	0.01	53.42	2.99	11.49	53.87	c	49.65	2.51	12.97	50.09	c
	100	0.10	55.18	24.00	67.21	78.62	-	30.67	21.46	65.39	70.77	-
	100	0.20	39.86	41.29	83.86	84.52	e	27.19	37.10	74.98	73.57	e
	100	0.30	22.18	52.90	86.94	87.70	-	48.51	49.50	71.94	74.65	e
	100	0.40	41.51	63.24	84.38	89.53	-	48.36	59.35	63.81	73.84	-
	100	0.50	57.71	71.07	78.55	90.86	-	62.12	68.03	54.06	74.36	e
	500	0.01	55.93	19.28	11.39	56.52	c	54.93	20.20	13.00	55.74	c
	500	0.10	58.53	70.87	68.59	80.18	-	38.84	67.57	68.49	77.89	b
	500	0.20	43.10	82.78	85.22	86.12	-	15.76	77.53	82.19	82.23	-
	500	0.30	24.88	86.36	88.88	89.32	-	51.75	79.62	82.26	83.41	-
	500	0.40	34.24	89.42	87.49	91.45	-	58.59	79.68	76.56	81.74	-
	500	0.50	58.31	91.32	82.64	93.04	-	53.44	79.68	66.79	80.91	d
	1,000	0.01	55.83	41.72	11.37	55.83	c	56.42	40.88	12.85	56.42	c
	1,000	0.10	58.65	75.38	68.77	80.56	-	39.63	79.15	68.92	79.08	-
	1,000	0.20	43.13	84.28	85.34	86.25	-	17.05	78.59	82.92	83.27	-
	1,000	0.30	25.17	82.93	89.05	89.50	-	53.65	73.81	83.63	84.53	-
	1,000	0.40	22.43	80.23	87.79	91.72	-	51.11	69.77	78.57	81.64	-
	1,000	0.50	58.70	82.53	83.13	93.24	b	55.16	67.19	69.00	80.98	-
Intron	100	0.01	29.78	2.58	10.42	29.78	c	22.97	2.30	10.60	23.03	c
	100	0.10	33.02	22.67	58.85	65.88	-	22.37	20.03	55.37	56.87	-
	100	0.20	24.40	38.55	75.33	75.52	-	45.84	35.58	66.46	64.52	-
	100	0.30	46.05	50.98	79.81	80.93	e	49.90	48.12	65.32	68.74	e
	100	0.40	57.02	61.64	78.17	84.29	-	58.93	58.62	58.43	71.22	b e
	100	0.50	59.74	69.54	72.61	86.73	-	67.18	67.55	49.58	73.06	e
	500	0.01	31.87	10.12	10.67	32.43	b c	30.50	11.09	11.37	30.98	b c
	500	0.10	37.34	54.86	59.88	68.13	-	30.42	54.54	61.23	67.45	-
	500	0.20	22.01	70.64	77.10	77.75	-	62.08	71.00	77.11	77.10	e
	500	0.30	49.38	78.99	82.47	83.10	-	60.99	79.18	80.91	82.23	-
	500	0.40	64.07	84.11	82.09	86.58	-	63.99	82.69	75.61	84.88	-
	500	0.50	61.30	87.43	77.73	89.29	-	71.53	83.62	63.76	85.99	-
	1,000	0.01	32.52	18.49	10.73	32.52	c	32.15	19.85	11.48	32.15	c
	1,000	0.10	37.41	67.26	59.93	68.42	-	31.77	68.33	61.97	68.91	d
	1,000	0.20	22.50	76.18	77.52	78.26	-	67.42	75.35	78.76	79.17	-
	1,000	0.30	40.18	77.58	82.85	83.38	-	60.57	73.98	83.18	83.94	-
	1,000	0.40	63.60	78.41	82.51	86.85	-	64.98	75.35	78.49	86.84	b
	1,000	0.50	55.56	80.72	78.41	89.64	-	69.07	71.70	66.27	88.30	a e
Mixed	100	0.01	35.41	2.77	11.57	35.41	c	32.37	2.38	12.61	32.37	c
	100	0.10	40.68	22.62	62.70	70.62	-	22.01	20.68	61.41	62.57	e
	100	0.20	25.74	40.00	79.05	79.32	-	42.46	36.10	70.81	68.74	e
	100	0.30	34.74	52.36	82.83	83.79	-	49.49	48.50	67.86	71.05	e
	100	0.40	57.21	62.03	80.86	86.67	-	54.71	58.92	60.68	71.83	a e
	100	0.50	57.76	70.23	74.93	88.71	-	65.58	67.78	51.14	73.46	e
	500	0.01	36.54	12.73	11.70	36.83	b c	36.30	12.71	12.50	36.39	b c
	500	0.10	44.77	59.51	63.69	72.69	-	27.90	57.75	64.76	70.49	-
	500	0.20	28.62	72.73	80.65	81.33	-	42.24	70.57	78.29	77.67	-
	500	0.30	16.58	81.80	85.41	85.98	-	60.31	76.39	78.96	80.95	-
	500	0.40	65.74	86.58	84.47	88.88	-	62.91	79.09	73.82	82.15	-
	500	0.50	56.34	88.83	79.66	91.02	-	65.49	80.80	64.43	82.76	a
	1,000	0.01	37.41	23.70	11.78	37.41	c	37.17	22.82	12.53	37.17	c
	1,000	0.10	44.70	71.75	63.73	72.85	-	28.90	71.10	64.81	71.09	-
	1,000	0.20	29.01	79.35	80.85	81.62	-	28.34	74.38	79.10	78.75	-
	1,000	0.30	14.75	80.97	85.63	86.15	-	60.21	72.73	80.44	82.05	-
	1,000	0.40	66.59	81.41	85.08	89.24	-	64.78	74.93	76.02	83.40	b
	1,000	0.50	55.60	82.46	80.53	91.42	-	66.31	68.91	66.95	83.41	a b e

Table D.1. Results of the comparison between the F measure obtained by PNB and PTAN setting the p at 0.25 and the actual value and setting it using the pseudo F and the Lee metric in datasets obtained from the database of acceptor sites.

Donor Sites F measure

			wPNB		PNB			wPTAN		PTAN		
	$\|\mathcal{D}_p\|$	rat.	ps.	Lee	0.25	actual	Wlx.	ps.	Lee	0.25	actual	Wlx.
Coding	100	0.01	28.91	3.38	8.80	28.91	c	26.03	3.07	9.65	26.05	c
	100	0.10	42.61	27.60	66.79	75.90	-	71.51	24.80	65.68	68.22	-
	100	0.20	75.08	45.06	84.32	83.95	-	66.68	40.21	78.61	76.29	-
	100	0.30	66.03	55.89	85.74	87.68	-	73.76	52.10	75.78	79.32	-
	100	0.40	75.46	66.15	81.21	90.15	-	75.08	61.91	65.72	79.81	-
	100	0.50	77.75	74.34	71.65	91.58	-	75.33	69.88	51.95	80.50	-
	500	0.01	30.00	16.73	8.02	30.00	c	29.19	16.55	7.70	29.16	c
	500	0.10	26.51	60.74	67.77	77.41	b	77.24	60.88	62.90	75.74	b
	500	0.20	77.51	78.20	85.37	84.91	-	70.67	72.65	81.12	83.23	e
	500	0.30	66.26	83.86	87.08	89.03	-	74.08	78.37	84.98	83.79	e
	500	0.40	75.86	87.12	82.83	91.47	-	78.40	81.73	80.94	82.60	b d
	500	0.50	75.66	90.59	74.33	93.13	a	75.70	84.59	71.47	83.57	-
	1,000	0.01	30.21	30.08	7.97	30.21	c d e	30.60	30.87	7.28	30.60	c d e
	1,000	0.10	19.06	77.65	67.80	77.28	-	78.71	78.43	61.20	76.73	e
	1,000	0.20	76.80	79.90	85.59	85.20	-	72.19	75.29	80.41	84.18	-
	1,000	0.30	65.66	79.71	87.16	89.19	-	73.26	78.08	86.22	83.34	c d e
	1,000	0.40	73.46	83.51	83.05	91.66	b	80.39	80.83	84.05	81.75	-
	1,000	0.50	73.23	82.62	74.80	93.35	a	74.75	74.88	75.08	83.97	a b e
Intron	100	0.01	24.47	3.39	10.15	24.41	c	20.59	3.09	10.19	18.56	-
	100	0.10	57.00	26.30	64.16	70.67	a c	67.25	24.33	62.35	64.88	-
	100	0.20	71.41	45.28	80.98	80.41	-	61.43	40.24	75.42	73.46	-
	100	0.30	63.67	57.42	83.31	85.30	-	71.82	51.99	74.91	77.76	-
	100	0.40	74.05	66.43	79.22	88.42	-	74.47	61.44	67.31	79.35	-
	100	0.50	75.11	73.71	70.12	90.19	e	75.43	69.34	51.95	79.98	-
	500	0.01	25.42	13.56	10.16	25.48	c	22.00	15.03	7.03	22.00	c
	500	0.10	28.01	60.92	64.68	71.93	b	73.31	55.51	54.83	71.52	b
	500	0.20	74.23	74.38	82.39	81.60	-	70.65	68.01	79.80	81.86	e
	500	0.30	63.97	82.90	84.52	86.67	-	68.15	76.01	83.69	83.86	-
	500	0.40	73.93	86.64	81.07	89.78	-	78.20	81.80	79.89	83.44	d
	500	0.50	74.88	89.81	73.25	91.89	a	75.18	84.50	71.21	84.69	d
	1,000	0.01	25.95	24.44	10.15	25.93	c	23.07	27.27	6.77	23.07	c
	1,000	0.10	23.60	72.27	64.75	72.30	-	75.23	74.04	51.54	72.38	-
	1,000	0.20	73.70	77.30	82.57	81.67	-	69.74	74.37	78.99	82.69	-
	1,000	0.30	63.54	77.57	84.59	86.93	-	63.70	75.70	84.56	83.81	-
	1,000	0.40	75.25	81.92	81.36	90.06	b	78.54	80.02	81.57	82.62	-
	1,000	0.50	72.42	79.41	73.77	92.22	a	73.79	72.70	73.71	83.86	a b e
Mixed	100	0.01	23.74	3.24	10.17	23.74	c	20.58	3.12	10.72	19.18	-
	100	0.10	49.98	26.32	65.79	73.04	a c	68.49	24.79	65.52	64.87	-
	100	0.20	72.58	45.30	82.61	82.04	-	66.35	40.33	76.95	74.37	-
	100	0.30	63.63	56.66	84.41	86.40	-	71.72	51.79	73.77	77.40	-
	100	0.40	73.42	65.89	79.91	89.14	-	75.05	61.04	64.54	78.98	-
	100	0.50	76.81	74.57	70.65	90.81	e	74.91	69.28	51.47	79.98	-
	500	0.01	24.61	16.02	10.19	24.56	c	22.06	14.90	8.34	21.92	c
	500	0.10	34.32	61.34	66.39	74.38	b	75.10	59.64	62.89	72.42	b
	500	0.20	75.17	75.49	83.80	82.96	-	69.48	70.59	81.33	81.78	e
	500	0.30	63.73	83.60	85.55	87.77	-	70.59	76.95	83.32	83.74	e
	500	0.40	76.41	86.56	81.77	90.66	-	77.00	81.48	79.31	83.86	-
	500	0.50	75.90	90.41	73.13	92.52	-	77.61	84.36	69.76	84.13	d
	1,000	0.01	25.14	27.24	10.17	25.14	c	23.44	30.10	7.72	23.08	c
	1,000	0.10	24.50	74.64	66.69	74.64	-	76.48	75.05	62.28	73.50	-
	1,000	0.20	74.76	78.02	84.09	82.98	-	68.09	73.59	80.77	82.53	-
	1,000	0.30	63.29	79.88	85.63	87.98	-	73.63	77.49	84.39	83.04	e
	1,000	0.40	74.72	82.62	81.88	90.81	b	77.14	78.60	81.32	83.55	-
	1,000	0.50	72.12	81.84	73.90	92.70	a	75.91	76.05	74.72	83.84	a b e

Table D.2. Results of the comparison between the F measure obtained by PNB and PTAN setting the p at 0.25 and the actual value and setting it using the pseudo F and the Lee metric in datasets obtained from the database of donor sites.

Letter Recognition F measure

			wPNB		PNB			wPTAN		PTAN		
	$\lvert\mathcal{D}_p\rvert$	rat.	ps.	Lee	0.25	actual	Wlx.	ps.	Lee	0.25	actual	Wlx.
D	100	0.01	33.70	2.56	10.70	40.72	-	18.79	2.96	30.23	19.52	c
	100	0.10	39.99	22.25	59.75	67.89	-	28.05	25.07	49.44	27.13	c d e
	100	0.20	37.79	38.55	74.11	73.93	e	40.71	42.23	32.29	25.94	e
	100	0.30	50.52	51.37	75.27	77.11	e	52.00	54.39	18.99	25.15	e
	100	0.40	61.41	61.57	70.60	79.64	e	60.65	62.45	10.42	25.85	e
	100	0.50	66.29	69.99	61.40	80.26	e	64.09	66.14	6.13	28.39	e
	200	0.01	42.77	11.63	10.17	46.40	b	36.86	8.95	24.33	40.65	c
	200	0.10	51.71	56.29	58.87	69.13	b	55.96	36.43	68.28	56.79	c
	200	0.20	42.54	69.60	76.56	77.26	-	66.17	49.98	64.40	58.74	a
	200	0.30	45.51	75.84	81.01	81.44	-	69.66	59.22	52.45	59.68	d
	200	0.40	65.50	78.46	80.35	84.07	b	71.21	66.76	38.93	59.40	-
	200	0.50	74.90	81.29	75.36	86.22	a	73.67	72.51	26.43	59.32	e
	300	0.01	45.87	31.16	10.14	45.87	c	48.22	13.53	20.70	46.79	c
	300	0.10	54.11	63.83	59.16	70.15	-	52.79	42.33	72.04	68.22	-
	300	0.20	35.97	76.08	76.55	77.66	d	72.69	57.77	75.38	71.83	a c
	300	0.30	46.40	79.01	82.08	82.35	-	76.53	65.78	68.30	72.94	-
	300	0.40	66.94	79.65	81.87	84.58	b	79.12	72.32	57.18	72.40	d
	300	0.50	67.76	84.40	80.02	87.76	-	79.87	76.76	45.04	73.64	-
P	100	0.01	42.59	2.28	12.89	53.27	-	28.90	3.62	35.89	33.99	c
	100	0.10	69.54	20.73	67.59	78.52	-	35.35	29.38	60.10	37.08	c
	100	0.20	59.33	36.72	82.48	83.23	-	37.05	47.60	45.27	37.18	b c
	100	0.30	44.98	49.91	84.27	85.13	e	45.43	58.76	26.61	34.03	-
	100	0.40	51.08	60.85	80.75	85.79	e	58.61	66.14	15.40	32.99	-
	100	0.50	60.47	70.02	72.57	85.69	a e	68.78	69.46	8.88	34.55	e
	200	0.01	52.10	38.96	13.67	54.26	-	54.08	22.32	29.90	54.96	c
	200	0.10	74.79	74.16	68.02	80.60	e	60.20	45.05	75.64	67.27	-
	200	0.20	64.90	80.90	83.90	84.85	b	71.25	56.03	72.18	66.91	a
	200	0.30	54.06	84.22	87.35	87.51	-	72.96	64.44	59.81	65.87	d
	200	0.40	57.71	81.57	86.52	88.57	-	75.87	70.40	44.10	64.57	-
	200	0.50	67.68	83.65	82.63	89.69	-	75.04	75.17	31.21	63.19	e
	300	0.01	53.94	53.94	14.25	53.94	c d e	61.88	56.56	27.80	63.56	c e
	300	0.10	76.89	77.09	69.42	81.49	e	52.57	54.85	77.73	75.82	e
	300	0.20	66.69	71.01	84.60	85.78	-	61.77	62.83	79.31	76.44	e
	300	0.30	44.49	73.40	87.78	87.99	-	72.93	70.81	72.72	76.20	a c
	300	0.40	48.64	83.79	87.73	89.42	-	80.11	74.59	63.96	76.83	-
	300	0.50	68.41	83.58	85.19	90.39	b	81.89	77.90	50.62	75.89	-
U	100	0.01	48.87	2.37	13.46	59.67	-	30.36	3.11	32.87	37.83	a
	100	0.10	61.08	20.56	66.89	75.56	-	32.77	26.48	57.09	44.38	-
	100	0.20	50.70	36.96	79.16	79.42	-	35.05	44.87	43.01	37.88	a b c
	100	0.30	44.52	49.50	80.23	81.27	e	46.67	55.66	29.70	33.75	-
	100	0.40	50.91	60.13	76.48	81.91	e	52.82	63.08	18.33	33.08	-
	100	0.50	63.77	69.44	68.66	82.27	a b e	65.04	67.47	10.05	33.66	e
	200	0.01	62.95	32.09	13.54	65.67	-	61.90	21.69	29.09	61.82	c
	200	0.10	69.53	61.84	69.56	78.37	a b	58.74	37.98	73.82	69.22	-
	200	0.20	58.61	75.08	81.48	81.74	-	63.26	50.36	70.92	66.99	c
	200	0.30	55.94	76.66	83.42	83.80	-	65.83	61.40	61.26	65.96	b c
	200	0.40	56.05	77.52	82.40	85.26	-	71.25	67.37	49.54	64.54	-
	200	0.50	61.06	81.89	79.04	86.82	-	74.34	73.21	34.83	62.13	-
	300	0.01	66.60	66.60	13.87	66.60	c d e	70.02	53.56	26.34	69.51	c
	300	0.10	71.33	71.86	70.36	78.89	a e	59.31	48.31	77.51	77.14	-
	300	0.20	60.78	67.76	81.87	82.31	-	52.12	56.08	79.07	77.41	e
	300	0.30	41.39	74.93	84.29	84.50	-	69.36	65.19	74.21	76.56	-
	300	0.40	55.36	79.83	84.22	86.85	-	76.42	72.15	66.89	76.66	c
	300	0.50	65.74	81.12	81.25	87.88	b	80.75	76.68	54.47	74.58	-

Table D.3. Results of the comparison between the F measure obtained by PNB and PTAN setting the p at 0.25 and the actual value and setting it using the pseudo F and the Lee metric in datasets obtained from the Letter Recognition database.

Nursery F measure

		wPNB		PNB			wPTAN		PTAN		
$\lvert\mathcal{D}_p\rvert$	rat.	ps.	Lee	0.25	actual	Wlx.	ps.	Lee	0.25	actual	Wlx.
100	0.01	32.17	3.10	17.40	0.00	-	19.48	2.97	15.31	15.76	-
100	0.10	63.27	26.03	70.66	31.94	-	63.39	24.30	68.41	51.00	-
100	0.20	68.83	43.30	74.14	62.96	-	69.10	40.49	76.87	70.64	-
100	0.30	75.15	54.37	67.92	77.20	c	72.95	51.41	71.56	78.69	-
100	0.40	77.30	62.58	56.57	83.09	-	75.86	60.67	57.87	81.71	-
100	0.50	78.42	71.00	43.09	86.33	-	75.87	68.59	46.69	82.32	-
200	0.01	31.10	16.47	18.93	0.00	-	23.07	18.54	16.09	15.86	b d
200	0.10	59.40	57.91	72.95	32.22	e	65.85	61.08	69.09	55.82	-
200	0.20	67.52	71.51	76.81	65.07	e	62.06	75.52	81.62	77.04	d
200	0.30	73.67	78.32	69.87	80.14	d	77.21	76.63	78.13	83.90	a b e
200	0.40	76.57	83.55	58.35	86.32	-	75.88	82.48	65.09	86.86	-
200	0.50	79.22	83.81	42.15	88.99	-	72.41	82.97	49.45	87.92	-
300	0.01	33.22	19.06	19.47	0.00	b	26.57	20.02	16.07	16.85	d
300	0.10	69.89	60.30	74.19	31.47	a	68.11	64.92	69.27	57.74	a
300	0.20	74.78	77.23	77.86	65.76	b e	76.92	78.54	82.64	79.65	a d e
300	0.30	79.38	82.58	70.38	81.17	c e	77.76	82.58	80.91	85.66	-
300	0.40	82.44	85.82	58.21	87.58	-	82.01	85.35	67.97	88.54	e
300	0.50	82.87	86.71	41.97	90.04	-	82.54	88.22	50.56	89.04	d

Table D.4. Results of the comparison between the F measure obtained by PNB and PTAN setting the p at 0.25 and the actual value and setting it using the pseudo F and the Lee metric in datasets obtained from the Nursery database.

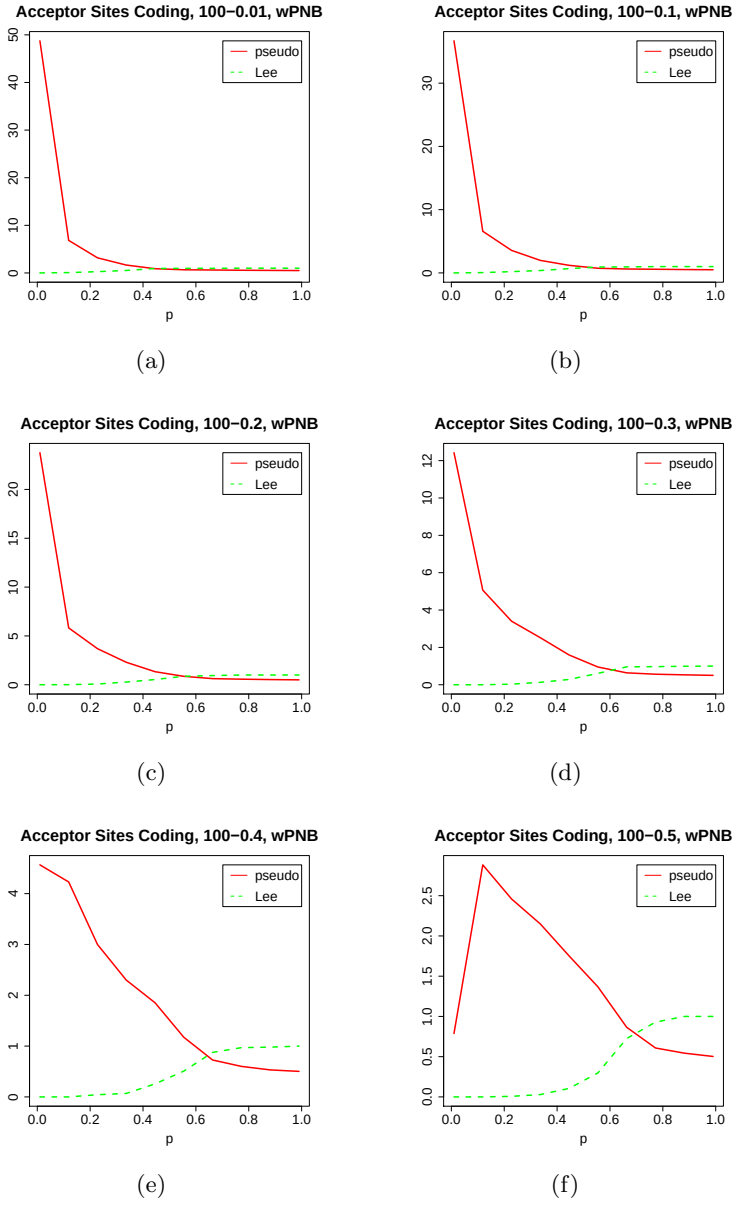
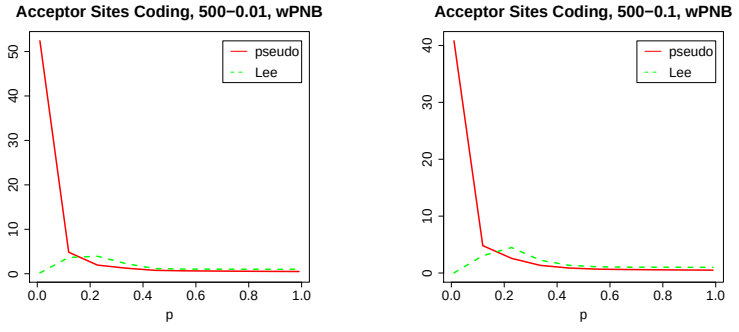
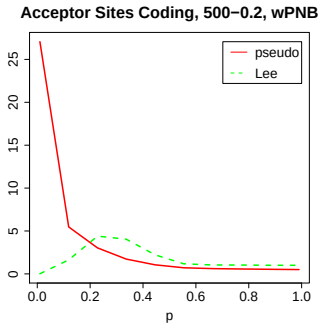


Fig. D.1. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Acceptor Sites Coding datasets where $|\mathcal{D}_p| = 100$.

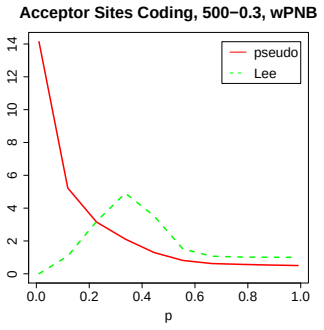


(a)

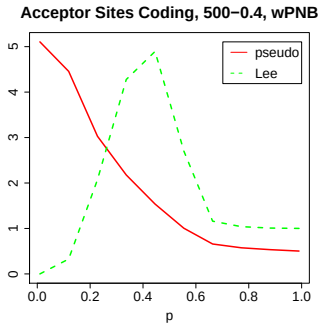
(b)



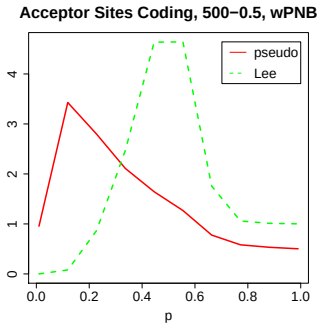
(c)



(d)



(e)



(f)

Fig. D.2. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Acceptor Sites Coding datasets where $|\mathcal{D}_p| = 500$.

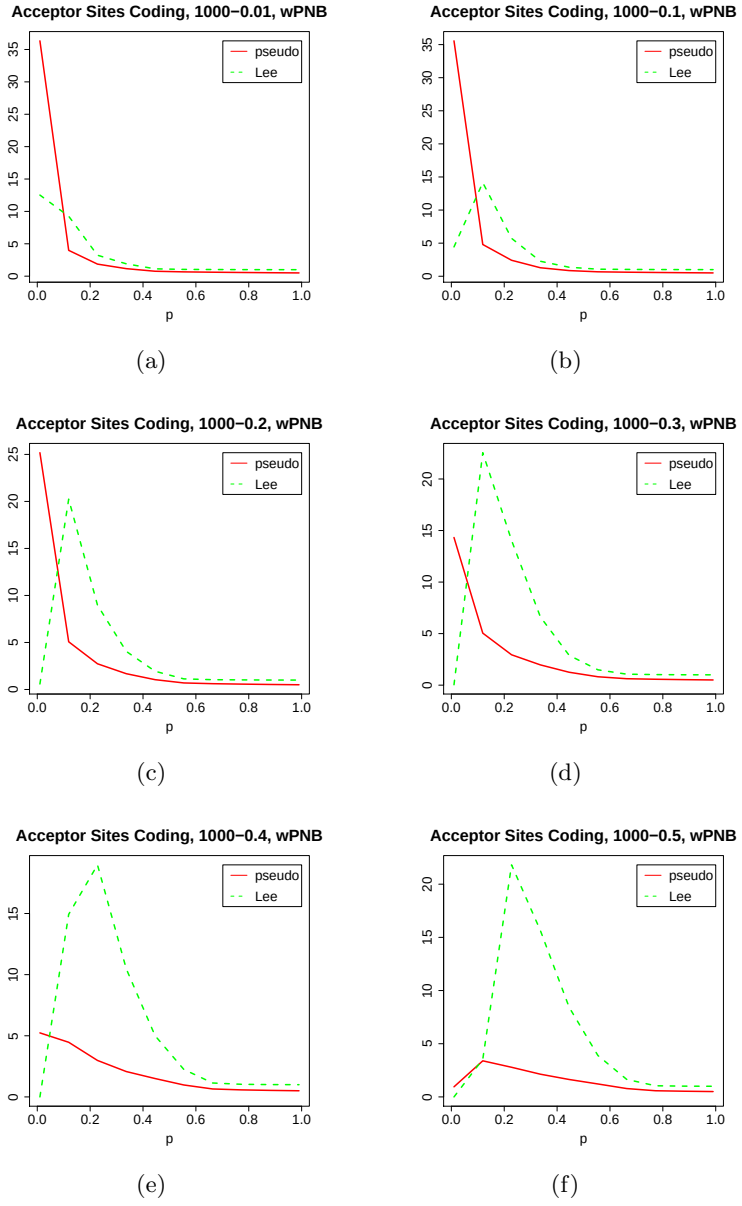


Fig. D.3. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Acceptor Sites Coding datasets where $|\mathcal{D}_p| = 1,000$.

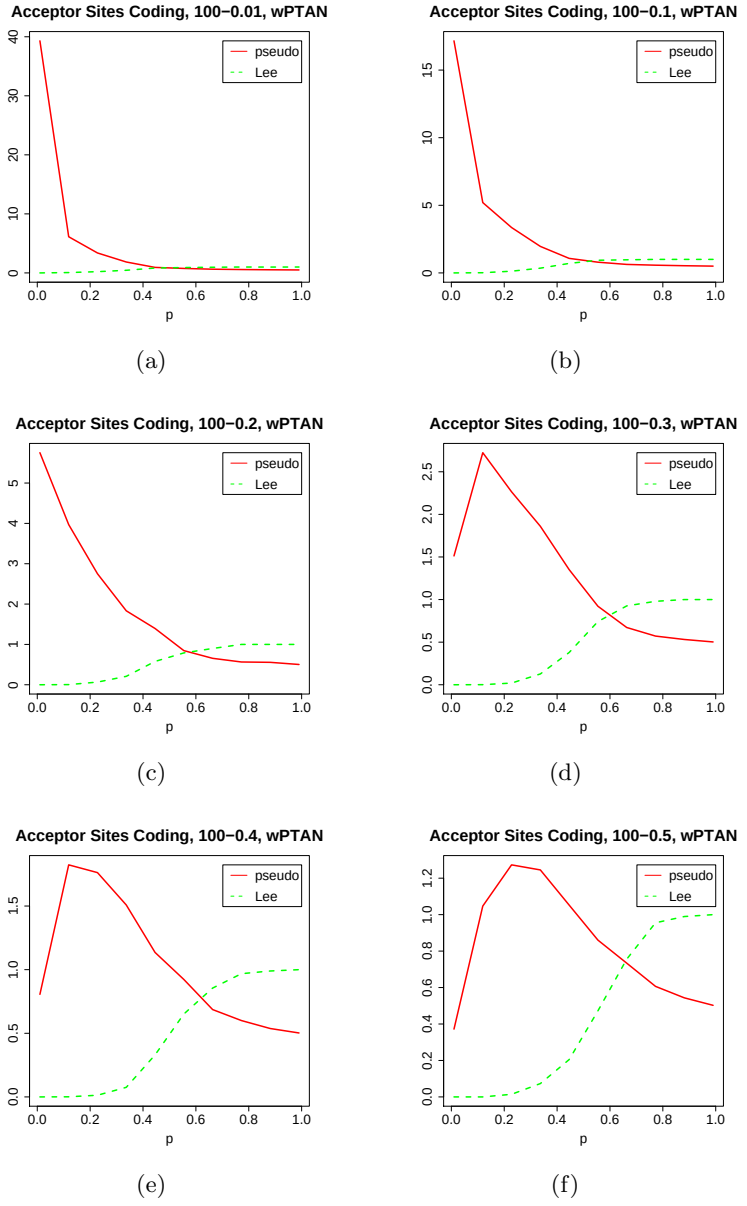


Fig. D.4. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Acceptor Sites Coding datasets where $|\mathcal{D}_p| = 100$.

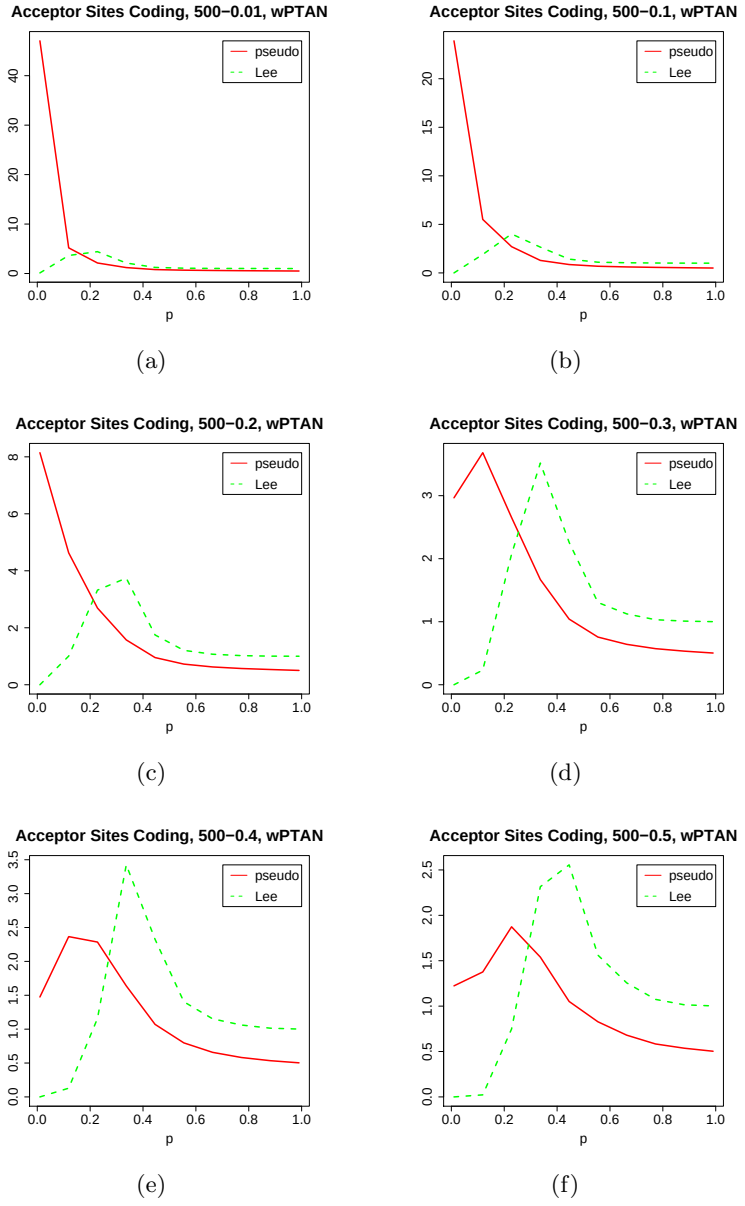


Fig. D.5. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Acceptor Sites Coding datasets where $|\mathcal{D}_p| = 500$.

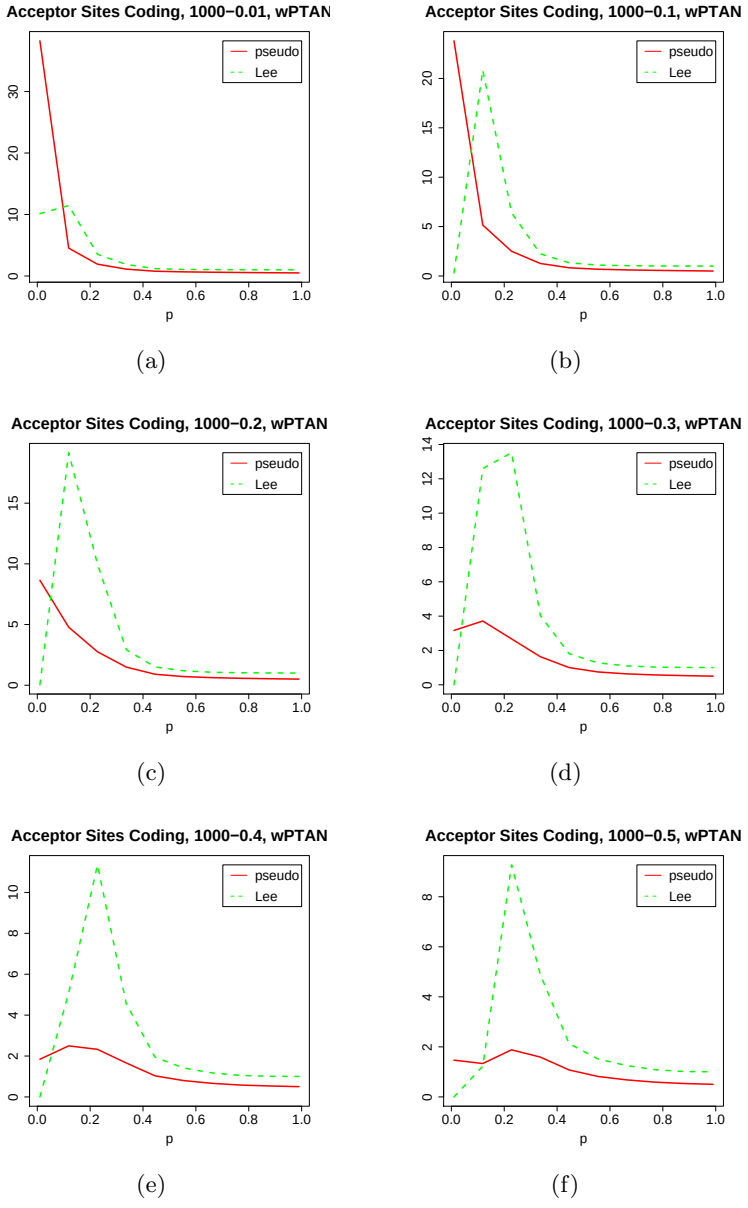


Fig. D.6. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Acceptor Sites Coding datasets where $|\mathcal{D}_p| = 1,000$.

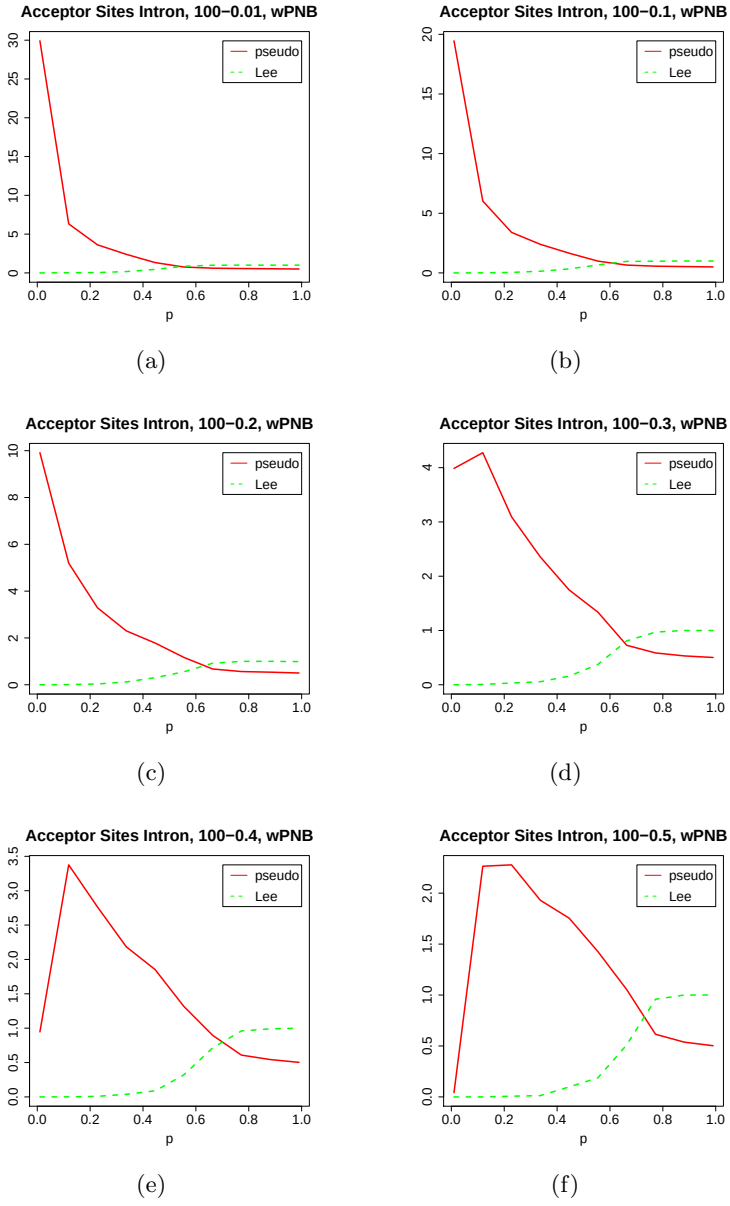
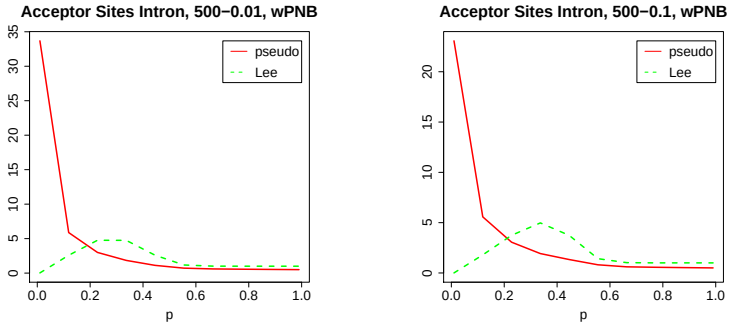
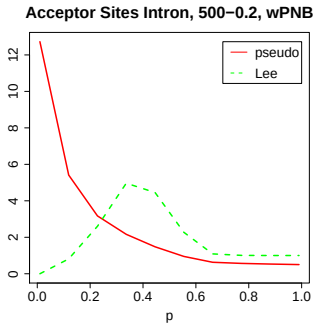


Fig. D.7. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Acceptor Sites Intron datasets where $|\mathcal{D}_p| = 100$.

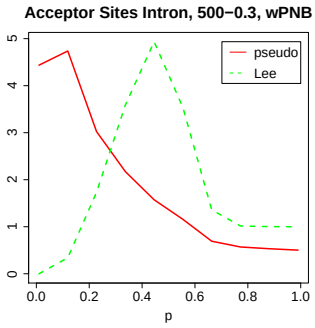


(a)

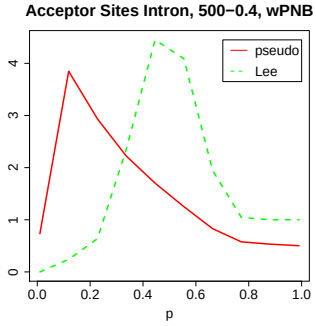
(b)



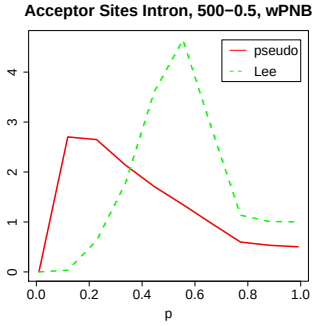
(c)



(d)



(e)



(f)

Fig. D.8. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Acceptor Sites Intron datasets where $|\mathcal{D}_p| = 500$.

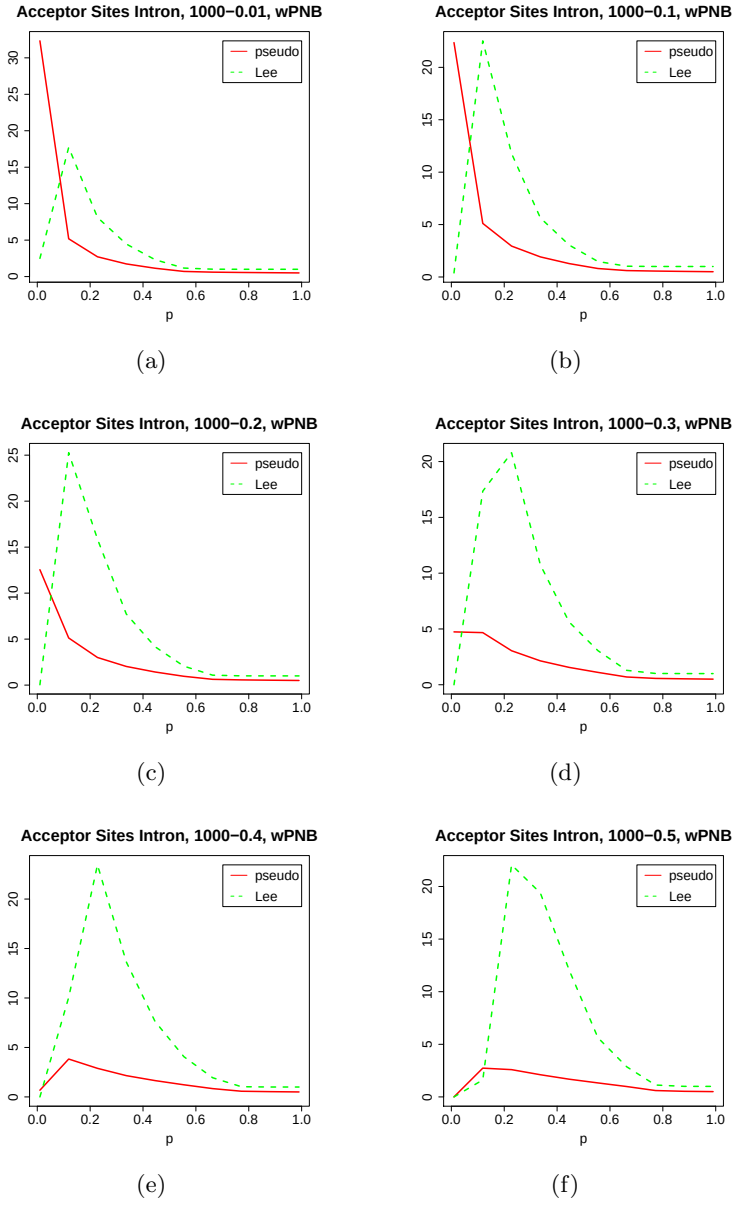


Fig. D.9. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Acceptor Sites Intron datasets where $|\mathcal{D}_p| = 1,000$.

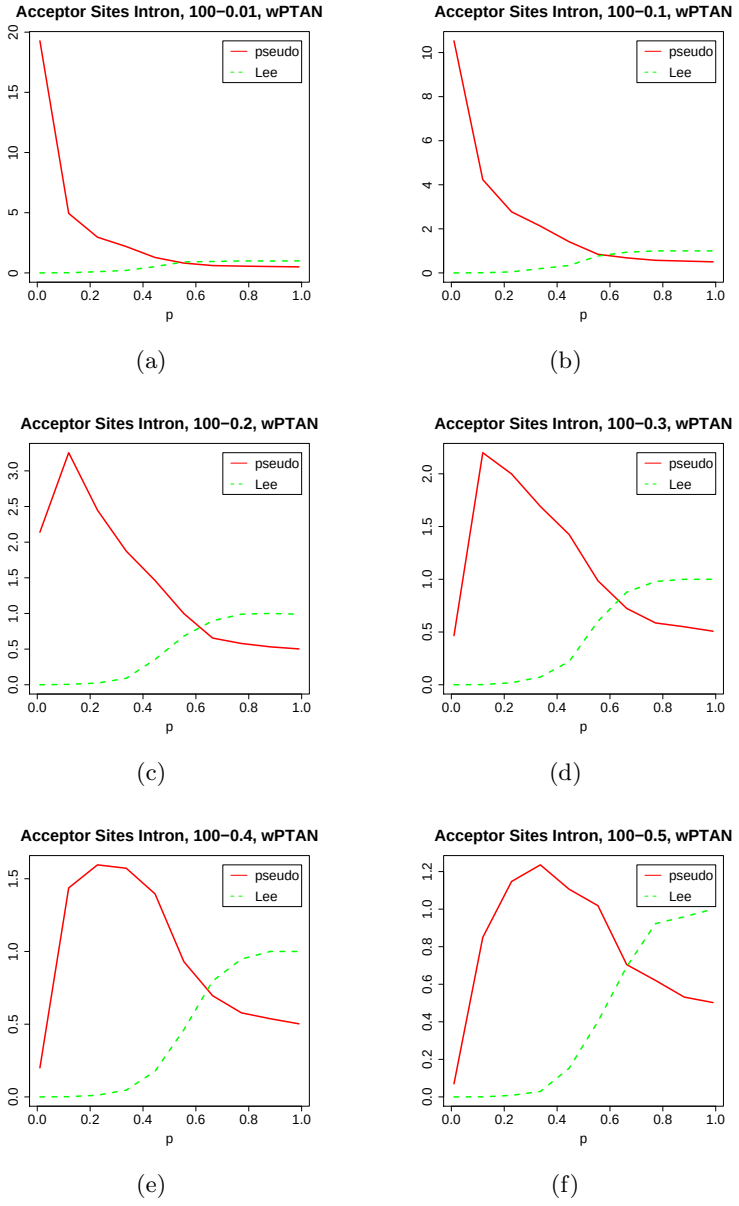


Fig. D.10. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Acceptor Sites Intron datasets where $|\mathcal{D}_p| = 100$.

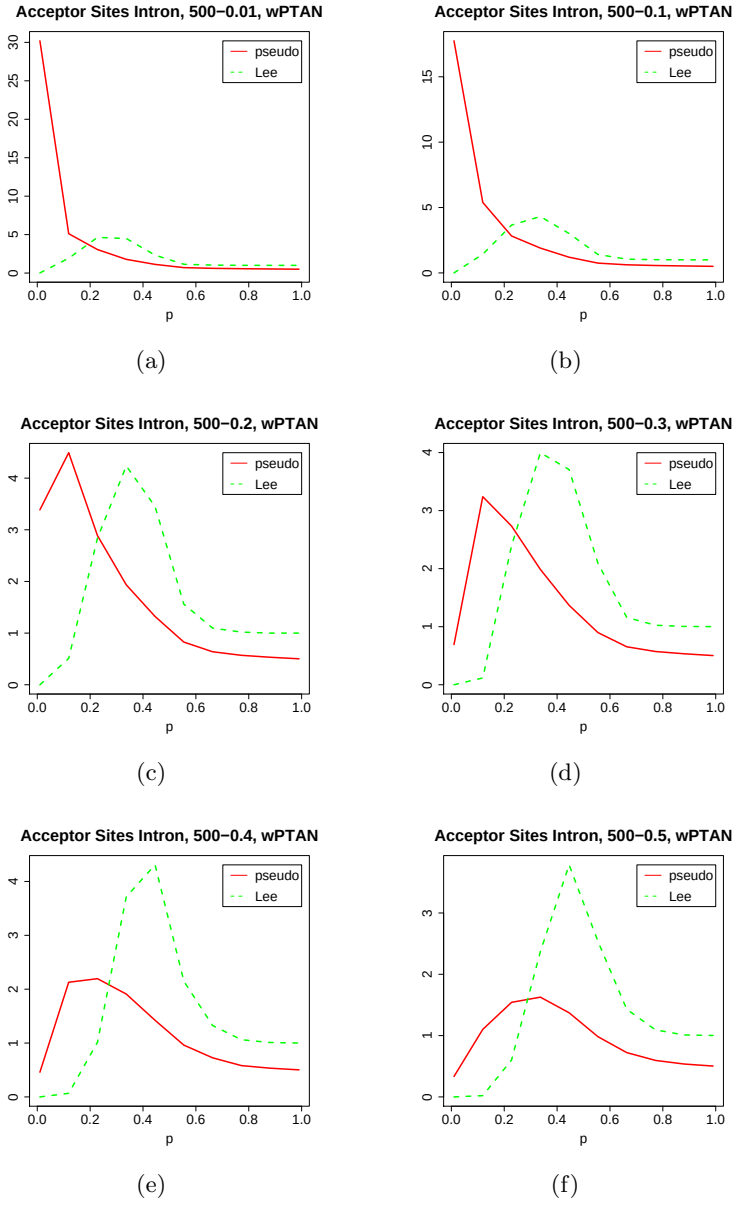


Fig. D.11. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Acceptor Sites Intron datasets where $|\mathcal{D}_p| = 500$.

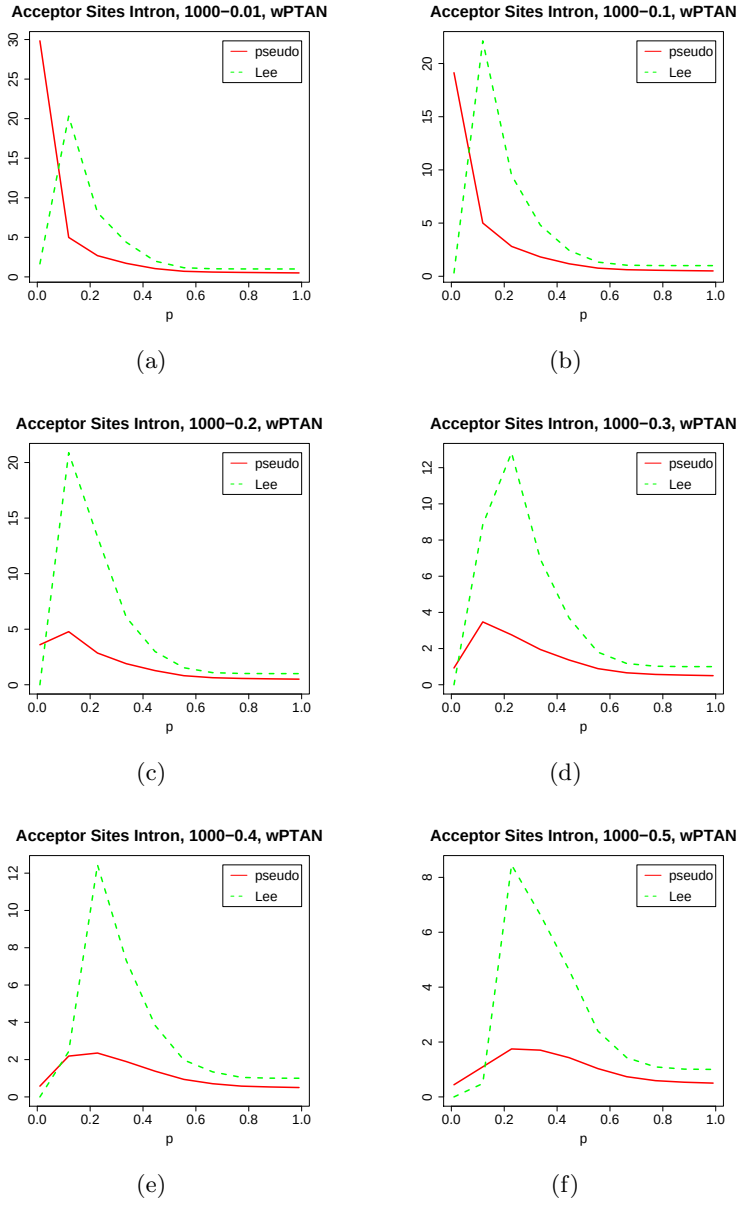


Fig. D.12. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Acceptor Sites Intron datasets where $|\mathcal{D}_p| = 1,000$.

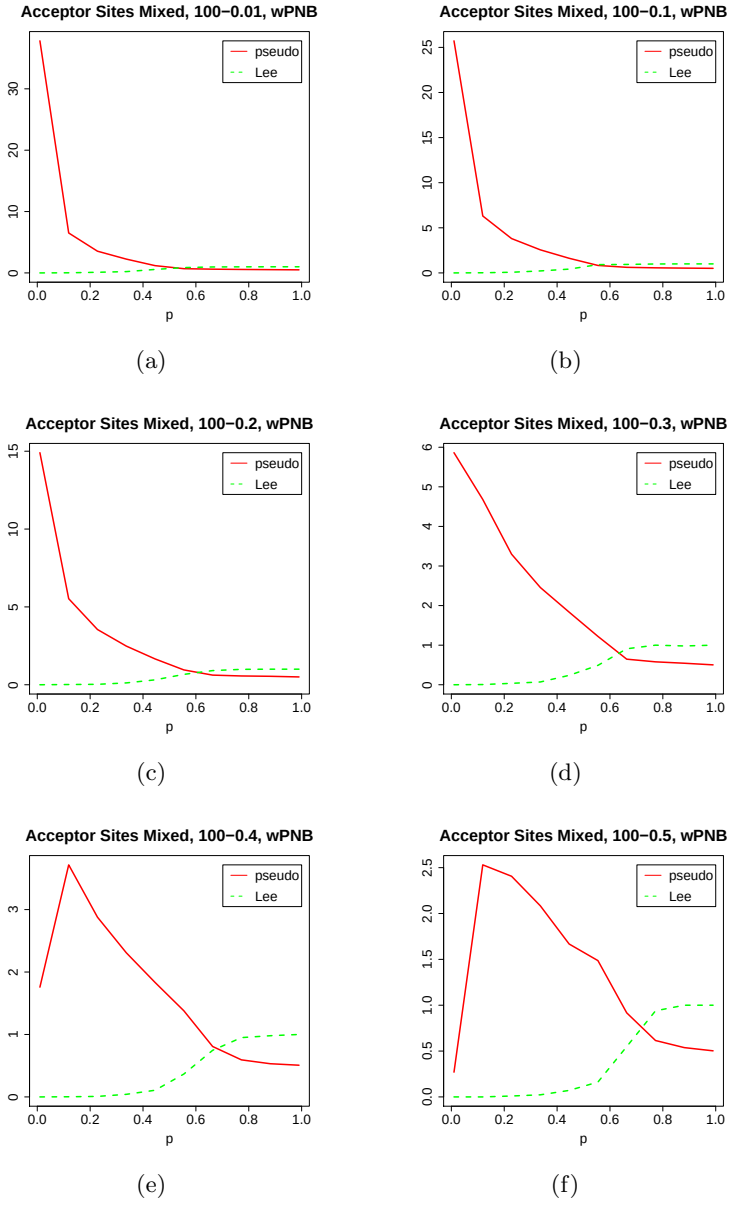
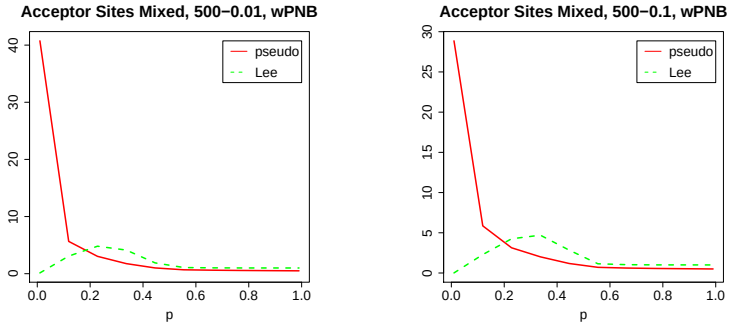
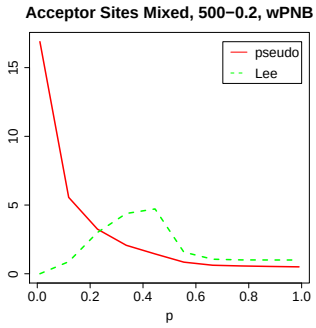


Fig. D.13. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Acceptor Sites Mixed datasets where $|\mathcal{D}_p| = 100$.

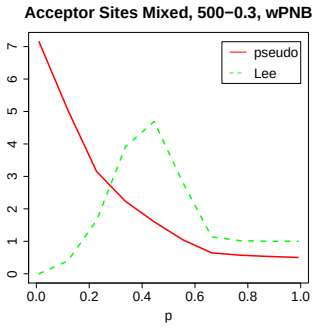


(a)

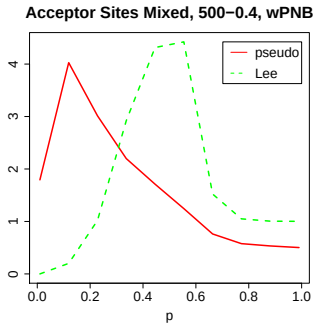
(b)



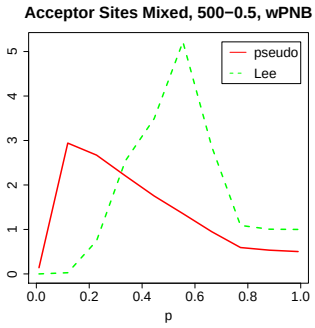
(c)



(d)



(e)



(f)

Fig. D.14. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Acceptor Sites Mixed datasets where $|\mathcal{D}_p| = 500$.

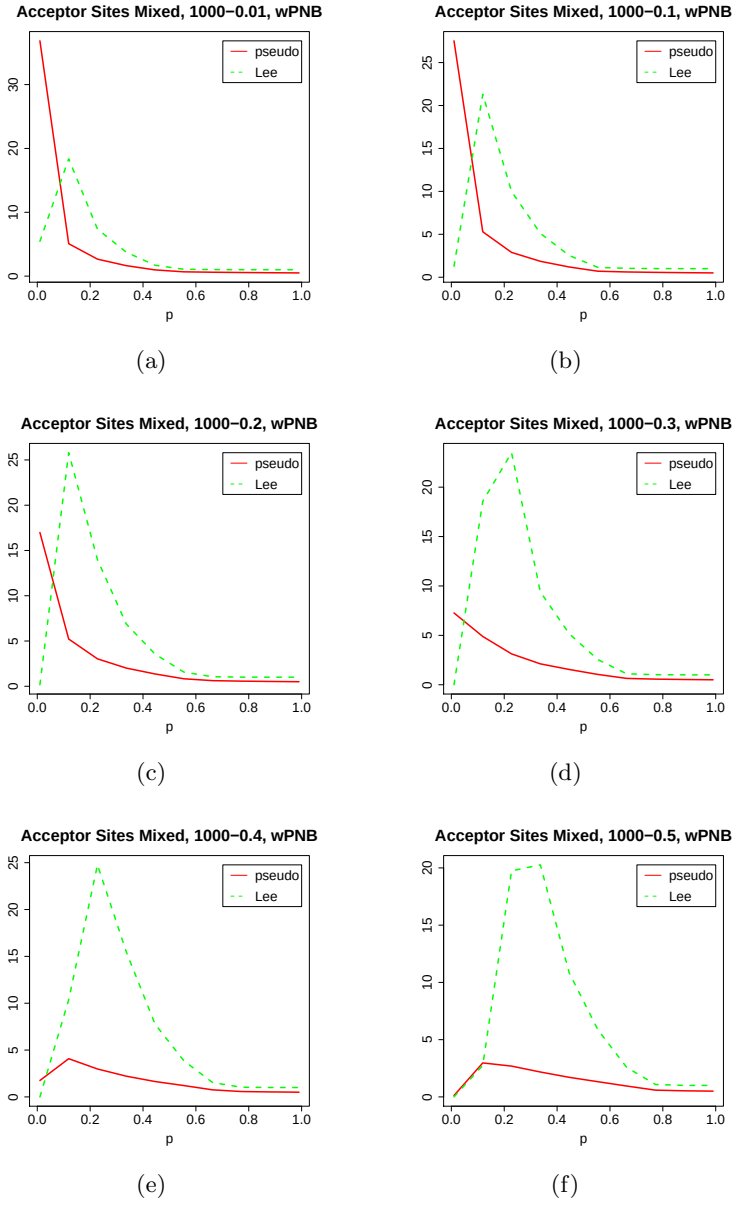
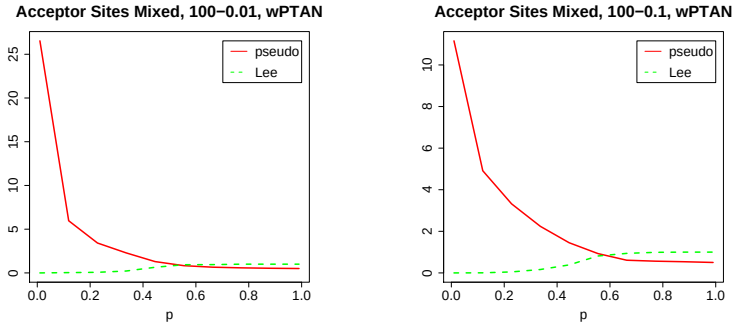
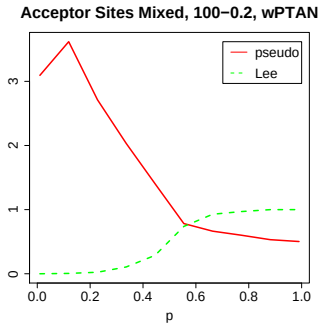


Fig. D.15. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Acceptor Sites Mixed datasets where $|\mathcal{D}_p| = 1,000$.

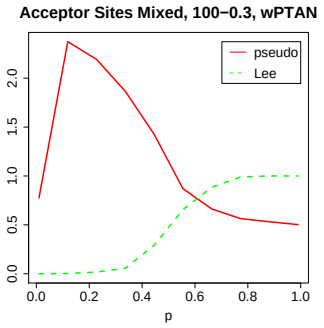


(a)

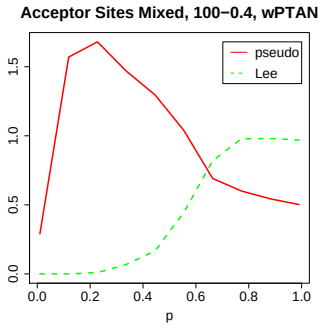
(b)



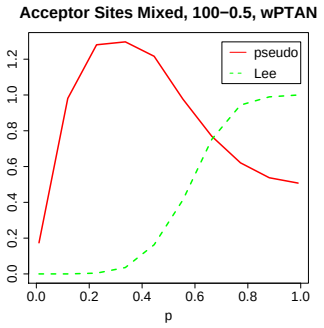
(c)



(d)



(e)



(f)

Fig. D.16. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Acceptor Sites Mixed datasets where $|\mathcal{D}_p| = 100$.

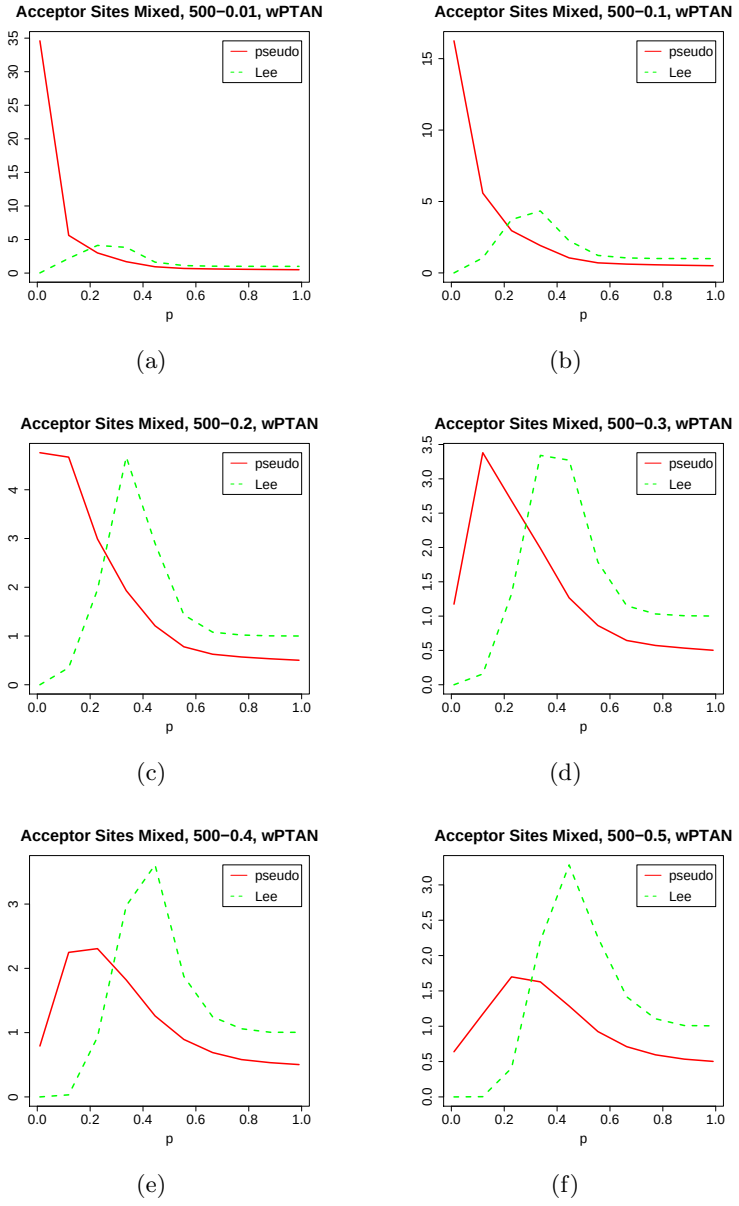


Fig. D.17. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Acceptor Sites Mixed datasets where $|\mathcal{D}_p| = 500$.

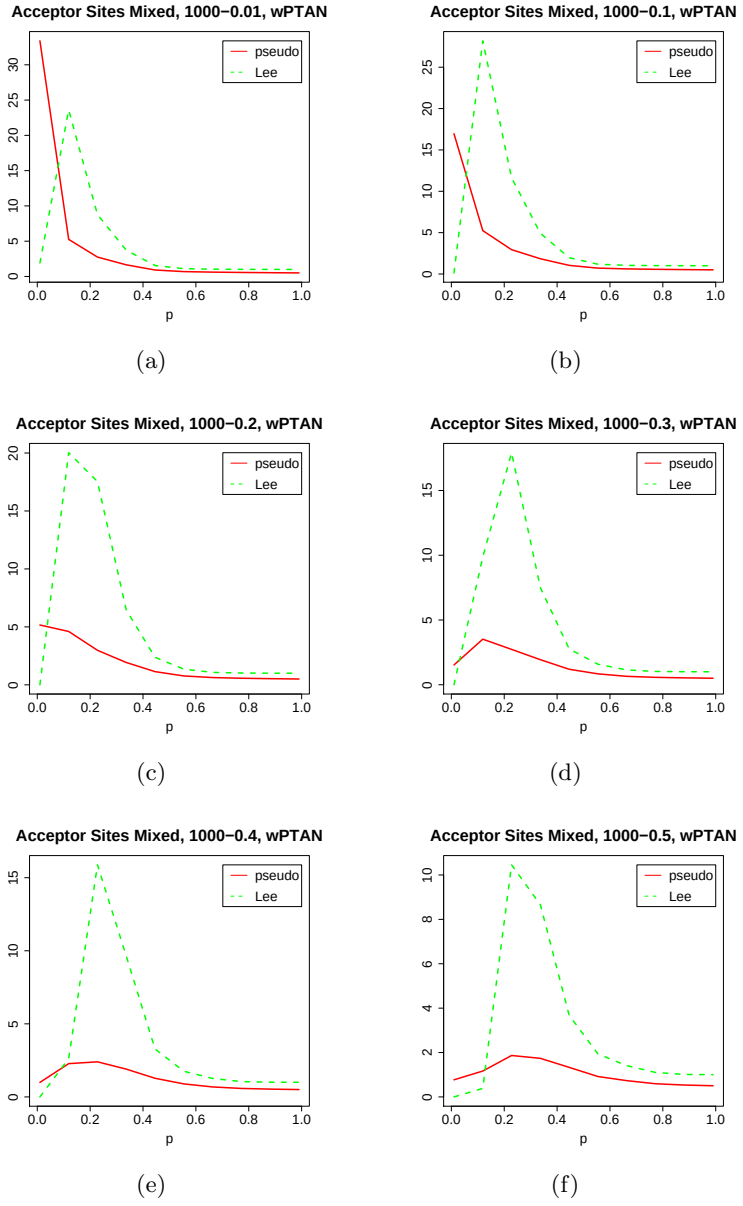


Fig. D.18. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Acceptor Sites Mixed datasets where $|\mathcal{D}_p| = 1,000$.

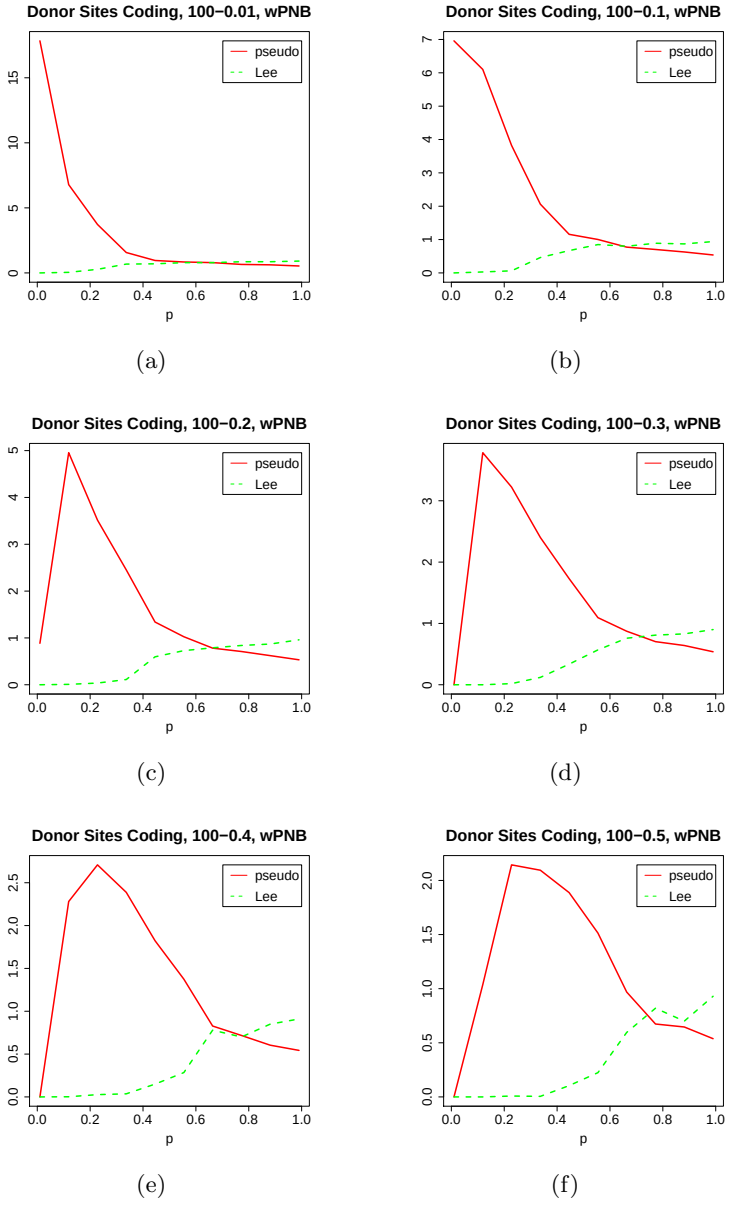
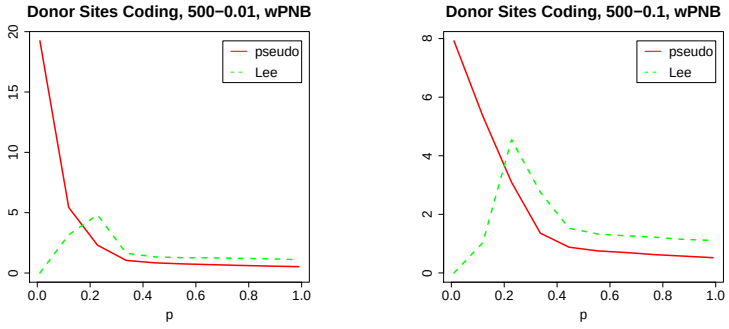
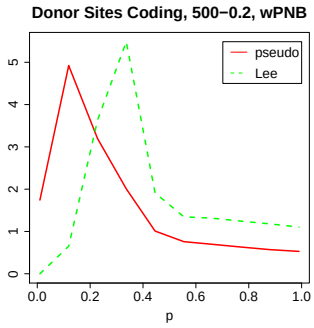


Fig. D.19. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Donor Sites Coding datasets where $|\mathcal{D}_p| = 100$.

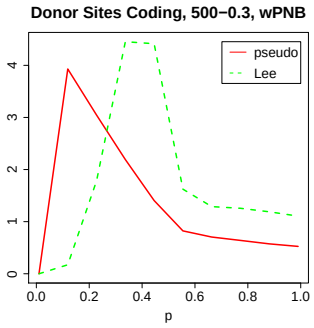


(a)

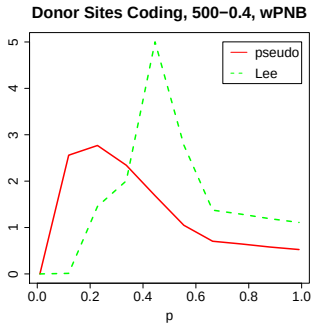
(b)



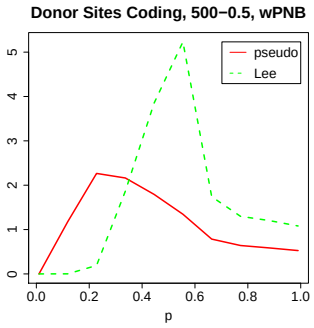
(c)



(d)



(e)



(f)

Fig. D.20. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Donor Sites Coding datasets where $|\mathcal{D}_p| = 500$.

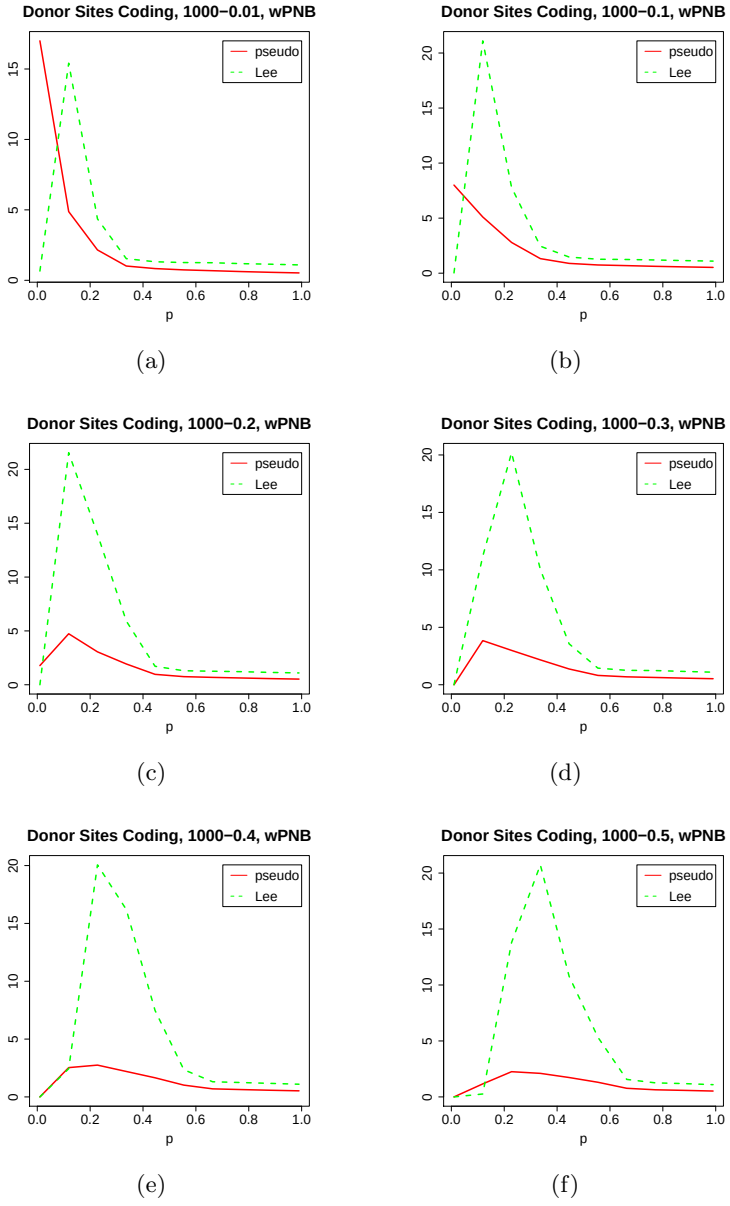
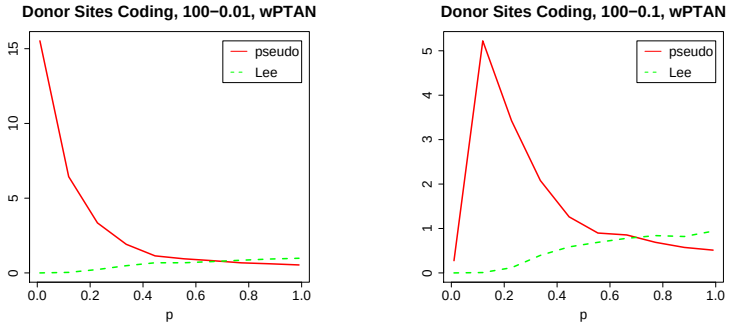
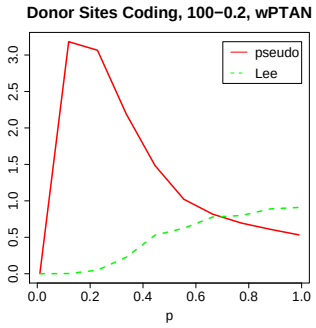


Fig. D.21. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Donor Sites Coding datasets where $|\mathcal{D}_p| = 1,000$.

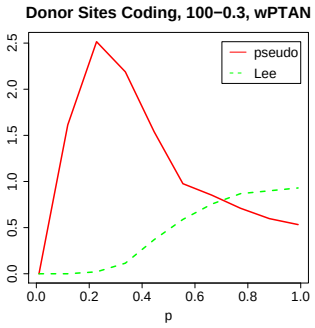


(a)

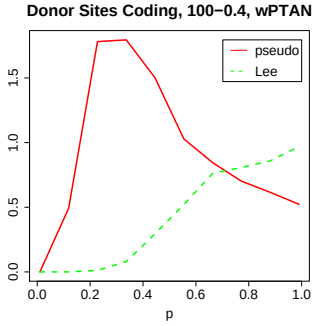
(b)



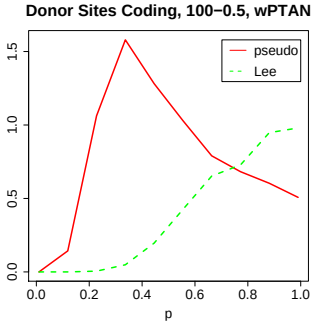
(c)



(d)



(e)



(f)

Fig. D.22. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Donor Sites Coding datasets where $|\mathcal{D}_p| = 100$.

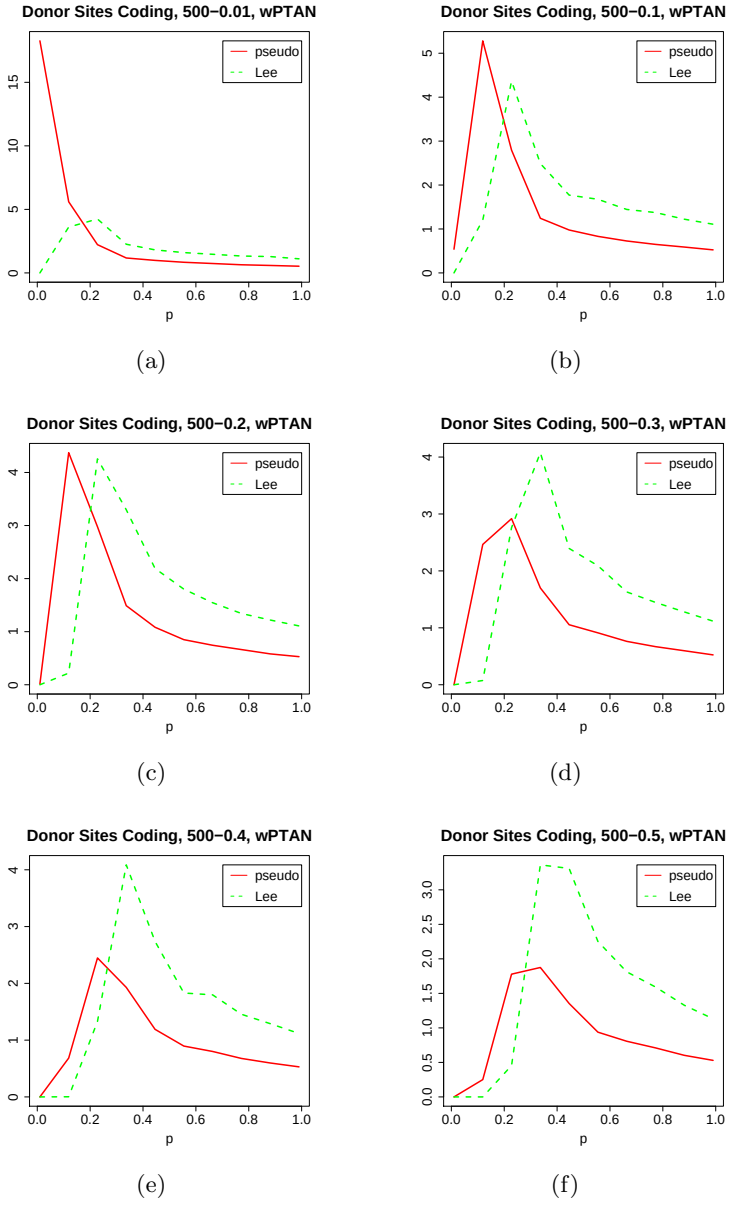
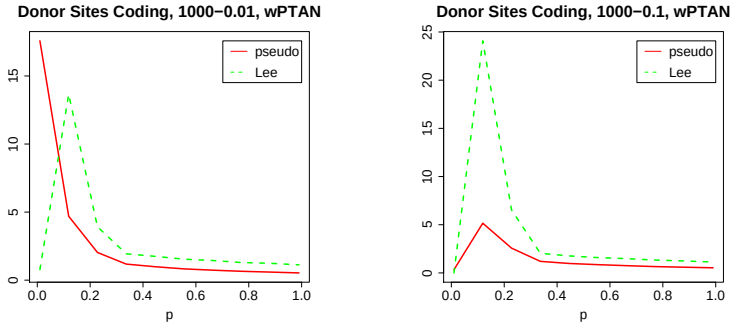
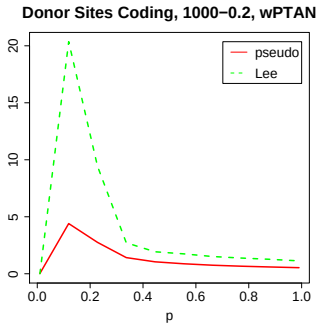


Fig. D.23. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Donor Sites Coding datasets where $|\mathcal{D}_p| = 500$.

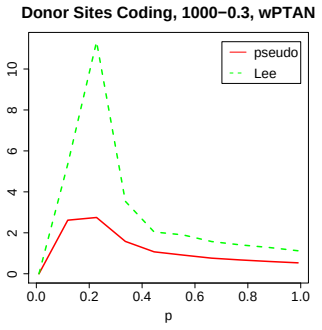


(a)

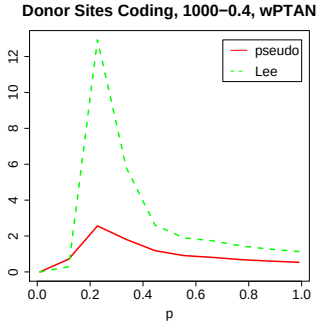
(b)



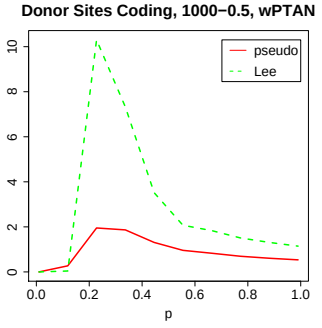
(c)



(d)



(e)



(f)

Fig. D.24. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Donor Sites Coding datasets where $|\mathcal{D}_p| = 1,000$.

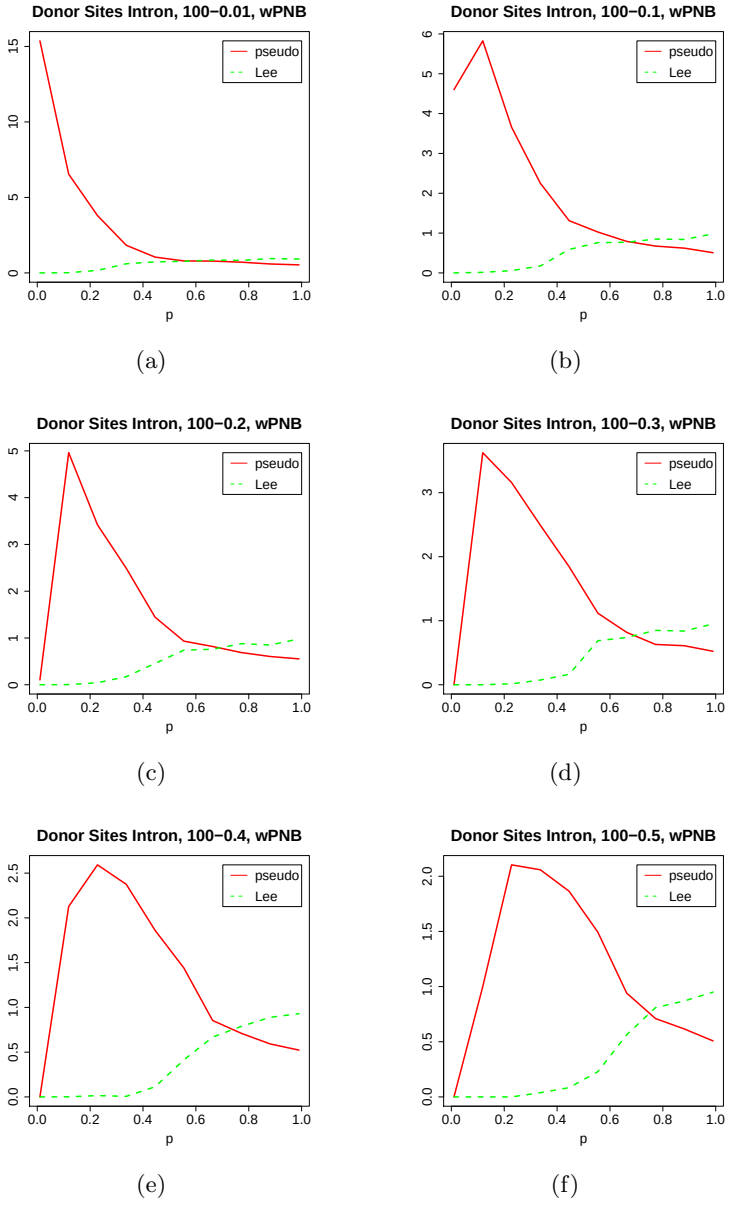
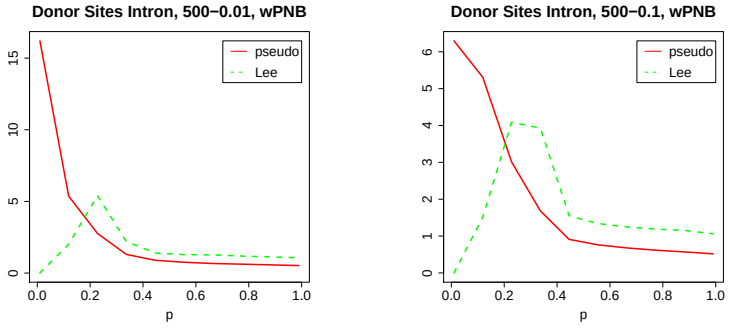
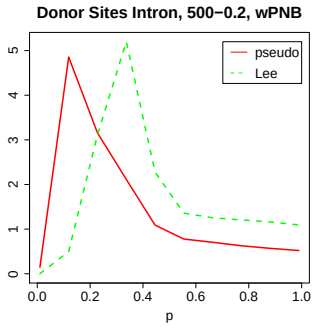


Fig. D.25. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Donor Sites Intron datasets where $|\mathcal{D}_p| = 100$.

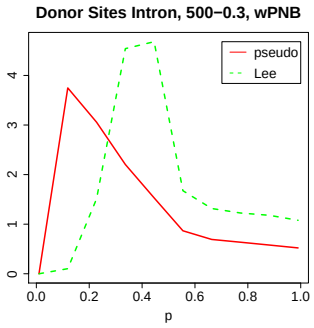


(a)

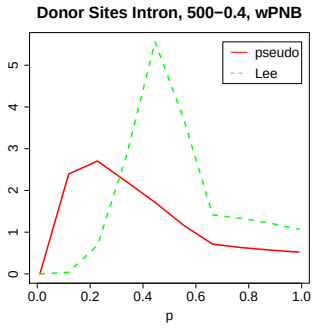
(b)



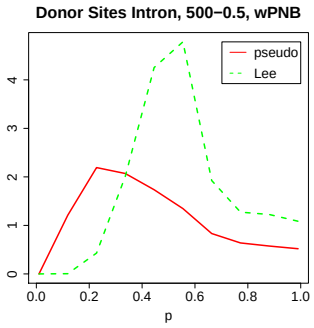
(c)



(d)



(e)



(f)

Fig. D.26. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Donor Sites Intron datasets where $|\mathcal{D}_p| = 500$.

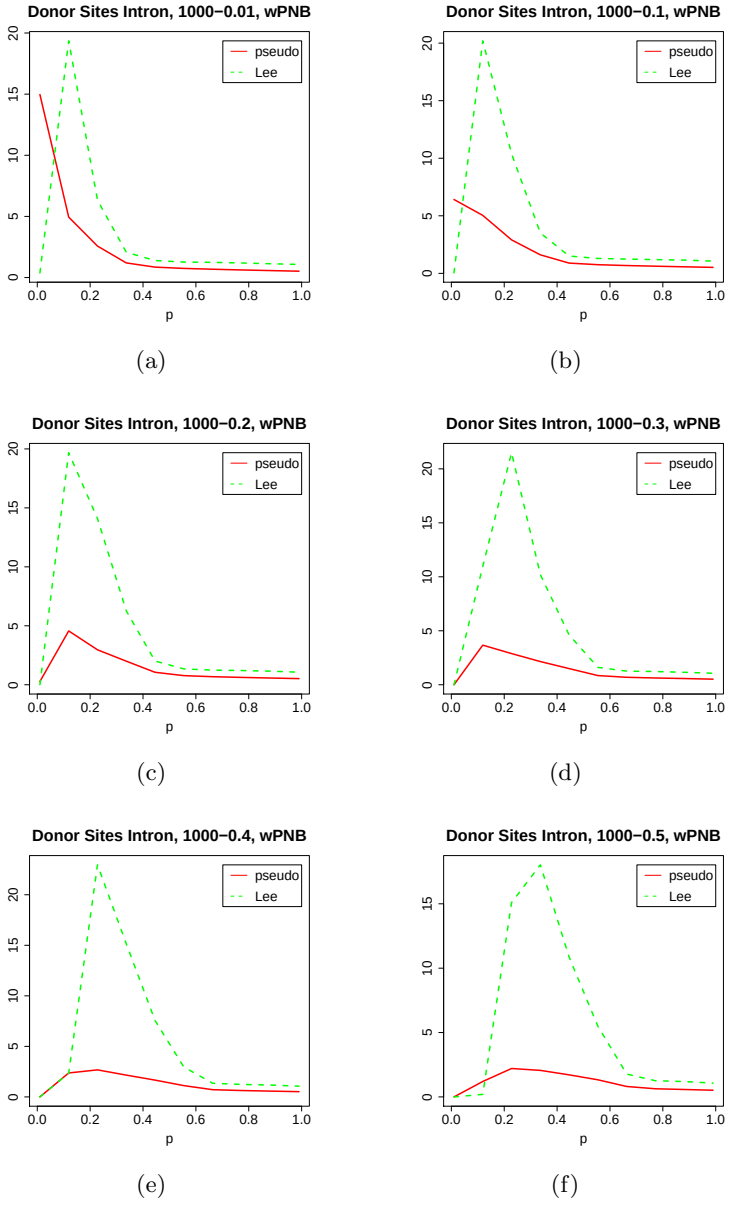
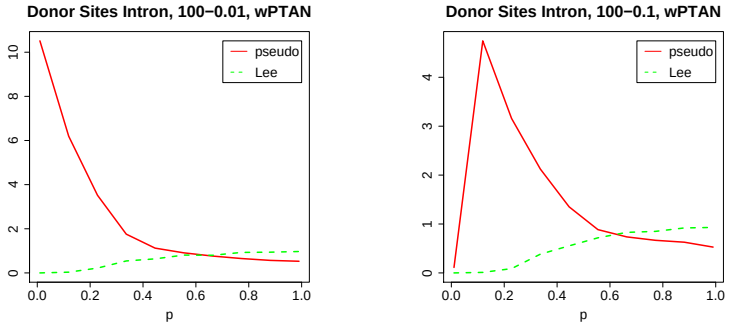
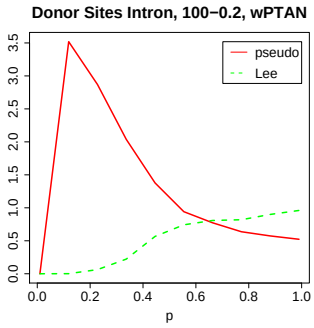


Fig. D.27. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Donor Sites Intron datasets where $|\mathcal{D}_p| = 1,000$.

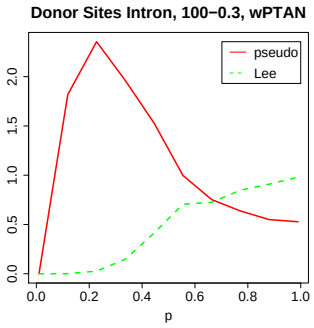


(a)

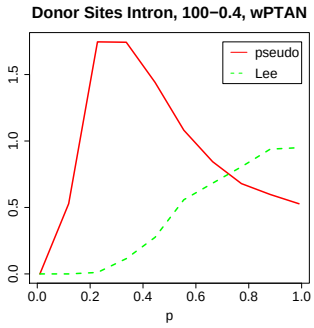
(b)



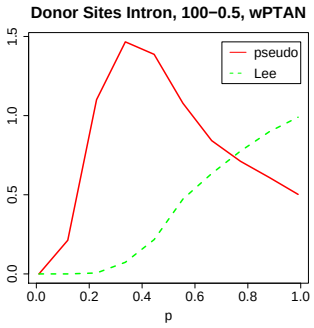
(c)



(d)

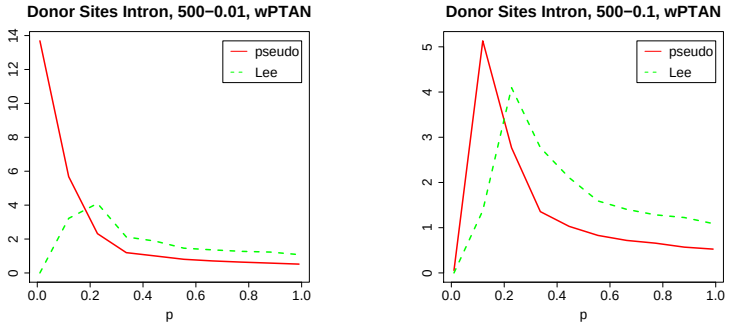


(e)



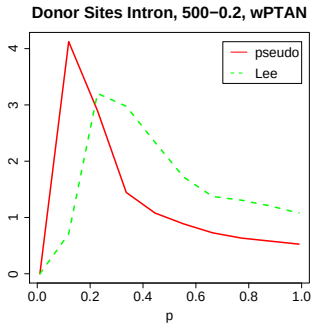
(f)

Fig. D.28. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Donor Sites Intron datasets where $|\mathcal{D}_p| = 100$.

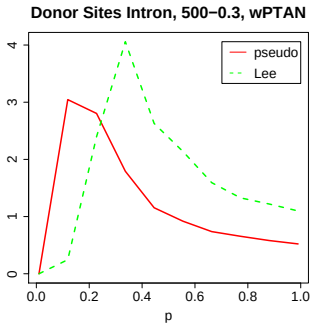


(a)

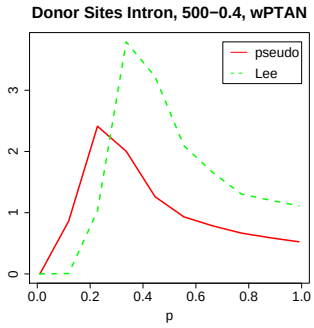
(b)



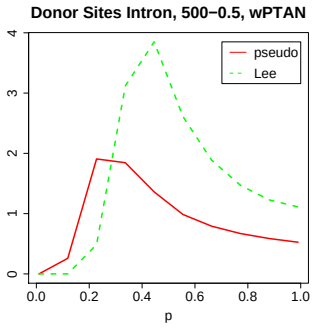
(c)



(d)

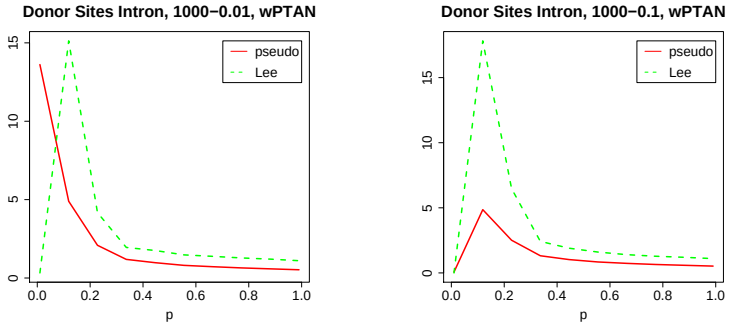


(e)



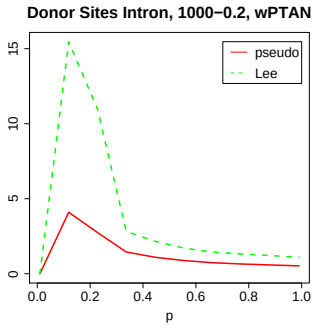
(f)

Fig. D.29. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Donor Sites Intron datasets where $|\mathcal{D}_p| = 500$.

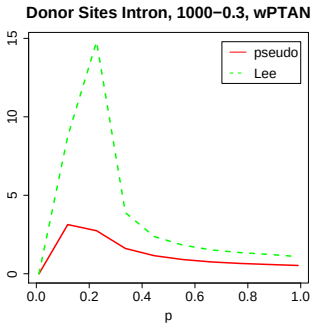


(a)

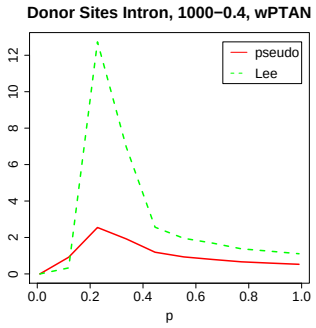
(b)



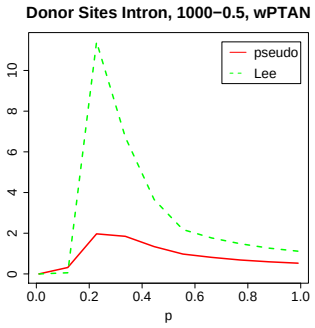
(c)



(d)



(e)



(f)

Fig. D.30. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Donor Sites Intron datasets where $|\mathcal{D}_p| = 1,000$.

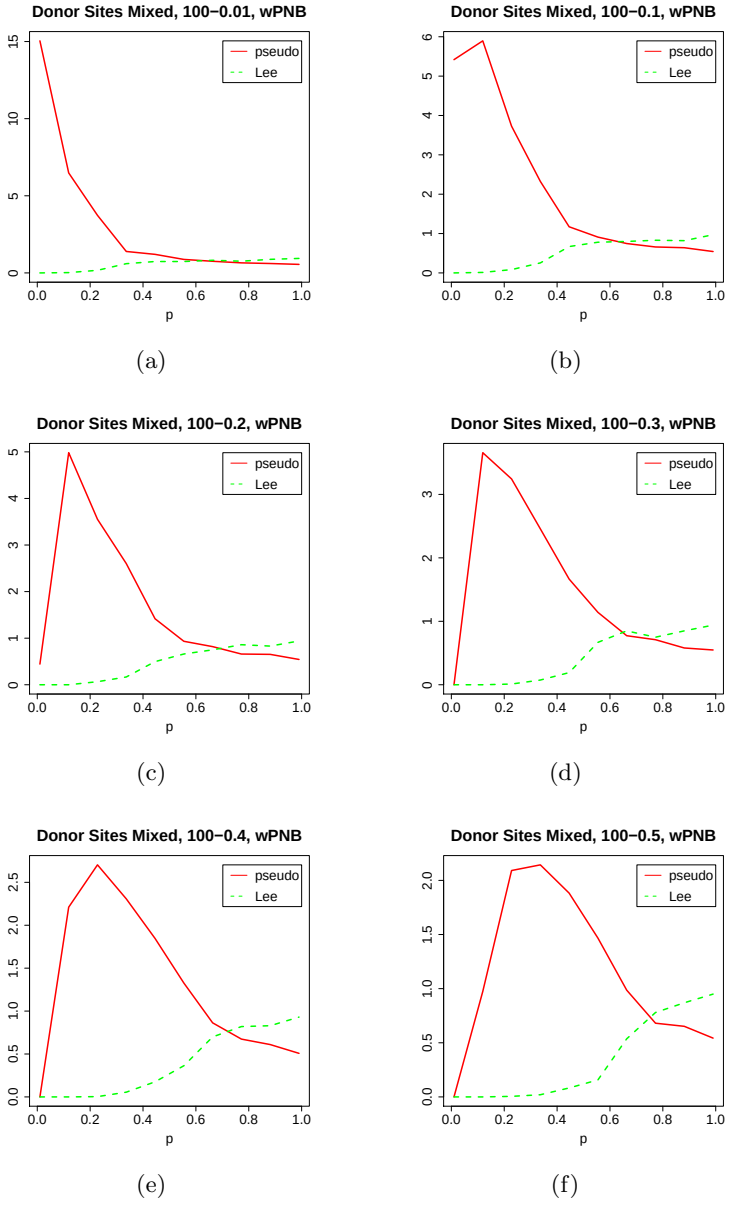
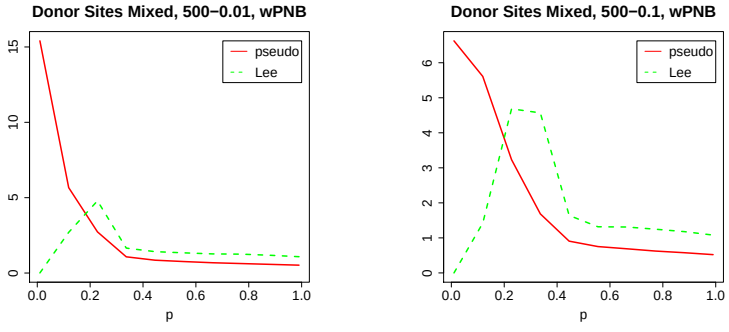
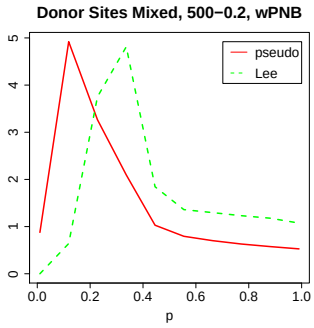


Fig. D.31. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Donor Sites Mixed datasets where $|\mathcal{D}_p| = 100$.

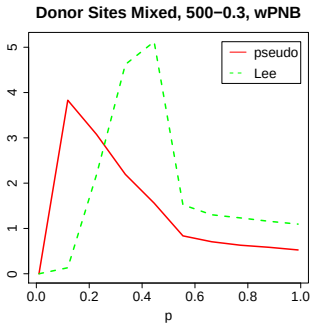


(a)

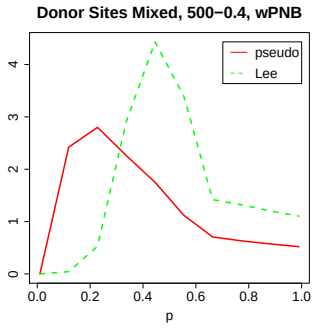
(b)



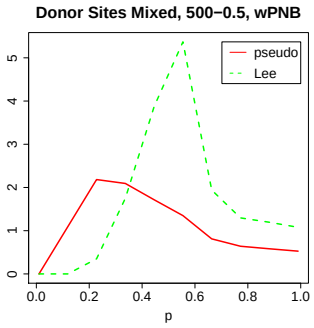
(c)



(d)



(e)



(f)

Fig. D.32. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Donor Sites Mixed datasets where $|\mathcal{D}_p| = 500$.

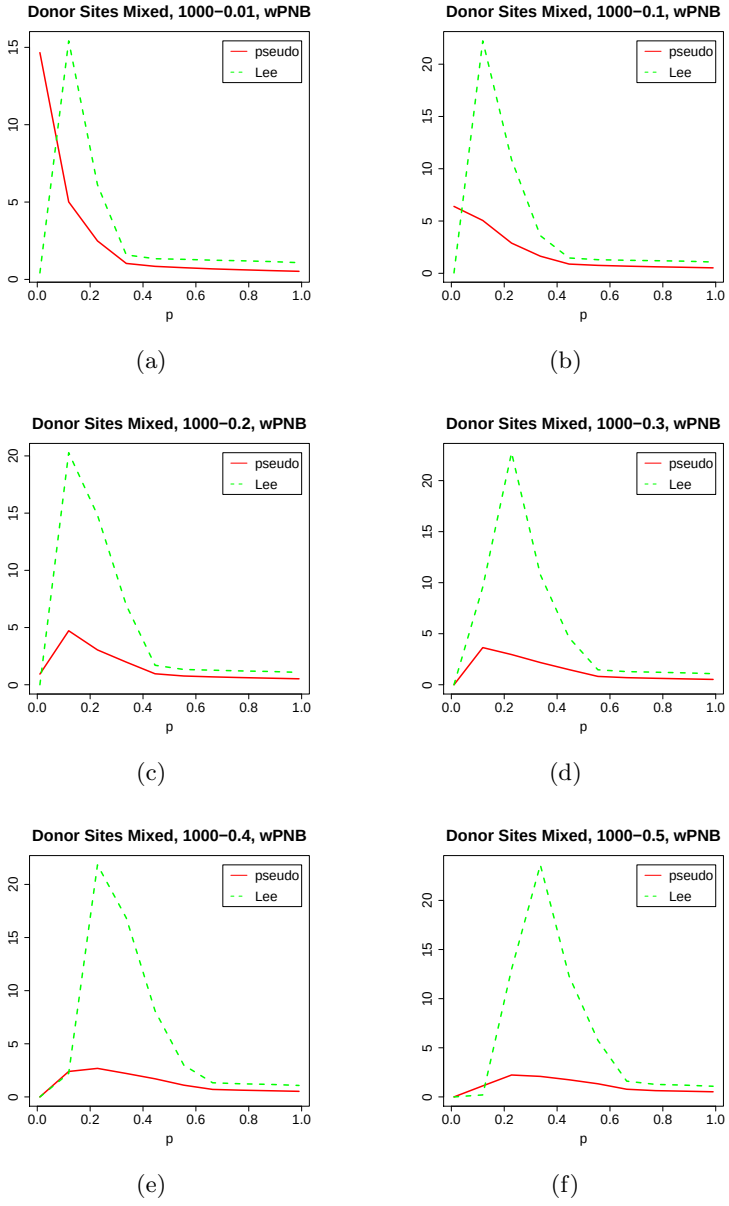
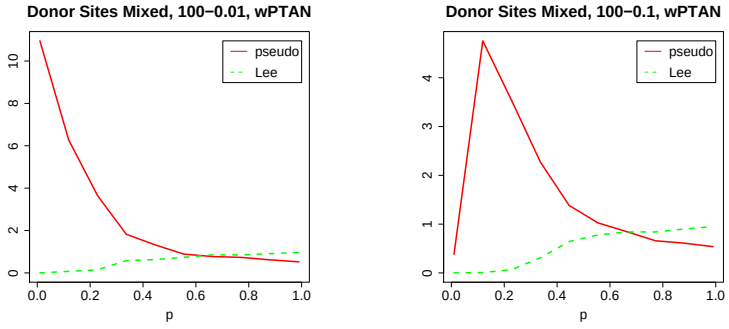
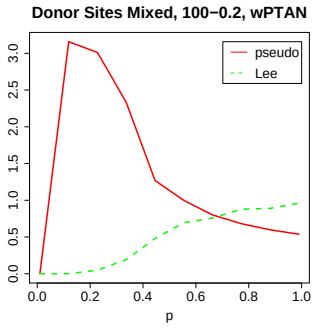


Fig. D.33. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Donor Sites Mixed datasets where $|\mathcal{D}_p| = 1,000$.

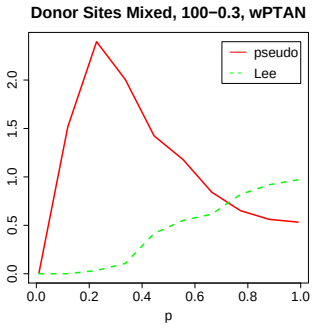


(a)

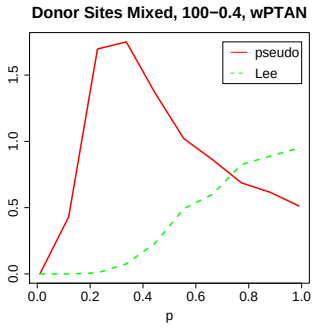
(b)



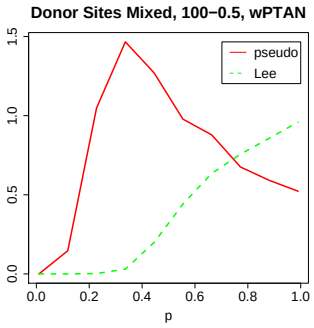
(c)



(d)



(e)



(f)

Fig. D.34. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Donor Sites Mixed datasets where $|\mathcal{D}_p| = 100$.

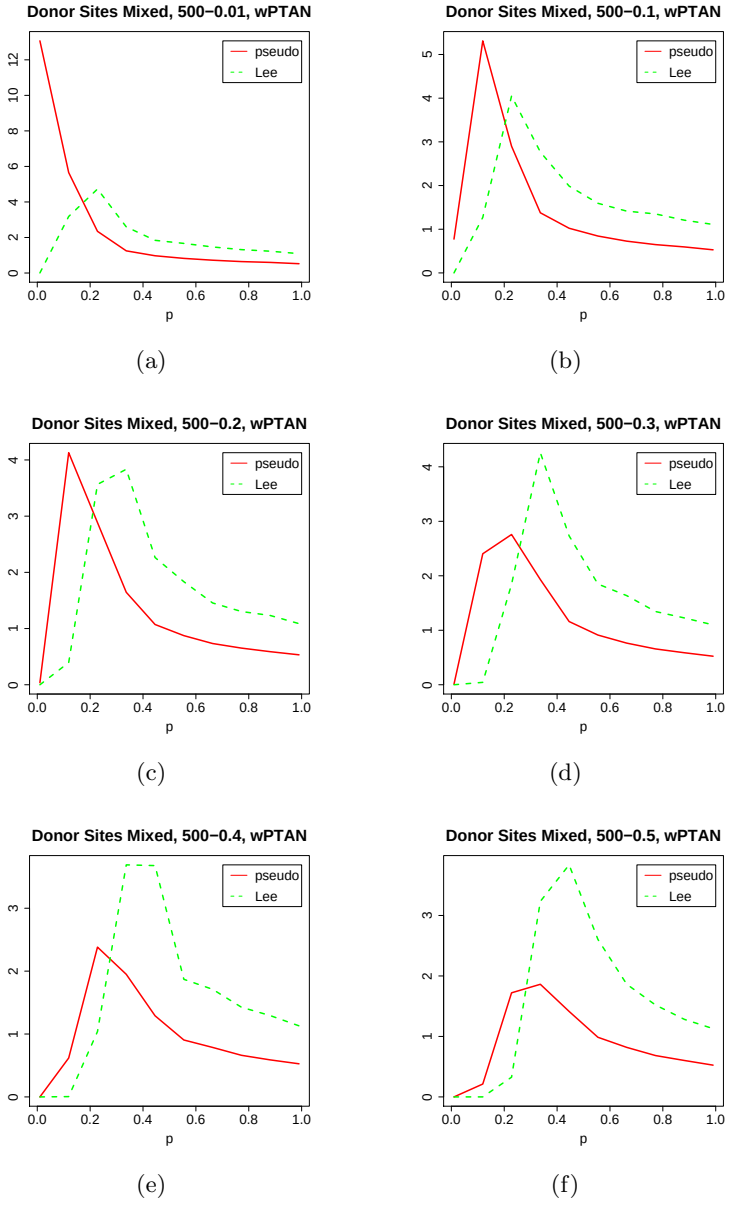
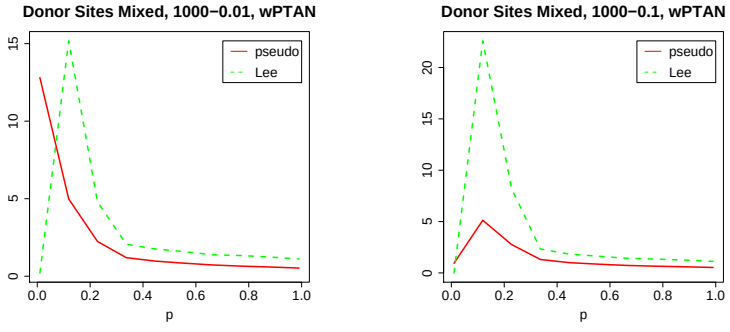
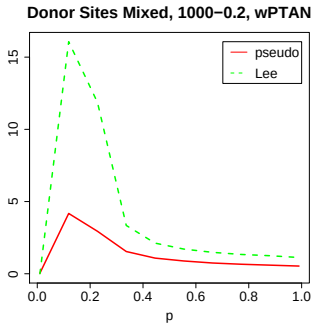


Fig. D.35. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Donor Sites Mixed datasets where $|\mathcal{D}_p| = 500$.

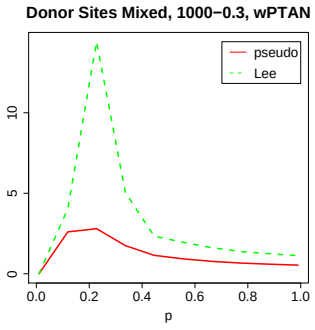


(a)

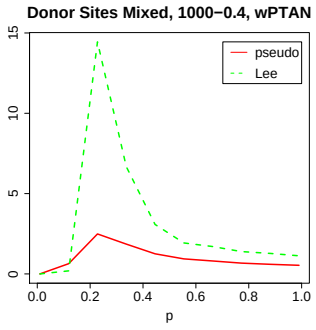
(b)



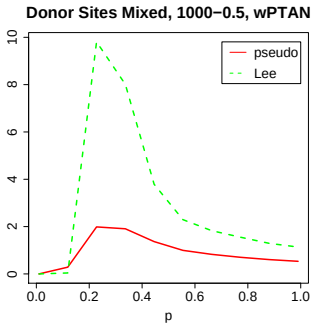
(c)



(d)



(e)



(f)

Fig. D.36. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Donor Sites Mixed datasets where $|\mathcal{D}_p| = 1,000$.

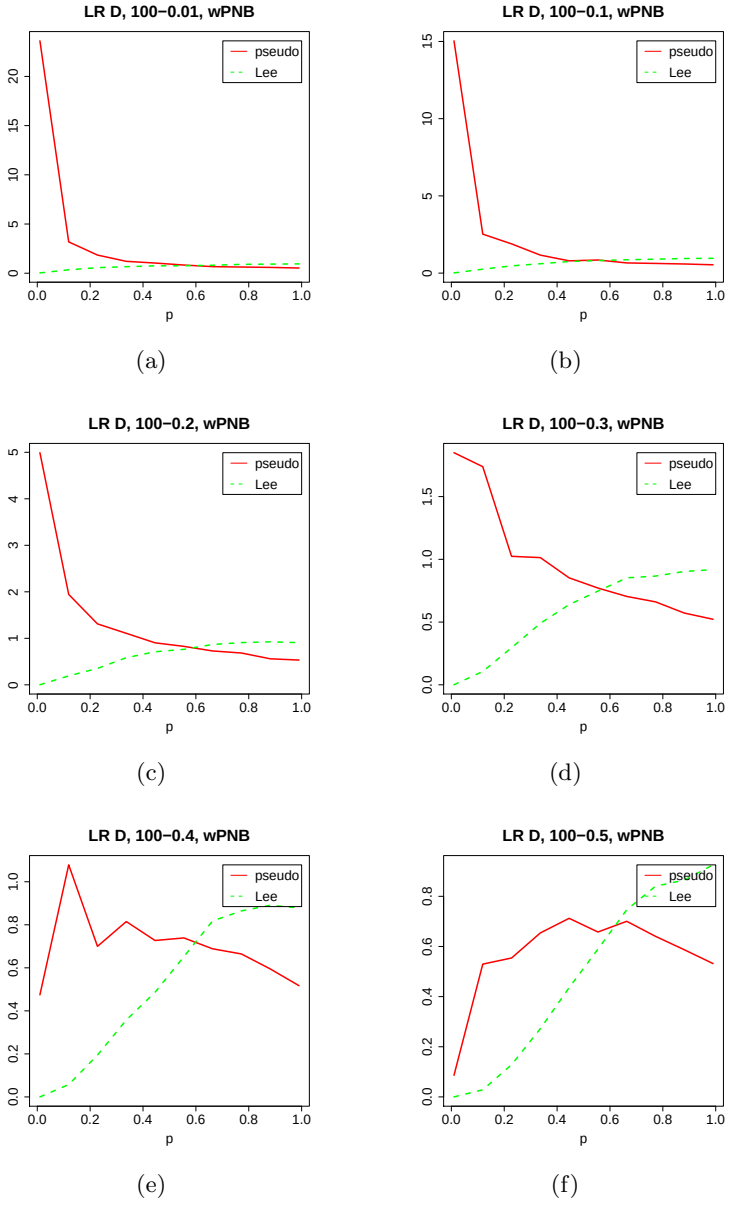


Fig. D.37. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Letter Recognition D datasets where $|\mathcal{D}_p| = 100$.

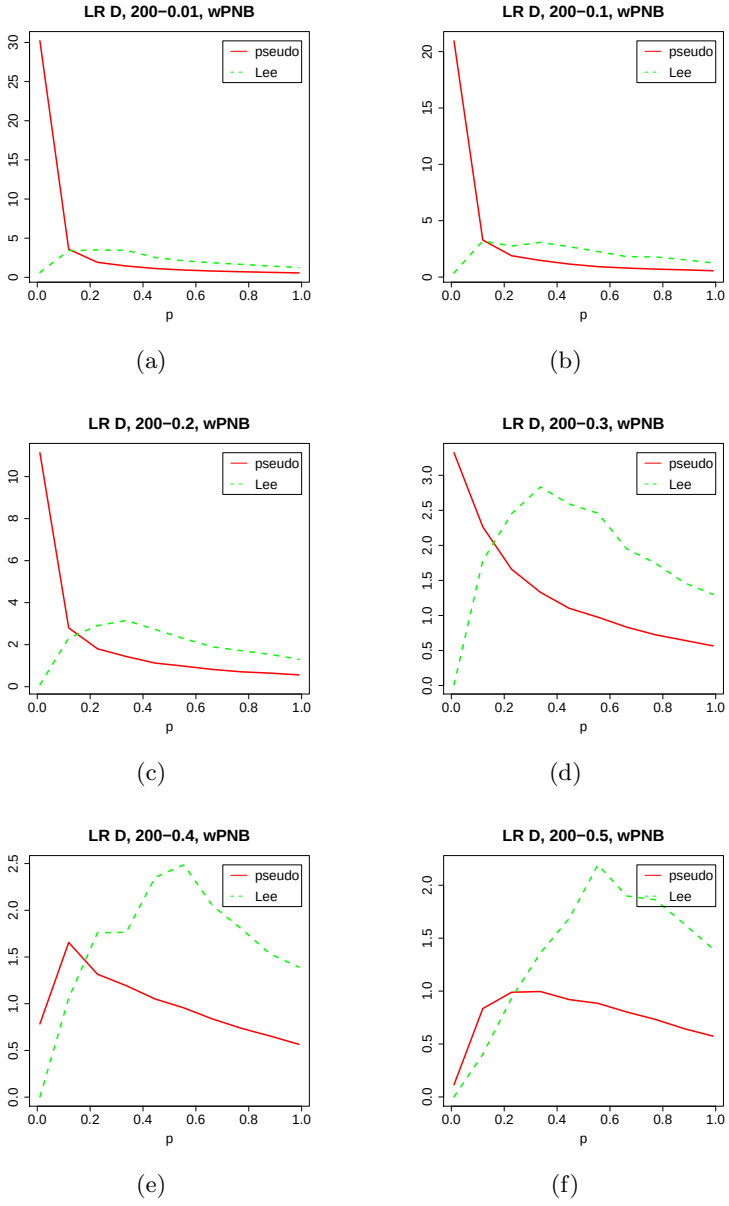


Fig. D.38. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Letter Recognition D datasets where $|\mathcal{D}_p| = 200$.

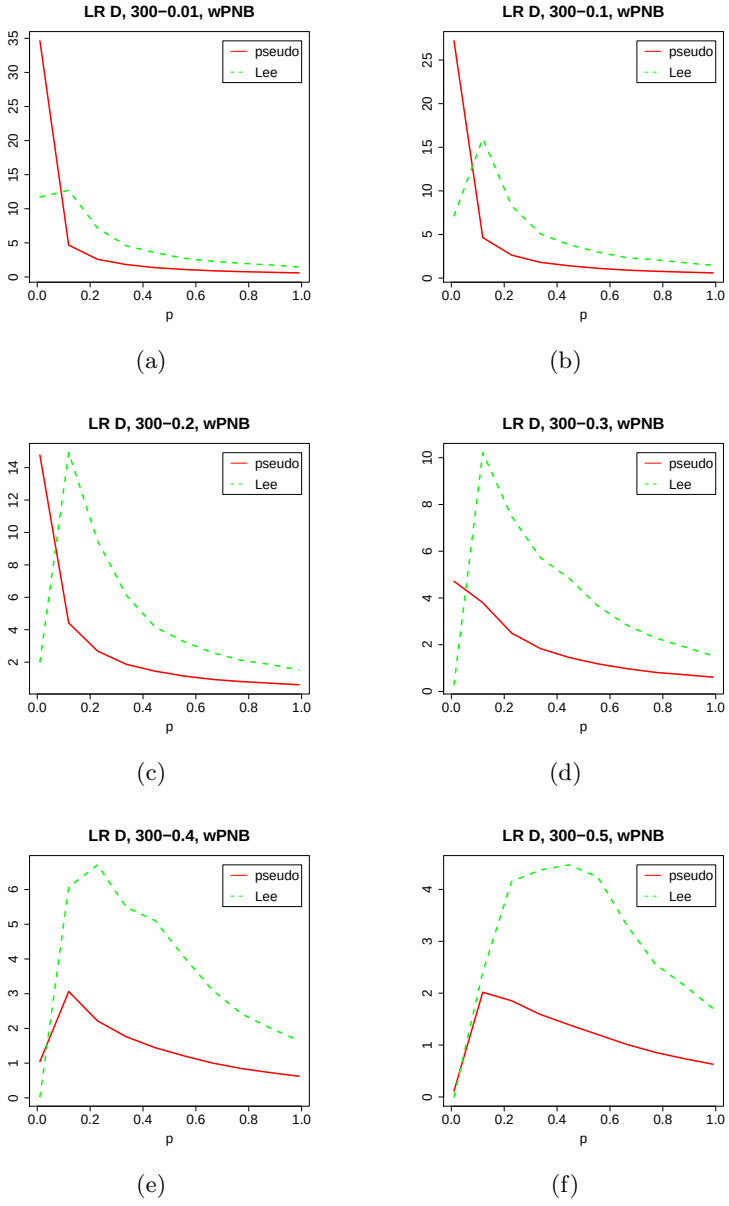


Fig. D.39. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Letter Recognition D datasets where $|\mathcal{D}_p| = 300$.

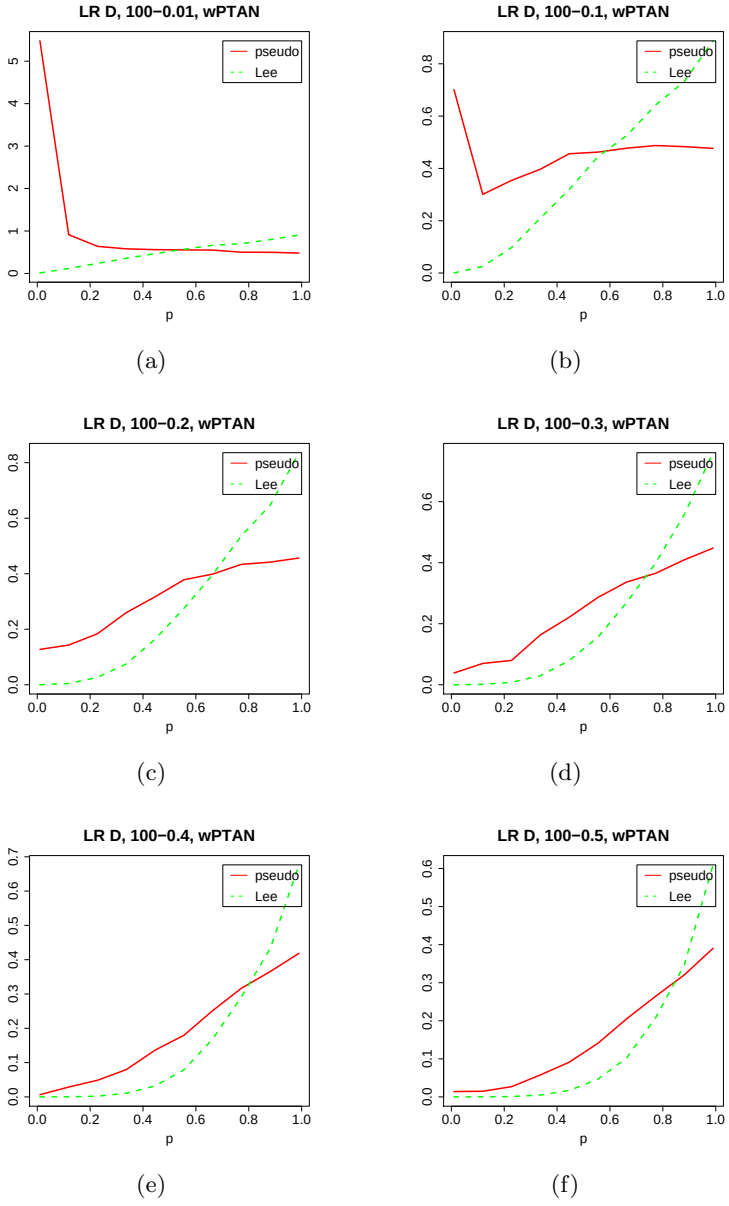


Fig. D.40. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Letter Recognition D datasets where $|\mathcal{D}_p| = 100$.

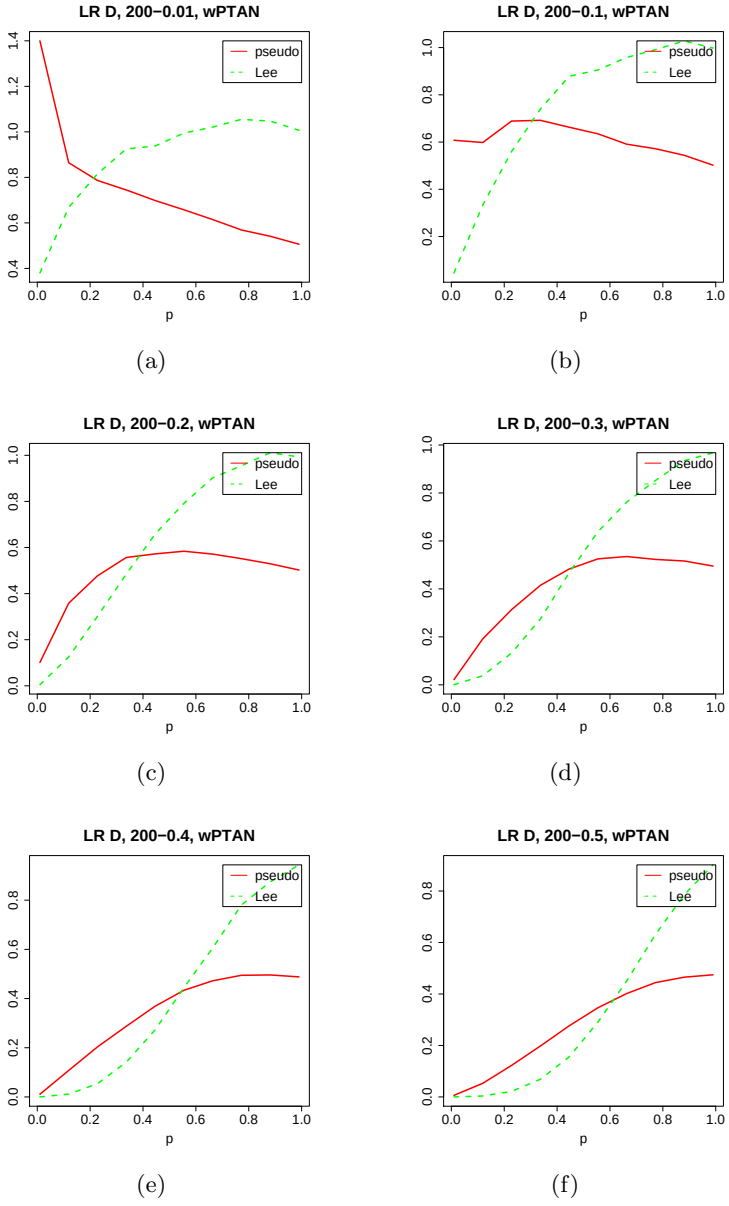


Fig. D.41. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Letter Recognition D datasets where $|\mathcal{D}_p| = 200$.

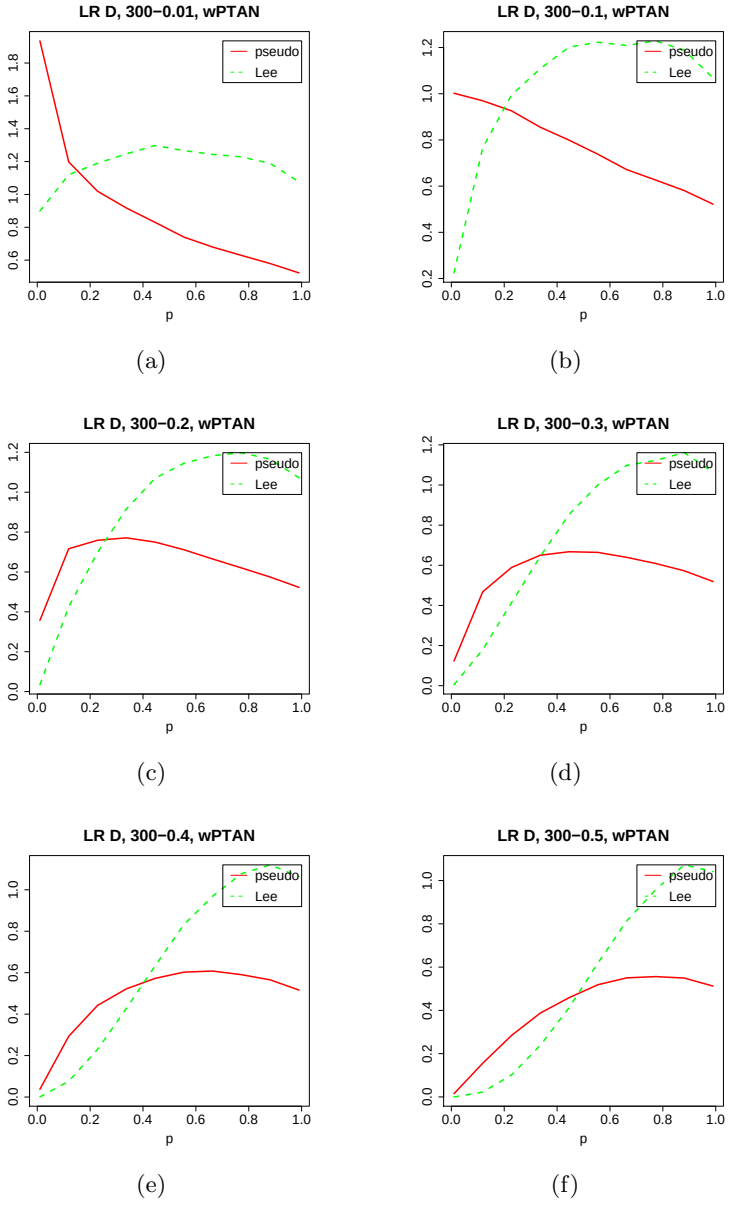


Fig. D.42. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Letter Recognition D datasets where $|\mathcal{D}_p| = 300$.

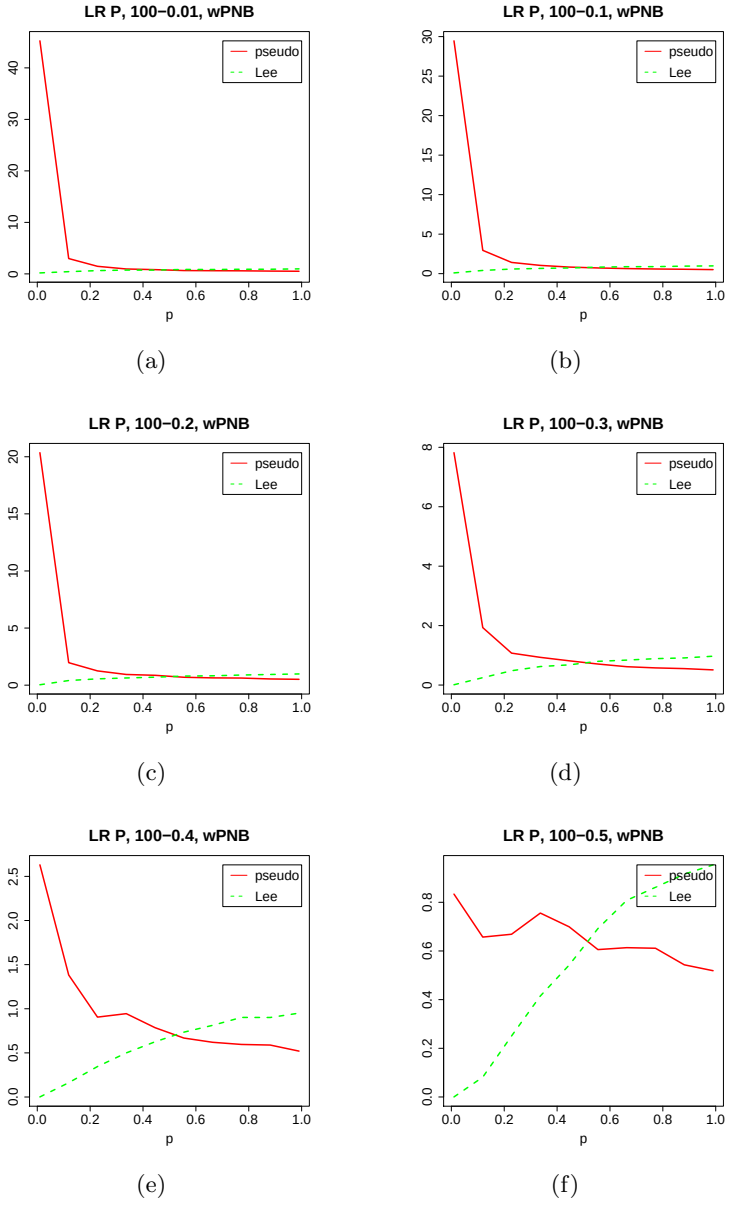


Fig. D.43. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Letter Recognition P datasets where $|\mathcal{D}_p| = 100$.

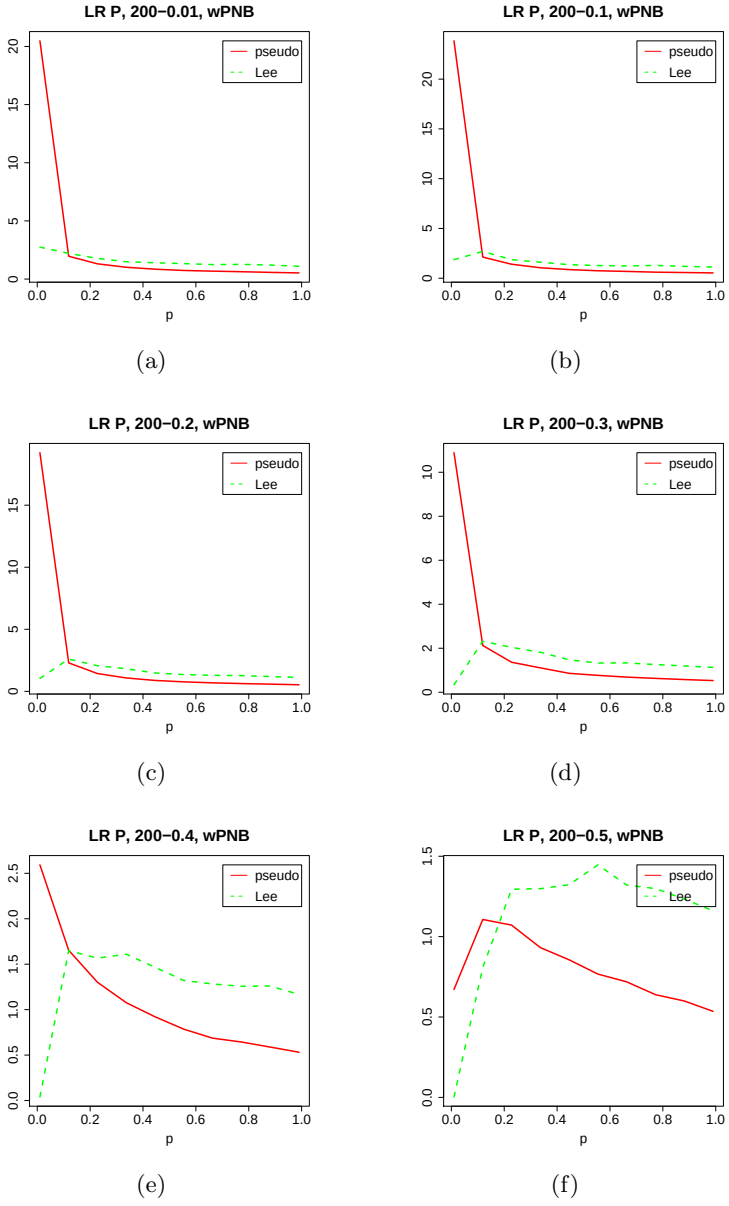


Fig. D.44. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Letter Recognition P datasets where $|\mathcal{D}_p| = 200$.

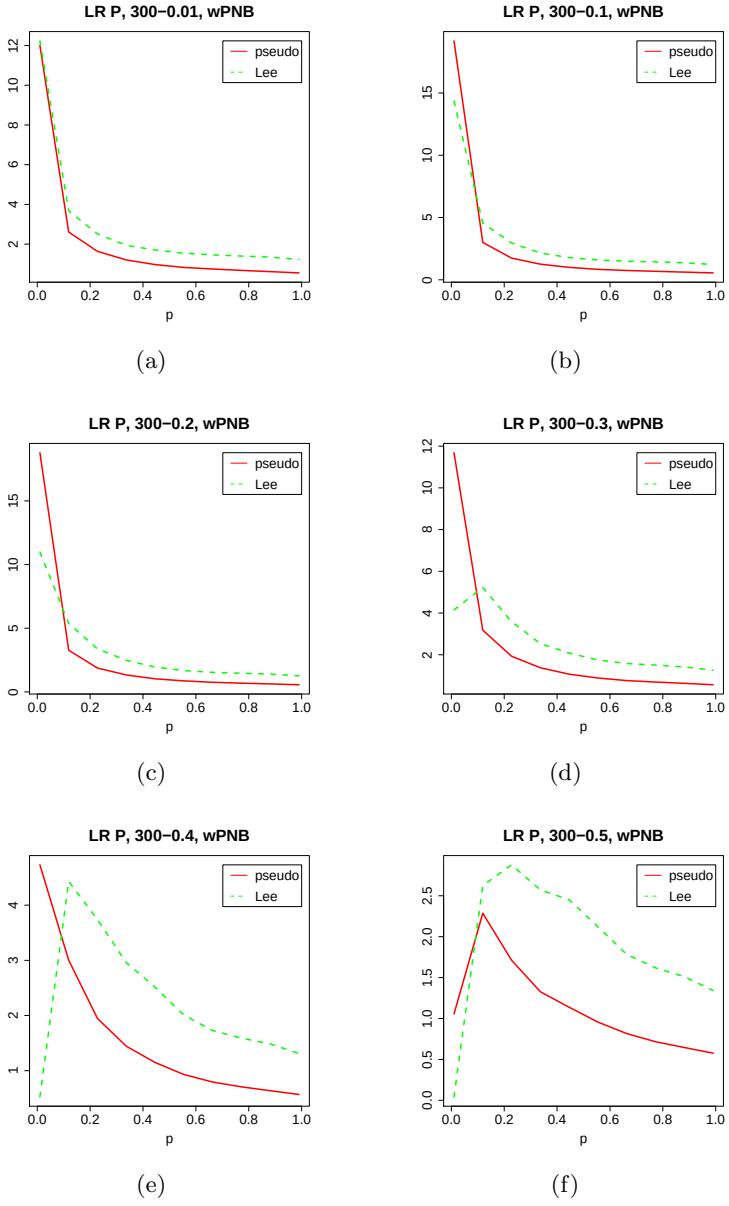
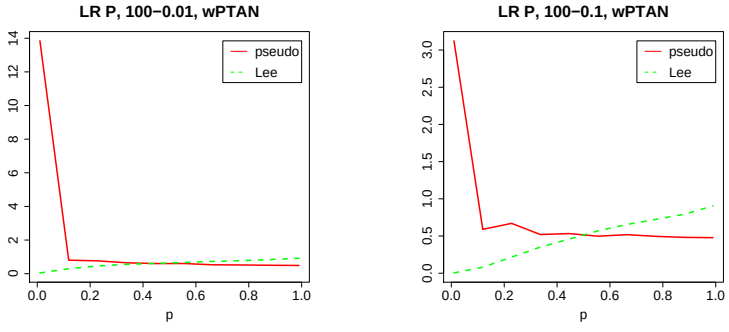
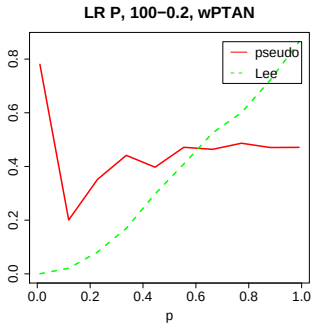


Fig. D.45. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Letter Recognition P datasets where $|\mathcal{D}_p| = 300$.

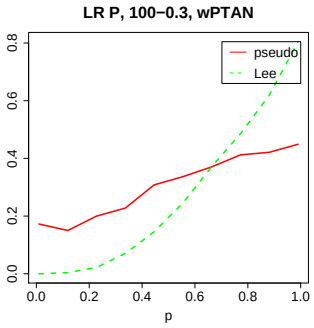


(a)

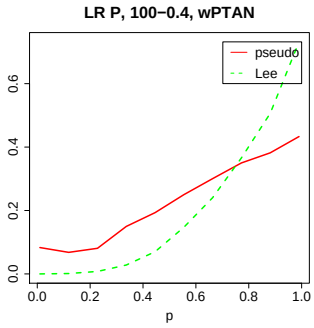
(b)



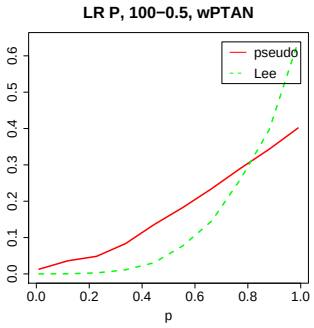
(c)



(d)



(e)



(f)

Fig. D.46. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Letter Recognition P datasets where $|\mathcal{D}_p| = 100$.

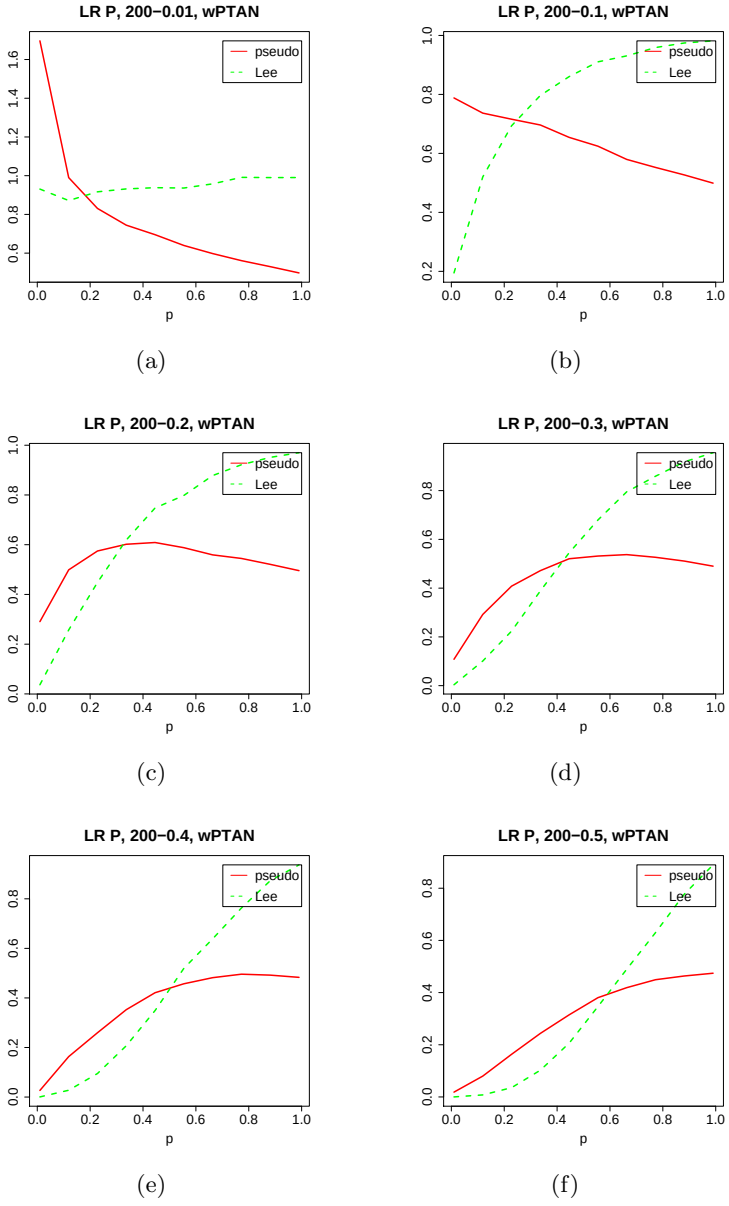


Fig. D.47. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Letter Recognition P datasets where $|\mathcal{D}_p| = 200$.

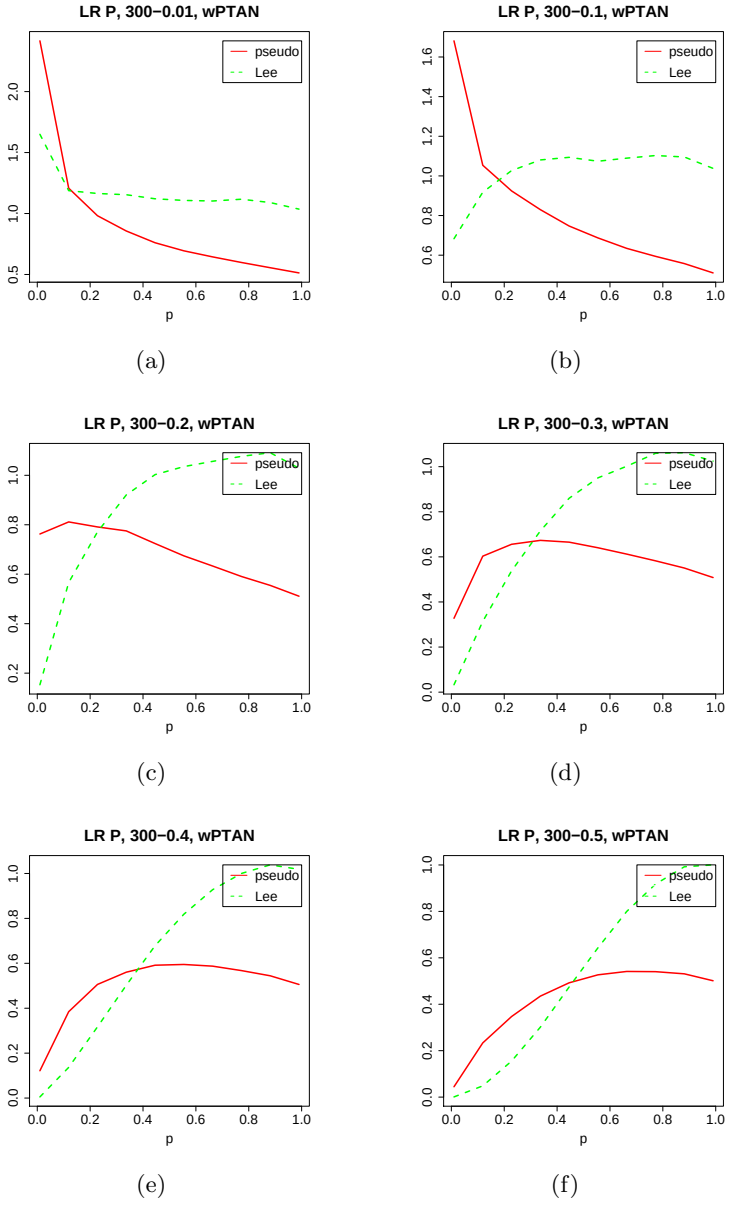


Fig. D.48. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Letter Recognition P datasets where $|\mathcal{D}_p| = 300$.

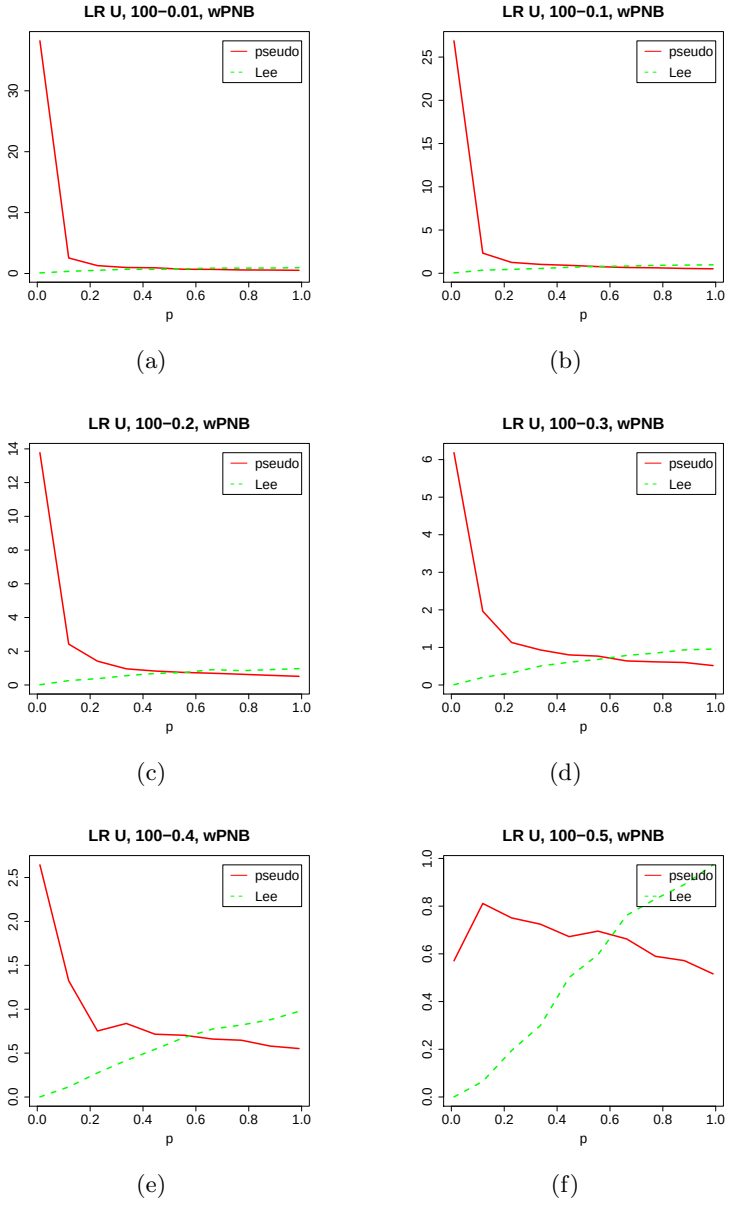


Fig. D.49. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Letter Recognition U datasets where $|\mathcal{D}_p| = 100$.

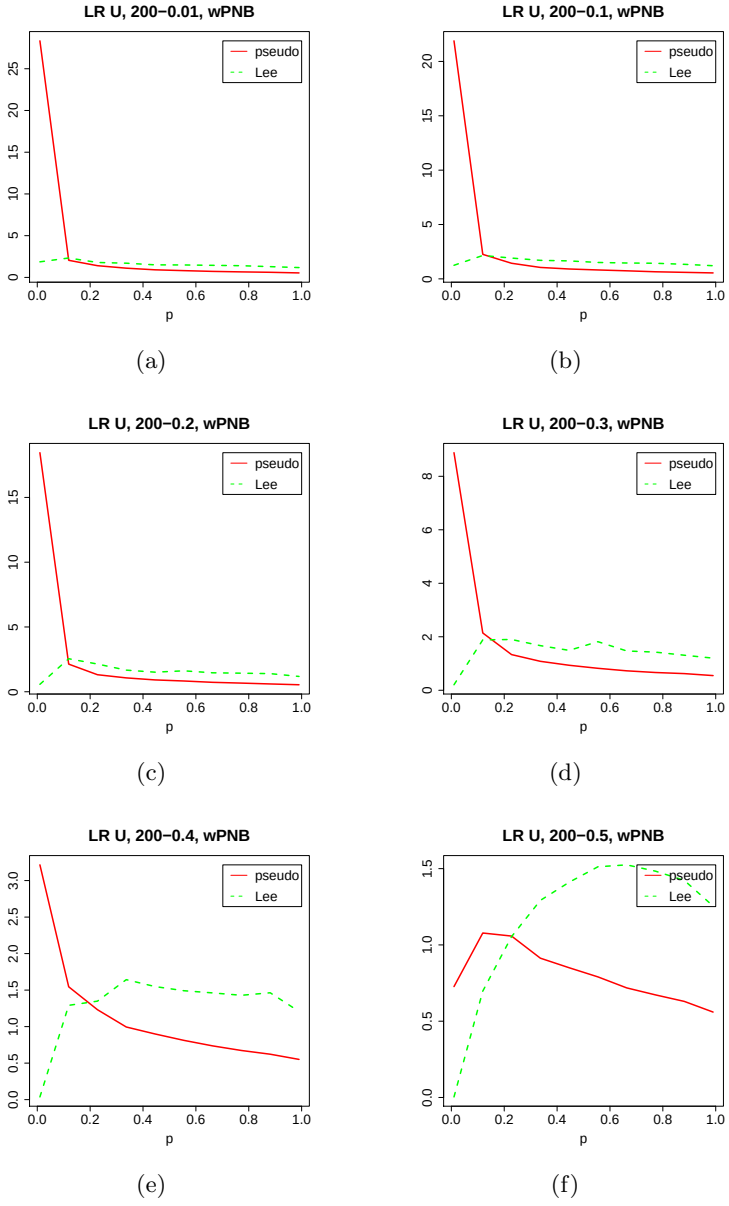


Fig. D.50. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Letter Recognition U datasets where $|\mathcal{D}_p| = 200$.

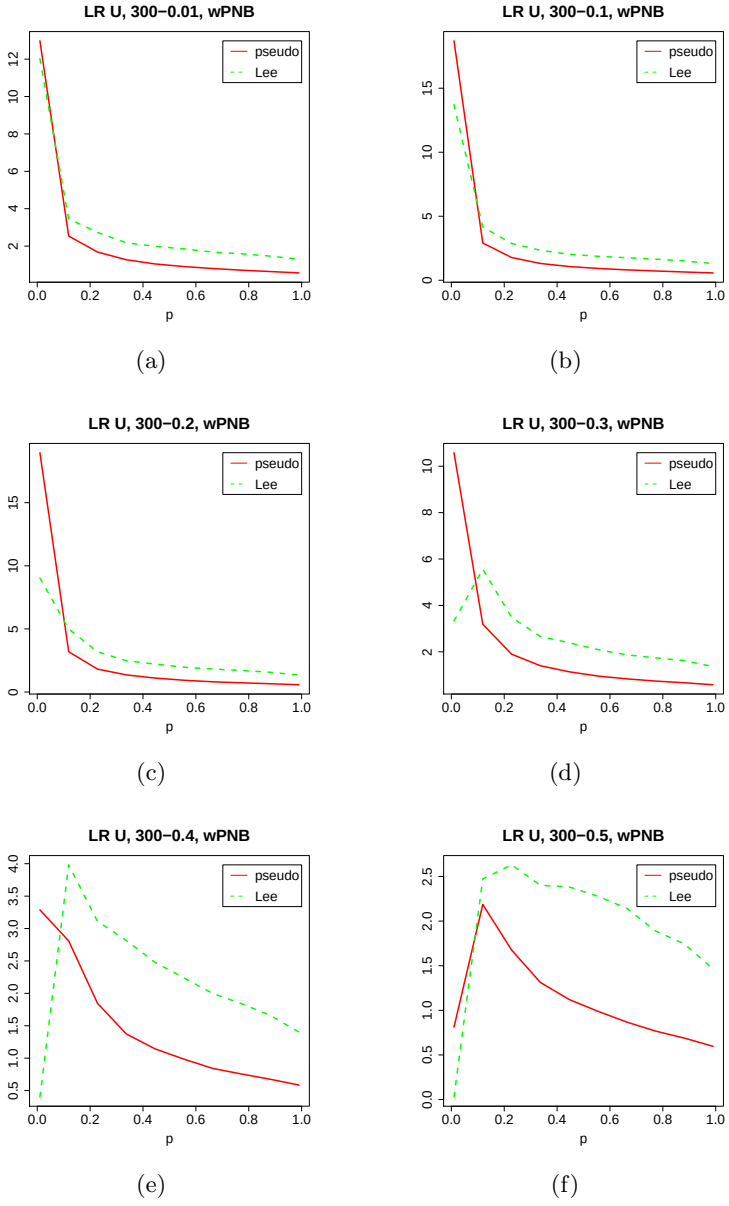


Fig. D.51. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Letter Recognition U datasets where $|\mathcal{D}_p| = 300$.

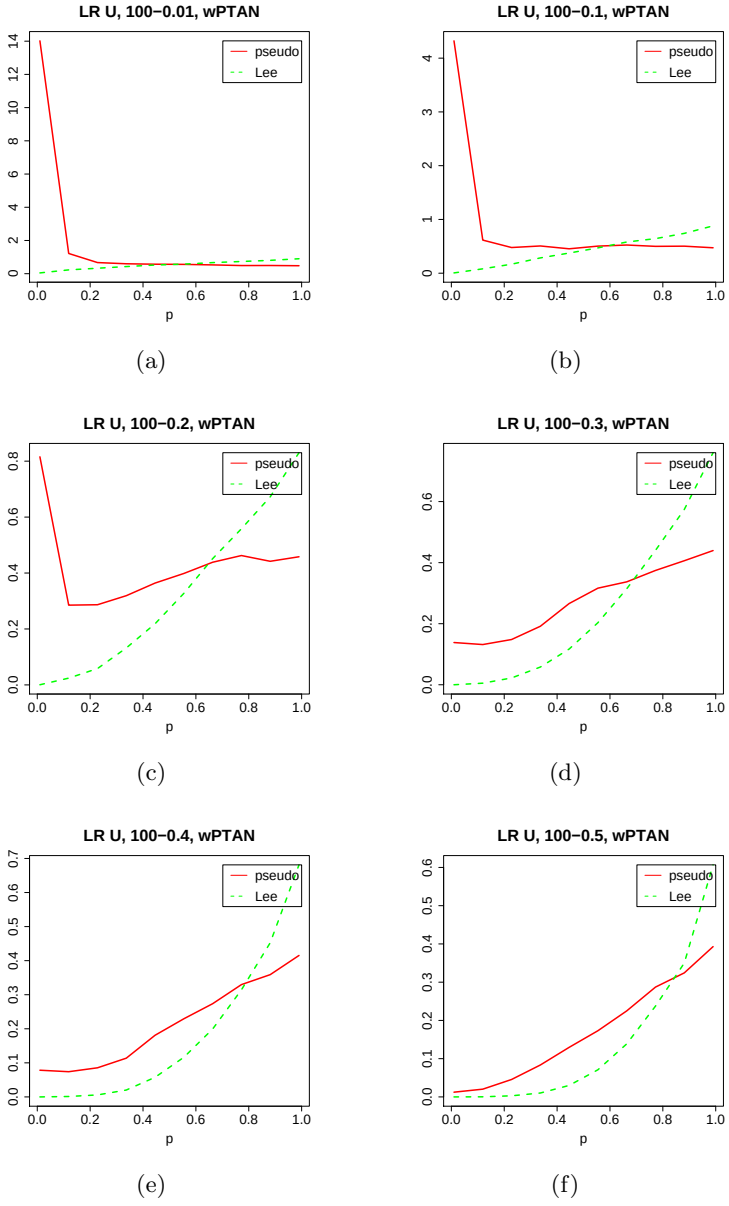


Fig. D.52. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Letter Recognition U datasets where $|\mathcal{D}_p| = 100$.

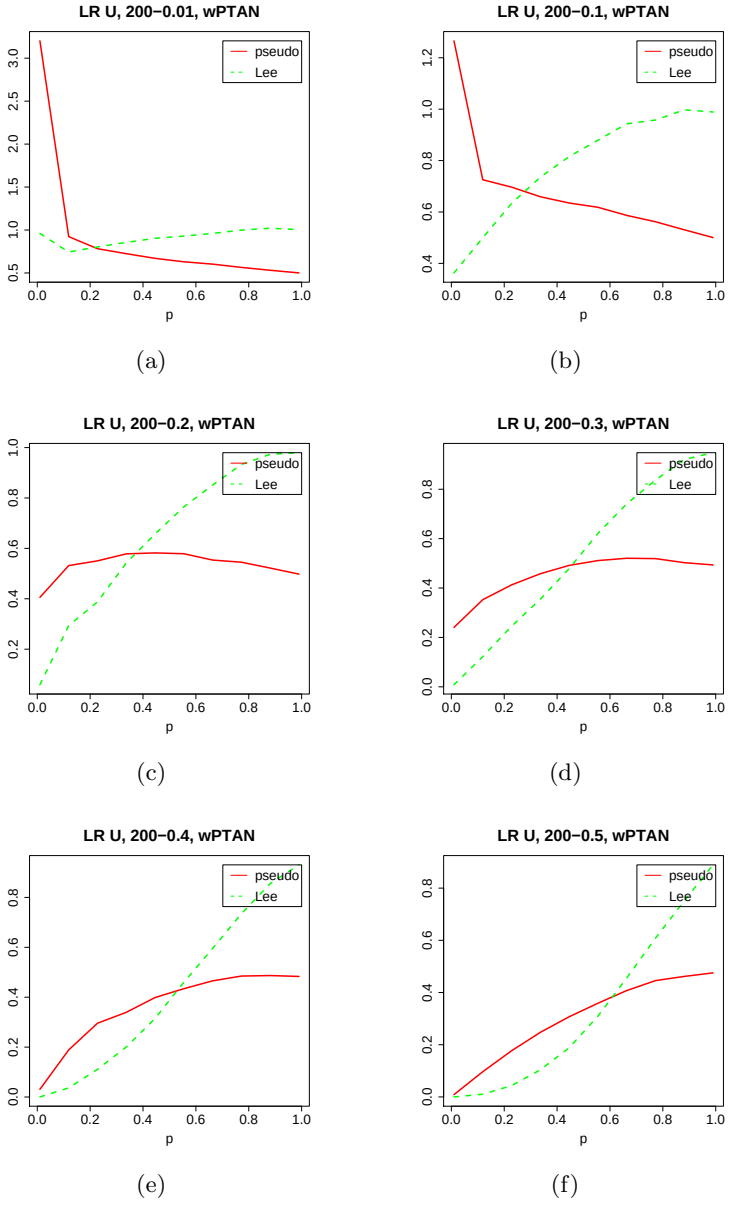


Fig. D.53. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Letter Recognition U datasets where $|\mathcal{D}_p| = 200$.

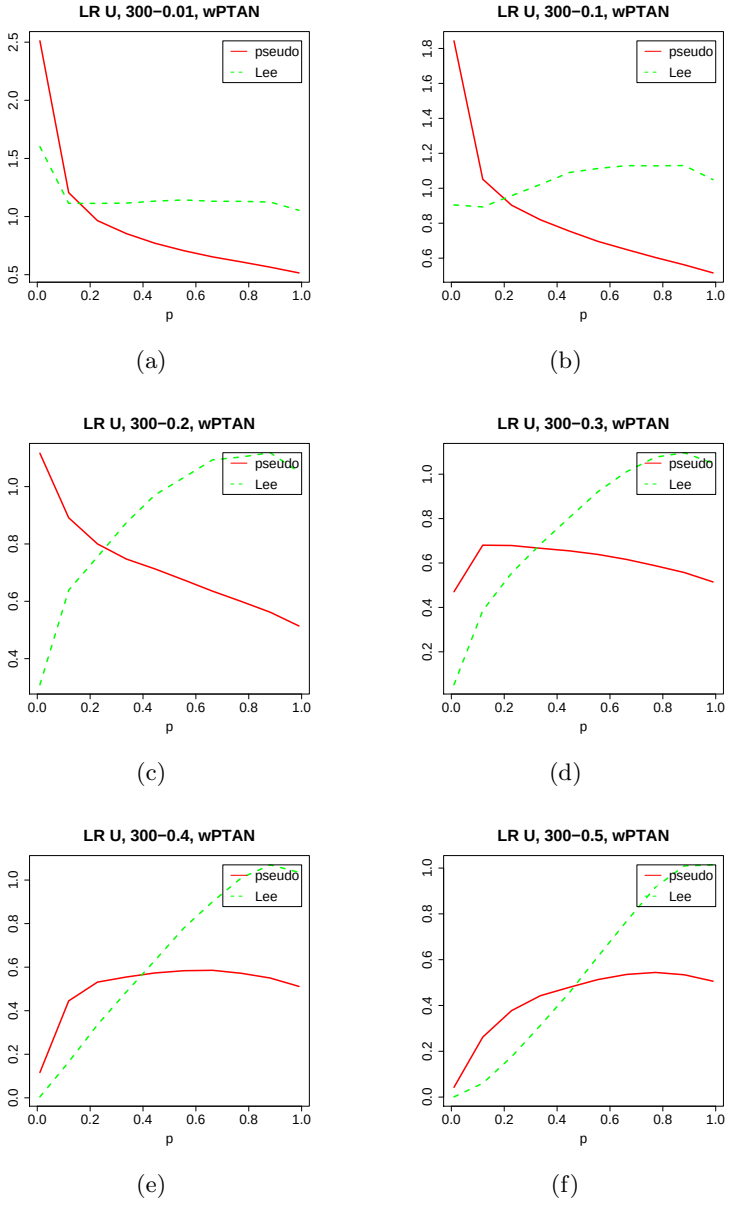


Fig. D.54. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Letter Recognition U datasets where $|\mathcal{D}_p| = 300$.

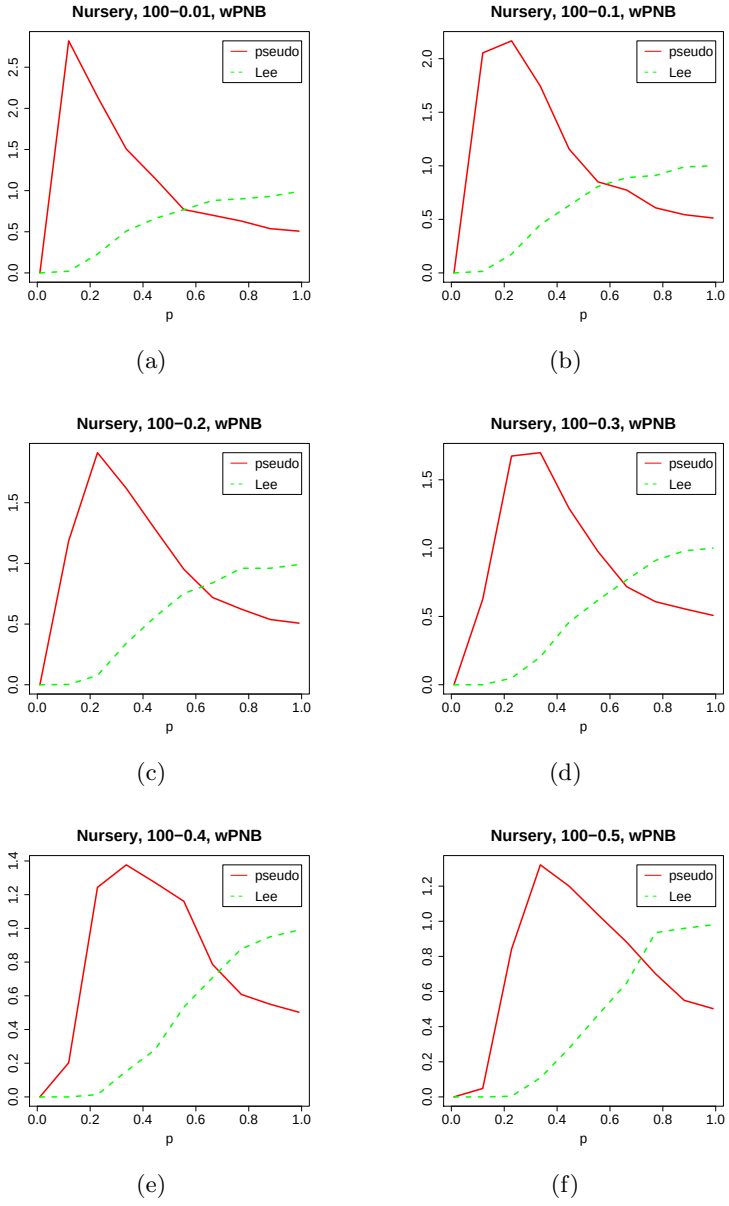


Fig. D.55. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Nursery datasets where $|\mathcal{D}_p| = 100$.

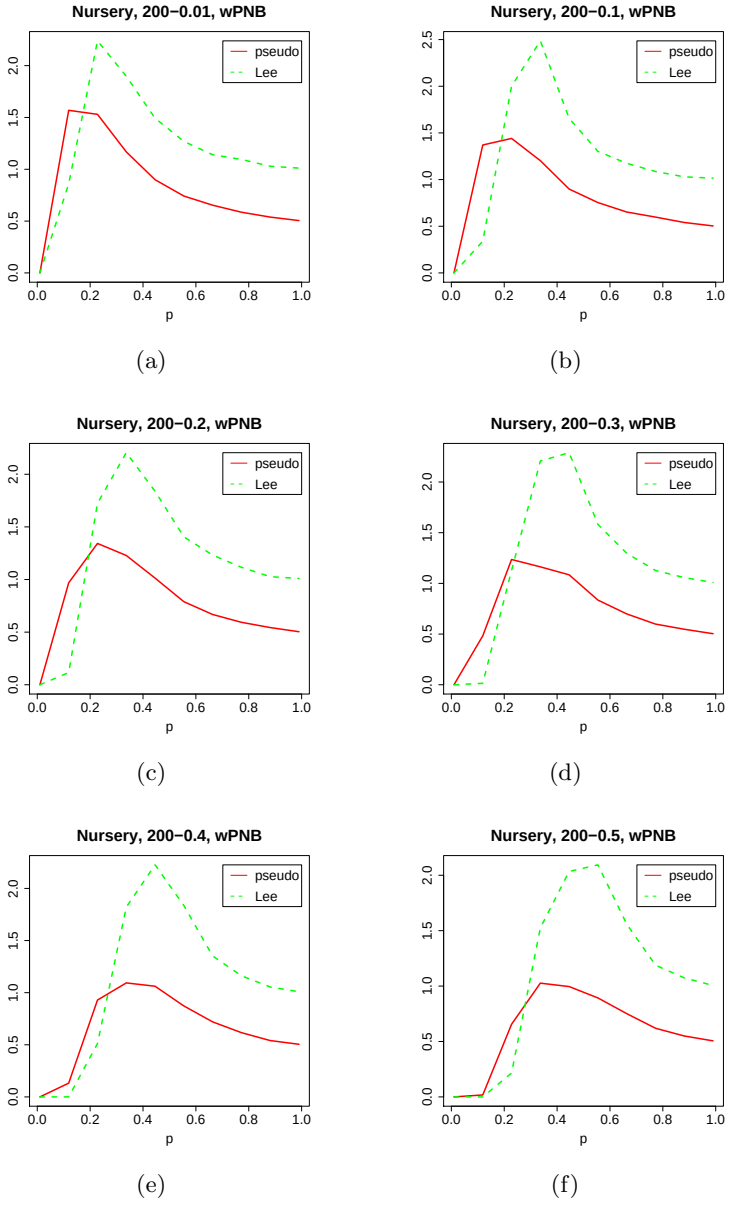


Fig. D.56. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Nursery datasets where $|\mathcal{D}_p| = 200$.

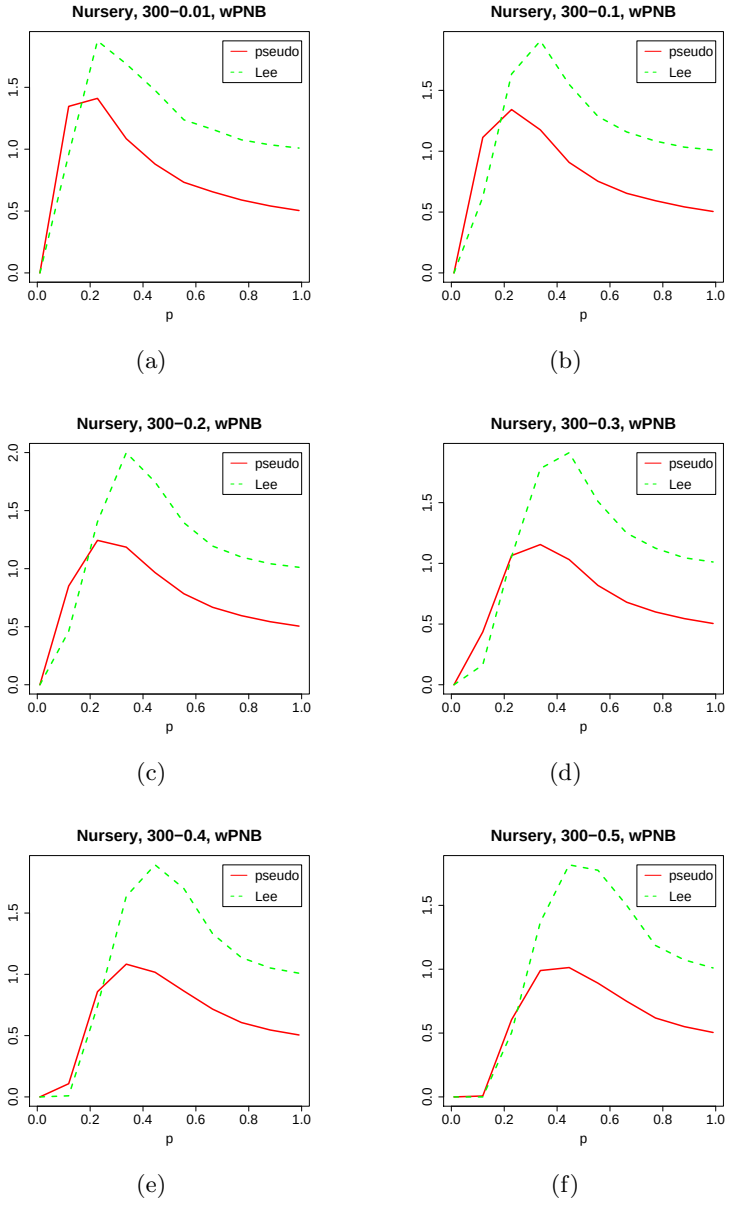
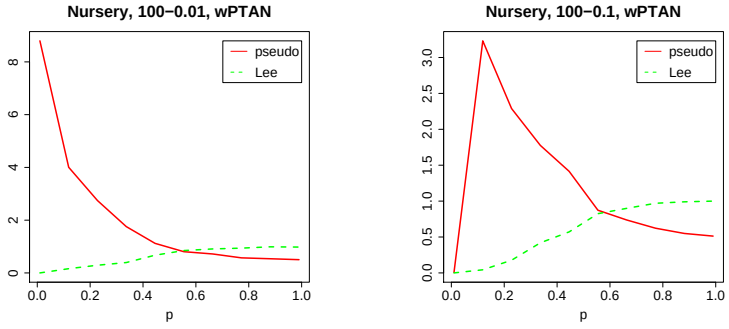
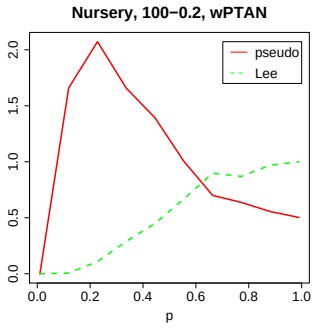


Fig. D.57. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PNB algorithm at different p values in Nursery datasets where $|\mathcal{D}_p| = 300$.

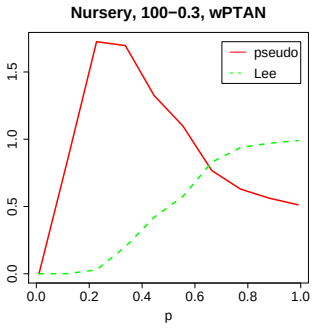


(a)

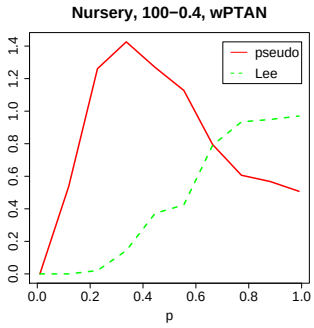
(b)



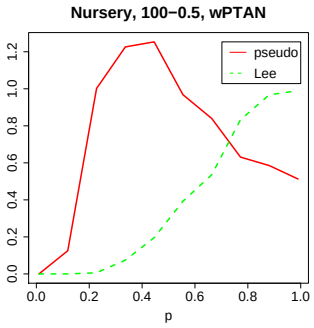
(c)



(d)



(e)



(f)

Fig. D.58. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Nursery datasets where $|\mathcal{D}_p| = 100$.

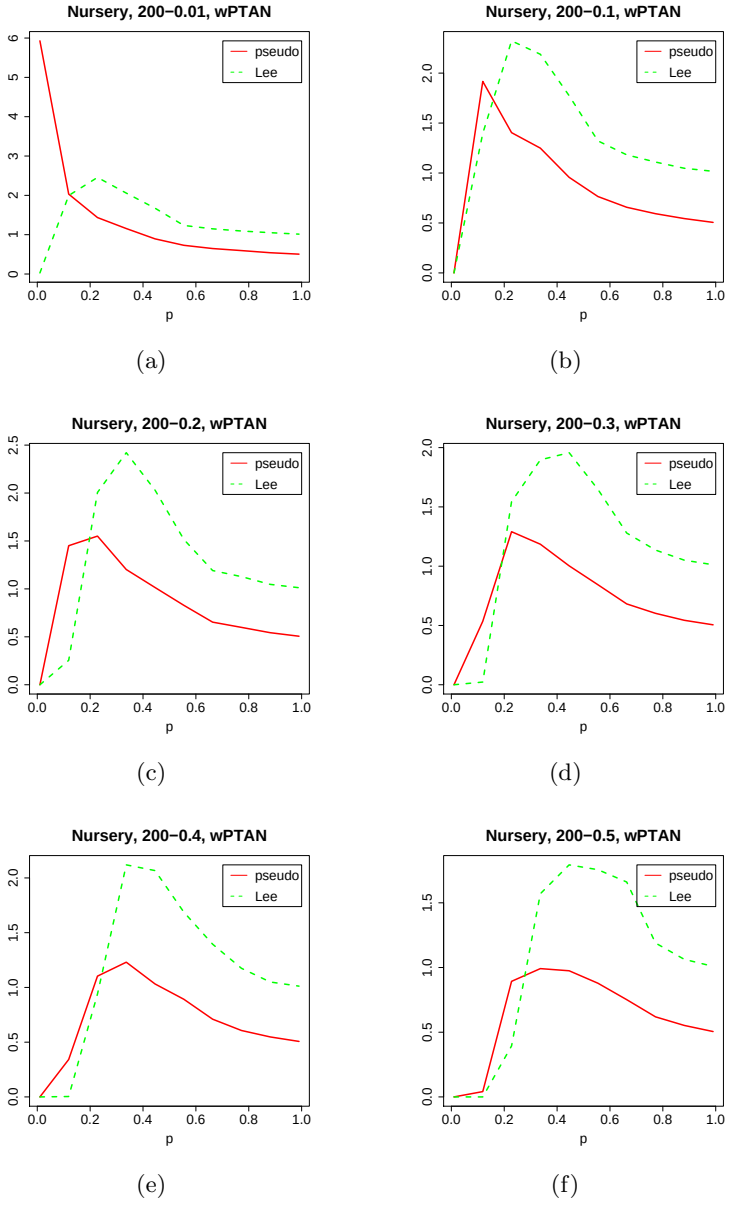
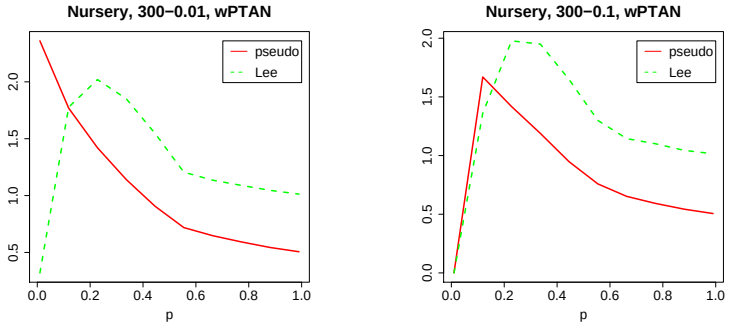
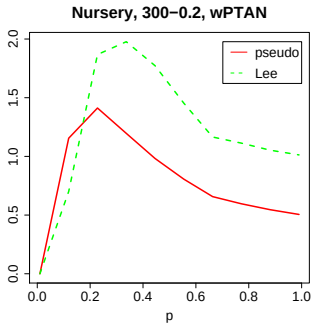


Fig. D.59. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Nursery datasets where $|\mathcal{D}_p| = 200$.

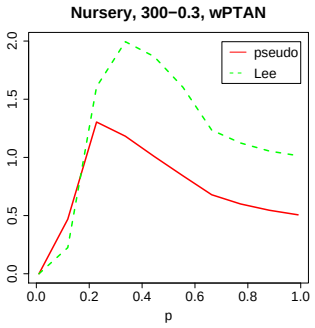


(a)

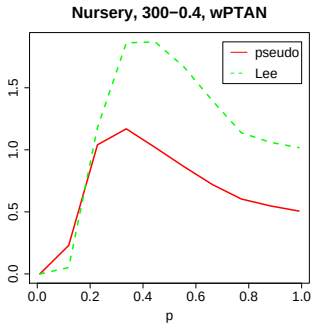
(b)



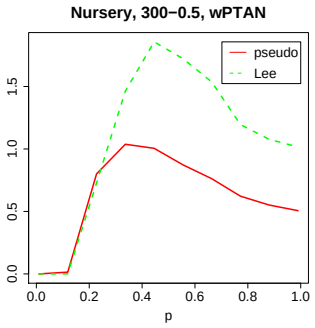
(c)



(d)



(e)



(f)

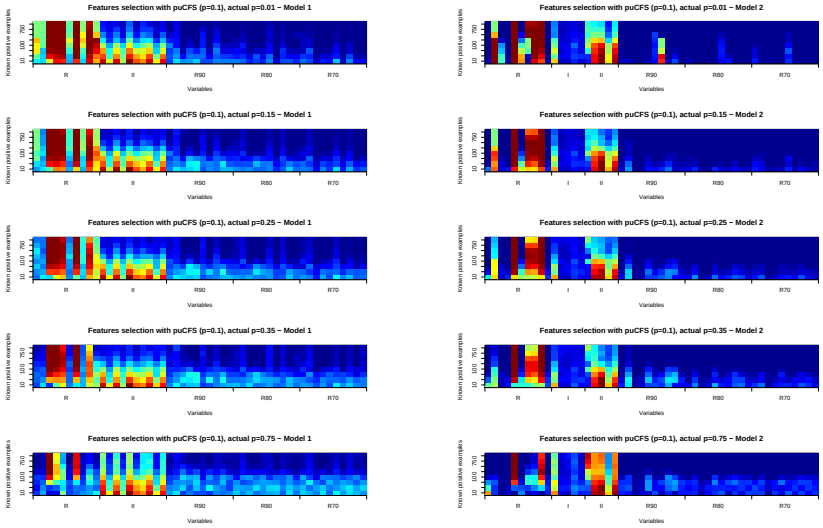
Fig. D.60. Representation of the pseudo F and the Lee metric obtained setting the p parameter in the PTAN algorithm at different p values in Nursery datasets where $|\mathcal{D}_p| = 300$.

E

Detailed results of the evaluation of puCFS and apuCFS

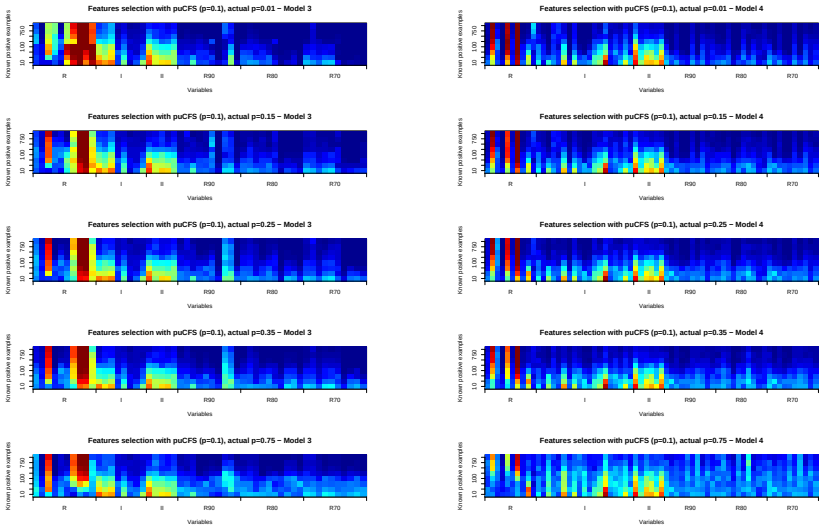
This appendix contains the detailed results obtained by puCFS and apuCFS. The X axis in the charts indicates the variable (identified as R (relevant), I (irrelevant), II (irrelevant and independent), R70 (redundant with noise degree of 30%), R80 (redundant with noise degree of 20%) and R90 (redundant with noise degree of 10%)) and the Y axis indicates the number of known positive examples. Each square in the charts represents the ratio of problems (with that amount of positive examples) where the variable was selected.

The first four figures correspond to the results obtained by puCFS ($p = 0.10$), puCFS ($p = 0.25$), puCFS ($p = 0.50$) and apuCFS in problems with different ratios of positive cases hidden in $\mathcal{D}_u$. The last figure shows the comparison between the results obtained by puCFS with different values of p (the charts show the average result obtained in problems with different ratios of positive cases hidden in $\mathcal{D}_u$).



(a) Model 1

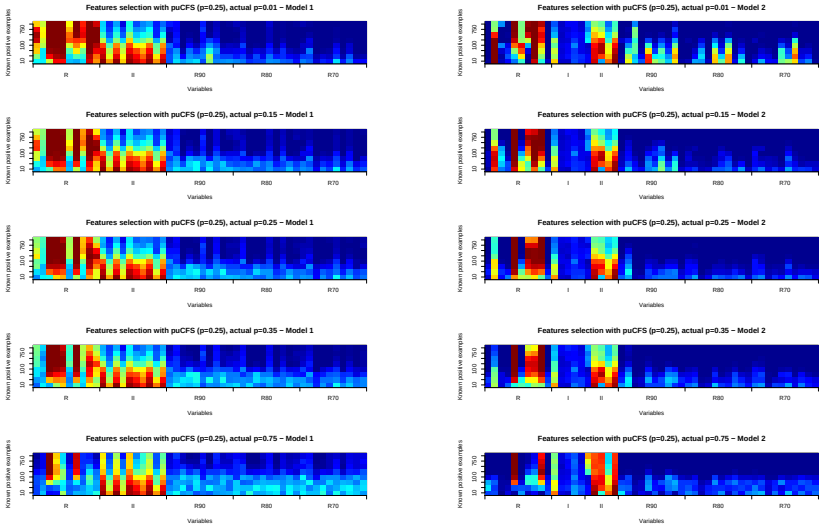
(b) Model 2



(c) Model 3

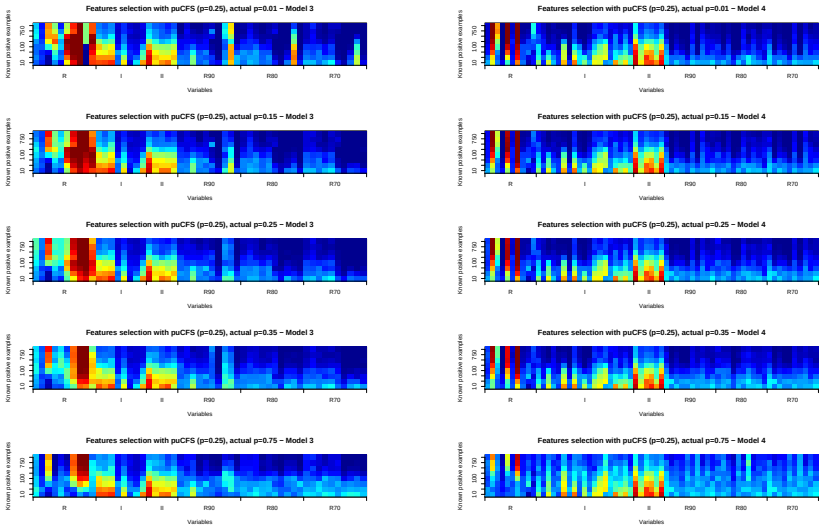
(d) Model 4

Fig. E.1. Results obtained by the puCFS algorithm when p was set at 0.1.



(a) Model 1

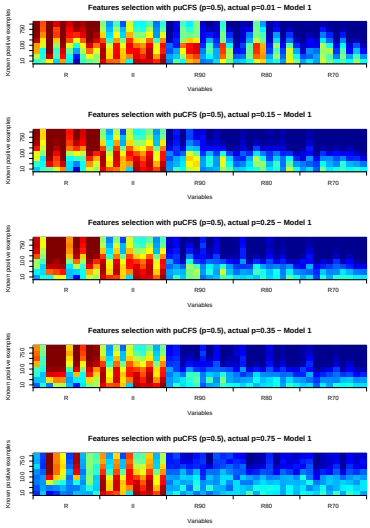
(b) Model 2



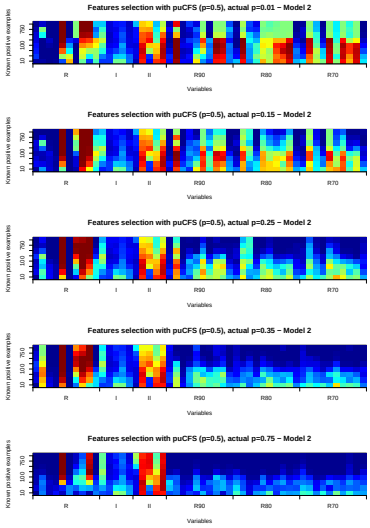
(c) Model 3

(d) Model 4

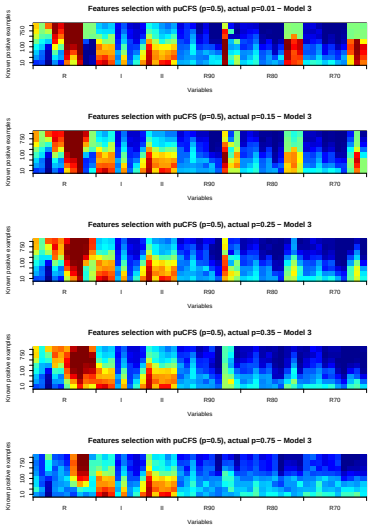
Fig. E.2. Results obtained by the puCFS algorithm when p was set at 0.25.



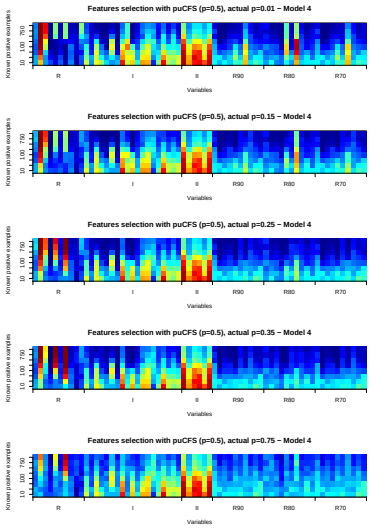
(a) Model 1



(b) Model 2

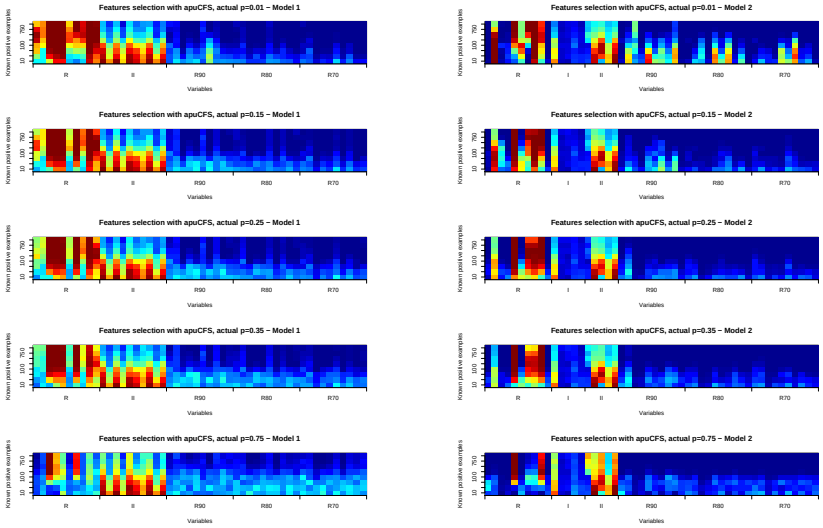


(c) Model 3



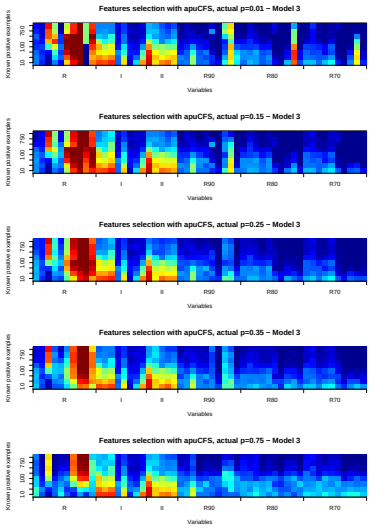
(d) Model 4

Fig. E.3. Results obtained by the puCFS algorithm when p was set at 0.5.

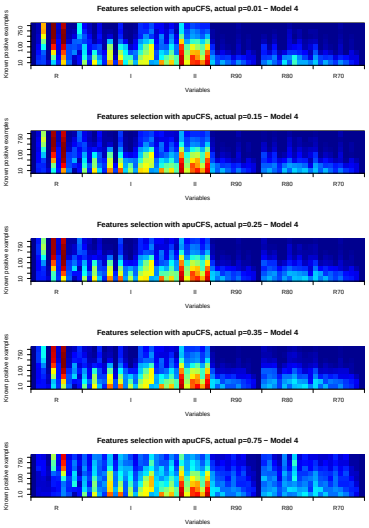


(a) Model 1

(b) Model 2

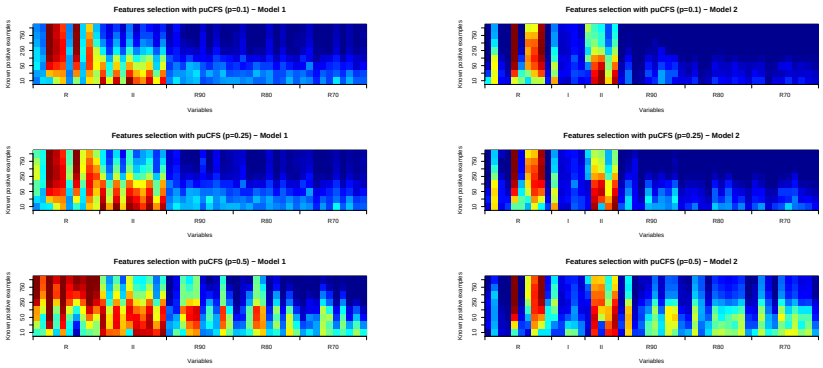


(c) Model 3



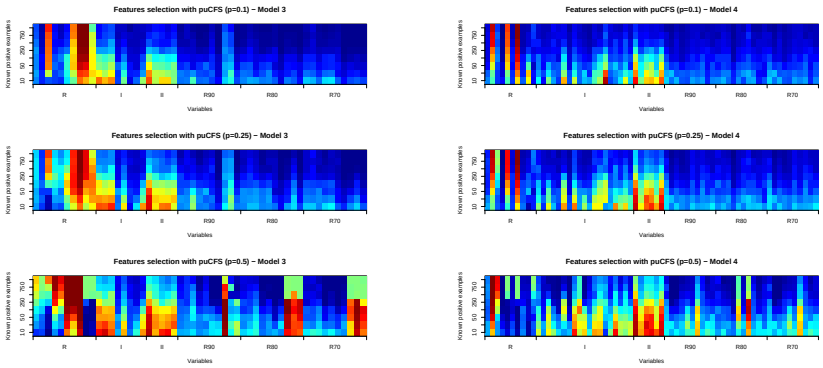
(d) Model 4

Fig. E.4. Results obtained by the apuCFS algorithm.



(a) Model 1

(b) Model 2



(c) Model 3

(d) Model 4

Fig. E.5. Comparison of the results obtained by the puCFS algorithms when p was set at 0.1, 0.25 and 0.5.

References

Acid, S., de Campos, L. M., 1995. Advances in Intelligence Computing. Vol. 945. Springer Verlag, Ch. Approximat, pp. 149–158.

Adie, E. A., Adams, R. R., Evans, K. L., Porteous, D. J., Pickard, B. S., 2005. Speeding disease gene discovery by sequence based candidate prioritization. BMC Bioinformatics 6, 55.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., Lipman, D. J., 1990. Basic local alignment search tool. Journal of Molecular Biology 215 (3), 403–410.

Amaldi, E., Kann, V., 1998. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. Theoretical Computer Science 209 (1-2), 237–260.

Asuncion, A., Newman, D. J., 2007. UCI machine learning repository.

Baak, J. P. A., Janssen, E. A. M., Soreide, K., Heikkilæ, R., 2005. Genomics and proteomics - the way forward. Annals of Oncology 16 (Suppl. 2), ii30.

Ben-Bassat, M., 1982. Use of distance measures, information measures and error bounds in feature evaluation. In: Krishnaiah, P. R., Kanal, L. N. (Eds.), Handbook of Statistics. Vol. 2. North-Holland Publishing Company, pp. 773–791.

Benvenuti, S., Arena, S., Bardelli, A., 2004. Identification of cancer genes by mutational profiling of tumor genes. FEBS Letters 579 (8), 11891–19984.

Bernardo, J. M., Smith, A. F. M., 2000. Bayesian Theory. Wiley.

Bezdek, J. C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press.

Birney, E., Andrews, D., Bevan, P., Caccamo, M., Cameron, G., Chen, Y., Clarke, L., Coates, G., Cox, T., Cuff, J., Cutts, V. C. T., Down, T., Durbin,

R., Eyras, E., Fernandez-Suarez, X. M., Gane, P., Gibbins, B., Gilbert, J., Hotz, M. H. H., Iyer, V., Kahari, A., Jekosch, K., Kasprzyk, A., Keefe, D., Keenan, S., Melsopp, H., G, L., C, M., Meidl, P., Mongin, E., Pettett, R., Potter, S., Proctor, G., Rae, M., Searle, S., Slater, G., Smedley, D., Smith, J., Spooner, W., Stabenau, A., Stalker, J., Storey, R., Ureta-Vidal, A., Woodwark, C., Clamp, M., Hubbard, T., 2004. Ensembl 2004. Nucleic Acids Research 32, D468–D470.

Bishop, C. M., 2006. Pattern Recognition and Machine Learning. Springer.

Blake, C. L., Merz, C. J., 1998. UCI repository of machine learning databases.

Blanco, R., Inza, I., Larrañaga, P., 2003. Learning Bayesian networks in the space of structures by estimation of distribution algorithms. International Journal of Intelligent Systems 18, 205–220.

Blum, A., Mitchell, T. M., 1998. Combining labeled and unlabeled data with co-training. In: Proceedings of the Eleventh Annual Conference on Computational Learning Theory. pp. 92–100.

Bouckaert, R. R., 1995. Bayesian belief networks: From construction to inference. Ph.D. thesis, University of Utrecht.

Breiman, L., 1996. Bagging predictors. Machine Learning 26 (2), 123–140.

Breiman, L., 2001. Random forest. Machine Learning 45 (1), 5–32.

Breiman, L., Friedman, J., Stone, C. J., Olshen, R. A., 1984. Classification and Regression Trees. Chapman & Hall/CRC.

Buntine, W., 1991. Theory refinement in Bayesian networks. In: Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence. pp. 52–60.

Buntine, W., 1996. A guide to the literature on learning probabilistic networks from data. IEEE Transactions on Knowledge and Data Engineering 8, 195–210.

Carter, S. L., Eklund, A. C., Kohane, I. S., Harris, L. N., Szallasi, Z., 2006. A signature of chromosomal instability inferred from gene expression profiles predicts clinical outcome in multiple human cancers. Nature Genetics 38 (9), 1043–1048.

Castelo, R., Guigó, R., 2004. Splice site identification by idlbns. Bioinformatics 4 (20), i69–i76.

Castillo, E., Gutiérrez, J. M., Hadi, A. S., 1997. Expert Systems and Probabilistic Network Models. Springer-Verlag.

Chapelle, O., Zien, A., Schökopf, B., 2006. Semi-supervised Learning. MIT Press.

Chickering, D. M., Geiger, D., Heckerman, D., 1995. Learning Bayesian networks: Search methods and experimental results. In: Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics. pp. 112–128.

Chow, C. K., Liu, C. N., 1968. Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory IT-14 (3), 462–467.

Cooper, G. F., Herskovits, E., 1992. A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9, 309–347.

Cover, T. M., Thomas, J. A., 2006. Elements of Information Theory. John Wiley & Sons, Inc.

Crichton, N., Hinde, J., Marchini, J., 1998. Models for diagnosing chest pain: is cart helpful? Statistics in Medicine 16 (7), 717–727.

Dave, S. S., Wright, G., Tan, B., Rosenwald, A., Gascoyne, R. D., Chan, W. C., Fisher, R. I., Braziel, R. M., Rimsza, L. M., Grogan, T. M., Miller, T. P., LeBlanc, M., Greiner, T. C., Weisenburger, D. D., Lynch, J. C., Vose, J., Armitage, J. O., Smeland, E. B., Kvaloy, S., Holte, H., Delabie, J., Connors, J. M., Lansdorp, P. M., Ouyang, Q., Lister, T. A., Davies, A. J., Norton, A. J., Muller-Hermelink, H. K., Ott, G., Campo, E., Montserrat, E., Wilson, W. H., Jaffe, E. S., Simon, R., Yang, L., Powell, J., Zhao, H., Goldschmidt, N., Michael, B. A., Chiorazzi, B. A., Staudt, L. M., 2004. Prediction of survival in follicular lymphoma based on molecular features of tumor-infiltrating immune cells. New England Journal of Medicine 351 (21), 2159–2169.

de Campos, L. M., 1998. Probabilistic Expert Systems. Ediciones de la Universidad de Castilla-La Mancha, Ch. Automatic, pp. 113–140.

de Campos, L. M., Fernández-Luna, J. M., Gámez, J. A., Puerta, J. M., 2002. Ant colony optimization for learning Bayesian networks. International Journal of Approximate Reasoning 31 (21), 291–311.

de Campos, L. M., Puerta, J. M., 2001. Stochastic local and distributed search algorithms for learning Bayesian networks. In: Proceedings of the Third Symposium on Adaptive Systems. pp. 109–115.

de Waal, P. R., van Der Gaag, L. C., 2007. Inference and learning in multi-dimensional Bayesian network classifiers. In: Mellouli, K. (Ed.), ECSQARU.

Vol. 4724. Springer, pp. 501–511.

Dempster, A. P., Laird, N. M., Rubin, D. B., 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society Series B 39, 1–38.

Denis, F., 1998. PAC learning from positive statistical queries. In: Proceedings of the 9th International Conference on Algorithmic Learning Theory. Springer-Verlag, pp. 112–126.

Denis, F., Gilleron, R., Tommasi, M., 2002. Text classification from positive and unlabeled examples. In: The 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2002. pp. 1927–1934.

Denis, F., Laurent, A., Gilleron, R., Tommasi, M., 2003. Text classification and co-training from positive and unlabeled examples. In: Proceedings of the ICML 2003 Workshop: The Continuum from Labeled to Unlabeled Data. pp. 80–87.

Domingos, P., Pazzani, M., 1997. On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning 29 (2-3), 103–130.

Duan, Q., Miao, D., Jin, K., 2007. A rough set approach to classifying web page without negative examples. In: Proceedings of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. Springer, pp. 481–488.

Duda, R. O., Hart, P. E., 1973. Pattern Classification and Scene Analysis. Jon Wiley and Sons.

Duda, R. O., Hart, P. E., Stork, D. G., 2001. Pattern Classification. Wiley Interscience.

Efron, B., 1983. Estimating the error rate of a prediction rule: Improvement on cross-validation. Journal of the American Statistical Association 78, 316–331.

Efron, B., Tibshirani, R. J., 1993. An Introduction to the Bootstrap. Chapman & Hall.

Etxeberria, R., Larrañaga, P., Pikaza, J. M., 1997. Analysis of the behavior of the genetic algorithms when searching Bayesian networks from data. Pattern Recognition Letters 18 (11-13), 1269–1273.

Fix, E., Hodges, J. L., 1951. Discriminatory analysis, non-parametric discrimination: Consistency properties. Tech. rep.

Forgy, E., 1965. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. Biometrics 21, 768–769.

Friedman, N., 2004. Inferring cellular networks using probabilistic graphical model. Science 303 (5659), 799–805.

Friedman, N., Geiger, D., Goldszmit, M., 1997. Bayesian network classifiers. Machine Learning 29, 131–163.

Friedman, N., Linial, M., Nachman, I., Pe'er, D., 2000. Using Bayesian networks to analyze expression data. Journal of Computational Biology 7 (3-4), 601–620.

Furney, S. J., del Mar Albá, Maria and López-Bigas, Núria, journal = BMC Genomics, n. . . p. . . t. . D. v. . . y. . ., ????

Furney, S. J., Madden, S. F., Kisiel, T. A., Higgins, D. G., López-Bigas, N., 2007. Distinct patterns in the regulation and evolution of human cancer genes. In Silico Biology 8, 4.

Futreal, P. A., Coin, L., Marshall, M., Down, T., Hubbard, T., Wooster, R., Rahman, N., Stratton, M. R., 2004. A census of human cancer genes. Nature Reviews Cancer 4 (3), 177–183.

Geiger, D., 1992. An entropy-based learning algorithm of Bayesian conditional trees. In: Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence. pp. 92–97.

Greenman, C., Stephens, P., Smith, R., Dalgliesh, G. L., Hunter, C., Bignell, G., Davies, H., Teague, J., Butler, A., Stevens, C., Edkins, S., Oïœ meara, S., Vastrik, I., Schmidt, E. E., Avis, T., Barthorpe, S., Bhamra, G., Buck, G., Choudhury, B., Clements, J., Cole, J., Dicks, E., Forbes, S., Gray, K., Halliday, K., Harrison, R., Hills, K., Hinton, J., Jenkinson, A., Jones, D., Menzies, A., Mironenko, T., Perry, J., Raine, K., Richardson, D., Shepherd, R., Small, A., Tofts, C., Varian, J., Webb, T., West, S., Widaa, S., Yates, A., Cahill, D. P., Louis, D. N., Goldstraw, P., Nicholson, A. G., Brasseur, F., Looijenga, L., Weber, B. L., Chiew, Y.-E., Defazio, A., Greaves, M. F., Green, A. R., Campbell, P., Birney, E., Easton, D. F., Chenevix-Trench, G., Tan, M.-H., Khoo, S. K., Teh, B. T., Yuen, S. T., Leung, S. Y., Wooster, R., Futreal, A. P., Stratton, M. R., 2007. Patterns of somatic mutation in human cancer genomes. Nature 446 (7132), 2153–2158.

Guyon, I., Elisseeff, André, j. . J. p. . . t. . A. v. . . y. . ., ????

Halkidi, M., Batistakis, Y., Vazirgiannis, M., 2001. On clustering validation techniques. Journal of Intelligent Information Systems 17 (2-3), 107–145.

Hall, M. A., Smith, L. A., 1997. Feature subset selection: A correlation based filter approach. In: Springer (Ed.), Proceedings of the International Conference on Neural Information Processing and Intelligent Information Systems. pp. 855–858.

Hamosh, A., Scott, A. F., Amberger, J., Valle, D., Mckusick, V. A., 2000. Online mendelian inheritance in man (omim). Human mutation 15 (1), 57–61.

Hand, D., You, K., 2001. Idiot's Bayes -not so stupid after all? International Statistical Review 69, 385–398.

Heckerman, D., 1995. A tutorial on learning with Bayesian networks. Tech. rep.

Heckerman, D., Geiger, D., Chickering, D. M., 1995. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning 20, 197–243.

Herskovits, E., Cooper, G., 1990. {K}utató: An entropy-driven system for construction of probabilistic expert systems from database. In: Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence. pp. 54–62.

Hubbard, T., Barker, D., Birney, E., Cameron, G., Chen, Y., Clark, L., Cox, T., Cuff, J., Curwen, V., Down, T., Durbin, R., Eyras, E., Gilbert, J., Hammond, M., Huminiecki, L., Kasprzyk, A., Lehvaslaiho, H., Lijnzaad, P., Melsopp, C., Mongin, E., Pettett, R., Pocock, M., Potter, S., Rust, A., Schmidt, E., Searle, S., Slater, G., Smith, J., Spooner, W., Stabenau, A., Stalker, J., Stupka, E., Ureta-Vidal, A., Vastrik, I., Clamp, M., 2002. The Ensembl genome database project. Nucleic Acids Research 30, 38–41.

Inza, I. n., Larrañaga, P., Etxeberria, R., Sierra, B., 2000. Feature subset selection by Bayesian networks based optimization. Artificial Intelligence 123 (1-2), 157–184.

Jardine, N., Sibson, R., 1971. Mathematical Taxonomy. Wiley.

Jensen, F. V., Nielsen, T. D., 2007. Bayesian Networks and Decision Graphs. Springer Verlag.

John, G., 1997. Enhancements to the data mining process. Ph.D. thesis, Computers Science Department, School of Engineering, Stanford University.

Jonsson, P. F., Bates, P. A., 2006. Global topological features of cancer proteins in the human interactome. Bioinformatics 22, 2291–2297.

Jordan, M. I. (Ed.), 1998. Learning in Graphical Models. Kluwer.

Karlin, S., Brocchieri, L., Bergman, A., Mrazek, J., Gentles, A. J., 2002. Amino acid runs in eukaryotic proteomes and disease associations. Proceedings of the National Academy of Science USA 99, 333–338.

Kearns, M., 1993. Efficient noise-tolerant learning from statistical queries. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing. pp. 392–401.

Kim, M., Gans, J. D., Nogueira, C., Wang, A., Paik, J.-H., Feng, B., Brennan, C., Hahn, W. C., Cordon-Cardo, C., Wagner, S. N., Flotte, T. J., Duncan, L. M., Granter, S. R., Chin, L., 2006. Comparative oncogenomics identifies nedd9 as a melanoma metastasis gene. Cell 125, 1269–1281.

Kleinbaum, D. G., Kupper, L. L., Chambless, L. E., 1982. Logistic regression analysis of epidemiologic data: Theory and practice. Comunications on Statistics 11 (5), 485–547.

Kohavi, R., John, G. H., 1997. Wrappers for feature subset selection. Artificial Intelligence 97 (1-2), 273–324.

Köhler, S., Bauer, S., Horn, D., Robinson, P. N., 2008. Walking the interactome for prioritization of candidate disease genes. American Journal of Human Genetics 82 (4), 949–958.

Kononenko, I., 1991. Semi-naive Bayes classifiers. In: Proceedings of the 6th European Working Session on Learning. pp. 206–219.

Korb, K. B., Nicholson, A. E., 2004. Bayesian Artificial Intelligence. Chapman & Hall / CRC.

Krawczak, M., Ball, E. V., Fenton, I., Stenson, P. D., Abeysinghe, S., Thomas, N., Cooper, D. N., 2000. Human gene mutation database - a biomedical information and research resource. Human Mutation 15 (1), 45–51.

Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I. n., Lozano, J. A., Armañanzas, R., Santafé, G., Pérez, A., Robles, V., 2006. Machine learning in bioinformatics. Briefings in Bioinformatics 7 (1), 86–112.

Larrañaga, P., Kuijpers, C., Murga, R., Yurramendi, Y., 1996. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. IEEE Transactions on System, Man and Cybernetics 26 (4), 487–493.

Larrañaga, P., Lozano, J. A. (Eds.), 2002. Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers.

Lauritzen, S., Dawid, A. P., Larsen, B. N., Leimer, H. G., 1990. Independence properties of directed Markov fields. Networks 20, 491–505.

Lauritzen, S. L., 1996. Graphical Models. Oxford University Press.

Lauritzen, S. L., 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.

Lee, W. S., Liu, B., 2003. Learning with positive and unlabeled examples using weighted logistic regression. In: Proceedings of the Twentieth International Conference on Machine Learning. pp. 448–455.

Letouzey, F., Denis, F., Gilleron, R., 2000. Learning from positive and unlabeled examples. In: ALT 2000, the Eleventh International Conference on Algorithmic Learning Theory.

Li, X., Liu, B., 2003. Learning to classify texts using positive and unlabeled data. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence 2003. pp. 587–594.

Li, X., Liu, B., Ng, S.-K., 2007a. Learning to classify documents with only a small positive training set. In: Proceedings of the European Conference on Machine Learning 2007. pp. 201–213.

Li, X., Liu, B., Ng, S.-K., 2007b. Learning to identify unexpected instances in the test set. In: Proceedings of the International Joint Conference on Artificial Intelligence 2007. pp. 2802–2807.

Li, X.-L., Liu, B., 2005. Learning from positive and unlabeled examples with different data distributions. In: Gama, J., Camacho, R., Brazdil, P., Jorge, A., Torgo, L. (Eds.), Proceedings of the European Conference on Machine Learning. Springer, pp. 218–229.

Liu, B., Dai, Y., Li, X., Lee, W. S., Yu, P. S., 2003. Building text classifiers using positive and unlabeled examples. In: Third IEEE International Conference on Data Mining (ICDM'03). pp. 179–186.

Liu, B., Lee, W. S., Yu, P. S., Li, X., 2002. Partially supervised classification of text documents. In: Proceedings of the Nineteenth International Conference on Machine Learning. pp. 387–394.

Liu, E., Kuznetsov, V., Miller, L., 2006. In the pursuit of complexity: Systems medicine in cancer biology. Cancer Cell 9 (4), 245–247.

Liu, H., Motoda, H., 1998. Feature Extraction, Construction and Selection: A Data Mining Perspective. Kluwer Academic Publishers.

Liu, H., Motoda, H. (Eds.), 2008. Computational Methods of Feature Selection. Chapman and Hall/CRC Press.

Liu, Z., Shi, W., Li, D., Qin, Q., 2005. Partially supervised classification - based on weighted unlabeled samples support vector machine. In: Proceedings of the First International Conference on Advanced Data Mining and Applications. pp. 118–129.

López-Bigas, N., Audit, B., Ouzounis, C., Parra, G., Guigó, R., 2005. Are splicing mutations the most frecuent cause of hereditary disease? FEBS Letters 579 (9), 1900–1903.

López-Bigas, N., Blencowe, B. B., Ouzounis, C. A., 2006. Highly consistent patterns for inherited human diseases at the molecular level. Bioinformatics 22 (3), 269–277.

López-Bigas, N., Ouzounis, C. A., 2004. Genome-wide identification of genes likely to be involved in human genetic disease. Nucleic Acids Research 32 (10), 3108–3114.

Lu, J., Getz, G., Miska, E. A., Alvarez-Saavedra, E., Lamb, J., Peck, D., Sweet-Cordero, A., Ebert, B. L., Mak, R. H., Ferrando, A. A., Downing, J. R., Jacks, T., Horvitz, R. R., Golub, T. R., 2005. MicroRNA expression profiles classify human cancers. Nature 435 (7043), 834–838.

Lucas, P., 2004. Advances in Bayesian Networks. Springer-Verlag, Ch. Restricted, pp. 217–234.

MacKay, D. J. C., 2003. Information Theory, Inference, and Learning Algorithms. Cambridge University Press.

Manevitz, L. M., Yousef, M., 2001. One-class SVMs for document classification. Journal of Machine Learning Research 2, 139–154.

McCulloch, W. S., Pitts, W., 1943. A logical calculus of ideas imminet in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133.

Minsky, M., 1961. Steps toward artificial intelligence. Proceedings of the Institute of Radio Engineers 49, 8–30.

Mühlenbein, H., Mahning, T., 2001. Advances in Evolutionary Synthesis of Intelligent Agents. MIT Press, Ch. Evolutiona, pp. 429–455.

Mulder, N. J., Apweiler, R., Attwood, T. K., Bairoch, A., Bateman, A., Binns, D., Bork, P., Buillard, V., Cerutti, L., Copley, R., Courcelle, E., Das, U., Daugherty, L., Dibley, M., Finn, R., Fleischmann, W., Gough, J., Haft, D., Hulo, N., Hunter, S., Kahn, D., Kanapin, A., Kejariwal, A., Labarga, A., Langendijk-Genevaux, P. S., Lonsdale, D., Lopez, R., Letunic, I., Madera,

M., Maslen, J., McAnulla, C., McDowall, J., Mistry, J., Mitchell, A., Nikolskaya, A. N., Orchard, S., Orengo, C., Petryszak, R., Selengut, J. D., Sigrist, C. J. A., Thomas, P. D., Valentin, F., Wilson, D., Wu, C. H., Yeats, C., 2006. New developments in the interpro database. Nucleic Acids Research 35 (Database issue), —-224.

Myers, J. W., Laskey, K. B., Levitt, T., 1999. Learning Bayesian networks from incomplete data with stochastic search algorithms. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence. pp. 476–485.

Neapolitan, E., 2003. Learning Bayesian Networks. Prentice Hall, Upper Saddle River, NJ.

Ohmann, C., Yang, Q., Kunneke, M., Stozing, H., Tohn, K., Lorenz, W., 1988. Bayes theorem and conditional dependence of symptoms: Different models applied to data of upper gastrointestinal bleeding. Methods of Information in Medicine 8, 23–36.

Pawlak, Z., 1991. Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers.

Pazzani, M., 1997. Searching for dependencies in Bayesian classifiers. Springer-Verlag.

Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers.

Pérez-Iratxeta, C., Bork, P., Andrade, M. A., 2002. Association of genes to genetically inherited diseases using data mining. Nature Genetics 31, 316–319.

Press, W. H., 1988. Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press.

Pruitt, K. D., Katz, K. S., Sicotte, H., Maglott, D. R., 2000. Introducing RefSeq and LocusLink: Curated human genome resources at the NCBI. Trends in Genetics 16 (1), 44–47.

Pruitt, K. D., Maglott, D. R., 2001. RefSeq and LocusLink: NCBI gene-centered resources. Nucleic Acids Research 29, 137–140.

Quinlan, J. R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.

Ramaswamy, S., Golub, T. R., 2002. DNA microarrays in clinical oncology. Journal of Clinical Oncology 20 (7), 1932–1941.

Rissanen, J., 1978. Modeling by shortest data description. Automatics 14, 465–471.

Rocchio, J., 1971. The smart retrieval system: Experiments in automatic document processing. Englenwood Cliffs, Ch. Relevant f.

Rodríguez, J. D., Lozano, J. A., 2008. Multi-objective learning of multi-dimensional Bayesian classifiers. In: International Conference on Hybrid Intelligent Systems. IEEE Computer Society, pp. 501–506.

Rosenblatt, F., 1962. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books.

Saeys, Y., Inza, I. n., Larrañaga, P., 2007. A review of feature selection techniques in bioinformatics. Bioinformatics (Oxford, England) 23 (19), 2507–17.

Sahami, M., 1996. Learning limited dependence Bayesian classifiers. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 335–338.

Schneider, K.-M., 2004. Learning to filter junk e-mail from positive and unlabeled examples. In: Proceedings of the 1st International Joint Conference on Natural Language Processing (IJCNLP-04). pp. 602–607.

Schwarz, G., 1978. Estimating the dimension of a model. Annals of Statistics 6, 461–464.

Silva, D. G., Schonbach, C., Brusic, V., Socha, L. A., Nagashima, T., Petrovsky, N., 2004. Identification of "pathologs" (disease-related genes) from the riken mouse cDNA dataset using human curation plus facts, a new biological information extraction system. BMC Genomics 5, 28.

Spiegelhalter, D., Lauritzen, S., 1990. Sequential updating of conditional probabilities on directed graphical structures. Networks 20, 579–605.

Spirtes, P., Glymour, C., Scheines, R., 1993. Causation, Prediction and Search. Vol. 81 of Lecture Notes in Statistics. Springer-Verlag.

Spirtes, S., Glymour, C., Scheines, R., 1991. An algorithm for fast recovery of sparse causal graphs. Social Science Computing Reviews 9, 62–72.

Stone, M., 1974. Cross-validatory choice and assessment of statistical predictions (with discussion). Journal of the Royal Statistical Society B 36, 147–211.

Tax, D. M. J., 2001. One-class classification. Ph.D. thesis, Technische Universiteit Delft.

Tax, D. M. J., Duin, R. P. W., 2002. Uniform object generation for optimizing one-class classifiers. Journal of Machine Learning Research 2 (2), 155–173.

Tiffin, N., Kelso, J. F., Pan, A. R. P. H., Bajic, V. B., Hide, W. A., 2005. Integration of text- and data-mining using ontologies successfully selects disease gene candidates. Nucleic Acids Research 33, 1544–1552.

Todd, B., Stamper, R., 1994. The relative accuracy of a variety of medical diganosis programs. Methods of Information Medicine 33, 402–416.

Tomlins, S. A., Mehra, R., Rhodes, D. R., Cao, X., Wang, L., Dhanasekaran, S. M., Kalyana-Sundaram, S., A., J. T. W. M., J., R. K., Pienta, Shah, R. B., Chinnaiyan, A. M., 2006. Integrative molecular concept modeling of prostate cancer progression. Nature Genetics 39 (1), 41–51.

Turner, F. S., Clutterbuck, D. R., Semple, C. A. M., 2003. Pocus: mining genomic sequence annotation to predict disease genes. Genome Biology 4, R75.

Valiant, L. G., 1984. A theory of the learnable. Communications of the ACM, 1134–1142.

van der Gaag, L. C., de Waal, P. R., 2006. Muti-dimensional Bayesian network classifiers. In: Proceedings of the Third European workshop in probabilistic graphical models. pp. 107–114.

van Driel, M. A., Cuelenaere, K., Kemmeren, P. P., Leunissen, J. A., Brunner, H. G., 2003. A new web-based data mining tool for the identification of candidate genes for human genetic disorders. European Journal of Human Genetics 11, 57–63.

van Driel, M. A., Cuelenaere, K., Kemmeren, P. P., Leunissen, J. A., Brunner, H. G., Vriend, G., 2005. {GeneSeeker}: extraction and integration of human disease-related information from web-based genetic databases. Nucleic Acids Research 33, —-758.

van 't Veer, ????

Vapnik, V., 1995. The Nature of Statistical Learning. Springer-Verlag.

Vogelstein, B., Kinzler, K. W., 1993. The multistep nature of cancer. Trends in Genetics 9 (4), 138–141.

Wang, C., Ding, C., Meraz, R. F., Holbrook, S. R., 2006. PSoL: A positive sample only learning algorithm for finding non-coding RNA genes. Bioinformatics 22 (21), 2590–2596.

Whittaker, J., 1991. Graphical Models in Applied Multivariate Statistics. Wiley Series in Probability and Mathematical Statistics.

Wood, L. D., Parsons, D. W., Jones, S., Lin, J., Sjöblom, T., Leary, R. J., Shen, D., Boca, S. M., Barber, T., Ptak, J., Silliman, N., Szabo, S., Dezso, Z., Ustyanksky, V., Nikolskaya, T., Nikolsky, Y., Karchin, R., Wilson, P. A., Kaminker, J. S., Zhang, Z., Croshaw, R., Willis, J., Dawson, D., Shipitsin, M., Willson, J. K. V., Sukumar, S., Polyak, K., Park, B. H., Pethiyagoda, C. L., Pant, P. V. K., Ballinger, D. G., Sparks, A. B., Hartigan, J., Smith, D. R., Suh, E., Papadopoulos, N., Buckhaults, P., Markowitz, S. D., Parmigiani, G., Kinzler, K. W., Velculescu, V. E., Vogelstein, B., 2007. The genomic landscapes of human breast and colorectal cancers. Science 318, 1108–1113.

Yu, H., 2003. SVMc: Single-class classification with support vector machines. In: Proceedings of the International Joint Conference on Artificial Intelligence. pp. 567–574.

Yu, H., 2005. Single-class classification with mapping convergence. Machine Learning 61 (1-3), 49–69.

Yu, H., Han, J., Chang, K. C.-C., 2002. PEBL: positive example based learning for web page classification using SVM. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 239–248.

Yu, H., Zuo, W., Peng, T., ???? In: Heidelberg, . (Ed.), Proceedings of the Fourth Mexican International Conference on Artificial Intelligence.

Zhang, D., Lee, W. S., 2005. A simple probabilistic approach to learning from positive and unlabeled examples. In: Proceedings of Fifth Annual UK Conference on Computational Intelligence (UKCI). pp. 83–87.

Zhao, X.-M., Wang, Y., Chen, L., Aihara, K., 2008. Gene function prediction using labeled and unlabeled data. BMC Bioinformatics 9, 57.

Zhu, X., 2005. Semi-supervised learning literature survey. Tech. Rep. 1530.