H24/3596

MONASH UNIVERSITY
THESIS ACCEPTED IN SATISFACTION OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
ON..... 5 August 2003 .....
.....[redacted].....
Sec. Research Graduate School Committee

Discretization for Naive-Bayes Learning

Ying Yang

A thesis submitted for

the degree of Doctor of Philosophy

to the School of Computer Science and

Software Engineering of Monash University

July 2003

Typeset in Palatino by TEX and LATEX 2ε.

To my husband, your love creates my spirit.

To my parents, your love supports my wellbeing.

Contents

Abstract

Naive-Bayes classifiers are widely employed for classification tasks because of their efficiency and efficacy. Real-world classification tasks often involve quantitative attributes. In naive-Bayes learning, quantitative attributes are usually discretized. We investigate the working mechanism of discretization in naive-Bayes learning. We prove a theorem that states particular conditions under which discretization will result in naive-Bayes classifiers delivering the same probability estimates as would be obtained if the correct probability density function were employed. We then analyze the factors that might affect the classification error of naive-Bayes classifiers that are trained on data processed by discretization. We suggest that the use of different discretization techniques can affect the classification bias and variance of the generated classifiers. We name such effects *discretization bias* and *variance*. We argue that by properly managing discretization bias and variance, we can effectively reduce the naive-Bayes classification error. However, according to the comprehensive literature review that we have conducted, existing discretization methods have potential problems when applied in naive-Bayes learning. Thus we aim at developing new discretization techniques that are able to improve classification efficacy and efficiency of naive-Bayes classifiers. Our new methods are informed by an analysis from the new perspective of managing discretization bias and variance. In particular, we propose *proportional discretization*, *fixed frequency discretization*, *non-disjoint discretization* and *weighted proportional dis-*

cretization. To validate our theoretical arguments, we conduct experiments across a large suite of real-world datasets. We empirically evaluate our new techniques against five existing key discretization methods, each of which was either designed especially for naive-Bayes learning or is in practice often used with naive-Bayes learning. The experimental results support our analysis by showing that with significant frequency, naive-Bayes classifiers coupled with our new discretization methods are able to achieve lower classification error than those coupled with previous discretization methods. This outstanding effectiveness is achieved with very low computational time and space overhead, which is desirable since the classification efficiency is one of naive-Bayes classifiers' characteristics that largely contributes to their popularity, especially with time-sensitive interactive applications.

Except where otherwise indicated, this thesis is my own original work. It contains no material that has been accepted for the award of any other degree or diploma in any university or other institution.

Ying Yang

17 March 2003

Preface

Prior publications

Our early thinking on many issues that are presented in this thesis has appeared in various publications. Chapter 4 contains materials that have been published in the proceedings of *the Seventh Pacific Rim International Conference on Artificial Intelligence, Knowledge Acquisition Workshop* [Yang and Webb 2002a]. Chapter 5 and Chapter 6 contain materials that have been published in the proceedings of *the Twelfth European Conference on Machine Learning* [Yang and Webb 2001], *the Nineteenth International Conference on Machine Learning* [Yang and Webb 2002b] and *the Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining* [Yang and Webb 2003]. In addition, Chapter 3 contains materials that have been submitted for publication.

Acknowledgements

Three years ago, I said farewell to my homeland China and came to Australia pursuing my Ph.D. degree in computer science. I always remember my Australian colleagues' comments when they first saw me appearing in the school: 'You look like a lost child looking for your mother.' It is a real challenge to turn that image into a Ph.D. candidate. However, I am lucky enough to have so many nice people around me. Their love and support irrigate endless courage and energy into my life. Thus, three years later, I am ready to submit my thesis. Without expressing my gratitude to all those people, this thesis cannot be complete.

The first 'thank you' goes to my supervisor Professor Geoffrey I. Webb. I am grateful for his contributions to the development of the theory and techniques that I present in this thesis. On the first day I met him, Geoff asked me to read through a whole Machine Learning textbook and finish all the exercises in ten days. 'A very strict professor', I said to myself and imagined what a solemn face I should keep to be his student. However, I soon found out that Geoff is more than what I imagined. He is knowledgeable while modest, friendly while inspirational, encouraging while of course always strict. Not only does he teach me how to do research, but also he teaches me how to hold ethics of being a scientist. Not only does he teach me how to write papers in a good manner, but also he teaches me how to write my life in a meaningful way. He is always keen to introduce me and promote my work to important

people in my research area. I owe a lot to Geoff because I cannot be whom I am today without his supervision.

A second 'thank you' goes to my dear parents and my extraordinarily good-tempered husband, whose generous love has spoiled me. My father was a soldier. He told me that I should be brave to achieve my goals. My mother is a judge. She told me that I should be honest to be successful. My husband just obtained his Ph.D. degree in computer science. He told me his experience and blazed my trial to make everything easier for me. I can never forget those days when we sat side by side to write our theses, like two birds hovering shoulder to shoulder in the sky of science. Having their love and support, I feel confident to face any 'ordeal' in my life, and always keep smiling.

The following thanks go to the school of computer science and software engineering, Monash university; and the school of information technology, Deakin university. They have provided the financial support to me throughout my Ph.D. student career. They have constructed such a united and genuine research community to accommodate me. The edification that I have obtained therefrom can always benefit me in research and outside.

It is impossible to name all the people for whom I feel grateful. I can only extend a sincere 'thank you' to everyone. Thank you for appearing in my life and turning my Ph.D. student career into an experience to memorize.

Chapter 1

Introduction

This thesis tackles the problem of discretization within the context of naive-Bayes learning. Discretization produces a qualitative attribute from a quantitative attribute. Naive-Bayes classifiers can be trained on the resulting qualitative attributes instead of the original quantitative attributes. This circumvents the dilemma that for real-world data we usually do not know the probability distribution of the class within quantitative data, which however we need to know for naive-Bayes learning. Although there exist a number of discretization methods in the research area of machine learning, most of them were initially developed in learning contexts other than naive-Bayes learning. In this thesis, we argue that naive-Bayes learning has requirements of effective discretization different from those of most other learning contexts. Hence the existing methods are not appropriate for naive-Bayes learning. This shortage of appropriate discretization techniques is a serious problem that needs to be addressed due to the widespread employment of naive-Bayes classifiers. Consequently, we believe that there is a real and immediate need for improving discretization effectiveness for naive-Bayes learning. We prove a theorem that explains why discretization can be effective for naive-Bayes learning. Discretization can affect the classification bias and variance of the generated naive-Bayes

classifiers, effects we name *discretization bias* and *variance*. We believe that by properly managing discretization bias and variance, we can effectively reduce the naive-Bayes classification error. We offer insights into the impact of discretization bias and variance, and accordingly propose our new discretization techniques that are able to enhance naive-Bayes classification efficacy and efficiency. In this chapter, we first present the background of our research. Next, we explain our motivation. We then describe our contributions. Finally we provide the organization of this thesis.

1.1 Background

Naive-Bayes classifiers have a widespread employment for real-world classification applications. The quantitative attributes involved are usually discretized before the naive-Bayes classifiers are trained.

1.1.1 Naive-Bayes classifiers are widely employed

When classifying an instance, naive-Bayes classifiers assume the attributes conditionally independent of each other given the class; then apply Bayes' theorem to estimate the probability of each class given this instance. The class with the highest probability is chosen as the class of this instance. Naive-Bayes classifiers are simple, effective, efficient, robust and support incremental training. These merits have seen them employed in numerous classification tasks. Naive-Bayes classifiers have long been a core technique in information retrieval [Maron and Kuhns 1960; Maron 1961; Lewis 1992; Guthrie and Walker 1994; Lewis and Gale 1994; Kalt 1996; Larkey and Croft 1996; Pazzani, Mura-

matsu, and Billsus 1996; Starr, Ackerman, and Pazzani 1996a; Joachims 1997; Koller and Sahami 1997; Li and Yamanishi 1997; Mitchell 1997; Pazzani and Billsus 1997; Lewis 1998; McCallum and Nigam 1998; McCallum, Rosenfeld, Mitchell, and Ng 1998; Nigam, McCallum, Thrun, and Mitchell 1998; Frasconi, Soda, and Vullo 2001]. They were first introduced into machine learning as a straw man, against which new algorithms were compared and evaluated [Cestnik, Kononenko, and Bratko 1987; Clark and Niblett 1989; Cestnik 1990]. But it was soon realized that their classification accuracy was surprisingly high compared with other more sophisticated classification algorithms [Kononenko 1990; Langley, Iba, and Thompson 1992; Domingos and Pazzani 1996; Domingos and Pazzani 1997; Zhang, Ling, and Zhao 2000]. Thus they have often been chosen as the base algorithm for bagging, boosting, wrapper, voting or hybrid methodologies [Kohavi 1996; Zheng 1998; Bauer and Kohavi 1999; Ting and Zheng 1999; Gama 2000; Kim, Hahn, and Zhang 2000; Tsymbal, Puuronen, and Patterson 2002]. Also, naive-Bayes classifiers have widespread employment in medical diagnosis [Kononenko 1993; Kohavi, Sommerfield, and Dougherty 1997; Kukar, Groselj, Kononenko, and Fettich 1997; McSherry 1997a; McSherry 1997b; Zelic, Kononenko, Lavrac, and Vuga 1997; Montani, Bellazzi, Portinale, Fiocchi, and Stefanelli 1998; Lavrac 1998; Lavrac, Keravnou, and Zupan 2000; Kononenko 2001; Zupan, Demsar, Kattan, Ohori, Graefen, Bohanec, and Beck 2001], email filtering [Pantel and Lin 1998; Provost 1999; Androutsopoulos, Koutsias, Chandrinos, and Spyropoulos 2000; Rennie 2000; Crawford, Kay, and Eric 2002], and recommender systems [Starr, Ackerman, and Pazzani 1996b; Miyahara and Pazzani 2000; Mooney and Roy 2000].

1.1.2 Discretization is usually used in naive-Bayes learning

Naive-Bayes learning needs to estimate probabilities for each attribute-class pair. For a qualitative attribute, its relevant probabilities can be estimated from the corresponding frequencies. For a quantitative attribute, its relevant probabilities can be estimated if we know the probability distributions from which the quantitative values are drawn. Unfortunately however, those distributions are usually unknown for real-world data. Thus how to deal with quantitative attributes is a key problem in naive-Bayes learning. Typically, there are two approaches to tackling this problem.

The first approach is probability density estimation that makes assumptions about the probability density function of a quantitative attribute given a class. The relevant probabilities can then be estimated accordingly. For instance, a conventional approach is to assume that a quantitative attribute's probability within a class has a normal distribution. This assumption is made because a normal distribution may provide a reasonable approximation to many real-world distributions [John and Langley 1995], or because the normal distribution is perhaps the most well-studied probability distribution in statistics [Mitchell 1997].

A second approach is discretization. Under discretization, a qualitative attribute is created for a quantitative attribute. Each value of the qualitative attribute corresponds to an interval of values of the quantitative attribute. The resulting qualitative attributes are used instead of the original quantitative attributes to train a classifier. Since the probabilities of a qualitative attribute can be estimated from its frequencies, it is no longer necessary to assume any form

of distributions for the quantitative attributes.

For naive-Bayes learning, discretization is more popular than assuming probability density function. The main reason is that naive-Bayes classifiers with discretization tend to achieve lower classification error than those with unsafe probability density assumptions [Dougherty, Kohavi, and Sahami 1995].

1.2 Motivation

Although there exist a number of discretization methods in the research area of machine learning, most of them were initially developed in learning contexts other than naive-Bayes learning, such as decision trees, decision rules, decision tables, decision lists, association rules or Bayes network structures. We argue that naive-Bayes learning's requirements of effective discretization differ from those of most other learning contexts. Hence these methods do not suit naive-Bayes learning very well. Although there also exist a few discretization techniques that were originally developed in the learning context of naive-Bayes learning, we suggest that they have potential problems when used in naive-Bayes learning. Since naive-Bayes classifiers are widely employed for classification tasks, and since discretization has a major effect on the naive-Bayes classification error [Pazzani 1995], we believe that there is a real and immediate need for improving discretization effectiveness for naive-Bayes learning.

Furthermore, most existing discretization methods have only been tested on small datasets with hundreds of instances. Since large datasets with high

dimensional attribute spaces and huge numbers of instances are increasingly used in real-world applications, a study of these methods' effectiveness on large datasets is necessary and desirable [Freitas and Lavington 1996; Provost and Aronis 1996]. In fact, naive-Bayes classifiers' computational efficiency has resulted in their popularity with applications involving large datasets. Thus it is particularly important that discretization in naive-Bayes learning is efficient so as to scale to large data.

Motivated by these observations, our research is devoted to developing purpose-designed discretization methods for naive-Bayes classifiers. Our goals are to improve both naive-Bayes classification efficacy and efficiency. These dual goals are of particular significance given naive-Bayes classifiers' widespread employment, and in particular their employment in time-sensitive interactive applications.

1.3 Contributions

The following is a list of contributions to the research area of machine learning, which we present in this thesis.

1. We clarify the differences among the various terms used to define discretization, and choose the most appropriate definition to use in this thesis (Chapter 2).

2. We develop a comprehensive set of taxonomies for discretization methods (Chapter 2).

3. We explain the working mechanism of discretization in naive-Bayes

learning. We prove a theorem that accounts for why discretization can be effective (Chapter 3).

4. We analyze factors that might affect discretization effectiveness (Chapter 3).

5. We propose the concept of discretization bias and variance (Chapter 3).

6. We present a comprehensive literature review of exiting discretization techniques in the research area of machine learning (Chapter 4).

7. We argue that naive-Bayes learning has requirements of effective discretization different from those of most other learning contexts. Existing discretization methods do not appropriately suit naive-Bayes learning (Chapter 5).

8. We propose *proportional discretization* for naive-Bayes learning, a discretization technique that equally weighs discretization bias reduction and variance reduction; and decreases both discretization bias and variance with the training data size increasing (Chapter 5).

9. We propose *fixed frequency discretization* for naive-Bayes learning, a discretization technique that controls discretization variance by setting a sufficient interval frequency, and decreases discretization bias as additional training data become available (Chapter 5).

10. We propose *non-disjoint discretization* for naive-Bayes learning, a discretization technique that very efficiently forms overlapping discretized intervals for a quantitative attribute, and then chooses the most appro-

priate interval for the attribute value of the present test instance (Chapter 5).

11. We evaluate the effectiveness of studied discretization techniques on a wide range of large datasets to test whether they can effectively reduce the naive-Bayes classification error and can efficiently scale to large data (Chapter 6).

12. Inspired by the experimental results, we propose *weighted proportional discretization* for naive-Bayes learning, a discretization technique that combines the advantages of the above *proportional discretization* and *fixed frequency discretization* (Chapter 6).

1.4 Organization

The remaining chapters are organized as follows.

In Chapter 2, we explain some important concepts used throughout this thesis. In particular, we clarify the definition of discretization. That is, we investigate what type of attribute is transformed to what type of attribute by discretization. This is of particular importance since there exists confusion on this issue in existing literature. Also there exist various proposals for the taxonomies of discretization techniques. We integrate our new perspectives with those preceding ideas, presenting a comprehensive view on this issue.

In Chapter 3, we analyze discretization in naive-Bayes learning. This offers the theoretical foundation of this thesis. We define naive-Bayes classifiers. We explain discretization's working mechanism in naive-Bayes learning. We

aim at finding out why discretization can be affective. In particular, we prove a theorem that states particular conditions under which discretization will result in naive-Bayes classifiers delivering the same probability estimates as would be obtained if the correct probability density functions were employed. We then analyze the *decision boundary* and the *error tolerance of probability estimation*, two factors that might affect discretization effectiveness. We propose the concept of *discretization bias* and *variance*. We believe that by properly managing discretization bias and variance, discretization can effectively reduce the naive-Bayes classification error.

In Chapter 4, we conduct a literature review of 34 discretization methods in the area of machine learning. The review comprises two parts. The first part is a detailed review of 6 methods that were developed or are often employed in the context of naive-Bayes learning. In particular, we discuss each method's effectiveness in terms of discretization bias and variance, which we think illuminating. The second part is a brief review of further methods that were developed in contexts other than naive-Bayes learning.

In Chapter 5, guided by the theoretical analysis of Chapter 3 and the literature review of Chapter 4, we argue that existing discretization methods have potential problems with respect to naive-Bayes learning. Thus naive-Bayes classifiers call for more appropriate discretization techniques. Accordingly, we develop three new discretization techniques, *proportional discretization*, *fixed frequency discretization* and *non-disjoint discretization*. All of these techniques focus on managing discretization bias and variance, the characteristics that we have valued. We argue that our techniques are appropriate for naive-Bayes learning.

In Chapter 6, we present the empirical evaluation of our new techniques, compared with existing key discretization methods for naive-Bayes learning. We examine whether our new techniques can enhance both the efficacy and the efficiency of naive-Bayes learning. We describe our experimental data, design and statistics employed. We then analyze the experimental results. Inspired by our empirical observations and analysis, we further propose the fourth new discretization method, *weighted proportional discretization*.

In Chapter 7, we present the conclusion of this thesis. We summarize the key issues presented in this thesis. We discuss some future work that we think is worth further exploration. Finally we highlight the major contributions of this thesis to the research area of machine learning.

Chapter 2

Terminology and taxonomy

The previous chapter has introduced the focus of this thesis, namely discretization in the context of naive-Bayes learning. In this chapter, we address some important concepts that are involved in naive-Bayes learning and discretization. These concepts will be frequently referred to throughout this thesis. First, we explain the terms used in classification learning. Second, we clarify the definition of discretization by differentiating diverse terminologies presented in the existing literature. Third, we present a comprehensive set of taxonomies of discretization methods, integrating various previous proposals and our new perspectives.

2.1 Terminology of classification learning

Naive-Bayes learning is a form of classification learning. In classification learning, each *instance* is described by a vector of *attribute* values and its *class* can take any value from some predefined set of values. An instance with its class known is called a *training instance* (also known as a labelled instance). An instance with its class unknown is called a *test instance* (also known as an unlabelled instance). A set of training instances, so-called the *training data*, are

provided. A test instance is presented. The learner is asked to predict the test instance's class according to the evidence provided by the training data.

2.2 Terminology of discretization

Discretization is a data processing procedure. It transforms an attribute from one type into another type. In the large amount of existing literature that address discretization, there is considerable variation in the terminology used to describe these two data types, including 'quantitative' *vs.* 'qualitative', 'continuous' *vs.* 'discrete', 'ordinal' *vs.* 'nominal', and 'numeric' *vs.* 'categorical'. We feel it necessary to make clear the difference among the various terms and accordingly choose the most suitable terminology for use in our thesis.

Turning to the authority of introductory statistical textbooks, [Bluman 1992; Samuels and Witmer 1999], there are two parallel ways to classify data into different types. Data can be classified into either *qualitative* or *quantitative*. Data can also be classified into different *levels of measurement scales*. Sections 2.2.1 and 2.2.2 summarize relevant materials from these textbooks.

2.2.1 Qualitative *vs.* quantitative

Attributes can be classified as either qualitative or quantitative. **Qualitative attributes**, also often referred to as **categorical attributes**, are attributes that can be placed into distinct categories, according to some characteristics. Qualitative attributes sometimes can be arrayed in a meaningful rank order. But no arithmetic operations can be applied to them. Examples of qualitative attributes are:

- blood type of a person: A, B, AB, O;
- sex of a fish: male, female;
- student evaluation: fail, pass, good, excellent;
- tenderness of beef: very tender, tender, slightly tough, tough.

Quantitative attributes are numerical in nature. They can be ranked in order. They can also have meaningful arithmetic operations. Quantitative attributes can be further classified into two groups, discrete or continuous.

A **discrete attribute** assumes values that can be counted. The attribute cannot assume all values on the number line within its value range. Examples of discrete attributes are:

- number of children in a family;
- number of bacteria colonies in a petri fish.

A **continuous attribute** can assume all values on the number line within the value range. The values are obtained by measuring. Examples of continuous attributes are:

- temperature;
- weight of a baby.

2.2.2 Levels of measurement scales

In addition to being classified as either qualitative or quantitative, attributes can also be classified by how they are categorized, counted or measured. This

type of classification uses **measurement scales**, and four common types of scales are used: nominal, ordinal, interval and ratio.

The **nominal** level of measurement scales classifies data into mutually exclusive (non-overlapping), exhaustive categories in which no order or ranking can be imposed on the data. Examples of nominal attributes are:

- blood type of a person: A, B, AB, O;
- sex of a fish: male, female.

The **ordinal** level of measurement scales classifies data into categories that can be ranked. However, the differences between the ranks cannot be calculated by arithmetic. Examples of ordinal attributes are:

- student evaluation: fail, pass, good, excellent;
- tenderness of beef: very tender, tender, slightly tough, tough.

It is meaningful to say that the student evaluation of pass ranks higher than that of fail. It is not meaningful in the same way to say that the blood type of A ranks higher than that of B.

The **interval** level of measurement scales ranks data, and the differences between units of measure can be calculated by arithmetic. However, *zero* in the interval level of measurement does not mean 'nil' or 'nothing' as *zero* in arithmetic means. Examples of interval attributes are:

- IQ, whose values are yielded by a standardized psychological test. There is a meaningful difference of one point between an IQ of 109 and an IQ of 110. But IQ tests do not measure people who have no intelligence;

- Fahrenheit temperature, there is a meaningful difference of one degree between each unit, such as 72 degrees and 73 degrees. But 0 degrees Fahrenheit does not mean no heat.

It is meaningful to say that the IQ of person A is two points higher than that of person B. It is not meaningful in the same way to say that the tenderness of piece of beef A is two points higher than the tenderness of piece B.

The **ratio** level of measurement scales possesses all the characteristics of interval measurement, and there exists a *zero* that, the same as arithmetic *zero*, means 'nil' or 'nothing'. In consequence, true ratios exist between different units of measure. Examples of ratio attributes are:

- number of children in a family;
- weight of a baby.

It is meaningful to say that the weight of child A is twice that of child B. It is not meaningful in the same way to say that the IQ of person A is twice that of person B.

The nominal level is the lowest level of measurement scales. It is the least powerful in terms of including data information. The ordinal level is higher. The interval level is even higher. The ratio level is the highest level. Any data conversion from a higher level of measurement scales to a lower level of measurement scales will lose information. Table 2.1 gives a summary of the characteristics of different levels of measurement scales.

2.2.3 Terminology employed in this thesis

In summary, the following taxonomy applies to attribute types:

Level	Ranking ?	Arithmetic operation ?	Arithmetic zero ?
Nominal	no	no	no
Ordinal	yes	no	no
Interval	yes	yes	no
Ratio	yes	yes	yes

Table 2.1: Measurement Scales

1. qualitative attributes:
   (a) nominal;
   (b) ordinal;
2. quantitative attributes:
   (a) interval, either discrete or continuous;
   (b) ratio, either discrete or continuous.

We believe that 'discretization' as it is usually applied in machine learning is best defined as the conversion of *quantitative* attributes to *qualitative* attributes. In consequence, we will refer to attributes as either quantitative or qualitative throughout this thesis.

Another term often used for describing discretization is 'cut point'. When discretizing a quantitative attribute, a cut point is a value of the attribute where an interval boundary is located by a discretization method.

2.3 Taxonomy of discretization methods

There exist diverse taxonomies in existing literature to classify discretization methods. Different taxonomies emphasize different aspects of the distinctions

among discretization methods.

Typically, discretization methods can be classified into either *primary* or *composite*. Primary methods accomplish discretization without reference to any other discretization method. Composite methods are built on top of a primary method.

Primary methods can be classified as per the following taxonomies.

1. **Supervised** *vs.* **Unsupervised** [Dougherty, Kohavi, and Sahami 1995]. Methods that use the class information of the training instances to select discretization cut points are supervised. Methods that do not use the class information are unsupervised. Supervised discretization can be further characterized as *error-based, entropy-based* or *statistics-based* according to whether intervals are selected using metrics based on error on the training data, entropy of the intervals, or some statistical measure.

2. **Univariate** *vs.* **Multivariate** [Bay 2000]. Methods that discretize each attribute in isolation are univariate. Methods that take into consideration relationships among attributes during discretization are multivariate.

3. **Parametric** *vs.* **Non-parametric**. Parametric discretization requires input from the user, such as the maximum number of discretized intervals. Non-parametric discretization only uses information from data and does not need input from the user.

4. **Hierarchical** *vs.* **Non-hierarchical**. Hierarchical discretization selects cut points in an incremental process, forming an implicit hierarchy over the value range. The procedure can be *split* or (and) *merge* [Kerber 1992].

Split discretization initially has the whole value range as an interval, then continues splitting it into sub-intervals until some threshold is met. Merge discretization initially puts each value into an interval, then continues merging adjacent intervals until some threshold is met. Some discretization methods utilize both split and merge processes. For example, intervals are initially formed by splitting, and then a merge process is performed to post-process the formed intervals. Non-hierarchical discretization does not form any hierarchy during discretization. For example, many methods scan the ordered values only once, sequentially forming the intervals.

5. **Global** *vs.* **Local** [Dougherty, Kohavi, and Sahami 1995]. Global methods discretize with respect to the whole training data space. They perform discretization once only, using a single set of intervals throughout a single classification task. Local methods allow different sets of intervals to be formed for a single attribute, each set being applied in a different classification context. For example, different discretizations of a single attribute might be applied at different nodes of a decision tree [Quinlan 1993].

6. **Eager** *vs.* **Lazy** [Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003]. Eager methods perform discretization *prior* to classification time. Lazy methods perform discretization during the classification time.

7. **Disjoint** *vs.* **Non-disjoint**. Disjoint methods discretize the value range of the attribute under discretization into disjoint intervals. No intervals overlap. Non-disjoint methods discretize the value range into intervals

that can overlap.

Composite methods first choose some primary discretization method to form the initial cut points. They then focus on how to adjust these initial cut points to achieve certain goals. The taxonomy of a composite method sometimes is flexible, depending on the taxonomy of its primary method.

To the best of our knowledge, we are the first to propose the taxonomies 'primary' *vs.* 'composite', 'parametric' *vs.* 'non-parametric', 'hierarchical' *vs.* 'non-hierarchical' and 'disjoint' *vs.* 'non-disjoint'.

2.4 Summary

In this chapter, we have explained the concepts that we will use throughout this thesis. For naive-Bayes learning, we have explained key terms including 'instance', 'attribute', 'class', 'training data' and 'test data'. For discretization, we have clarified the difference among different types of data and accordingly chosen to define discretization as transforming 'quantitative' attributes into 'qualitative' attributes. We have also presented a comprehensive set of taxonomies of discretization methods that integrates our new perspectives and previous proposals. We think our work is of particular necessity, since there is considerable confusion regarding the terminology and taxonomy of discretization in the existing literature.

In the next chapter, we will analyze discretization in naive-Bayes learning.

Chapter 3

Theoretical analysis of discretization in naive-Bayes learning

The previous chapter has explained the terms involved in this thesis. These terms will be frequently referred to in this chapter, which analyzes discretization in naive-Bayes learning and thus offers the theoretical foundation of this thesis. In this chapter, we first define naive-Bayes classifiers and describe their characteristics. Next, we address the working mechanism of discretization in naive-Bayes learning. We prove a theorem that provides accounts for why discretization can be effective. We then analyze factors that might affect the effectiveness of discretization. We suggest that discretization can affect the classification bias and variance of the generated naive-Bayes classifiers, effects we name discretization bias and variance. We believe that by properly managing discretization bias and variance, we can effectively reduce the naive-Bayes classification error. In particular, we offer insights into managing discretization bias and variance by tuning interval frequency and interval number formed by discretization.

3.1 Naive-Bayes classifiers

In naive-Bayes learning, we define:

- C as a random variable denoting the class of an instance,
- $\mathbf{X} < X_1, X_2, \cdots, X_k >$ as a vector of random variables denoting the observed attribute values (an instance),
- c as a particular class label,
- $\mathbf{x} < x_1, x_2, \cdots, x_k >$ as a particular observed attribute value vector (a particular instance),
- $\mathbf{X}{=}\mathbf{x}$ as shorthand for $X_1{=}x_1 \wedge X_2{=}x_2 \wedge \cdots \wedge X_k{=}x_k$.

Suppose a test instance $\mathbf{x}$ is presented. The learner is asked to predict its class according to the evidence provided by the training data. Expected classification error can be minimized by choosing $argmax_c(p(C{=}c \,|\, \mathbf{X}{=}\mathbf{x}))$ for each $\mathbf{x}$ [Duda and Hart 1973; Domingos and Pazzani 1997]. We start with Bayes' theorem:

$$p(C{=}c \,|\, \mathbf{X}{=}\mathbf{x}) = \frac{p(C{=}c)p(\mathbf{X}{=}\mathbf{x} \,|\, C{=}c)}{p(\mathbf{X}{=}\mathbf{x})}. \tag{3.1}$$

Since the denominator in (3.1) is invariant across classes, it does not affect the final choice and can be dropped:

$$p(C{=}c \,|\, \mathbf{X}{=}\mathbf{x}) \propto p(C{=}c)p(\mathbf{X}{=}\mathbf{x} \,|\, C{=}c). \tag{3.2}$$

Probabilities $p(C{=}c)$ and $p(\mathbf{X}{=}\mathbf{x} \,|\, C{=}c)$ need to be estimated from the training data. Unfortunately, since $\mathbf{x}$ is usually an unseen instance which does not appear in the training data, it may not be possible to directly estimate

$p(\mathbf{X}{=}\mathbf{x}|C{=}c)$. So a simplification is made: if attributes $X_1, X_2, \cdots, X_k$ are conditionally independent of each other given the class, then:

$$\begin{aligned} p(\mathbf{X}{=}\mathbf{x}|C{=}c) &= p(\wedge_{i=1}^{k} X_i{=}x_i|C{=}c) \\ &= \prod_{i=1}^{k} p(X_i{=}x_i|C{=}c). \end{aligned} \tag{3.3}$$

Combining (3.2) and (3.3), one can further estimate the most probable class by using:

$$p(C{=}c|\mathbf{X}{=}\mathbf{x}) \propto p(C{=}c)\prod_{i=1}^{k} p(X_i{=}x_i|C{=}c). \tag{3.4}$$

Classifiers using (3.4) are *naive-Bayes classifiers*. The assumption embodied in (3.3) is the *attribute independence assumption*. The probability $p(C{=}c|\mathbf{X}{=}\mathbf{x})$ denotes the conditional probability of a class c given an instance $\mathbf{x}$. The probability $p(C{=}c)$ denotes the prior probability of a particular class c. The probability $p(X_i{=}x_i|C{=}c)$ denotes the conditional probability that an attribute X_i takes a particular value x_i given the class c.

In naive-Bayes learning, the class C is qualitative, and an attribute X_i can be either qualitative or quantitative. Since quantitative data have characteristics different from qualitative data [Bluman 1992; Samuels and Witmer 1999], the practice of estimating probabilities in (3.4) when involving qualitative data is different from that when involving quantitative data.

3.1.1 Calculating frequency for qualitative data

The class, as well as a qualitative attribute, usually takes a small number of values [Bluman 1992; Samuels and Witmer 1999]. Thus there are usually

many instances of each value in the training data. The probability $p(C{=}c)$ can be estimated from the frequency of instances with $C{=}c$. The probability $p(X_i{=}x_i \mid C{=}c)$, when X_i is qualitative, can be estimated from the frequency of instances with $C{=}c$ and the frequency of instances with $X_i{=}x_i \wedge C{=}c$. These estimates are strong consistent estimates according to the strong law of large numbers [Casella and Berger 1990; John and Langley 1995].

In practice, a typical approach to estimating $p(C{=}c)$ is to use the Laplace-estimate [Cestnik 1990]: $\frac{n_c+k}{N+n\times k}$, where n_c is the number of instances satisfying $C{=}c$, N is the number of training instances, n is the number of classes, and k equals 1. A typical approach to estimating $p(X_i{=}x_i \mid C{=}c)$ is to use the M-estimate [Cestnik 1990]: $\frac{n_{ci}+m\times p}{n_c+m}$, where n_{ci} is the number of instances satisfying $X_i{=}x_i \wedge C{=}c$, n_c is the number of instances satisfying $C{=}c$, p is $p(X_i{=}x_i)$ (estimated by the Laplace-estimate), and m is a constant that is set as 2 as in Cestnik's study [1990].

3.1.2 Probability density estimation for quantitative data

When it is quantitative, X_i often has a large or even an infinite number of values [Bluman 1992; Samuels and Witmer 1999]. Thus the probability of a particular value x_i given the class c, $p(X_i{=}x_i \mid C{=}c)$ can be infinitely small. Accordingly, there usually are very few training instances for any one value. Hence it is unlikely that reliable estimation of $p(X_i{=}x_i \mid C{=}c)$ can be derived from the observed frequency. Consequently, in contrast to qualitative attributes, probability density estimation models each quantitative attribute by some continuous probability distribution over the range of its values [John and Langley

1995]. Hence $p(X_i{=}x_i \,|\, C{=}c)$ is completely determined by a probability density function f, which satisfies:

1. $f(X_i{=}x_i \,|\, C{=}c) \geq 0, \forall x_i \in S_i$;
2. $\int_{S_i} f(X_i \,|\, C{=}c) \mathrm{d}X_i = 1$;
3. $\int_{a_i}^{b_i} f(X_i \,|\, C{=}c) \mathrm{d}X_i = p(a_i \leq X_i \leq b_i \,|\, C{=}c), \forall [a_i, b_i] \in S_i$;

where S_i is the value space of X_i [Scheaffer and McClave 1995].

When involving quantitative attributes, naive-Bayes classifiers can manipulate $f(X_i{=}x_i \,|\, C{=}c)$ instead of $p(X_i{=}x_i \,|\, C{=}c)$. According to John and Langley [1995], supposing X_i lying within some interval $[x_i, x_i + \Delta]$, we have $p(x_i \leq X_i \leq x_i + \Delta \,|\, C{=}c) = \int_{x_i}^{x_i+\Delta} f(X_i \,|\, C{=}c) \mathrm{d}X_i$. By the definition of a derivative, $\lim_{\Delta \to 0} \frac{p(x_i \leq X_i \leq x_i + \Delta \,|\, C{=}c)}{\Delta} = f(X_i{=}x_i \,|\, C{=}c)$. Thus for very small constant Δ, $p(X_i{=}x_i \,|\, C{=}c) \approx p(x_i \leq X_i \leq x_i + \Delta \,|\, C{=}c) \approx f(X_i{=}x_i \,|\, C{=}c) \times \Delta$. The factor Δ then appears in the numerator of (3.4) for each class. They cancel out when normalization is performed. Thus

$$p(X_i{=}x_i \,|\, C{=}c) \widetilde{\propto} f(X_i{=}x_i \,|\, C{=}c); \tag{3.5}$$

$$\text{and} \quad p(C{=}c \,|\, \mathbf{X}{=}\mathbf{x}) \widetilde{\propto} p(C{=}c) \prod_{i=1}^{k} f(X_i{=}x_i \,|\, C{=}c). \tag{3.6}$$

The density function f gives a description of the distribution of X_i within the class c, and allows probabilities associated with $X_i \mid C{=}c$ to be found [Silverman 1986]. Unfortunately however, f is usually unknown for real-world data. In consequence, *probability density estimation* is used to construct $\hat{f}$, an estimate of f from the training data.

A conventional approach to constructing $\hat{f}$ is to assume that the values of X_i within the class c are drawn from a normal (Gaussian) distribution [Dougherty, Kohavi, and Sahami 1995; Mitchell 1997]. Thus

$$\hat{f} = N(X_i; \mu_c, \sigma_c) = \frac{1}{\sqrt{2\pi}\sigma_c} e^{-\frac{(X_i-\mu_c)^2}{2\sigma_c^2}},$$

where μ_c is the *mean* and σ_c is the *standard deviation* of the attribute values from the training instances whose class equals c. In this case, training involves learning the parameters μ_c and σ_c from the training data. The normal distribution assumption is made because it may provide a reasonable approximation to many real-world distributions [John and Langley 1995], or because it is perhaps the most well-studied probability distribution in statistics [Mitchell 1997]. This approach is *parametric*, that is, it assumes that the data are drawn from one of a known parametric family of distributions [Silverman 1986]. The major problem of this method is that when the attribute data do not follow a normal distribution, which is often the case in real-world data, the probability estimation of naive-Bayes classifiers is not reliable and thus can lead to inferior classification performance [Dougherty, Kohavi, and Sahami 1995; Pazzani 1995].

A second approach is *less parametric* in that it does not constrain $\hat{f}$ to fall in a given parametric family. Thus less rigid assumptions are made about the distribution of the observed data [Silverman 1986]. A typical approach is kernel density estimation [John and Langley 1995]. $\hat{f}$ is averaged over a large

set of normal (Gaussian) kernels,

$$\hat{f} = \frac{1}{n_c} \sum_j N(X_i; \mu_{ij}, \sigma_c),$$

where n_c is the total number of training instances with class c, μ_{ij} is the jth value of X_i within class c, and $\sigma_c = \frac{1}{\sqrt{n_c}}$. It has been demonstrated that kernel density estimation results in higher naive-Bayes classification accuracy than the former method in domains that violate the normal distribution assumption, and causes only small decreases in accuracy in domains where the assumption holds. However, this approach tends to incur high computational memory and time. Whereas the former method can estimate μ_c and σ_c by storing only the sum of the observed x_i and the sum of their squares, this approach must store every x_i. Whereas the former method only has to calculate $N(X_i)$ once for each $X_i = x_i | C = c$, this approach must perform this calculation n_c times. Thus it has a potential problem that undermines the efficiency of naive-Bayes learning.

3.1.3 Merits of naive-Bayes classifiers

Naive-Bayes classifiers are simple and efficient. They need only to collect information about individual attributes, which contrasts to most learning systems that must consider attribute combinations. Thus naive-Bayes' computational time complexity is only linear with respect to the size of the training data. This is much more efficient than the exponential complexity of non-naive Bayes approaches [Yang and Liu 1999; Zhang, Ling, and Zhao 2000]. They are also space efficient. They do not require the training data be retained in mem-

ory during classification and can record all required information using only tables of no more than two dimensions.

Naive-Bayes classifiers are effective. They are optimal methods of supervised learning if the independence assumption holds and the estimates of the required probabilities are accurate [Duda and Hart 1973]. Even when the independence assumption is violated, their classification performance is still surprisingly good compared with other more sophisticated classifiers. One reason for this is because that the classification estimation under zero-one loss is only a function of the sign of the probability estimation. In consequence, the classification accuracy can remain high even while the probability estimation is poor [Domingos and Pazzani 1997].

Naive-Bayes classifiers are robust to noisy data such as irrelevant attributes. They take all attributes into account simultaneously. Hence, the impact of a misleading attribute can be absorbed by other attributes under zero-one loss [Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003].

Naive-Bayes classifiers support incremental training [Rennie 2000; Roy and McCallum 2001; Zaffalon and Hutter 2002]. One can map an existing naive-Bayes classifier and a new training instance to a new naive-Bayes classifier which is identical to the classifier that would have been learned from the original data augmented by the new instance. Thus it is trivial to update a naive-Bayes classifier whenever a new training instance becomes available. This contrasts to the non-incremental methods which must build a new classifier from scratch in order to utilize new training data. The cost of incremental update is far lower than that of retraining. Consequently, naive-Bayes classifiers are particularly attractive for classification tasks where the training infor-

mation updates frequently.

These merits have led to naive-Bayes learning's widespread employment for real-world classification applications.

3.2 How discretization works

Discretization provides an alternative to probability density estimation for naive-Bayes learning. Under probability density estimation, if the assumed density is not a proper estimate of the true density, the naive-Bayes classification performance tends to degrade [Dougherty, Kohavi, and Sahami 1995; John and Langley 1995]. Since the true density is usually unknown for real-world data, unsafe assumptions unfortunately often occur. Discretization can circumvent this problem. Under discretization, a qualitative attribute X_i^* is formed for X_i. Each value x_i^* of X_i^* corresponds to an interval $(a_i, b_i]$ of X_i. Any original quantitative value $x_i \in (a_i, b_i]$ is replaced by x_i^*. All relevant probabilities are estimated with respect to x_i^*. Since probabilities of X_i^* can be properly estimated from frequencies as long as there are enough training instances, there is no need to assume the probability density function any more. However, because qualitative data have a lower level of measurement scales than quantitative data as we have addressed in Chapter 2, discretization can suffer information loss.

3.2.1 Why discretization can be effective

Dougherty, Kohavi, and Sahami [1995] conducted an empirical study to show that naive-Bayes classifiers resulting from discretization achieved

lower classification error than those resulting from unsafe probability density assumptions. With this empirical support, Dougherty et al. suggested that discretization could be effective because they did not make any assumption about the form of the probability distribution from which the quantitative attribute values were drawn. Hsu, Huang, and Wong [2000, 2003] further analyzed this issue from a theoretical base. Specifically, they assumed a discretized attribute given a class $X_i^* | C=c$ to have a multinomial distribution with parameters n_c and $p_1, p_2, \cdots, p_v$, where n_c is the number of the training instances with class c, v is the number of discretized values of X_i^*, x_{ij}^* is the jth value of X_i^*, and $p_j = p(X_i^*=x_{ij}^* | C=c)$ for $j=1,2,\cdots,v$. The parameters $p_1, p_2, \cdots, p_v$ have a Dirichlet distribution which conjugates to the multinomial distribution. 'Perfect aggregation' of a Dirichlet distribution implies that one can estimate $p(X_i^*=x_i^* | C=c)$ with arbitrary accuracy, independent of the shape of the curve of the density function $f(X_i | C=c)$ in the interval $(a_i, b_i]$ to which x_i^* corresponds. They suggested that discretization would achieve optimal effectiveness by forming x_i^* for x_i such that $p(X_i^*=x_i^* | C=c)$ simulates the role of $f(X_i=x_i | C=c)$ by distinguishing the class that gives x_i high density from the class that gives x_i low density. However, they did not supply any insight into why there exist differences in the effectiveness among different discretization methods. Instead, they suggested that 'well-known' discretization methods, such as equal width discretization [Catlett 1991; Kerber 1992; Dougherty, Kohavi, and Sahami 1995] and entropy minimization discretization [Fayyad and Irani 1993], were unlikely to degrade naive-Bayes classification performance. In contrast, we do not believe in this *unconditional* excellence. Rather, we believe that discretization for naive-Bayes learning should focus

on the accuracy of $p(C{=}c|X_i^*{=}x_i^*)$ as an estimate of $p(C{=}c|X_i{=}x_i)$. As we will prove in Theorem 1, as long as $p(C{=}c|X_i^*{=}x_i^*)$ is an accurate estimate of $p(C{=}c|X_i{=}x_i)$, discretization can be effective to the degree that the probability estimates are the same as would be obtained if the correct probability density functions were employed.

Theorem 1 *Assume the first l of k attributes are quantitative and the remaining attributes are qualitative*[1]. *Suppose instance* $\mathbf{X}^*$ *is the discretized version of instance* $\mathbf{X}$, *resulting from substituting qualitative attribute* X_i^* *for quantitative attribute* X_i $(1 \leq i \leq l)$. *If* $\forall_{i=1}^{l}(p(C{=}c|X_i{=}x_i) = p(C{=}c|X_i^*{=}x_i^*))$, *and the naive-Bayes attribute independence assumption (3.3) holds, we have* $p(C{=}c|\mathbf{X}{=}\mathbf{x}) \propto p(C{=}c|\mathbf{X}^*{=}\mathbf{x}^*)$.

Proof:

According to Bayes theorem, we have:

$$\begin{aligned} & p(C{=}c|\mathbf{X}{=}\mathbf{x}) \\ = \; & p(C{=}c)\frac{p(\mathbf{X}{=}\mathbf{x}|C{=}c)}{p(\mathbf{X}{=}\mathbf{x})}; \end{aligned}$$

since the naive-Bayes attribute independence assumption (3.3) holds, we continue:

$$= \frac{p(C{=}c)}{p(\mathbf{X}{=}\mathbf{x})}\prod_{i=1}^{k} p(X_i{=}x_i|C{=}c);$$

[1] *In naive-Bayes learning, the order of the attributes does not matter. We make this assumption only to simplify the expression of our proof. This does not affect the theoretical analysis at all.*

using Bayes theorem:

$$= \frac{p(C{=}c)}{p(\mathbf{X}{=}\mathbf{x})}\prod_{i=1}^{k}\frac{p(X_i{=}x_i)p(C{=}c|X_i{=}x_i)}{p(C{=}c)}$$
$$= \frac{p(C{=}c)}{p(C{=}c)^k}\frac{\prod_{i=1}^{k}p(X_i{=}x_i)}{p(\mathbf{X}{=}\mathbf{x})}\prod_{i=1}^{k}p(C{=}c|X_i{=}x_i);$$

since the factor $\frac{\prod_{i=1}^{k}p(X_i=x_i)}{p(\mathbf{X}=\mathbf{x})}$ is invariant across classes:

$$\propto p(C{=}c)^{1-k}\prod_{i=1}^{k}p(C{=}c|X_i{=}x_i)$$
$$= p(C{=}c)^{1-k}\prod_{i=1}^{l}p(C{=}c|X_i{=}x_i)\prod_{j=l+1}^{k}p(C{=}c|X_j{=}x_j);$$

since $\forall_{i=1}^{l}(p(C{=}c|X_i{=}x_i) = p(C{=}c|X_i^*{=}x_i^*))$:

$$= p(C{=}c)^{1-k}\prod_{i=1}^{l}p(C{=}c|\mathbf{X}^*{}_i{=}\mathbf{x}^*{}_i)\prod_{j=l+1}^{k}p(C{=}c|X_j{=}x_j);$$

using Bayes theorem again:

$$= p(C{=}c)^{1-k}\prod_{i=1}^{l}\frac{p(C{=}c)p(X_i^*{=}x_i^*|C{=}c)}{p(X_i^*{=}x_i^*)}\prod_{j=l+1}^{k}\frac{p(C{=}c)p(X_j{=}x_j|C{=}c)}{p(X_j{=}x_j)}$$
$$= p(C{=}c)\frac{\prod_{i=1}^{l}p(X_i^*{=}x_i^*|C{=}c)\prod_{j=l+1}^{k}p(X_j{=}x_j|C{=}c)}{\prod_{i=1}^{l}p(X_i^*{=}x_i^*)\prod_{j=l+1}^{k}p(X_j{=}x_j)};$$

since the denominator $\prod_{i=1}^{l}p(X_i^*{=}x_i^*)\prod_{j=l+1}^{k}p(X_j{=}x_j)$ is invariant across classes:

$$\propto p(C{=}c)\prod_{i=1}^{l}p(X_i^*{=}x_i^*|C{=}c)\prod_{j=l+1}^{k}p(X_j{=}x_j|C{=}c);$$

since the naive-Bayes attribute independence assumption (3.3) holds:

$$= p(C{=}c)p(\mathbf{X}^*{=}\mathbf{x}^* \,|\, C{=}c)$$

$$= p(C{=}c \,|\, \mathbf{X}^*{=}\mathbf{x}^*)p(\mathbf{X}^*{=}\mathbf{x}^*);$$

since $p(\mathbf{X}^*{=}\mathbf{x}^*)$ is invariant across classes:

$$\propto p(C{=}c \,|\, \mathbf{X}^*{=}\mathbf{x}^*). \quad \square$$

Theorem 1 assures us that in naive-Bayes learning, if the attribute independence assumption holds, and if for each quantitative attribute X_i, discretization can form a qualitative X_i^* such that $p(C{=}c \,|\, X_i^*{=}x_i^*)$ is a reasonable approximation of $p(C{=}c \,|\, X_i{=}x_i)$, we can expect that $p(C{=}c \,|\, \mathbf{X}^*{=}\mathbf{x}^*)$ is a reasonable approximation of $p(C{=}c \,|\, \mathbf{X}{=}\mathbf{x})$. Since $p(C{=}c \,|\, X_i^*{=}x_i^*)$ is estimated as for qualitative attributes, it allows naive-Bayes learning not to assume any form of the probability density of the quantitative data.

This analysis of discretization that focuses on $p(C{=}c \,|\, X_i{=}x_i)$ instead of $f(X_i{=}x_i \,|\, C{=}c)$ is derived from Kononenko's analysis [1992]. However, Kononenko's analysis requires that the attributes be assumed *unconditionally* independent of each other, which entitles $\prod_{i=1}^{k} p(X_i{=}x_i) = p(\mathbf{X}{=}\mathbf{x})$. This assumption is much stronger than the naive-Bayes attribute independence assumption embodied in (3.3). Thus we suggest that our deduction in Theorem 1 more accurately captures the mechanism by which discretization works.

3.3 What affects discretization effectiveness

We have proved that the accuracy of the probability $p(C=c|X_i^*=x_i^*)$ as an estimate of $p(C=c|X_i=x_i)$ for each class c plays a key role on discretization's effectiveness. Two important factors that relate to this accuracy are the *decision boundary* and the *error tolerance of probability estimation*. Different discretization methods have different ways to deal with these two factors, and thus have different effects on the classification bias and variance of the generated classifiers. We name these effects *discretization bias* and *variance*. We suggest that discretization methods that can well manage discretization bias and variance are of great utility. According to (3.4), besides $p(C=c|X_i=x_i)$, the prior probability of each class $p(C=c)$ also affects the final choice. To simplify our analysis, we here assume that each class has the same prior probability. That is, $p(C=c)$ is identical for each c. Thus we can cancel out the effect of $p(C=c)$. However, our analysis extends straightforwardly to non-uniform cases.

In addition, two important terms throughout our analysis are *interval frequency* and *interval number*. Interval frequency is the number of training instances in an interval formed by discretization. Interval number is the total number of intervals formed by discretization.

3.3.1 Classification bias and variance

When we talk about the effectiveness of a discretization method, we in fact mean the performance of naive-Bayes classifiers that are trained on data discretized by this discretization method. Thus it is essential to explain how the performance of naive-Bayes classifiers is measured.

The performance of a classifier is usually measured by its classification *error*. The error can be partitioned into a *bias* term, a *variance* term and an *irreducible* term [Kong and Dietterich 1995; Breiman 1996; Kohavi and Wolpert 1996; Friedman 1997; Webb 2000]. Bias describes the component of error that results from systematic error of the learning algorithm. Variance describes the component of error that results from random variation in the training data and from random behavior in the learning algorithm, and thus measures how sensitive an algorithm is to changes in the training data. As the algorithm becomes more sensitive, the variance increases. Irreducible error describes the error of an optimal algorithm (the level of noise in the data), which is usually aggregated with the bias term or (and) the variance term.

Consider a classification learning algorithm A applied to a set S of training instances to produce a classifier to classify an instance $\mathbf{x}$. Suppose we could draw a sequence of training sets $S_1, S_2, ..., S_l$, each of size m, and apply A to construct classifiers. The error of A at $\mathbf{x}$ with respect to this sequence of training sets of size m can be defined as:

$$Error(A,m,\mathbf{x}) = Bias(A,m,\mathbf{x}) + Variance(A,m,\mathbf{x}) + Irreducible(A,m,\mathbf{x}).$$

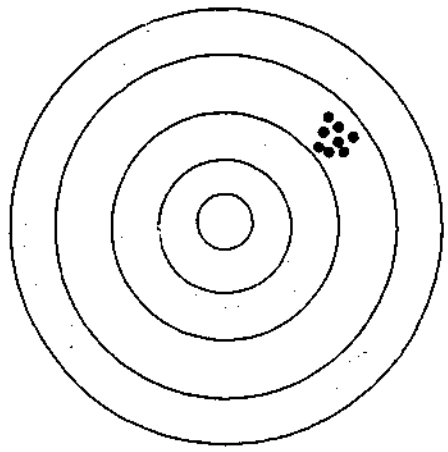
There is often a 'bias and variance trade-off' [Kohavi and Wolpert 1996]. All other things being equal, as one modifies some aspect of the learning algorithm, it will have opposite effects on bias and variance. For example, usually as one increases the number of degrees of freedom in the algorithm, the bias decreases but the variance increases. The optimal number of degrees of freedom (as far as the expected loss is concerned) is the number that optimizes

this trade-off between bias and variance.

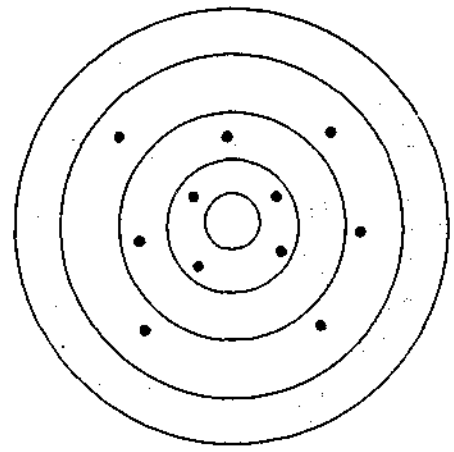
Moore and McCabe [2002] illustrated bias and variance through the analogy of shooting arrows at a target, as reproduced in Figure 3.1. We can think of the perfect model as the bull's-eye on a target, and the algorithm learning from some set of training data as an arrow fired at the bull's-eye. Bias and variance describe what happens when an archer fires many arrows at the target. Bias means that the aim is off and the arrows land consistently off the bull's-eye in the same direction. The learned model does not center about the perfect model. Variance means that repeated shots are widely scattered on the target. They do not give similar results but differ widely among themselves. A good learning scheme, like a good archer, must have both low bias and low variance.

3.3.2 Decision boundary

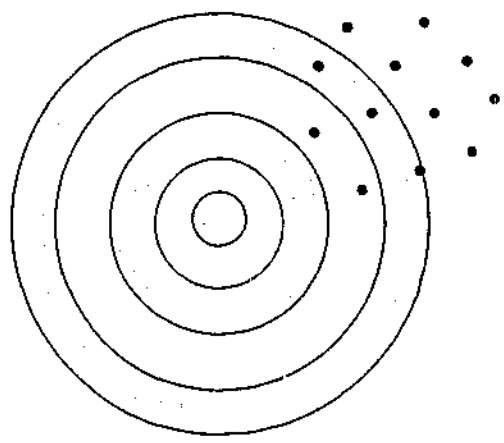
This factor in our analysis is inspired by Hsu, Huang, and Wong's study and analysis of discretization effectiveness [2000, 2003]. Hsu et al. addressed this factor in the context of estimating the probability density function $f(X_i|C{=}c)$ of a quantitative attribute X_i given each class c. They defined decision boundaries of X_i as intersection points of the curves of $f(X_i|C)$, where ties occurred among the largest conditional densities. They suggested that the optimal classification for an instance with $X_i{=}x_i$ was to pick the class c such that $f(X_i{=}x_i|C{=}c)$ was the largest, and the pick of the class was different when x_i was on different sides of a decision boundary. Hsu et al.'s analysis only



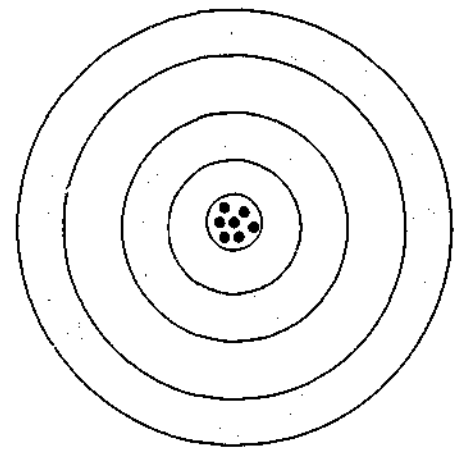
(a) High bias, low variance



(b) Low bias, high variance



(c) High bias, high variance



(d) Low bias, low variance

Figure 3.1: Bias and variance in shooting arrows at a target. Bias means that the archer systematically misses in the same direction. Variance means that the arrows are scattered.

addressed one-attribute classification problems[2], and only suggested that the analysis could be extended to multi-attribute applications without indicating how this might be so.

In our analysis we employ a definition of a decision boundary different from that of Hsu et al.'s because:

1. We believe that better insights are obtained by focusing on the values of X_i at which the class that maximizes $p(C{=}c \mid X_i{=}x_i)$ changes rather than those that maximize $f(X_i{=}x_i \mid C{=}c)$.

2. The condition that ties occur among the largest conditional probabilities is neither necessary nor sufficient for a decision boundary to occur. For example, suppose that we have probability distributions as plotted in Figure 3.2 that depicts a domain with two classes (*positive* vs. *negative*) and one attribute X_1. We have $p(positive \mid X_1){=}1.0$ (if $X_1 \geq d$); or 0.0 other-

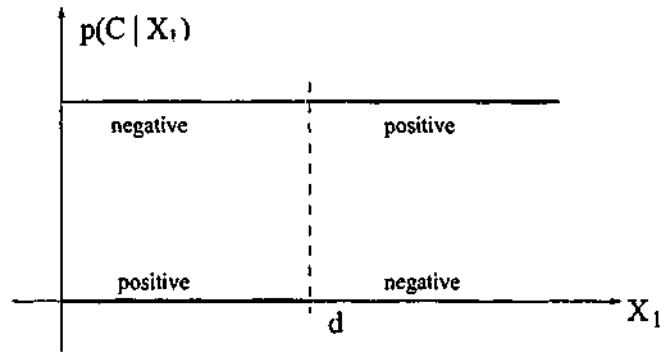


Figure 3.2: A tie in conditicnal probabilities is not a necessary condition for a decision boundary to exist.

wise. $X_1{=}d$ should be a decision boundary since the most probable class changes from *negative* to *positive* when X_i crosses the value d. However,

[2]By default, we talk about quantitative attributes.

there is no value of X_1 at which the probabilities of the two classes are equal. Thus the condition requiring ties is not necessary. Consider a second example as plotted in Figure 3.3. The conditional probabilities for c_1

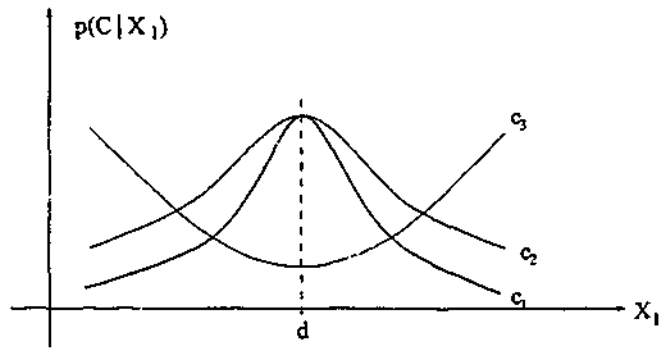


Figure 3.3: A tie in conditional probabilities is not a sufficient condition for a decision boundary to exist.

and c_2 are equally largest at X_1=d. However, d is not a decision boundary because c_2 is the most probable class on both sides of X_1=d. Thus the condition requiring ties is not sufficient either.

3. It is possible that a decision boundary is not a single value, but a value region. For example as plotted in Figure 3.4, the two classes c_1 and c_2

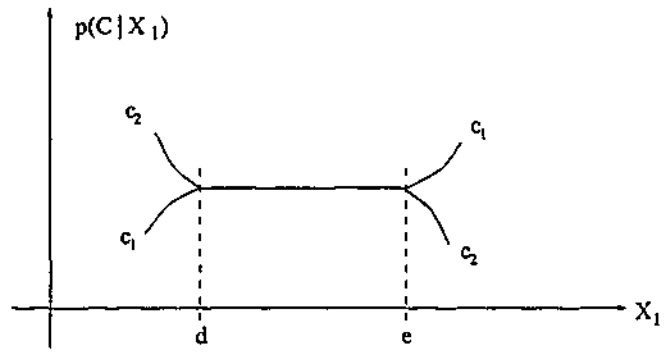


Figure 3.4: Decision boundaries may be regions rather than points.

are both most probable through the region $[d, e]$. In addition, the region's

width can be *zero*, as illustrated in Figure 3.2.

4. Decision boundaries of a quantitative attribute are expected to vary from test instance to test instance, depending on the precise values of other attributes presented in the test instance, as we will explain later in this section. However, Hsu et al. defined the decision boundaries of a quantitative attribute in such a way that they are independent of other attributes.

In view of these issues we propose a new definition for decision boundaries. This new definition is central to our study of discretization effectiveness in naive-Bayes learning. As we have explained, motivated by Theorem 1, we focus on the probability $p(C=c|X_i)$ of each class c given a quantitative attribute X_i rather than on the density function $f(X_i|C=c)$.

To define a decision boundary of a quantitative attribute X_i, we first define a *most probable class*. When classifying an instance $\mathbf{x}$, a most probable class c_m given $\mathbf{x}$ is the class that satisfies $\forall c \in C, P(c|\mathbf{x}) \leq P(c_m|\mathbf{x})$. Note that there may be multiple most probable classes for a single $\mathbf{x}$ if the probabilities of those classes are equally largest. In consequence, we define a *set of most probable classes* whose elements are all the most probable classes for a given instance $\mathbf{x}$. We use $mpc(\mathbf{x})$ to represent the set of most probable classes for $\mathbf{x}$. As a matter of notational convenience we define $\mathbf{x}\backslash X_i{=}v$ to represent an instance $\mathbf{x}'$ that is identical to $\mathbf{x}$ except that $X_i{=}v$ for $\mathbf{x}'$.

A *decision boundary* of a quantitative attribute X_i given an instance $\mathbf{x}$ in our analysis is an interval (l,r) of X_i (that may be of zero width) such that

$$\forall(w \in [l,r), u \in (l,r]), \neg(w{=}l \wedge u{=}r) \Rightarrow mpc(\mathbf{x}\backslash X_i{=}w) \cap mpc(\mathbf{x}\backslash X_i{=}u) \neq \emptyset$$

$$\wedge$$

$$mpc(\mathbf{x}\backslash X_i{=}l) \cap mpc(\mathbf{x}\backslash X_i{=}r) = \emptyset.$$

When analyzing how decision boundaries affect the discretization effectiveness, we suggest that the analysis involving only one attribute differs from that involving multiple attributes, since the final choice of the class is decided by the product of each attribute's probability in the later situation.

Consider a simple learning task with one quantitative attribute X_1 and two classes c_1 and c_2. Suppose $X_1 \in [0,2]$, and suppose that the probability distribution function for each class is $p(C{=}c_1 \,|X_1) = 1-(X_1-1)^2$ and $p(C{=}c_2 \,|X_1) = (X_1-1)^2$ respectively, which are plotted in Figure 3.5. The consequent decision

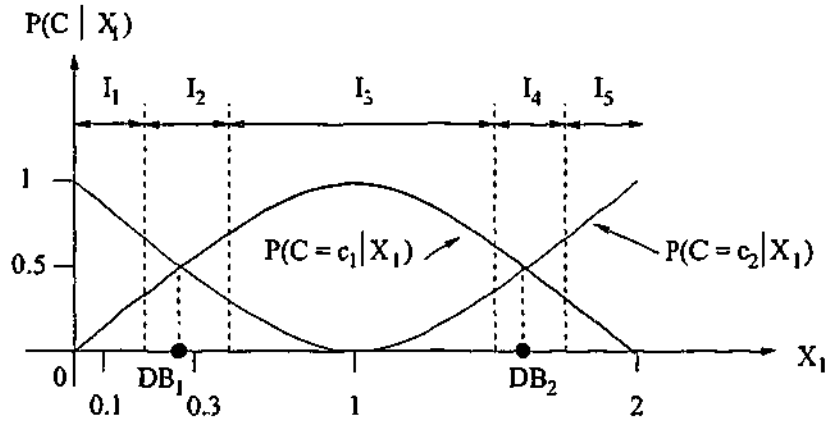


Figure 3.5: Probability distribution in one-attribute classification problem

boundaries are labelled as DB_1 and DB_2 respectively in Figure 3.5. The most-probable class for a value x_1 changes each time its location crosses a decision boundary. Assume a discretization method to create intervals I_i ($i{=}1,\cdots,5$) as in Figure 3.5. I_2 and I_4 each contain a decision boundary while the remaining intervals do not. For any two values in I_2 (or I_4) but on different sides

of a decision boundary, the optimal naive-Bayes learning under zero-one loss should select a different class for each value[3]. But under discretization, all the values in the same interval cannot be differentiated and we will have the same class probability estimate for all of them. Consequently, a naive-Bayes classifier with discretization will assign the same class to all of them, and thus values at one of the two sides of the decision boundary will be misclassified with respect to the optimal classification under zero-one loss. The larger the interval frequency, the more likely that the value range of the interval is larger, thus the more likely that the interval contains a decision boundary. The larger the interval containing a decision boundary, the more instances to be misclassified, thus the greater the expected bias of the generated naive-Bayes classifiers. In other words, larger interval frequency tends to incur higher discretization bias.

In a one-attribute classification problem, the locations of decision boundaries of the attribute X_1 depend on the distribution of $p(C{=}c\,|\,X_1)$ for each class c. However, for a multi-attribute application, the decision boundaries of an attribute, say X_1, are not only decided by the distribution of $p(C|X_1)$, but also vary from test instance to test instance depending upon the precise values of other attributes of the current test instance.

Consider another learning task with two quantitative attributes X_1 and X_2, and two classes c_1 and c_2. The probability distribution of each class given each attribute is depicted in Figure 3.6, of which the probability distribu-

[3]Please note that since naive-Bayes classification is a probabilistic problem, some instances will be misclassified even when optimal classification is performed. An optimal classifier is one that minimizes the Bayes classification error under zero-one loss [Duda and Hart 1973]. Hence even though it is optimal, it still can misclassify instances on both sides of a decision boundary.

tion of each class given X_1 is identical with that in the above one-attribute classification problem in Figure 3.5. We assume that the attribute indepen-

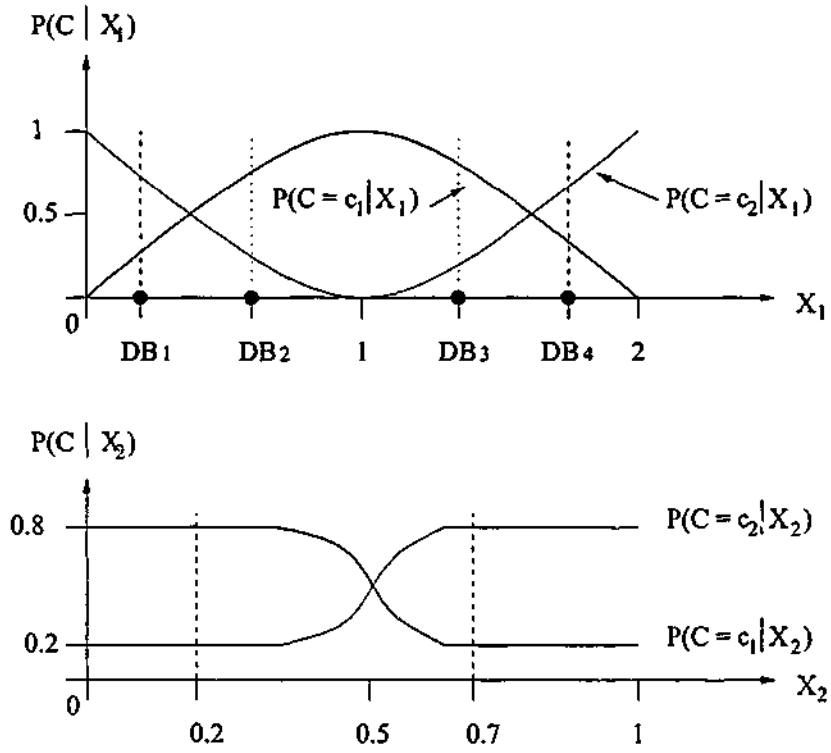


Figure 3.6: Probability distribution in two-attribute classification problem

dence assumption holds. We analyze the decision boundaries of X_1 for an example. If X_2 does not exist, X_1 has decision boundaries as depicted in Figure 3.5. However, because of the existence of X_2, those might not be decision boundaries any more. Suppose there comes a test instance $\mathbf{x}$ with X_2=0.2. When X_1 falls on any of the single attribute decision boundaries as presented in Figure 3.5, $p(C{=}c_1 \,|\, \mathbf{X}{=}\mathbf{x})$ does not equal $p(C{=}c_2 \,|\, \mathbf{X}{=}\mathbf{x})$. This is because $p(C{=}c_1 \,|\, X_2{=}0.2){=}0.8 \neq p(C{=}c_2 \,|\, X_2{=}0.2){=}0.2$, and $p(C{=}c \,|\, \mathbf{X}{=}\mathbf{x}) \propto \prod_{i=1}^{2} p(C{=}c \,|\, X_i{=}x_i)$ for each class c according to Theorem 1. Instead X_1's decision boundaries change to be DB_1 and DB_4 as in Figure 3.6. Suppose another test instance with X_2=0.7. By the same reasoning X_1's decision boundaries

change to be DB_2 and DB_3 as depicted in Figure 3.6. When there are more than two attributes, each combination of values of the attributes other than X_1 results in a corresponding decision boundary of X_1. Thus in multi-attribute applications the decision boundaries of one attribute can only be identified with respect to each specific combination of values of the other attributes. As we increase either the number of attributes or the number of values of an attribute, we will increase the number of combinations of attribute values, and thus the number of decision boundaries. Since many real-world applications involve training data of large size, each attribute may have a very large number of potential decision boundaries. Nevertheless, for the same reason as we have discussed in the one-attribute context, intervals containing decision boundaries have the potential negative impact on discretization bias.

Consequently, discretization bias can be reduced by identifying the decision boundaries and setting the interval boundaries close to them. However, identifying the correct decision boundaries depends on finding the true form of $p(C|X_i)$ for each quantitative attribute X_i. Ironically, if we have already found $p(C|X_i)$, we can resolve the classification task directly; thus there is no need to consider discretization any more.

Without knowing $p(C|X_i)$, a solution is to increase the interval number so as to decrease the interval frequency. An extreme approach is to set each (different) value as an interval. Although this most likely guarantees that no interval contains a decision boundary, it usually results in very few instances per interval. As a result, the estimation of $p(C{=}c|X_i^*{=}x_i^*)$ for each c might be so unreliable that we cannot identify the truly most probable class even if there is no decision boundaries in the interval. The larger the interval number

for probability estimation, the greater the expected variance of the generated naive-Bayes classifiers, since even a small change to the training data might substantially change the probability estimation. In other words, larger interval number tends to incur higher discretization variance.

A possible solution to this problem is to require that the interval frequency should be sufficient to ensure stability in the probability estimated therefrom. This raises the question, how reliable must the probability be? That is, when estimating $p(C{=}c|X_i{=}x_i)$ by $p(C{=}c|X_i^*{=}x_i^*)$, how much error can be tolerated without altering the classification? This motivates our following analysis.

3.3.3 Error tolerance of probability estimation

To investigate this factor, we return to our example depicted in Figure 3.5. We suggest that different values have different error tolerances of their probability estimation. For example, for a test instance $\mathbf{x}$ with $X_1{=}0.1$ and thus with c_2 being its most probable class, its true class probability distribution is $p(C{=}c_1|\mathbf{X}{=}\mathbf{x}) = p(C{=}c_1|X_1{=}0.1) = 0.19$ and $p(C{=}c_2|\mathbf{X}{=}\mathbf{x}) = p(C{=}c_2|X_1{=}0.1) = 0.81$. According to naive-Bayes learning, as long as $p(C{=}c_2|X_1{=}0.1) > 0.50$, c_2 will be correctly chosen as the class and the classification is optimal under zero-one loss. This means that the error tolerance of estimating $p(C|X_1{=}0.1)$ can be as big as $0.81{-}0.50 = 0.31$. However, for another test instance $\mathbf{x}$ with $X_1{=}0.3$ and thus with c_1 being its most probable class, its probability distribution is $p(C{=}c_1|\mathbf{X}{=}\mathbf{x}) = p(C{=}c_1|X_1{=}0.3) = 0.51$ and $p(C{=}c_2|\mathbf{X}{=}\mathbf{x}) = p(C{=}c_2|X_1{=}0.3) = 0.49$. The error tolerance of estimating $p(C|X_1{=}0.3)$ is only $0.51{-}0.50 = 0.01$. In the learning context of multi-

attribute applications, the analysis of the error tolerance of probability estimation is even more complicated. The error tolerance of a value of an attribute affects, as well as is affected by those of the values of the other attributes since it is the product of $p(C{=}c|X_i{=}x_i)$ of each x_i that decides the final probability of each class.

The larger an interval's frequency, the lower the expected error of probability estimates pertaining to that interval. Hence, the lower the error tolerance for a value, the larger the ideal frequency for the interval from which its probabilities are estimated. Since all the factors that affect error tolerance vary from case to case, there cannot be a universal, or even a domain-wide constant that represents the ideal interval frequency, which thus will vary from case to case. Further, the error tolerance can only be calculated if the true probability distribution of the training data is known. If it is not known, the best we can hope for is heuristic approaches to managing error tolerance that work well in practice.

3.3.4 Summary

By this line of reasoning, optimal discretization can only be performed if the probability distribution of $p(C{=}c|X_i)$ for each c within each X_i is known, and thus the decision boundaries are known. If the decision boundaries are not known, which is often the case for real-world data, we want to maximize the interval number so as to minimize the risk that an instance is classified using an interval containing a decision boundary. By this means we expect to reduce the discretization bias, thus to reduce the classification bias of the generated

naive-Bayes classifiers. On the other hand, however, we want to ensure that the interval frequency is sufficiently large so as to minimize the risk that the error of estimating $p(C{=}c|X_i{=}x_i)$ by $p(C{=}c|X_i^*{=}x_i^*)$ will exceed the current error tolerance. By this means we expect to reduce the discretization variance, thus to reduce the classification variance of the generated naive-Bayes classifiers.

However, all other things being equal, there is a trade-off between interval frequency and interval number. That is, the larger the interval frequency, the smaller the interval number, and vice versa. A larger interval frequency leads to lower discretization variance but higher discretization bias, while a larger interval number leads to lower discretization bias but higher discretization variance. Hence, tuning interval frequency and interval number can be an approach to finding a good trade-off between discretization bias and variance, thus to achieving low naive-Bayes classification error. We argue that there is no universal solution to this problem. That is, the optimal trade-off between interval frequency and interval number will vary greatly from test instance to test instance.

Another illuminating issue arising from our study is that since the decision boundaries of a quantitative attribute value depend on the precise values of other quantitative attributes given a particular test instance, we can not develop optimal discretization by any apriori methodology, that is, by forming intervals prior to the classification time. However, even if we adopt a lazy methodology [Zheng and Webb 2000], that is, taking into account the values of other attributes when classifying an instance during classification time, we still cannot guarantee optimal discretization unless we know the true probability distribution of the quantitative attributes. These insights reveal that,

while discretization is desirable when the true underlying probability density functions are not available, practical discretization techniques are necessarily heuristic in nature. The holy grail of an optimal universal discretization strategy for naive-Bayes learning is unobtainable.

3.4 Summary

In this chapter, we have conducted a theoretical analysis of discretization in naive-Bayes learning. Naive-Bayes learning is probabilistic learning. It needs to estimate the probability of each class given an instance in order to classify this instance. When the learning involves quantitative attributes, we need to know the true probability distribution of each class within each quantitative attribute. However, this is usually unknown for real-world data. Consequently, there are two typical approaches to dealing with quantitative attributes, probability density estimation or discretization. Among the two approaches, discretization is more popular because of its efficacy and efficiency for naive-Bayes learning. Nonetheless, we do not think that discretization can be 'unconditionally' effective as previous research suggested [Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003]. Instead we have proved Theorem 1 that states particular conditions under which discretization will result in naive-Bayes classifiers delivering the same probability estimates as would be obtained if the correct probability density function were employed. According to Theorem 1, we have proposed two factors, the *decision boundary* and the *error tolerance of probability estimation* to be the key factors that are able to affect discretization's effectiveness. Different discretization methods have different

ways to handle these two factors, thus have different impacts on the classification bias and variance of the generated naive-Bayes classifiers. We name these effects *discretization bias* and *variance*. We believe that by better managing discretization bias and variance, we can achieve lower naive-Bayes classification error. In particular, we have related discretization bias and variance to discretized interval frequency and interval number. We have suggested that by properly adjusting interval frequency and number, we can better manage discretization bias and variance.

In the next chapter, we will present a comprehensive literature review of existing discretization methods in the area of machine learning. We are particularly interested in those methods that are usually employed in naive-Bayes learning.

Chapter 4

Review of previous discretization methods

The previous chapter has analyzed discretization in naive-Bayes learning. This chapter will present a comprehensive literature review of existing discretization methods in the research area of machine learning.

Many real-world data tackled by machine learning algorithms involve quantitative data. However, there exist many machine learning algorithms that are more oriented to handle qualitative attributes than quantitative attributes [Kerber 1992; Dougherty, Kohavi, and Sahami 1995; Kohavi and Sahami 1996]. Even for algorithms that can directly deal with quantitative attributes, learning is often less efficient and less effective for quantitative data than for qualitative data [Catlett 1991; Kerber 1992; Richeldi and Rossotto 1995; Frank and Witten 1999]. Since larger and larger datasets are becoming routinely available for most modern research, the learning efficiency is of particular importance. Thus discretization has attracted much attention.

Over the years, many discretization algorithms have been proposed and tested to show that discretization helps improve the performance of learning and helps understand the learning result. In this chapter, we review 34 dis-

cretization methods. We break down the review to methods that are typically used for naive-Bayes learning, and to methods that are used for learning contexts other than naive-Bayes learning.

4.1 Methods for naive-Bayes learning

We here review six discretization methods, each of which was either designed especially for naive-Bayes classifiers or is in practice often used for naive-Bayes classifiers. Since we have valued the bias-variance characteristic of discretization in naive-Bayes learning, we are particularly interested in analyzing each method's effectiveness in terms of discretization bias and variance, which we believe illuminating. The methods are ordered by the year that they each were published.

4.1.1 Equal width discretization & Equal frequency discretization

When discretizing a quantitative attribute, equal width discretization (EWD) [Catlett 1991; Kerber 1992; Dougherty, Kohavi, and Sahami 1995] divides the number line between v_{min} and v_{max} into k intervals of equal width, where v_{min} is the minimum observed value, v_{max} is the maximum observed value, and k is a user predefined parameter. Thus the intervals have width $w = (v_{max} - v_{min})/k$ and the cut points are at $v_{min} + w, v_{min} + 2w, \cdots, v_{min} + (k-1)w$.

When discretizing a quantitative attribute, equal frequency discretization (EFD) [Catlett 1991; Kerber 1992; Dougherty, Kohavi, and Sahami 1995] di-

vides the sorted values into k intervals so that each interval contains approximately the same number of training instances. k is a user predefined parameter. Thus each interval contains n/k training instances with adjacent (possibly identical) values. Note that training instances with identical values must be placed in the same interval. In consequence it is not always possible to generate k equal frequency intervals.

Both EWD and EFD are often used for naive-Bayes classifiers because of their simplicity and reasonable effectiveness [Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003]. However both EWD and EFD fix the number of intervals to be produced (decided by the user predefined parameter k). When the training dataset is very small, intervals tend to have small frequency and thus tend to incur high discretization variance. When the training data size is very large, intervals tend to have large frequency and thus tend to incur very high discretization bias. Thus we anticipate that they do not control either discretization bias or discretization variance well.

4.1.2 Fuzzy learning discretization

Fuzzy learning discretization[1] (FLD) [Kononenko 1992; Kononenko 1993] initially discretizes X_i into k equal width intervals $(a_i, b_i]$ $(1 \leq i \leq k)$ using EWD, where k is a user predefined parameter. For each discretized value x_i^* corresponding to $(a_i, b_i]$, FLD estimates $p(X_i^*{=}x_i^* \,|\, C{=}c)$ from all training instances rather than only from instances that have values of X_i in $(a_i, b_i]$. The in-

[1]This is one of the three versions of fuzzy discretization proposed by Kononenko [1992, 1993] for naive-Bayes classifiers. We present here only the version that, according to our experiments, is most effective to reduce the naive-Bayes classification error.

fluence of a training instance with value v of X_i on $(a_i, b_i]$ is assumed to be normally distributed with the mean value equal to v and is proportional to $P(v, \sigma, i) = \int_{a_i}^{b_i} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-v}{\sigma})^2} dx$. σ is a parameter to the algorithm and is used to control the 'fuzziness' of the interval bounds. According to Kononenko's experiments, setting σ to $0.7 \times \frac{v_{max}-v_{min}}{k}$ achieved the best results. Suppose there are n_c training instances with known values for X_i and with class c, each with influence $P(v_j, \sigma, i)$ on $(a_i, b_i]$ $(j=1, \cdots, n_c)$:

$$p(X_i^*=x_i^* \mid C=c) = \frac{p(a_i < X_i \leq b_i \wedge C=c)}{p(C=c)} \approx \frac{\frac{\sum_{j=1}^{n_c} P(v_j, \sigma, i)}{n}}{p(C=c)}. \quad (4.1)$$

The idea behind fuzzy learning discretization is that small variation of the value of a quantitative attribute should have small effects on the attribute's probabilities, whereas under non-fuzzy discretization, a slight difference between two values, such that one is above and one is below the cut point, can have drastic effects on the estimated probabilities. However, we suspect that when the training instances' influence on each interval does not follow the normal distribution, FLD's effectiveness can degrade. As for discretization bias and variance, since FLD takes into consideration all values in a quantitative attribute's value range for the naive-Bayes probability estimation, we anticipate it to be good at reducing discretization variance but reversely, bad at reducing discretization bias. Besides, its primary method can have impact on its discretization bias and variance. For example, when its primary method is EWD, FLD tends to incur high discretization bias when the training data size is large where EWD forms intervals of large frequency.

4.1.3 Entropy minimization discretization

Entropy minimization discretization (EMD) [Fayyad and Irani 1993] evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For evaluating each candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidates. The binary discretization is applied recursively, always selecting the best cut point. A minimum description length criterion (MDL) is applied to decide when to stop discretization.

An often-cited contribution of Fayyad and Irani's work is that it defined *boundary cut points* of a quantitative attribute. Boundary cut points are values between two instances with different classes in the sequence of instances sorted by this quantitative attribute. It was proved that evaluating only the boundary cut points is sufficient for finding the minimum class information entropy. Many methods in our review refer to the concept of boundary cut points.

Although it has demonstrated strong effectiveness for naive-Bayes [Dougherty, Kohavi, and Sahami 1995; Perner and Trautzsch 1998], EMD was developed in the context of top-down induction of decision trees. It uses MDL as the termination condition. According to An and Cercone [1999], this has an effect to form qualitative attributes with few values. For decision tree learning, it is important to minimize the number of values of an attribute, so as to avoid the fragmentation problem [Quinlan 1993]. If an attribute has many values, a split on this attribute will result in many branches, each of which receives

relatively few training instances, making it difficult to select appropriate subsequent tests. Naive-Bayes learning assumes that attributes are independent of one another given the class, and hence is not subject to the same fragmentation problem. As a result, we anticipate that EMD used in naive-Bayes learning is good at reducing discretization variance, but does not control discretization bias so successfully. This might work well for the training data of small size, for which it is credible that variance reduction can contribute more to lowering naive-Bayes learning error than bias reduction [Friedman 1997]. However, when the training data size is large, it is very likely that the loss through discretization bias increase will soon overshadow the gain through discretization variance reduction, resulting in inferior learning performance.

A second issue is that when it is used in naive-Bayes learning, EMD discretizes a quantitative attribute by calculating the class information entropy as if the naive-Bayes classifiers only use that *single* attribute after discretization. Thus EMD might be effective at identifying decision boundaries in the learning context of one-attribute applications. But in the context of multi-attribute applications, the resulting cut points can easily diverge from the true ones as we have explained in Section 3.3. If this happens, we anticipate that EMD will control neither discretization bias nor discretization variance well.

4.1.4 Iterative-improvement discretization

Iterative-improvement discretization (IID) [Pazzani 1995] was purposely designed for naive-Bayes classifiers. It initially forms a set of intervals using EWD or EMD, and then iteratively adjusts the intervals to minimize the naive-

Bayes classification error on the training data. It defines two operators: merge two contiguous intervals, or split an interval into two intervals by introducing a new cut point that is midway between each pair of contiguous values in that interval. In each loop of the iteration, for each quantitative attribute, IID applies both operators in all possible ways to the current set of intervals and estimates the classification error of each adjustment using leave-one-out cross validation. The adjustment with the lowest error is retained. The loop stops when no adjustment further reduces the error.

A potential disadvantage of IID results from its iterative nature. When the training data size is large, the possible adjustments by applying the two operators will be numerous. Consequently the repetitions of the leave-one-out cross validation will be numerous so that IID has high computational time overhead, which defeats a principle advantage of naive-Bayes classifiers for many application: its computational efficiency.

IID can split as well as merge discretized intervals. How many intervals will be formed and where the cut points are located are decided by the error of the cross validation. This is a case by case problem and thus it is not clear what IID's systematic impact on discretization bias and variance is. However, as cross-validation has frequently proved successful in finding a model that minimizes error, we might infer that IID can obtain a good discretization bias-variance trade-off.

4.1.5 Lazy discretization

Lazy discretization (LD) [Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003] defers discretization until classification time. It waits until a test instance is presented to determine the cut points and then to estimate $f(X_i{=}x_i|C{=}c)$ by $p(X_i^*{=}x_i^*|C{=}c)$ for each quantitative attribute of the test instance. When classifying an instance, LD creates only one interval for each quantitative attribute containing its value from this instance, and leaves other value regions untouched. In particular, it selects a pair of cut points for each quantitative attribute such that the value is in the middle of its corresponding interval. The interval frequency is equal to that produced by its primary discretization method. In Hsu et al.'s implementation, the interval frequency is the same as created by EWD with $k{=}10$. However, as already noted, $k{=}10$ is an arbitrary value.

The motivation of LD is to save the training effort of naive-Bayes learning, since correct classification of an instance only depends on intervals that contain values of this instance, and is completely independent of how other value regions are discretized [Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003]. Ironically, however, LD tends to have high computational memory and time requirements because of its lazy methodology. Eager approaches carry out discretization at training time. Thus the training instances can be discarded before classification time. In contrast, LD needs to keep the training instances for use during classification time. This demands high memory expenses when the training data size is large. Furthermore, where a large number of instances need to be classified, LD will incur large computational

overhead since it must estimate probabilities from the training data for each attribute of each instance individually. Although LD achieves comparable accuracy to EWD and EMD [Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003], the high memory and computational overhead might impede it from feasible implementation for classification tasks with large training or test data.

From the perspective of discretization bias and variance, we value the idea of 'placing a quantitative value at the middle of its interval'. As we will explain in detail later in Section 5.2.3, we believe it can contribute to reducing discretization bias and variance in naive-Bayes learning. However, a precondition of this idea's success is that a proper interval frequency strategy is employed. Without forming intervals of proper frequency, as LD's implementation did, sub-optimal effectiveness can be expected. For example, if LD chooses EWD as its primary method, we anticipate that LD tends to have high discretization bias when the training data size is large, where EWD forms intervals of large frequency.

4.2 Methods for learning contexts other than naive-Bayes learning

The majority of existing discretization methods in machine learning, however, have been developed for learning contexts other than naive-Bayes learning, such as decision trees, decision rules, decision tables, decision lists, association rules and Bayes networks. We are interested in whether these methods can

be appropriate for naive-Bayes learning. Thus we conduct a comprehensive review of these methods and present them by the order of their publication years.

1. **Discretizer-2** [Catlett 1991]. Discretizer-2 was proposed for ID3 [Quinlan 1986], an inductive learning program that constructs classification rules in the form of decision trees. The first cut point is chosen in the same way as ID3, using a formula that assesses the gain in information theoretic terms of all possible cut points. The subsequent cut points are recursively chosen based on the subsets of training instances between the obtained cut points. A minimum number of instances in each discretized interval, a maximum number of discretized intervals and a minimum information gain decide when to stop discretization. D-2 chooses all the cut points of a quantitative attribute in one step before the tree construction, rather than jumping back and forth as ID3 does. As a result it offers large reduction in learning time at the cost of little loss of classification accuracy.

2. **ChiMerge discretization** [Kerber 1992]. ChiMerge uses the χ^2 statistics to determine if the relative class frequencies of adjacent intervals are distinctly different or if they are similar enough to justify merging them into a single interval. The ChiMerge algorithm consists of an initialization process and a bottom-up merging process. The initialization process contains two steps: (1) ascendingly sort the training instances according to their values for the attributes being discretized, (2) construct the initial discretization, in which each instance is put into its own interval.

The interval merging process contains two steps, repeated continuously: (1) compute the χ^2 for each pair of adjacent intervals, (2) merge the pair of adjacent intervals with the lowest χ^2 value. Merging continues until all pairs of intervals have χ^2 values exceeding a χ^2-*threshold*. That is, all intervals are considered significantly different by the χ^2 independence test. The user can also override the χ^2-*threshold* through two parameters *min-intervals* and *max-intervals* which specify a lower and upper limit on the number of intervals to create. The standard recommended procedure for using ChiMerge is to set χ^2-*threshold* at the 0.90, 0.95 or 0.99 significant level and to set *max-interval* to a value around 10 or 15 to prevent an excessive number of intervals from being created.

3. **1-Rules discretization** [Holte 1993]. In order to propose the 'simplicity first' research methodology, Holte [1993] described a kind of rules, called '1-rules', which classify an instance on the basis of a single attribute (that is, they are 1-level decision trees). To deal with quantitative attributes, a discretization method, 1-rules discretization is proposed. It sorts the observed values of a quantitative attribute and then divides the values into a finite number of intervals. In dividing, it attempts to make each interval 'pure' (that is, containing instances that are all of the same class). Since overfitting may result from such a scheme (that is, one interval for each observed value), 1-rules discretization requires all intervals (except the rightmost) to contain more than a predefined number of instances in the same class. Based on the empirical results from Holte, Acker, and Porter [1989], the threshold is set at 6 for all datasets except for the datasets with

fewest examples where the threshold is set at 3.

4. **Error-based discretization** [Maass 1994; Kohavi and Sahami 1996]. Error-based discretization optimally discretizes a quantitative attribute with respect to error on the training set. In the work of [Maass 1994], it produces an optimal set of k or fewer intervals that results in the minimum error on the training set if the instances were to be classified using that single attribute after discretization. The maximum number of intervals k is a user-predefined parameter. This method was implemented as part of the T2 induction algorithm which is guaranteed to produce close to optimal 2-level decision trees from sufficiently large training datasets for any distribution of data [Auer, Holte, and Maass 1995]. T2 sets the value of k to be the number of classes plus one. In their research comparing error-based and entropy-based discretization, Kohavi and Sahami [1996] set k to be the same number of intervals proposed by running the entropy minimization discretization [Fayyad and Irani 1993]. They showed that the error-based discretization method has a deficiency, that is, it will never generate two adjacent intervals when a particular class prevails in both intervals, even when the class frequency distribution differs in both intervals.

5. **C' uster-based discretization** [Chmielewski and Grzymala-Busse 1996]. This method consists of two steps. The first step is *cluster formation* to determine initial intervals for the quantitative attributes. The second step is *post-processing* to minimize the number of discretized intervals. Instances here are deemed as points in n-dimensional space which is defined by n

attribute values. During cluster formation, the median cluster analysis method is used. Clusters are initialized by allowing each instance to be a cluster. New clusters are formed by merging two existing clusters that exhibit the greatest similarity between each other. The cluster formation continues as long as the level of consistency (defined in the study) of the partition is not less than the level of consistency of the original data. Once this process is completed, instances that belong to the same cluster are indiscernible by the subset of quantitative attributes, thus a partition on the set of training instances is induced. Clusters can be analyzed in terms of all attributes to find out cut points for each attribute simultaneously. After discretized intervals are formed, post-processing picks a pair of adjacent intervals among all quantitative attributes for merging whose resulting class entropy is the smallest. If the consistency of the dataset after the merge is above a given threshold, the merge is performed. Otherwise this pair of intervals are marked as non-mergable and the next candidate is processed. The process stops when each possible pair of adjacent intervals are marked as non-mergable.

6. **StatDisc discretization** [Richeldi and Rossotto 1995]. This method was proposed to overcome some 'undesirable' properties of ChiMerge discretization [Kerber 1992], including that ChiMerge examines pairs of adjacent intervals only, ignoring other surrounding intervals; and ChiMerge produces a fixed partition of the attribute values. It consists of three steps: initialization, interval hierarchy creation and selection of the best discretization. The initialization step sorts the training instances

according to their values being discretized, and then groups adjacent instances labelled by the same class into the same interval. The interval hierarchy, in the form of a tree, is created bottom-up during a merging process. Intervals that are created in the initialization step are associated to leaf nodes. The merging process contains two steps, repeated continuously: (1) compute the Φ statistic for any N-uples of adjacent intervals where N is selected by the user, (2) merge the N-uples with the lowest Φ value. A new non-leaf node is added to the tree whenever a merge occurs. This node, associated to the new interval, is connected to the nodes that represent the intervals that have been merged. Thus each level of the tree represents a discretization of the quantitative attribute. Merging continues until all N-uples of intervals have a Φ value greater than the Φ distribution at a desired level of significance and degrees of freedom for a sample. A two-step heuristic is then applied to force the merging process to continue until a one-interval partition (tree root) is obtained. The final step is the selection of the best discretization. StatDisc seeks the largest partition that is obtained before decreasing the significance level. If this search fails, it returns the partition which, on average, contains the largest adjacent intervals whose relative class frequencies are the most dissimilar.

7. **Compression-based Discretization** [Pfahringer 1995]. In this research, it was argued that discretization methods which merge or split an interval into a *predefined* number of intervals sometimes miss the opportunity of forming larger uniform intervals. Consequently these methods may

produce superfluous intervals, which may have a detrimental influence on both learning efficiency and accuracy. In addition, they cannot distinguish truly irrelevant attributes from what Kerber [1992] called *second-order* correlated attributes, that is, attributes correlating only in the presence of some other condition. Compression-based discretization was developed to solve those problems. It is an application of the *minimum description length* (MDL) principle [Rissanen 1978] as an objective function for evaluating a specific discretization as a whole. The basic assumption used is as follows. If some discretization is synthesized taking class information into account, then such a discretization of a single attribute can be viewed as a set of rules classifying instances. Classifying an instance determines the interval that covers the instance's respective attribute value and just returns that interval's majority class. Adopting this interpretation, the MDL principle can be straightforwardly used to assign a numerical measure of quality to any single discretization. Pfahringer defined a set of formulae to calculate the cost (bit size) for defining a discretization plus the cost for encoding the instances in terms of the discretization. The discretization which minimize the sum cost, that is, the one with the smallest bit cost (also called the most *compressive* theory) is the most probable theory given the class distribution of the training instances. Compression-based discretization consists of two steps. First a set of promising cut points is heuristically found by an efficient supervised discretization. In particular, D-2 [Catlett 1991] is employed with the stopping criteria as finding at most $(2^5 - 1)$ cut points. Second this set is searched thoroughly by a hill-climbing search for the most compres-

sive discretization according to the MDL measure defined in this study. Due to the nature of the MDL measure, most superfluous cut points are discarded. As for possible second-order attributes, the result of the MDL measure is one large interval covering all training instances. Thus it is easy to tell that these attributes are problematic and an unsupervised method is accordingly used to produce at least a reasonable discretization.

8. **InfoMerge discretization** [Freitas and Lavington 1996]. InfoMerge discretization is based on information theory, rather than based on a statistical measure of dependency (or significance test) as ChiMerge discretization [Kerber 1992] or StatDisc discretization [Richeldi and Rossotto 1995]. Statistical measures of dependency are not designed for classification. Rather, they are designed for measuring the dependency between two attributes in a *symmetric* way. That is, none of the two attributes being analyzed (the attribute to be discretized and the class) is given special treatment when computing the measure. On the other hand, classification is an *asymmetric* task with respect to the two attributes being analyzed. It wants to predict the value of the class attribute *given* the discretized attribute, not the reverse. Accordingly InfoMerge uses a measure of information loss. The information loss is calculated as the amount of information necessary to identify the class of an instance after merging minus the corresponding amount of information before merging. First, InfoMerge sorts the attribute values and calculates the class frequency distribution for each value. Second, InfoMerge iteratively calculates the information

loss associated with the interval-merging process for every group of N adjacent intervals (where N is a user-supplied parameter). Besides information loss, the size of the intervals to be merged is also taken into consideration. Merging intervals with smaller size is preferable since it will tend to produce less information loss in the dataset as a whole. Hence the information loss is furthermore weighted with the interval frequency. InfoMerge merges the groups of N adjacent intervals with the minimum value of weighted information loss, following the bottom-up paradigm of ChiMerge and StatDisc. The stopping criterion of the merging process depends on user-specified maximum and minimum number of intervals to be produced and on a given threshold which are set as 12, 5, and 0.001 respectively.

9. **K-means clustering discretization** [Torgo and Gama 1997]. This method aims at building k intervals that minimize the sum of the distances of each element of an interval to its gravity center [Dillon and Goldstein 1984]. This method starts with the EW approximation and then moves the elements of each interval to contiguous intervals whenever these changes reduce the sum of distances. The k is found by searching using a classification algorithm where the loss function considers a cost matrix given by the distance between the centroids of the clusters.

10. **Search-based discretization** [Torgo and Gama 1997]. This method was developed in the context of using classification for regression, where the regression target variable is discretized into class intervals and some classification algorithm is used to predict the class interval instead of a par-

ticular target value. When choosing the cut points, each of three candidate discretization methods can be used: equal width discretization, equal frequency discretization [Catlett 1991; Kerber 1992; Dougherty, Kohavi, and Sahami 1995] and k-means clustering [Torgo and Gama 1997]. When deciding how many intervals to be formed, a wrapper technique [John, Kohavi, and Pfleger 1994; Kohavi 1995b] is used as a method for finding the near-optimal set of intervals. Each of two alternative search operators can be provided to the wrapper approach. One is to increase the previously tried number of intervals by a constant amount. The other is to improve the previous set of intervals taking into account their individual evaluation. That is, the median of these individual error estimates is calculated; then all intervals whose error is above the median are further split by dividing them into two and all the other intervals remain unchanged. The evaluation strategy of the wrapper approach is an N-fold cross validation [Stone 1974]. The two wrapper operators together with the three cut-point-finding strategies make six alternative discretization methods. The one that gives the best estimated result for the learning system is selected.

11. **Distance-based discretization** [Cerquides and Lopez de Mantaras 1997]. This method is based on the Mantaras distance between partitions [Lopez de Mantaras 1991]. It is iterative, considering all training instances for the selection of each new cut point, in comparison to the recursive 'divide and conquer' technique adopted by many alternative discretization methods.

Given a quantitative attribute, suppose that P_D denotes the partition on training instances induced by a discretization D and that $P_{D\cup\{T\}}$ denotes the partition on training instances induced by adding a new cut point T to the current discretization D. The requirement is to find a cut point T_A so that it accomplishes:

$$\forall T, Dist(P_C, P_{D\cup\{T\}}) \geq Dist(P_C, P_{D\cup\{T_A\}}),$$

where P_C is the partition on the training instances generated by the class attribute and *Dist* stands for Mantaras normalized distance, defined as:

$$Dist(P_C, P_D) = \frac{I(P_C|P_D) + I(P_D|P_C)}{I(P_C \cup P_D)},$$

where I is the standard Shannon measures of information [Lopez de Mantaras 1991].

Once T_A is found, the minimum description length principle (MDL) [Fayyad and Irani 1993] is used to decide whether the improvement resulting from introducing T_A is significant enough to accept T_A or if otherwise no further cut points are considered necessary for the discretization. Given two discretization, one with p and the other with $p+1$ cut points, the one with the minimal description length will be chosen. If it is the one with p cut points, the algorithm stops and no more cut points are added to the discretization. If it is the one with $p+1$ cut points, the process continues by considering introducing a new cut point. Since the complexity

	A_1	A_2	$\cdots$	A_k
C_1	n_{11}	n_{12}	$\cdots$	n_{1k}
C_2	n_{21}	n_{22}	$\cdots$	n_{2k}
$\cdots$				
C_k	n_{k1}	n_{k2}	$\cdots$	n_{kk}

Table 4.1: Zeta Discretization

of the algorithm tends to be high, the process of selecting new cut points has also been parallelized to obtain a high improvement in the efficiency of the algorithm.

12. **Zeta discretization** [Ho and Scott 1997]. Zeta is a measure of strength of association between the class and an attribute. It is defined as the maximum accuracy achievable when each value of the attribute predicts a different class value. Suppose there are N training instances with a k-valued attribute A and a k-valued class C whose distribution is given by Table 4.1, where n_{ij} is the number of instances with class value C_i and attribute value A_j and $N = \sum_{i=1}^{k} \sum_{j=1}^{k} n_{ij}$. Zeta is defined as:

$$Zeta = \frac{\sum_{i=1}^{k} n_{f(i),i}}{N} \times 100\%,$$

where $f(i)$ is the class index that has the *highest* frequency of instances with attribute value A_i.

When discretization, given a k-valued class variable, a quantitative attribute could be partitioned into k intervals by calculating zeta for each possible assignment of the $k-1$ cut points and selecting the combination of the cut points that gives the largest value of zeta. In order to evade

examining all possible cut points which is computationally expensive when k is large, a stepwise hill-climbing procedure is implemented to recursively find out the cut points in the formed intervals.

13. **ConMerge discretization** [Wang and Liu 1998]. This method consists of an initialization step and a bottom-up merging process. In the initialization step, each quantitative value is put into its own interval. In the merging process, the best pair of adjacent intervals from *all* quantitative attributes are repeatedly selected according to a goodness criterion (defined by inconsistency rate in this study). The selected pair are merged if doing so does not exceed a user-specified inconsistency threshold. If the pair are not merged, the merging of this pair is excluded from further consideration. The merging process is repeated until no more merging is possible. Since scanning all pairs of adjacent intervals for all attributes for each merge step is not acceptable in terms of computation overhead, especially for large datasets, a merge-tree structure, a modified B-tree is proposed to find out the best merging in a constant time that is independent of the number of intervals. Thus the algorithm can scale up well in large datasets.

14. **Dynamic discretization** [Gama, Torgo, and Soares 1998]. This method looks at all possible discretizations of a quantitative attribute as a hierarchy. The most general discretization is at the top of this hierarchy, and consists of one interval containing all the observed values. At the bottom of the hierarchy is the most specific discretization which is a set of intervals, each containing a single value. For more than one quanti-

tative attribute, a set of hierarchies exist. This method performs an A^* search over the set of hierarchies defined by all the possible combination of attribute discretizations. By proceeding this way the discretization of one quantitative attribute depends on the discretization of other quantitative attributes. Thus it is able to explore inter-dependencies between attributes.

For each quantitative attribute, dynamic discretization considers only the boundary cut points [Fayyad and Irani 1993]. The search space consists of $n_1 * n_2 * \cdots * n_m$ states, where n_i is the number of boundary cut points of attribute i and m is the number of quantitative attributes. A state can be described by a vector of integers, $< v_1, v_2, \cdots, v_m >$, where v_i is the number of discretized intervals of attribute i and attribute i is discretized by k-means clustering discretization with k=v_i [Torgo and Gama 1997]. Each state is evaluated by the classification error resulting from a 10-fold cross validation on C4.5 or naive-Bayes classifiers with the corresponding discretized data. The initial state of search consists of the vector $< 1, 1, \cdots, 1 >$. In the search space, each node (vector) has m descendants. On each descendant, the number of intervals of one attribute grows by a user defined parameter. The next node to expand is the one not yet expanded and with lowest value of the classification error. If several nodes have the lowest error, the one with the least number of discretized intervals is chosen to be expanded. If the classification error returns 0, or the classification performance does not improve in a certain number of expansions, the discretization process stops.

15. **Berka's discretization** [Berka and Bruha 1998]. The basic idea behind this method is to create intervals for which the *aposteriori* distribution of classes $P(C|interval)$ significantly differs from the *apriori* distribution of classes $P(C)$ in the whole training data. This can be achieved by simply merging those values for which most instances belong to the same class. The number of resulting intervals is controlled by giving a threshold for minimal number of instances in one interval. For each quantitative attribute, the discretization process is:

    (a) Sort values of the attribute;

    (b) For each value:

        i. Compute frequencies of occurrence of instances with respect to each class;

        ii. If for the given value all instances belong to the same class, assign that class to the value;

        iii. Else if for the given value the distribution of instances with respect to classes significantly differs (according to χ^2 or relative frequency criterion) from the *apriori* distribution of classes, assign the most frequent class to the value;

        iv. Else assign the class *UNKNOWN* to the value;

    (c) Create the intervals from values by:

        i. If a sequence of values belong to the same class, then create the interval $INT_i = [LBound_i, UBound_i]$ by grouping these values;

        ii. If the interval INT_i belongs to the class *UNKNOWN*, then:

A. If its neighboring intervals INT_{i-1} and INT_{i+1} belong to the same class, then create the interval by joining $INT_{i-1} \cup INT_i \cup INT_{i+1}$;

B. Else create the interval either by joining $INT_{i-1} \cup INT_i$ or by joining $INT_i \cup INT_{i+1}$ according to a given criterion (for χ^2 test, join the intervals with the higher value of χ^2; for relative frequency criterion, join the intervals with the higher relative frequency of the majority class);

C. Create continuous coverage of the attribute by assigning $LBound_i := (LBound_i + UBound_{i-1})/2$ and $UBound_{i-1} := LBound_i$.

16. **Semi-optimal discretization** [Chlebus and Nguyen 1998]. This method was developed in the context of decision tables with two attributes. It can be generalized to handle more attributes.

    Let $\mathbf{A} = (U, \{a, b\} \cup \{d\})$ be a consistent decision table with two attributes a and b. Assume that the decision d classifies U into m classes. Such a decision table can be represented as a set of points $S = \{(a(u_i), b(u_i)) : u_i \in U\}$ in a plane, painted by m colors. The task is to find a set of horizontal and vertical lines that divide the plane into the minimum number of rectangular regions, such that every one contains points of the same color.

    Let L be a set of all possible horizontal and vertical lines for the set of points S. The main idea of the algorithm is to reduce L by removing from it many useless lines without loss of consistency (defined in the study). The use of a partition line l is characterized by the number of regions

defined by l and the number of point pairs discerned by l. l is *useless* if both numbers are 'small'. A given region is adjacent to line l if l is one of its boundaries. Every line l has a function $Density(l)$ being a density of regions adjacent to l. Let R^l_{left} and R^l_{right} be the sets of points belonging to regions adjacent to l on the left and right respectively. Let $N(l)$ be the number of regions adjacent to l. The density function is defined as follows:

$$Density(l) = \frac{cardinality(R^l_{left}) + cardinality(R^l_{right})}{N(l)}.$$

For every partition line l, define two functions measuring its global and local discernibility degrees as follows:

$$GlobalDisc(l) = cardinality\{(u,v) : d(u) \neq d(v), u \in R^l_{left}, v \in R^l_{right}\};$$

$$LocalDisc(l) = cardinality\{(u,v) : d(u) \neq d(v) \wedge u, v \text{ are discernible by } l \text{ only}\}.$$

A line l can be rejected if $LocalDisc(l) = 0$. A line l is a preferable candidate to be rejected if both $Density(l)$ and $GlobalDisc(l)$ are 'small'. The algorithm is in four steps:

(a) Start with the set L with all possible lines;

(b) Find a partition line l with $LocalDisc(l) = 0$ of minimum values of $Density(l)$. If there are several such lines then select the one with the minimum value of $GlobalDisc(l)$;

(c) Set L to $L - \{l\}$ and update the structure of the regions;

(d) If L is reducible (if L contains redundant lines (such ones with LocalDisc = 0) that can be rejected from L), then go to step (b). Otherwise stop.

17. **Cost-sensitive discretization** [Brijs and Vanhoof 1998]. This method by introducing a misclassification cost function, aims at solving an important deficiency of error-based discretization, that is, error-based discretization will never generate two adjacent intervals when in both intervals a particular class prevails even when the class frequency distributions differs in both intervals [Kohavi and Sahami 1996]. Discretizing a quantitative attribute involves searching for a discretization of the attribute value range that minimizes a given cost function. The specification of this cost function depends on the costs assigned to the different error types which show their relative importance against each other. For an authentic dataset the cost parameters can be found in the business context of the dataset. However, in many cases, exact cost parameters are not known, thus is assigned by the user. This study was developed in the context of detecting bankruptcy. Candidate cut points are evaluated against the cost function to minimize the overall misclassification cost of the false positive and false negative errors instead of just the total sum of the errors. False positive (respectively false negative) errors in the study are companies incorrectly classified as not bankrupt (bankrupt) although they actually are bankrupt (not bankrupt).

First, all boundary cut points [Fayyad and Irani 1993] for the attribute under discretization are found out as candidate cut points. Second, the

cost parameters of the cost function are constructed to specify the cost of making false positive and false negative errors. Third, costs are calculated and assigned to all potential intervals (split as well as merge) by multiplying the false positive (respectively false negative) cost by the false positive (false negative) errors made as a result of assigning one of the two classes to the interval and picking the minimal cost of both assignments. Fourth, the maximum number of intervals k is specified. The value of k may depend on the problem being studied, but the author suggested that it is advisable to keep k relatively low. Fifth, a shortest route network can be constructed from all potential intervals with their corresponding minimal costs. Finally the shortest route linear programming approach can be applied on the network to identify the optimal number and placement of the intervals that yield the overall minimal cost for the discretization of the quantitative attribute.

The cost-sensitive discretization thus does not suffer the above-mentioned deficiency of error-based discretization because it can take into account the class frequency difference of two intervals. By increasing the error-cost of the minority class, the frequency of the minority class is leveraged so that, eventually, different class labels will be assigned to both intervals, indicating a potential cut point.

18. **LVQ-based discretization** [Perner and Trautzsch 1998]. This method is based on learning vector quantization (LVQ) [Kohonen 1995]. LVQ is a supervised algorithm. It attempts to define class regions in the input data space. Initially, a number of codebook vectors (for details, refer to

the original paper) W_i labelled by a class are placed into the input space. Usually several codebook vectors are assigned to each class. After the initialization of the neural net, each learning instance **X**, as an input vector, is presented one or several times to the net. **X** will be compared to all codebook vectors in order to find the closest codebook vector W_c. If **X** represents the same class as W_c, the learning algorithm will try to optimize the similarity between the codebook vectors and the learning instances by shifting W_c in the direction of **X**. If W_c and **X** have different classes, W_c gets shifted away from **X**, so that the similarity between these two decreases. All other codebook vectors remain unchanged. The following equations represent this idea:

$$\begin{aligned}
\text{for identical classes} &: W_c(t+1) = W_c(t) + \alpha(t) \cdot [X(t) - W_c(t)]; \\
\text{for different classes} &: W_c(t+1) = W_c(t) - \alpha(t) \cdot [X(t) - W_c(t)]; \\
\text{for } W_j \text{ other than } W_c &: W_j(t+1) = W_j(t).
\end{aligned}$$

LVQ-based discretization employs the LVQ algorithm to carry out discretization during decision tree learning, either combined with or without the minimal description length principle. A potential cut point might be in the middle of the learned codebook vectors of two different classes. Since the algorithm tries to optimize the misclassification probability, good classification performance can be expected. However the proper initialization of the codebook vectors and the choice of learning rate $\alpha(t)$ are crucial problems.

19. **Histogram-based discretization** [Perner and Trautzsch 1998]. This method carries out discretization during decision tree learning. To discretize a quantitative attribute A, the distribution $p(A|A \in C_k)p(C_k)$ of attribute A according to class C_k is calculated. The curve of the distribution is approximated by a first order polynom and the minimum square error method is used for calculating the coefficients:

$$\begin{aligned} E &= \sum_{i=1}^{n}(a_1 x_i + a_0 - y_i)^2; \\ a_1 &= \frac{\sum_{i=1}^{n} x_i \cdot i}{\sum_{i=1}^{n} i^2}. \end{aligned}$$

a_0 is either the starting point of this interval or the last point of the preceding interval.

The cut points are selected by finding two maxima of different classes situated next to each other.

Another version of this method can further combine with the entropy-based minimization criteria. The potential boundary cut points [Fayyad and Irani 1993] are determined by finding the peaks of the distribution. If two peaks belong to different classes, the entropy-based minimization criteria is used in order to find the exact cut point between these two classes by evaluating each boundary point K with $Peak_i \leq K \leq Peak_{i+1}$ between these two peaks.

20. **Evolutionary discretization** [Kwedlo and Kretowski 1999]. This method was implemented in EDRL-MD, an evolutionary-algorithm-based system for learning decision rules. EDRL-MD simultaneously searches for

the cut points for all quantitative attributes during the generation of decision rules. An evolutionary algorithm (EA) [Michalewicz 1996] is used. The success of EA is attributed to the ability to avoid local optima, which is its main advantage over greedy search methods. A decision rule R takes the form $t_1 \wedge t_2 \wedge \cdots \wedge t_r \rightarrow c_k$, where c_k is the kth class and t_r is the rth attribute. A rule set RS^{c_k} for a class c_k is defined as a disjunction of decision rules whose right hand side is c_k. In EDRL-MD the EA is called separately for each class c_k to find the rule set RS^{c_k}. Each rule set is encoded as a concatenation of fixed-length strings as illustrated in Fig. 4.1. Each string presents the left hand side of one decision rule. Because EA is called to find a rule set for the given class c_k, there is no need for encoding the right hand side. Each string is composed of N sub-strings where N is the number of attributes. Each sub-string encodes a condition related to one attribute. In case of a quantitative attribute A_i, the substring encodes the lower l_i and the upper u_i cut point of the condition $l_i < A_i \leq u_i$. Both l_i and u_i are selected from the finite set of all boundary cut points [Fayyad and Irani 1993]. For a qualitative attribute, the sub-string consists of binary flags, each of which corresponds to one value of the attribute. Each RS^{c_k} is initialized using a randomly chosen positive instance, that is, the initial rule set consists of a single rule which covers the instance. Then six genetic operators are employed to produce rules: *changing condition*, *positive instance insertion*, *negative instance removal*, *rule drop*, *crossover* and *rule copy* (details in paper). These operators form the cut points of the quantitative attributes at the same time as when the rule set is formed.

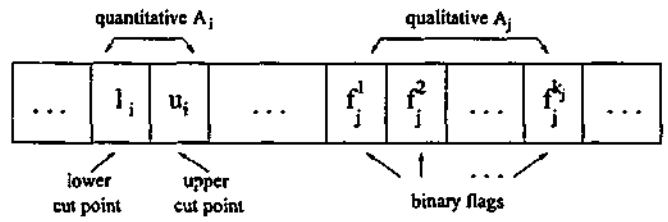


Figure 4.1: String encoding the left hand side of a decision rule

The search criterion, called *fitness function* in EA is given by:

$$f(RS^{c_k}) = \frac{pos - neg}{\log_{10}(L+\alpha)+\beta},$$

where *pos* is the number of positive instances, *neg* is the number of negative instances, L is the total number of conditions in RS^{c_k}, and α=β=10 is chosen on the experimental basis. Note that the maximization of the numerator is equivalent to maximization of the probability of correct classification of an instance. The denominator is a measure of complexity of RS^{c_k}. An increase of the complexity results in a reduction of the fitness and thus the rules tends to overfit. Thus the search criterion prefers rule sets consisting of few conditions, which cover many positive instances and very few negative ones.

21. **Max-m discretization** [An and Cercone 1999]. This method was developed in the context of learning classification rules. The authors argued that the entropy minimization heuristic discretization with minimum description length principle (MDLP) as stopping criterion [Fayyad and Irani 1993] might not be appropriate. One possible reason is that, when the training set is small, the instances are not sufficient to make the

MDLP criterion valid and meaningful so that the criterion causes the discretization process to stop too early before producing useful cut points. Another possible reason is that, even if the recursive MDLP method is applied to the entire instance space to find the first cut point, it is applied 'locally' in finding subsequence cut points due to the recursive nature of the method. Local regions represent smaller samples of the instance space and the estimation based on small samples using the MDLP criterion may not be reliable. Accordingly Max-m discretization was proposed. It uses the same entropy criterion as Fayyad and Irani [1993] did for selecting cut points before rule induction, but it simply chooses a maximum number of m entropy-lowest cut points without recursive application of the method. m is set to be $max(2, k * \log_2 l)$, where k is the number of classes and l is the number of distinct observed values for the attribute being discretized. The empirical results showed that MDLP was not superior to Max-m in most of the tested data sets.

The post-hoc analysis of Max-m discretization uncovered that for several attributes the selected cut points concentrated on a small local area of the entire value space. This problem might be caused by the way that Max-m discretization selects cut points. Max-m first selects the cut point that has the lowest entropy and then selects as the next one the point with the second lowest entropy, and so on. This strategy may result in a large number of cut points being selected near the first cut point because they have entropy values closer to the entropy value of the first cut point than the cut points located far from the first cut point. Thus the selected cut points are among a small area and offer very little additional discrimi-

nating power because the difference between them involves only a few instances.

22. **EDA-DB** [An and Cercone 1999]. This method, entropy-based discretization according to distribution of boundary cut points, is a revised version of Max-m discretization [An and Cercone 1999]. It was proposed to avoid selecting cut points only within a small area which is a deficiency of max-m discretization [An and Cercone 1999]. It chooses cut points according to both information entropy and the distribution of boundary cut points (defined by Fayyad and Irani [1993]) over the instance space. Similar to max-m discretization, EDA-DB selects a maximum number of m cut points, where m is defined as in the max-m method. However rather than taking the first m entropy-lowest cut points, EDA-DB divides the value range of the attribute into intervals and selects in ith interval m_i cut points based on the entropy calculated over the entire instance space. m_i is determined by estimating the probability distribution of the boundary cut points over the instance space.

    To discretize a quantitative attribute A, let l be the number of distinct observed values of A, b be the total number of the boundary cut points of A, and k be the number of classes in the training data, EDA-DB does:

    (a) Calculate m as $max\{2, k * \log_2(l)\}$;

    (b) Estimate the probability distribution of boundary cut points:

        i. Divide the value range of A into d intervals, where $d = max\{1, \log_2(l)\}$;

ii. Calculate the number b_i of boundary cut points in each interval iv_i, where i=1,2,$\cdots$,d and $\sum_{i=1}^{d} b_i = b$;

iii. Estimate the probability of boundary cut points in each interval iv_i as $p_i = \frac{b_i}{b}$;

(c) Calculate the quota q_i of cut points for each interval iv_i by $q_i = p_i * m$;

(d) Rank the boundary cut points in each interval by increasing order of the class information entropy of the partition induced by the boundary point. The entropy for each point is calculated globally over the entire instance space;

(e) For each interval iv_i, select the first q_i points in the above ordered sequence. A total of m cut points are selected.

23. **Dynamic qualitative discretization** [Mora, Fortes, Morales, and Triguero 2000]. This method was developed for data analysis by time series whose goal was to find models which were able to reproduce the statistical characteristics of the series and to use the models to predict next values of the series from its predecessors. Most models are restricted to input attributes with qualitative values so that quantitative attributes need to be discretized. Typical discretization algorithms form discretized intervals according to statistical stationary properties of the values, not taking into account the evolution of these values. Contrarily, this method is called dynamic since the qualitative value associated to a particular quantitative value can change along the time, that is, the same quantitative value can be discretized into different values, depending on the previous values observed in the series. This method is also called

qualitative since only those changes which are qualitatively significant appear in the discretized series. Two approaches are individually proposed to implement dynamic qualitative discretization.

The first approach is to use statistical information about the preceding values observed from the series to select the qualitative value which corresponds to a new quantitative value of the series. The new quantitative value will be associated to the same qualitative value as its preceding values if they belong to the same population. Otherwise, it will be assigned a new qualitative value. To decide if a new quantitative value belongs to the same population as the previous ones, a statistic with Student's t distribution is computed.

The second approach is to use distance functions. Two consecutive quantitative values, v_i and v_j, correspond to the same qualitative value when the distance between them ($distance(v_i, v_j) = |v_i - v_j|$) is smaller than a threshold significant distance (defined in the study). The first quantitative value of the time series is used as reference value. The next values in the series are compared with this reference. When the distance between the reference and a specific value is greater than the threshold (there is a significant difference between them), the comparison process stops. For each value between the reference and the last value which has been compared, the following distances are computed: distance between the value and the first value of the interval, and distance between the value and the last value of the interval. If the former is lower than the latter, the qualitative value assigned is the one corresponding to the first value; otherwise,

the qualitative value assigned is the one corresponding to the last value.

24. **Relative unsupervised discretization** [Ludl and Widmer 2000a]. This method was designed with a view to association rule mining. It is unsupervised because there is no class attribute in rule mining. But the cut points for one quantitative attribute are constructed dependent on all the other attributes. The causal intuition is that for a particular quantitative attribute, a good 'discretization' should create cut points that correlate strongly with changes in the value distributions of other attributes. It comprises three steps, preprocessing, structure projection and postprocessing. Suppose the attribute to be discretized is the *target attribute*, and attributes other than target attributes are the *source attributes*. In the preprocessing step, all source attributes are discretized via some unsupervised discretization method. In structure projection, for each source attribute a_i, the training instances are filtered so as to group by the different values of a_i; for each such filtering, a clustering procedure is performed on values of the target attribute, and the cut points thereby created are gathered. In the postprocessing step, the cut points are merged according to some pre-defined standard so that they lie far enough from each other. The clustering algorithm has its root in the concept of *edge detection* in gray scale image processing. It opens a 'window' of a fixed size around each value in an ordered sequence and determines whether this value lies at an 'edge'.

The same discretization method is also used for a regression task which predicts an exact quantitative target value [Ludl and Widmer 2000b].

The basic idea is to discretize the target attribute by splitting its range into some pre-defined number of intervals and learn to classify examples with a classification learner (for example, C4.5). Then the median of the predicted interval is returned as the exact target value for the test instance.

25. **Multivariate discretization** [Bay 2000]. This method was developed for set mining whose emphasis is on finding previously unknown and insightful patterns in data. Univariate discretization methods which consider only a single attribute at a time are sub-optimal for set mining since they can destroy hidden patterns among attributes. As a result, discretization should consider the effects on all attributes. Two value ranges X and Y of a quantitative attribute should only be in the same interval after discretization if the instances in those ranges have similar multivariate distributions (F_x, F_y) across all attributes and combinations of attributes.

    A contrast set miner STUCCO [Bay and Pazzani 1999] is used to test the difference between two multivariate distributions. Given two groups of instances G_1 and G_2, STUCCO can find all conjunctions of attribute value pairs C meeting two constraints: $P(C|G_1) \neq P(C|G_2)$, and $|support(C|G_1) - support(C|G_2)| \geq \delta$. *support* is the percentage of examples where C is true for the given instance group and δ is set as 0.01 denoting the minimum allowed difference of support. If any C is returned by STUCCO, F_x is deemed substantially different from F_y. Otherwise F_x is similar to F_y.

The discretization process is:

(a) Initially discretize all quantitative attributes using equal width discretization or equal frequency discretization [Catlett 1991; Kerber 1992; Dougherty, Kohavi, and Sahami 1995].

(b) Select two adjacent intervals X and Y that have the minimum combined support and do not have a known cut point between them as candidates for merging.

(c) If F_x is similar to F_y, merge X and Y into a single interval. Otherwise place a cut point between them.

(d) If there are no intervals to merge, stop. Otherwise go to step (a).

26. **Fuzzy discretization** [Ishibuchi, Yamamoto, and Nakashima 2001]. Fuzzy discretization was proposed for generating linguistic association rules. It is motivated by the fact that many linguistic terms cannot be appropriately represented by an interval. For example 'young', 'middle age' or 'old' may not be well defined on intervals with sharp cut points. Each quantitative attribute can be discretized into homogeneous fuzzy intervals using symmetric triangular membership functions. It also can be discretized into inhomogeneous fuzzy intervals based on the entropy criterion. Each quantitative value has a *compatibility grade* with the linguistic terms resulting from the discretization. Such a compatibility grade is mathematically described by a membership function in fuzzy logic. Experiments showed that the generalization ability of linguistic rules with fuzzy discretization is better than that of standard association rules with non-fuzzy discretization.

27. **MODLEM** [Grzymala-Busse and Stefanowski 2001]. MODLEM is a rule induction algorithm which performs discretization and rule induction simultaneously, so that it can be directly applied to quantitative data. For each quantitative attribute q, values are sorted in increasing order. The discretization considers only boundary cut points [Fayyad and Irani 1993]. Any candidate cut point is evaluated and the best cut point v is found out, either using the minimal class entropy technique [Fayyad and Irani 1993] or using Laplacian accuracy [Clark and Boswell 1991]. Once the best cut point is obtained, one of the two conditions $q < v$ or $q \geq v$ that covers more positive instances is chosen. The same procedure is repeated for each attribute. The best condition for all compared attributes is chosen as a new condition of the rule. If it is not sufficient for completing the rule, the strategy is repeated until the complete rule is induced. Since MODLEM discretizes quantitative attributes during the rule induction, the search space is bigger than when generating rules from already discretized attributes. Thus the rules induced by MODLEM are simpler and stronger.

28. **Ordinal discretization** [Frank and Witten 1999; Macskassy, Hirsh, Banerjee, and Dayanik 2001]. Ordinal discretization aims at taking advantage of the ordering information implicit in quantitative attributes, so as not to make values 1 and 2 as dissimilar as values 1 and 1000. Most discretization methods transform quantitative attributes into nominal ones (qualitative attributes without order). Frank and Witten [1999] proposed a transformation of discretized data that is able to preserve the ordering

information. For each discretized attribute A^* with n values $(v_1, v_2, \cdots, v_n)$, $n-1$ boolean ones are introduced, one for each of the attribute's first $n-1$ values. The ith boolean attribute represents the test $A^* \leq v_i$. These boolean attributes are substituted for the original discretized attribute and are input to the learning process.

A very similar idea was proposed by Macskassy, Hirsh, Banerjee, and Dayanik [2001]. By introducing the boolean attributes, each value is converted into a bag of tokens. The closer two values are, the more overlapping their bags. The degree of 'overlap' measures the ordinal dissimilarity between two quantitative values. Information-retrieval-based text classification methods then apply to these converted quantitative attributes.

4.3 Summary

In this chapter, we have conducted a literature review of 34 discretization methods. We summarize in Table 4.2 all these methods in terms of the taxonomies presented in Section 2.3. The methods are sorted by the year that they were published. Some explanations of Table 4.2 are:

1. Abbreviations are used for the names of some taxonomies because of the space limits. 'Sup.' is supervised. 'Uns.' is unsupervised. 'Uni.' is univariate. 'Mul.' is multivariate. 'P.' is parametric. 'Np.' is non-parametric. 'H.' is hierarchical among which 'H. (Sp.)' is split and 'H. (Me.)' is merge. 'Nh.' is non-hierarchical. 'D.' is disjoint. 'Nd.' is non-disjoint.

2. Methods with both 'P.' and 'Np.' ticked can be applied in either way.

3. Methods with both 'H. (Sp.)' and 'H. (Me.)' ticked use the combination of the two processes, for example, they initially discretize by splitting, then post-process by merging.

4. For a composite method, if it is flexible in terms of a taxonomy under the preliminary discretization, the taxonomy will not be ticked; otherwise the taxonomy will be ticked.

In the next chapter, we will combine Chapter 3 and Chapter 4 to suggest that existing discretization methods have potential problems if employed in naive-Bayes learning. We will accordingly present our research on devising new discretization techniques that are more appropriate for naive-Bayes classifiers.

Index	Methods	Primary															Composite
		Sup.	Uns.	Uni.	Mul.	P.	Np.	H. (Sp.)	H. (Me.)	Nh.	Global	Local	Eager	Lazy	D.	Nd.	
1	Equal width		√	√		√				√	√		√		√		
2	Equal frequency		√	√		√				√	√		√		√		
3	Discretizer-2	√		√		√	√	√			√		√		√		
4	ChiMerge	√		√		√	√		√		√		√		√		
5	Fuzzy learning			√							√		√			√	√
6	1R	√		√		√				√	√		√		√		
7	Entropy minimization	√		√			√	√			√		√		√		
8	Error-based	√		√		√					√		√		√		√
9	Cluster-based	√			√		√	√	√		√		√		√		
10	Iterative-improvement			√				√	√		√		√		√		√
11	StatDisc	√		√			√		√		√		√		√		
12	Compression-based	√		√							√		√		√		√
13	InfoMerge	√		√		√	√		√		√		√		√		
14	K-means clustering		√	√			√		√		√		√		√		
15	Search-based												√		√		√
16	Distance-based	√		√			√			√	√		√		√		
17	Zeta	√		√		√		√			√		√		√		
18	ConMerge	√			√		√		√		√		√		√		
19	Dynamic	√			√		√	√			√		√		√		
20	Berka's	√		√		√			√		√		√		√		
21	Semi-optimal	√			√		√		√		√		√		√		
22	Cost-sensitive	√		√		√		√	√		√		√		√		
23	LVQ-base	√			√	√	√			√	√		√		√		
24	Histogram-based	√		√			√	√				√	√		√		
25	Evolutionary	√			√	√				√		√	√		√		
26	Max_m	√		√		√				√	√		√		√		
27	EDA-DB	√		√		√				√	√		√		√		
28	Lazy													√		√	√
29	Dynamic qualitative		√	√			√	√	√			√		√		√	
30	Relative unsupervised		√		√	√		√	√		√		√		√		
31	Multivariate		√		√		√	√	√		√		√		√		
32	Fuzzy			√							√		√			√	√
33	MODLEM	√		√			√	√				√	√		√		
34	Ordinal			√							√		√		√		√

Table 4.2: Discretization Methods Review

Chapter 5

Improving discretization effectiveness for naive-Bayes learning

Chapter 3 has addressed the characteristics of naive-Bayes learning and the working mechanism of discretization. We have argued that there are no universally optimal discretization methods, and thus the best one can hope for is approaches that work well with certain types of real-world applications. However, because usually the nature of the real-world data is not well known, it is difficult to categorize a particular application into a certain type and accordingly choose a corresponding discretization method (if there is any). In this situation, there may be two alternative approaches. One approach is to try every available discretization method and choose the one that is proved to be most effective in the training data of this application. Another approach is to choose a discretization method that has been approved effective in a wide-range of applications. We suggest that the first approach is not practical concerning that there exist a large number of discretization methods, as we have reviewed in Chapter 4. In contrast, the second approach can be both

feasible and reasonable. Thus although a universally optimal method is not achievable, discretization techniques that can work well for a wide-range of real-world applications are useful and desirable.

However, combining what we have learned from the previous two chapters, in this chapter we will argue that existing discretization methods have potential problems if employed for naive-Bayes learning, and thus are less likely to have widespread effectiveness. Thus naive-Bayes classifiers call for new, special-purpose, discretization techniques. Spurred by these understandings, we aim at devising new discretization techniques that improve both naive-Bayes classification efficiency and efficacy for a wide-range of applications. Since we have valued the *bias-variance* characteristic of discretization, our new methods will focus on managing discretization bias and variance. We argue that our techniques can be effective by explaining their working mechanisms. We then argue that they can be efficient by calculating their computational time complexity.

5.1 Why to improve discretization effectiveness

From our literature review presented in Chapter 4, we can see that the majority of the existing discretization methods were designed for learning contexts other than naive-Bayes learning. Naive-Bayes classifiers are probabilistic, selecting the class with the highest probability given the instance to be classified. It is plausible that it is less important to form intervals dominated by a single class for naive-Bayes classifiers than for decision trees or decision rules. Thus discretization methods that pursue pure intervals (containing instances dom-

inated by a single class) [Catlett 1991; Kerber 1992; Fayyad and Irani 1993; Holte 1993; Richeldi and Rossotto 1995; Freitas and Lavington 1996; Ho and Scott 1997; An and Cercone 1999] might not suit naive-Bayes classifiers. Besides, naive-Bayes classifiers deem attributes conditionally independent of each other given the class, so there is no need to calculate the joint probabilities of multiple attribute-values. Thus discretization methods that seek to capture inter-dependencies among attributes [Chmielewski and Grzymala-Busse 1996; Gama, Torgo, and Soares 1998; Monti and Cooper 1998; Perner and Trautzsch 1998; Wang and Liu 1998; Kwedlo and Kretowski 1999; Bay 2000; Ludl and Widmer 2000a] might be less applicable to naive-Bayes classifiers. Furthermore, methods that were designed to capture the ordinal information of discretized attributes [Frank and Witten 1999; Macskassy, Hirsh, Banerjee, and Dayanik 2001] will create a large number of inter-dependent attributes, hence they are likely to compound naive-Bayes' attribute inter-dependence problem when its independence assumption is violated. Thus they are not appropriate for naive-Bayes learning. In summary, it is plausible that those previous techniques may not well suit naive-Bayes learning.

As discussed in our literature review, there also exist a very small number of discretization methods that were developed particularly for naive-Bayes classifiers. We have analyzed their behavior in naive-Bayes learning in terms of discretization bias and variance in Chapter 4. Although these methods take into consideration the nature of naive-Bayes learning, they still have potential problems. One weakness of many methods is that they produce a fixed number of intervals, or tend to minimize the number of intervals. However, we have argued that when the true distribution of $p(C|X_i)$, and thus the true

decision boundaries of X_i are unknown, it is advisable to form as many intervals as constraints on adequate probability estimation accuracy allow. Thus these methods' strategy of interval number might not be desirable. Another weakness, as a consequence of the first one, is that when the training data size increases, the interval frequency increases but the interval number does not tend to. Thus when the training data size becomes large, the increase in discretization bias tends to overshadow the decrease in discretization variance, and leads to an increase in the learning error. This contradicts our normal expectation that the more data we have, the better we learn; and is of particular disadvantage since large data are increasingly common in modern classification applications. Besides, another disadvantage of some methods is that they tend to incur high computational overhead. Thus they are inappropriate for naive-Bayes classifiers whose classification efficiency is one key factor of their popularity. Concerned by these weaknesses, we suggest that these existing discretization methods for naive-Bayes learning do not work effectively or efficiently enough.

This shortage of appropriate discretization techniques is further exacerbated by the widespread employment of naive-Bayes classifiers. Thus we believe that there is a real and immediate need for improving discretization effectiveness for naive-Bayes learning. This motivated our research into specially tailored discretization methods for naive-Bayes classifiers, which will be the focus of the remainder of this chapter.

5.2 Manage discretization bias and variance

We have valued the *bias-variance* characteristic of discretization. We have argued that discretization methods that can well manage discretization bias and variance can be of great utility for naive-Bayes learning. In particular, we have provided the insights that the interval frequency and interval number formed by a discretization method can affect that method's discretization bias and variance. Also, a number of previous authors have realized that the interval frequency and interval number have a major effect on the naive-Bayes classification error. Pazzani [1995] and Mora, Fortes, Morales, and Triguero [2000] have mentioned that if the interval number is too small, important distinctions are missed; if it is too big, the probability estimation may become unreliable. Torgo and Gama [1997] have noticed that an interesting effect of increasing the interval number is that after some threshold the learning algorithm's performance decreases. They suggested that it might be caused by the decrease of the interval frequency leading to unreliable estimates due to overfitting the data. Gama, Torgo, and Soares [1998] have suggested that discretization with fewer intervals tends to have greater utility. By minimizing the number of intervals, the dependence of the set of intervals on the training data will be reduced. This fact will have positive influence on the variance of the generated classifiers. In contrast, if there are a large number of intervals, high variance tends to be produced since small variation on the training data will be propagated on to the set of intervals. Hussain, Liu, Tan, and Dash [1999] have proposed that there is a trade-off between the interval number and its effect on the accuracy of classification tasks. A lower number can improve

'understanding' of an attribute but lower learning accuracy. A higher number can degrade 'understanding' but increase learning accuracy. Hsu, Huang, and Wong [2000, 2003] have observed that as the interval number increases, the classification accuracy will improve and reach a plateau. When the interval number becomes very large, the accuracy will drop gradually. How fast the accuracy drops will depend on the size of the training data. The smaller the training data size, the earlier and faster the accuracy drops. As a result, we anticipate that one effective way to manage discretization bias and variance is to adjust interval frequency and interval number. Accordingly we propose three new discretization techniques, *proportional discretization*, *fixed frequency discretization*, and *non-disjoint discretization*. To the best of our knowledge, these are the first techniques that explicitly manage discretization bias and variance by tuning interval frequency and interval number.

5.2.1 Proportional discretization

Since a good learning scheme should have both low bias and low variance [Moore and McCabe 2002], it would be advisable to equally weigh discretization bias reduction and variance reduction. As we have analyzed in Chapter 3, discretization resulting in large interval frequency tends to have low variance but high bias; conversely, discretization resulting in large interval number tends to have low bias but high variance. Thus a way to achieve equal bias reduction and variance reduction is to set interval frequency equal to interval number. This understanding leads to a new discretization methods, *proportional discretization* (PD).

When discretizing a quantitative attribute for which there are n training instances with known values, supposing that the desired interval frequency is s and the desired interval number is t, PD employs (5.1) to calculate s and t. It then sorts the quantitative values in ascending order and discretizes them into intervals of frequency s. Thus each interval contains approximately[1] s training instances with adjacent (possibly identical) values.

$$\begin{aligned} s \times t &= n \\ s &= t. \end{aligned} \tag{5.1}$$

By setting interval frequency and interval number equal, PD equally weighs discretization bias reduction and variance reduction. By setting them proportional to the training data size, PD can use an increase in the training data to lower both discretization bias and variance. As the number of training instances increases, bias can decrease because the interval number increases, thus the decision boundaries of the original quantitative values are more likely to be close to the interval boundaries; while variance can decrease because the interval frequency increases, thus the naive-Bayes probability estimation is more stable and reliable. This means that PD has greater capacity to take advantage of the additional information inherent in large volumes of training data than previous methods.

[1]It is 'approximately' because we should put all identical values into one interval. Thus sometimes the interval frequency has to be bigger than s to accommodate all the identical values.

5.2.2 Fixed frequency discretization

As we have explained in Chapter 3, ideal discretization for naive-Bayes learning should first ensure that the interval frequency is sufficiently large enough so that the error of the probability estimation falls within quantitative data's error tolerance of probability estimation. On top of that, ideal discretization should maximize the interval number so that the formed intervals are less likely to contain decision boundaries. This understanding leads to an alternative approach to managing discretization bias and variance, *fixed frequency discretization* (FFD).

To discretize a quantitative attribute, FFD sets a *sufficient interval frequency*, m. Then it discretizes the ascendingly sorted values into intervals of frequency m. Thus each interval has approximately[2] the same number m of training instances with adjacent (possibly identical) values.

By introducing m, FFD aims to ensure that in general the interval frequency is sufficient so that there are enough training instances in each interval to reliably estimate the naive-Bayes probabilities. Thus FFD can preventing discretization variance from being very high. As we have explained in Chapter 3, the optimal interval frequency varies from test instance to test instance, and varies from application to application. Nonetheless, we have to choose a frequency so that we can implement and evaluate FFD. Particularly, in our study we choose the frequency m as 30 since it is commonly held to be the minimum sample size from which one should draw statistical inferences [Weiss 2002]. By not limiting the number of intervals formed, more intervals can be formed

[2]As explained in PD, 'approximately' is because of the existence of identical values.

as the training data size increases. This means that FFD can make use of additional data to reduce discretization bias. Thus intervals of high bias are not associated with large datasets any more. In this way, FFD can prevent both high bias and high variance.

It is important to distinguish our new method, fixed frequency discretization (FFD) from equal frequency discretization (EFD) [Catlett 1991; Kerber 1992; Dougherty, Kohavi, and Sahami 1995], both of which form intervals of equal frequency. EFD fixes the interval number. It arbitrarily chooses the interval number k and then discretizes a quantitative attribute into k intervals such that each interval has the same number of training instances. Since it does not control the interval frequency, EFD is not good at managing discretization bias and variance. In contrast, FFD fixes the interval frequency. It sets an interval frequency m that is sufficient for the naive-Bayes probability estimation. It then sets cut points so that each interval contains m training instances. By setting m, FFD can control discretization variance. On top of that, FFD forms as many as intervals as constraints on adequate probability estimation accuracy allow, which is advisable for reducing discretization bias.

5.2.3 Non-disjoint discretization

Both PD and FFD, as well as most of the previous methods that we have reviewed in Chapter 4 are *disjoint* discretization techniques. For a quantitative attribute, they partition its value range offered by the training data into *disjoint* (non-overlapping) intervals, and then apply these intervals to the whole set of test instances. However, as we have argued in Chapter 3, the optimal

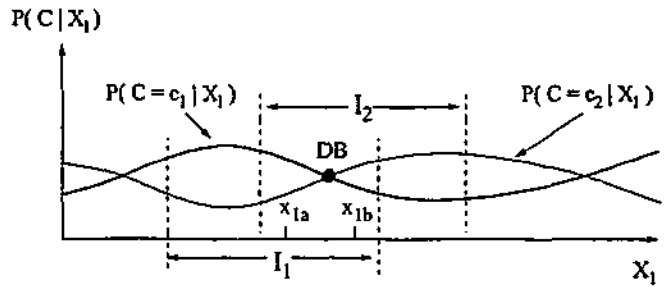


Figure 5.1: Favorable and unfavorable intervals.

discretization of a quantitative attribute varies from value to value, depending on each specific test instance. Thus it is advisable to form an interval most appropriate for the single value offered by the current test instance, instead of forming intervals with respect to all values from the whole set of test instances.

Because of the existence of decision boundaries, discretization can result in one of two types of intervals for a quantitative attribute value: a *favorable* interval and an *unfavorable* interval. Suppose that the present test instance's truly most probable class is c. Its value of the quantitative attribute to be discretized is x_i. A favorable interval composes values such that the most probable class for the majority of the values is c. An unfavorable interval composes values such that c is not the most probable class for the majority of the values. To illustrate these two types of intervals, we demonstrate for example a learning task with two classes c_1 and c_2, and k quantitative attributes. Suppose that one quantitative attribute X_1 is under discretization and the class probability distribution within X_1 given a combination of values of the remaining $k-1$ attributes as depicted in Figure 5.1. Suppose a test instance[3] with value $X_1=x_{1a}$

[3]A test instance here means a test instance having the same combination of values of the remaining $k-1$ attributes as in Figure 5.1.

turns up, whose truly most probable class is c_1. As illustrated in Figure 5.1, I_1 is a favorable interval for x_{1a}, while I_2 is an unfavorable interval of x_{1a}. Choosing c_1 as the class of the test instance will minimize the naive-Bayes classification error under zero-one loss. This is more likely to be achieved by substituting I_1 for x_{1a}. Thus discretization resulting in a favorable interval for a value is more reliable in terms of naive-Bayes probability estimation. However, under disjoint discretization, this is difficult to be *generally* achieved with respect to all the values from the whole set of test instances. If an interval containing a decision boundary is a favorable interval for values on one side of the decision boundary, it will be an unfavorable interval of values on the other side. As also illustrated in Figure 5.1, I_1 is not a favorable interval for value x_{1b}, since the most probable class of x_{1b} is c_2 while the majority of the values in I_1 have the most probable class as c_1.

The above analysis motivates *non-disjoint discretization* (NDD), which forms overlapping (non-disjoint) intervals for a quantitative attribute and always tries to choose a favorable interval for the value provided by the present test instance. Although many learning algorithms require values of an attribute to be disjoint, that is the set of instances covered by one value of X_i^* cannot overlap that covered by another value of X_i^*, naive-Bayes classifiers do not have that requirement. One implementation of NDD is to always locate a value at the middle of its corresponding interval. If the value itself is a decision boundary, each class is equally probable for the instance. Thus there is no truly most probable class. What the most probable class for the interval is chosen does not matter too much. If the interval contains a decision boundary that is not the value, locating this value at the middle means at least more than half of

the value range in this interval has their most probable class the same as this value. Thus this interval is a favorable one for this value. However, this argument is correct only when there is no more than one decision boundary in the interval. If there exist multiple decision boundaries, the most probable class will change each time the value crosses a decision boundary. In this case, locating the value to the middle does not ensure that the instance's most probable class takes up the majority of the values in the interval. Thus another important issue for NDD is the interval frequency strategy. A strategy that is able to exclude most decision boundaries from an interval while still retaining sufficient instances in the interval for reliable probability estimation will be of great utility for NDD. This is again a discretization bias and variance management problem.

When discretizing a quantitative attribute, suppose the number of training instances[4] is n and the desired interval frequency is s, NDD identifies among the sorted values t' *atomic intervals*, $(a'_1, b'_1], (a'_2, b'_2], ..., (a'_{t'}, b'_{t'}]$, each with frequency equal to s', so that[5]

$$\begin{aligned} s' &= \frac{s}{3} \\ s' \times t' &= n. \end{aligned} \tag{5.2}$$

One interval is formed for each set of three consecutive atomic intervals, such that the kth ($1 \leq k \leq t' - 2$) interval $(a_k, b_k]$ satisfies $a_k = a'_k$ and $b_k = b'_{k+2}$. Each value v is assigned to interval $(a'_{i-1}, b'_{i+1}]$ where i is the index of the atomic

[4]We only consider instances with known value of the quantitative attribute.

[5]Theoretically any odd number k besides 3 is acceptable in (5.2) as long as the same number k of atomic intervals are grouped together later for the probability estimation. For simplicity, we take k=3 for demonstration.

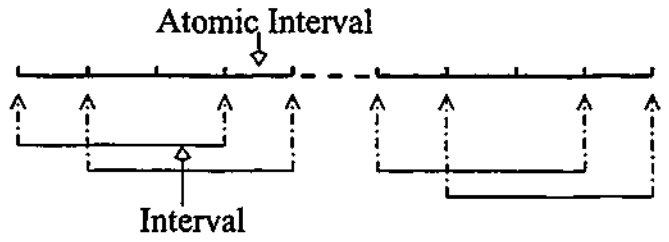


Figure 5.2: *Atomic intervals* compose actual intervals.

interval $(a'_i, b'_i]$ such that $a'_i < v \leq b'_i$, except when i=1 in which case v is assigned to interval $(a'_1, b'_3]$ and when i=t' in which case v is assigned to interval $(a'_{t'-2}, b'_{t'}]$. Figure 5.2 illustrates the procedure.

As a result, each interval has frequency equal to s by comprising three atomic intervals; and except in the case of falling into the first or the last atomic interval, a quantitative value is always towards the middle of its corresponding interval. From the perspective of the whole set of test instances, the discretized intervals formed for two different values of a quantitative attribute might *overlap* with each other. However because the values are used for different test instances independently, NDD's overlapping intervals will not cause any confusion in classification.

As we have explained, how to choose the interval frequency s is important. NDD can be employed with a primary strategy for selecting interval frequency. The strategy can be based on those of many *unsupervised* discretization methods, such as equal frequency discretization [Catlett 1991; Kerber 1992; Dougherty, Kohavi, and Sahami 1995], proportional discretization [Section 5.2.1] and fixed frequency discretization [Section 5.2.2]. In our implementation, we chose s as 30 which equals the *sufficient interval frequency* of FFD, since it had been demonstrated to well manage discretization bias and vari-

ance according to our previous experiments on FFD.

It is important to compare NDD with lazy discretization (LD) [Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003] that we have reviewed in Chapter 4. Since LD locates a value exactly at the middle of its interval, it should be the ideal form of NDD if its computational resources are not a consideration. However, when proposing LD, Hsu et al.'s study did not explain why the scheme of 'placing a value at the middle of its interval' can be effective in naive-Bayes learning. Neither did it recognize the importance of the interval frequency strategy in this scheme. In contrast, our reasoning leading to NDD has explained why LD can be effective at reducing the naive-Bayes classification error, and has argued that the interval frequency is essential for this scheme's success. Besides, Hsu et al. stated that the goal of LD is to save training effort by only working on the interval containing the single value from the current test instance. The other value ranges are ignored since they are not needed for classifying the instance. Ironically, because of its lazy methodology as we have analyzed in Chapter 4, LD tends to have high overhead of computational memory and time. This is especially inappropriate for naive-Bayes learning, which is popular with applications involving large data. In contrast, NDD is an 'eager' approach. It carries out discretization at training time. Thus the training instances can be discarded prior to classification time and require no high memory expenses. It has computational time complexity as low as the simplest discretization methods like EWD or EFD, which we will present in detail in Section 5.3. Thus we anticipate NDD to scale to large data very well.

Table 5.1: Taxonomy of each new method

Method	Taxonomy
PD	primary, unsupervised, univariate, non-parametric, non-hierarchical, global, eager, disjoint
FFD	primary, unsupervised, univariate, parametric, non-hierarchical, global, eager, disjoint
NDD	composite, unsupervised, univariate, non-hierarchical, global, eager, non-disjoint

5.2.4 Summary

In Table 5.1, we summarize each of our new methods in terms of our taxonomies proposed in Section 2.3.

5.3 Time complexity comparison

We here calculate the time complexities of our new discretization methods as well as the previous ones discussed in Section 4.1. Because of their computational efficiency, naive-Bayes classifiers are very attractive for applications with large data. Thus discretization methods for naive-Bayes learning are necessary to be efficient so that they can scale to large data.

To discretize a quantitative attribute, suppose the number of training instances[6], test instances, attributes and classes are n, l, v and m respectively.

- EWD, EFD, FLD, PD, FFD and NDD are dominated by sorting. Their complexities are of order $O(n \log n)$.

- EMD does sorting first, an operation of complexity $O(n \log n)$. It then goes through all the training instances a maximum of $\log n$ times, recursively applying 'binary division' to find out at most $n-1$ cut points. Each time, it will estimate $n-1$ candidate cut points. For each candidate point,

[6]We only consider instances with known value of the quantitative attribute.

probabilities of each of m classes are estimated. The complexity of that operation is $O(mn\log n)$, which dominates the complexity of the sorting and thus results in complexity of order $O(mn\log n)$.

- IID's operators have $O(n)$ possible ways to adjust cut points in each iteration. For each adjustment, the leave-one-out cross validation has complexity of order $O(nmv)$. The number of times the iteration will repeat depends on the initial discretization as well the error estimation. It varies from case to case, which we denote by u here. Thus the complexity of IID is of order $O(n^2mvu)$.

- LD sorts the values once and performs discretization separately for each test instance and hence its complexity is $O(n\log n)+O(nl)$.

Thus EWD, EFD, FLD, PD and FFD have lower complexity than EMD. LD tends to have high complexity when the training or testing data size is large. IID's complexity is prohibitively high when the training data size is large.

5.4 Summary

In this chapter, we have proposed three new discretization techniques, *proportional discretization*, *fixed frequency discretization* and *non-disjoint discretization*. All of them focus on managing discretization bias and variance by adjusting interval frequency and interval number. In addition, non-disjoint discretization is able to form overlapping intervals such that for each quantitative value to be discretized, it is most likely to choose a favorable interval. Theoretically we have anticipated our new techniques to effectively lower the naive-Bayes

classification error. We have also argued that they are very efficient in computational memory and time. These merits are desirable for naive-Bayes learning.

In the next chapter, we will conduct the empirical evaluation to assess the degree to which our new techniques have the merits that we have anticipated in theory.

Chapter 6

Experimental evaluation

In the previous chapter, we have proposed three new discretization techniques, *proportional discretization* (PD), *fixed frequency discretization* (FFD) and *non-disjoint discretization* (NDD). In theory, we have argued that these techniques can be both effective and efficient in naive-Bayes learning.

In this chapter, we empirically validate our arguments, using real-world data. We evaluate whether PD, FFD and NDD can better reduce the naive-Bayes classification error, compared with previous discretization methods, equal width discretization (EWD) and equal frequency discretization (EFD) [Catlett 1991; Kerber 1992; Dougherty, Kohavi, and Sahami 1995], fuzzy learning discretization (FLD) [Kononenko 1992; Kononenko 1993], entropy minimization discretization (EMD) [Fayyad and Irani 1993] and lazy discretization (LD) [Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003][1]. Since iterative-improvement discretization (IID) [Pazzani 1995] tends to have high computational complexity, while our experiments frequently involve large datasets (up to 166 quantitative attributes and up to half million

[1]EWD, EFD and FLD are implemented with the parameter k=10. The original LD in Hsu et al.'s implementation chose EWD with k=10 as its primary discretization method. That is, it forms *interval width* equal to that produced by EWD with k=10. Since we manage discretization bias and variance through interval frequency (and interval number), which is relevant but not identical to interval width, we implement LD with EFD being its primary method. That is, LD forms *interval frequency* equal to that produced by EFD with k=10.

training instances), we do not implement IID for the sake of feasibility. Due to the importance of its efficiency in many applications, we instead focus on computationally efficient techniques of discretization for naive-Bayes learning.

First, we describe our experimental data, design and statistics. Then, we report the experimental results and analyze what information we can learn from the results.

6.1 Data

We run our experiments on 29 natural datasets from UCI machine learning repository [Blake and Merz 1998] and KDD archive [Bay 1999]. This experimental suite comprises 3 parts. The first part is composed of all the UCI datasets used by Fayyad and Irani [1993] when publishing the entropy minimization discretization (EMD). The second part is composed of all the UCI datasets with quantitative attributes used by Domingos and Pazzani [1996] for studying naive-Bayes classifiers. In addition, as large data are becoming more and more common in modern applications, and the first two parts are mainly confined to small data size, we further augment our data collection with datasets that we can identify containing quantitative attributes, with emphasis on those having more than 5000 instances. Table 6.1 describes each dataset, including the number of instances (Size), quantitative attributes (Qa.), qualitative attributes (Ql.) and classes (Class). The datasets are increasingly ordered by the size.

6.2 Design

In our study, the effectiveness of a discretization method is presented by the performance of naive-Bayes classifiers that are trained on data discretized by this method. We use cross validation to estimate the naive-Bayes classification performance. The performance is recorded as the classification error, bias and variance; and the computational time.

6.2.1 Cross validation

According to Kohavi [1995a], and Kohavi and Provost [1998], in k-fold cross validation, the dataset $\mathcal{D}$ is randomly split into k mutually exclusive subsets (the folds) of approximately equal size, $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_k$. The classifier is trained and tested k times. Each time $t \in \{1, 2, \cdots, k\}$, it is trained on $\mathcal{D} - \mathcal{D}_t$ and tested on $\mathcal{D}_t$. The cross validation estimate of error is the overall number of incorrect classifications, divided by the number of instances in the dataset. Repeating cross validation multiple times (trials) using different splits of the instances into folds provides a reasonable estimate of the error of the single classifier produced from all the instances. Cross validation can be either stratified or unstratified. In stratified cross validation, the folds are stratified so that they contain approximately the same proportion of classes as the original datasets. In unstratified cross validation, the folds are randomly formed without considering the class proportion.

To evaluate a discretization method, for each dataset, we implement naive-Bayes learning by conducting a 10-trial, 3-fold unstratified cross validation. For each fold, the training data are discretized by this method. The intervals

so formed are applied to the test data. We repeat 10 trials because we need classify each instance several times to estimate the classification bias and variance. We form 3 folds for each trial because we try to reduce the computation overhead which otherwise can be heavy because of the many trials[2]. We do not use stratification because we are not sure that the true class distribution of the data will be always the same as the current training data.

6.2.2 Performance metrics

The following metrics are used to record the performance of naive-Bayes classifiers.

- **Classification error** is the percentage of incorrect predictions of naive-Bayes classifiers in the test averaged across all folds in all trials of the cross validation.

- **Classification bias and variance** are estimated by the method described by Webb [2000]. They equate to those defined by Breiman [1996], except that irreducible error is aggregated into bias and variance. An instance is classified once in each trial and hence ten times in all. The central tendency of the learning algorithm is the most frequent classification of an instance. Total error is the classification error defined as above. Bias is that portion of the total error that is due to errors committed by the central tendency of the learning algorithm. This is the portion of classifications that are both incorrect and equal to the central tendency. Variance is that portion of the total error that is due to errors that are deviations

[2]Although naive-Bayes learning itself is very efficient, some discretization methods are not.

from the central tendency of the learning algorithm. This is the portion of classifications that are both incorrect and not equal to the central tendency. Bias and variance sum to the total error.

- **Computational time** is the computational time of discretizing data, and training and testing a naive-Bayes classifier. It is averaged across all folds in all trails of the cross validation.

6.3 Statistics

Three statistics are employed to evaluate the performance of naive-Bayes classifiers.

- **Mean** is the arithmetic mean of the classification error, bias or variance respectively across all datasets. It provides a gross indication of the relative performance of competitive methods. It is debatable whether errors in different datasets are commensurable, and hence whether averaging errors across datasets is very meaningful. Nonetheless, a low mean error is indicative of a tendency towards low errors for individual datasets.

- **Geometric mean** is the geometric mean of the classification error, bias or variance respectively across all datasets. Webb [2000] suggested that geometric mean error ratio is a more useful statistic than mean error ratio across datasets. Geometric mean error functions the same as geometric mean error ratio to indicate the relative performance of two methods. For two methods A and B, the ratio of their geometric mean errors is their geometric mean error ratio. That A's geometric mean error is lower

than B's is equivalent to that the geometric mean error ratio of A against B is smaller than 1, and vice versa.

- **Win/lose/tie record** comprises three values that are respectively the number of datasets for which the naive-Bayes classifier trained with one discretization method obtains lower, higher or equal learning error, compared with the naive-Bayes classifier trained with another discretization method. A one-tailed sign test can be applied to each record. If the test result is significantly low (here we use the 0.05 critical level), it is reasonable to conclude that the outcome is unlikely to be obtained by chance and hence the record of wins to losses represents a systematic underlying advantage of the winning discretization method with respect to the type of datasets studied.

6.4 Results and analysis

We here present and analyze the experimental results. For each alternative discretization method on each dataset, Table 6.1 lists its classification error under the column 'Classification error'; Table 6.2 lists its classification bias and variance under the column 'Classification bias' and 'Classification variance' respectively; and both tables list its mean and geometric mean of corresponding metrics in the row 'ME' and 'GM' respectively. For each new technique that we have proposed, Table 6.3 presents its win/lose/tie records on classification error compared with each previous discretization method.

6.4.1 Proportional discretization (PD)

- PD achieves both lower mean and lower geometric mean of classification error than all previous methods.
- With respect to the win/lose/tie records, PD achieves lower classification error than each previous method with frequency significant at the 0.05 level.
- PD better reduces classification bias than previous methods. It achieves lower mean and lower geometric mean of classification bias. Its advantage in bias reduction grows more apparent with the training data size increasing. This outstanding effectiveness in bias reduction is achieved without incurring high variance. This supports our suggestion that PD can use additional data to decrease both discretization bias and variance by setting both interval frequency and interval number proportional to the training data size.

6.4.2 Fixed frequency discretization (FFD)

- FFD achieves both lower mean and lower geometric mean of classification error than all previous methods.
- With respect to the win/lose/tie records, FFD achieves lower classification error than each previous method with frequency significant at the 0.05 level.
- FFD better reduces classification bias than previous methods. It achieves lower mean and lower geometric mean of classification bias. Its advan-

tage in bias reduction grows more apparent with the training data size increasing. This supports our suggestion that FFD can use additional data to decrease discretization bias, and thus high bias no longer attaches to large training data.

- FFD also demonstrates a good control on discretization variance. Especially among smaller datasets where naive-Bayes probability estimation has a higher risk to suffer insufficient training data, FFD usually achieves lower variance than alternative methods. This supports our suggestion that by controlling the frequency of the interval, FFD can prevent the discretization bias from being very high. However, we have also observed that FFD does have higher variance especially in some very large datasets. We suggest the reason is that m=30 might not be the optimal size for those datasets, since there is no universally optimal interval frequency as we have argued. Nonetheless, the gain through FFD's bias reduction is more than the loss through its variance increase, thus FFD still achieves lower naive-Bayes classification error in large datasets compared to previous discretization methods.

6.4.3 Non-disjoint discretization (NDD)

- NDD achieves both lower mean and lower geometric mean of classification error than all previous methods.

- NDD also achieves the lowest mean and the lowest geometric mean of classification error among our three new techniques.

- With respect to the win/lose/tie records, NDD achieves lower classification error than all previous methods with frequency significant at the 0.05 level.

It is illuminating to compare NDD with FFD. Although we base the frequency strategy of NDD on that of FFD, we expect NDD to obtain additional advantage because of its 'non-disjoint' strategy. This is supported by our experimental results.

- Compared with FFD, the win/lose/tie record of classification error of NDD is 19/5/5. NDD's wins are of frequency significant at the 0.05 level.

It is also important to compare NDD with LD since they are similar in terms of locating a quantitative value towards the middle of its discretized interval. NDD has been demonstrated to significantly outperform the version of LD that has EFD as its primary method. However, this LD's disadvantage might be caused by its frequency strategy which in our experiments is based on that of EFD with k=10. Since NDD's frequency strategy is based on FFD's, and FFD has been shown to be better than EFD at managing discretization bias and variance, for the sake of fair comparison, we here implement another version of lazy discretization, named LD_FFD, whose frequency strategy is based on that of FFD. We consequently augment Table 6.1 and Table 6.2 by adding columns for LD_FFD.

- The win/lose/tie records of NDD against LD_FFD is 13/9/7. The win to loss is not significant at the 0.05 level. Thus when they take the same frequency strategy, NDD and LD have competitive effectiveness at reducing the naive-Bayes classification error. However, from the perspective of

feasibility, NDD is overwhelmingly superior over LD_FFD. For the four largest datasets in out study, Table 6.4 lists the averaged computational time of training and testing a naive-Bayes classifier on data discretized by NDD and LD_FFD respectively per fold out of 10-trial 3-fold cross validation. NDD is much faster than LD_FFD.

6.4.4 Previous methods

We are interested in verifying our analysis of previous discretization methods' effectiveness in terms of discretization bias and variance. We first focus on primary methods. Then we address composite methods.

6.4.4.1 Primary methods

Among previous methods, there are three primary methods: EWD, EFD and EMD. Among them, EMD is supervised while the other two are unsupervised. We analyze their effectiveness, comparing with our new primary methods PD and FFD. We break down our analysis into unsupervised methods and supervised methods. Compared with PD and FFD, we list in Table 6.5 and Table 6.6 each previous primary method's win/lose/tie record of bias and variance respectively.

With respect to previous unsupervised primary methods EWD and EFD, we have suggested that one common weakness is that they fix the interval number ignorant of the training data size. Thus we have predicted that they tend to obtain high discretization bias when the training data size is large. This is particularly undesirable given that applications involving large data

are more and more common. Our experimental results support our prediction. According to Table 6.5, both EWD and EFD obtain higher bias than PD with frequency significant at the 0.05 level. EWD obtains higher bias than FFD with frequency significant at the 0.05 level. Although overall EFD does not lose to FFD in bias reduction with significant frequency, it uniformly obtains higher bias than FFD in all the 14 largest datasets. As for discretization variance, according to Table 6.6, the win/lose/tie records of EWD compared with PD and FFD are not significant at the 0.05 level, which suggests that EWD does not have an advantage in variance reduction. Although EFD obtains lower discretization variance than PD (but not than FFD) with frequency significant at the 0.05 level, this advantage in variance reduction is usually overshadowed by its disadvantage in bias reduction. As a result, EFD is still sub-optimal in reducing the naive-Bayes classification error.

With respect to the previous supervised primary method EMD, according to Table 6.1, it achieves lower mean and lower geometric mean of naive-Bayes classification error than the previous unsupervised primary methods EWD and EFD. This might indicate that EMD enjoys an advantage because it is able to make use of the class information. However, we have suggested that one of EMD's weakness is that it always tends to minimize the interval number. This tends to form intervals of high bias when naive-Bayes learning involves large data. Another weakness, as we have suggested, is that although EMD might be effective at identifying decision boundaries for one-attribute applications, the resulting cut points can easily diverge from the true decision boundaries when in multi-attribute applications where the decision boundaries of one attribute alter depending on the values of the other attributes. Thus we

suggest that EMD might be inappropriate for real-world applications where multi-attribute cases dominate. These suggestions are supported by our observations. According to Table 6.5, EMD obtains higher bias than PD with frequency significant at the 0.05 level. Although without statistically significant frequency across all the datasets, EMD obtains higher bias than FFD in 9 out of the 10 largest datasets. According to Table 6.6, EMD does not demonstrate any significant wins in variance reduction compared with either PD or FFD.

6.4.4.2 Composite methods

Among previous methods, there are two composite discretization methods, FLD and LD. FLD has EWD as its primary discretization method. LD has EFD as its primary discretization method. In Table 6.7, we list the win/lose/tie record of classification error, bias and variance of FLD and LD compared with their own primary method respectively.

FLD first uses EWD to form an interval $(a_i, b_i]$ for a quantitative attribute value x_i. It then estimates $p(a_i < X_i \leq b_i | C{=}c)$ from all training instances rather than only from instances that have values of X_i in $(a_i, b_i]$. The influence of a training instance with value v of X_i on $(a_i, b_i]$ is assumed to be normally distributed with the mean value equal to v. FLD was claimed to be advisable since a slight difference between two values, such that one is above and one is below the cut point, does not have drastic effects on the estimated probabilities, which happens otherwise with 'non-fuzzy' methods. However, we suspect that when the training instances' influence on each interval does not follow the normal distribution which is often the case for real-world applica-

tions, FLD's effectiveness can degrade. Our experimental results suggest that this indeed occurs in practice. According to Table 6.1, compared with its primary method EWD that is 'non-fuzzy', FLD obtains both higher mean and higher geometric mean of the naive-Bayes classification error than EWD. According to Table 6.7, the win/lose/tie record of FLD compared with EWD on classification error is 10/18/1, which demonstrates FLD obtains higher error more often than not compared with EWD. We have also suggested that since FLD takes into consideration all values in a quantitative attribute's value range for the naive-Bayes probability estimation, it is expected to be good at reducing discretization variance but reversely, bad at reducing discretization bias. According to Table 6.2, FLD obtains both higher mean and higher geometric mean of bias than EWD. According to Table 6.7, its win/lose/tie record of bias reduction against EWD demonstrate a significant loss at the 0.05 level. In contrast, it achieves both lower mean and lower geometric mean of variance than EWD; and its win/lose/tie record of variance reduction against EWD demonstrates a significant win at the 0.05 level. However, the bias increase usually outweighs the variance reduction, thus FLD still results in inferior effectiveness of reducing naive-Bayes classification error compared with its primary method EWD.

We are particularly interested in checking LD's effectiveness, since we value the idea of 'placing a quantitative value at the middle of its interval', and deem that LD should be the ideal format of our new strategy NDD if its computational resources are not a consideration. According to Table 6.1, LD achieves both lower mean and lower geometric mean of the naive-Bayes classification variance than its primary method EFD. According to Table 6.7,

its win/lose/tie record against EFD on classification error is significant at the 0.05 level. Most striking is LD's effectiveness on variance reduction. It obtains lower mean and lower geometric mean of classification variance than EFD as well as all the other previous methods. Its win/lose/tie record against EFD on variance reduction is significant at the 0.05 level. These observations support our suggestion that always looking for favorable intervals by placing a quantitative value at the middle of its interval can enjoy an advantage. However, according to Table 6.7, LD does not significantly improve on EFD with respect to classification bias reduction, since the win/lose/tie record is 14/12/3 with sign test equal to 0.42. Also, according to Table 6.2, LD's mean and geometric mean of classification bias are equal to those of EFD, which are higher than most of the other methods. We attribute LD's disadvantage of bias reduction to the fact that LD did not identify the proper interval frequency strategy and only arbitrarily followed that of EFD's. Thus we have proposed LD_FFD and predicted that LD_FFD could better reduce naive-Bayes classification error than LD since it is coupled with FFD's frequency strategy which we believe although not universally optimal, is more advisable than EFD's. The experimental results support our prediction. According to Table 6.1, LD_FFD achieves both lower mean and lower geometric mean of the classification error than LD and all other previous methods. According to Table 6.7, its win/lose/tie against LD on classification error is significant at the 0.05 level. As for bias reduction, according to Table 6.2, it also achieves the lowest mean and the lowest geometric mean of classification bias among the previous methods, which LD failed to. Its win/lose/tie record against LD on bias reduction is significant at the 0.05 level. In addition, compared with LD, LD_FFD achieves competi-

tive effectiveness of variance reduction. The win/lose/tie record according to Table 6.7 is 12/14/3. These observations support our analysis that in order to improve the discretization effectiveness, the frequency strategy is as essential as locating a quantitative value at the middle of its interval.

6.4.5 Further discussion and weighted proportional discretization

PD and FFD manage discretization bias and variance from two different perspectives. Although both have clearly demonstrated advantage over previous discretization methods in reducing the naive-Bayes classification error, there is no statistically significant difference between their own effectiveness with respect to all the datasets. According to Table 6.1, they obtain the same mean errors (18.6%), and obtain very similar geometric mean errors (13.8% and 13.7% respectively). The win/lose/tie record of PD compared with FFD is 15/12/2, which is insignificant with sign test equal to 0.35. However, from the perspective of discretization bias and variance, PD and FFD demonstrate different effectiveness according to different training data sizes. In order to analyze this issue, we split our datasets into 'smaller' ones whose sizes are smaller than 1200; and 'larger' ones whose sizes are larger than 1200. The reason of this split is that in 3-fold cross validation as our experiments have employed, the training data size will be 900 if the dataset size is 1200. FFD with m=30 and PD will produce identical intervals for a quantitative attribute if there are 900 training instances with known values for this attribute. If the dataset size is smaller than 1200 and thus the training data size is smaller than 900, PD

produces smaller interval frequency (larger interval number) than FFD. If the dataset size is larger than 1200 and thus the training data size is larger than 900, PD produces larger interval frequency (smaller interval number) than FFD[3]. In Table 6.8, we list the mean and geometric mean of naive-Bayes classification error, bias and variance of PD and FFD respectively on smaller datasets. We also list the win/lose/tie record of classification error, bias and variance respectively of PD against FFD on smaller datasets in Table 6.8. All the statistics on larger datasets are listed in Table 6.9.

From these statistics we can find that among smaller datasets, PD achieves lower mean and lower geometric mean of naive-Bayes classification error than FFD, although its wins are not of statistically significant frequency with the win/lose/tie record as 11/5/1. It is PD's effectiveness in bias reduction that contributes to its advantage in error reduction. Its mean and geometric mean of classification bias are both lower than those of FFD's. It also obtains lower bias more often than not compared with FFD. While in larger datasets, FFD demonstrates advantage over PD at reducing classification error. It achieves both lower mean and lower geometric mean of error than PD. It also obtains lower error in 7 out of 12 larger datasets. We attribute this advantage to FFD's effectiveness in bias reduction. FFD achieves both lower mean and lower geometric mean of classification bias than PD. Its wins of bias reduction are of significant frequency with sign test equal to 0.03.

However, it can be observed that PD is less effective at reducing variance for smaller dataset. Compared with FFD in smaller datasets, PD obtains higher

[3]This is true only if there is no unknown values for any attribute. Otherwise, it is possible that although there are more than 900 training instances, there are less than 900 known values for some attribute. Hence PD will instead produce smaller interval frequency than FFD.

variance in 11 out of 17 datasets; and it also obtains higher mean and higher geometric mean of classification variance. We suggest that this disadvantage is because that PD produces many intervals of small frequency when the training data size is small. For example, with 100 training instances, FFD will produce 3 intervals containing approximately at least 30 instances each, while PD will produce 10 intervals containing only 10 instances each. On the other hand, it can also be observed that FFD is less effective at reducing variance for larger datasets. Compared with PD in larger datasets, FFD obtains higher classification variance in 9 out of 12 datasets; and it also obtains higher mean and higher geometric mean of classification variance. We suggest that this disadvantage is because although we choose m=30 for FFD in our experiments, 30 can not always be an optimal sufficient interval frequency independent of each specific dataset, as we have explained in Chapter 3.

These understandings raise an interesting question: can we combine PD and FFD together so that each can use its advantage to complement the other's disadvantage? For example, we may set for PD a sufficient interval frequency like that of FFD's so as to constrain PD's discretization variance when the training data size is small. Or we may make the sufficient interval frequency of FFD take into consideration the increasing size of the training data as PD can. Accordingly, we propose *weighted proportional discretization* (WPD). WPD follows PD in terms of setting both interval frequency and interval number proportional to the training data size. However, instead of setting interval frequency equal to interval number, WPD sets a *sufficient interval frequency*, m that follows the variance control strategy of FFD. As the training data size increases, both the interval frequency *above* m and the interval number increase. As a result,

WPD does not equally weigh bias reduction and variance reduction. Instead, WPD weighs variance reduction more than bias reduction by firstly controlling interval frequency to be no less than m, by what reason it is named.

When discretizing a quantitative attribute for which there are n training instances with known values, supposing that the desired interval frequency is s and the desired interval number is t, WPD employs (6.1) to calculate s and t. It then sorts the quantitative values in ascending order and discretizes them into intervals of frequency s. Thus each interval contains approximately[4] s training instances with adjacent (possibly identical) values.

$$\begin{aligned} s \times t &= n \\ s - m &= t. \end{aligned} \tag{6.1}$$

We anticipate that on one hand, WPD can make up PD's disadvantage for smaller datasets since it can prevent the discretization variance from being high by setting interval frequency above m. On the other hand, we anticipate WPD to make up FFD's disadvantage for larger datasets since its interval frequency may have a better chance to be suitable for each specific dataset by reacting to the training data size. We implement WPD the same way as for the other discretization methods. We record the resulting classification error, bias and variance in columns 'WPD' in Table 6.1 and Table 6.2 respectively. The win/lose/tie records on classification error of WPD against alternative methods are listed in Table 6.10.

Our anticipations have been supported by these statistics. Among smaller

[4] Again, 'approximately' is because the existence of identical values.

datasets, WPD's mean and geometric mean of classification variance are 3.3% and 2.2% respectively, both being lower than PD's (3.7% and 2.3% respectively). WPD obtains lower variance than PD more often than not with the win/lose/tie record as 10/5/2. Among larger datasets, WPD's mean and geometric mean of classification variance are 1.7% and 1.2%, both being lower than FFD's (2.0% and 1.5% respectively). WPD obtains lower variance than FFD more often than not with the win/lose/tie record as 8/2/2. Across all the datasets, if compared with every previous discretization methods, WPD is able to achieve the lowest mean and the lowest geometric mean of the naive-Bayes classification error. Its win/lose/tie records against each previous method are all of frequency significant at the 0.05 level. However, if compared with our new methods, WPD does not demonstrate statistically significant advantage over PD or FFD. Another illuminating observation is that with frequency significant at the 0.05 level, WPD loses to NDD in error reduction. We suggest the reason is that WPD produces disjoint intervals; while NDD produces non-disjoint intervals, each of which tends to be a favorable interval for a quantitative attribute value and each of which has a reliable interval frequency for naive-Bayes probability estimation by employing FFD as its primary method.

6.5 Conclusion

This chapter has focused on evaluating our proposed discretization techniques PD, FFD, WPD and NDD against previous key discretization methods in naive-Bayes learning, using a large suite of real-world data.

Both PD and FFD manage discretization bias and variance by adjusting

interval frequency and interval number. Our experimental results have suggested that compared with previous methods, PD and FFD enjoy an advantage at reducing the naive-Bayes classification error with statistically significant frequency; and compared with each other, there is no statistically significant difference between PD and FFD's effectiveness of error reduction. However, since they take different approaches to managing discretization bias and variance, PD and FFD have demonstrated different effectiveness according to different training data sizes. We have suggested that it is sensible to combine PD and FFD's strategies since each one's advantages can make up the other's disadvantages. This understanding leads to another discretization technique WPD. The experimental results have shown that WPD also achieves lower naive-Bayes classification error than every previous method with significant frequency. Again there is no significant difference among the effectiveness of WPD, and PD and FFD. This observation supports our analysis in Chapter 3 that the optimal interval frequency (interval number) strategy varies from case to case. There is not a single universal method that can always achieve the lowest naive-Bayes classification error ignorant of specific applications. Thus the best for which one can hope is to develop heuristic approaches that work well with certain types of real-world applications.

Another new method that we have proposed is NDD. Contrasting to the above three new methods that are disjoint, NDD is non-disjoint discretization that forms overlapping intervals. NDD always locates a quantitative attribute value towards the middle of its interval, thus the interval is most likely to be favorable for this value. We have argued that besides the 'non-disjoint' strategy, the interval frequency strategy is also essential to NDD's success. Thus

we choose FFD as NDD's primary method whose frequency strategy has been demonstrated effective by our experiments. We anticipate NDD to be of great utility in naive-Bayes learning. Our experimental results have supported our anticipation by showing that NDD is significantly more effective than every previous method and is most effective among our new methods in terms of reducing the naive-Bayes classification error.

The empirical observation has also supported our analysis on previous methods' effectiveness in terms of discretization bias and variance.

Having already evaluated our research in both theory and practice, we will bring a conclusion to this thesis in the next chapter.

Table 6.1: Experimental datasets and classification error

Dataset	Size	Qa.	Ql.	Class	Classification error %									
					EWD	EFD	FLD	EMD	LD	PD	FFD	NDD	LD.FFD	WPD
Labor-Negotiations	57	8	8	2	12.3	8.9	12.8	9.5	9.6	7.4	9.3	5.4	7.7	9.3
Echocardiogram	74	5	1	2	29.6	30.0	26.5	23.8	29.1	25.7	25.7	24.6	24.1	25.7
Iris	150	4	0	3	5.7	7.7	5.4	6.8	6.7	6.4	7.1	7.5	6.1	6.9
Hepatitis	155	6	13	2	14.3	14.2	14.3	13.9	13.7	14.1	15.7	15.2	14.8	15.3
Wine-Recognition	178	13	0	3	3.3	2.4	3.2	2.6	2.9	2.4	2.8	1.9	1.9	2.0
Sonar	208	60	0	2	25.6	25.1	25.8	25.5	25.8	25.7	23.3	22.8	22.5	23.6
Glass-Identification	214	9	0	6	39.3	33.7	40.7	34.9	32.0	32.6	39.1	33.2	32.9	38.4
Heart-Disease(Cleveland)	270	7	6	2	18.3	16.9	15.8	17.5	17.6	17.4	16.9	16.7	17.1	16.7
Liver-Disorders	345	6	0	2	37.1	36.4	37.6	37.4	37.0	38.9	36.5	35.5	36.8	35.7
Ionosphere	351	34	0	2	9.4	10.3	8.7	11.1	10.8	10.4	10.7	10.5	11.1	10.4
Horse-Colic	368	7	14	2	20.5	20.8	20.8	20.7	20.8	20.3	20.6	20.4	20.6	20.7
Credit-Screening(Australia)	690	6	9	2	15.6	14.5	15.2	14.5	13.9	14.4	14.2	14.2	14.3	14.4
Breast-Cancer(Wisconsin)	699	9	0	2	2.5	2.6	2.8	2.7	2.6	2.7	2.6	2.6	2.7	2.7
Pima-Indians-Diabetes	768	8	0	2	24.9	25.6	25.2	26.0	25.4	26.0	26.5	25.9	26.2	25.2
Vehicle	846	18	0	4	38.7	38.8	42.4	38.9	38.1	38.1	38.3	38.3	38.4	38.2
Annealing	898	6	32	6	3.8	2.4	3.7	2.1	2.3	2.1	2.3	2.1	2.1	2.1
German	1000	7	13	2	25.1	25.2	25.2	25.0	25.1	24.7	25.4	24.7	24.5	25.2
Multiple-Features	2000	3	3	10	31.0	31.8	30.9	32.9	31.0	31.2	31.7	31.8	31.9	31.4
Hypothyroid	3163	7	18	2	3.6	2.8	2.7	1.7	2.4	1.8	1.8	1.6	1.6	2.0
Satimage	6435	36	0	6	18.8	18.8	20.2	18.1	18.4	17.8	17.7	17.6	17.6	17.8
Musk	6598	166	0	2	13.7	18.4	23.0	9.4	15.4	8.2	6.9	6.8	6.9	8.6
Pioneer-MobileRobot	9150	29	7	57	13.5	15.0	21.2	19.3	15.3	4.6	3.2	3.3	3.4	4.9
Handwritten-Digits	10992	16	0	10	12.5	13.2	13.3	13.5	12.8	12.0	12.5	12.6	12.5	11.9
Australian-Sign-Language	12546	8	0	3	38.3	37.7	38.7	36.5	36.4	35.8	36.0	36.0	36.0	36.0
Letter-Recognition	20000	16	0	26	29.5	29.8	34.7	30.4	27.9	25.7	25.5	25.5	25.4	25.7
Adult	48842	6	8	2	18.2	18.6	18.5	17.3	18.1	17.1	16.2	16.1	16.0	17.0
Ipums-la-99	88443	20	40	13	21.0	21.1	37.8	21.3	20.4	20.6	18.4	17.0	17.4	20.5
Census-Income	299285	8	33	2	24.5	24.5	24.7	23.6	24.6	23.3	20.0	19.2	19.2	23.4
Forest-Covertype	581012	10	44	7	32.4	33.0	32.2	32.1	32.3	31.7	31.9	32.0	32.0	31.7
ME	-	-	-	-	20.1	20.0	21.5	19.6	19.6	18.6	18.6	18.0	18.1	18.7
GE	-	-	-	-	16.0	15.6	16.8	15.0	15.3	13.8	13.7	13.0	13.1	14.0

Table 6.2: Classification bias and variance

Dataset	Size	Classification bias %										Classification variance %									
		EWD	EFD	FLD	EMD	LD	PD	PFD	NDD	LD_PFD	WPD	EWD	EFD	FLD	EMD	LD	PD	PFD	NDD	LD_PFD	WPD
Labor-Negotiations	57	7.7	5.4	9.6	6.7	6.3	5.1	6.1	2.5	5.3	6.5	4.6	3.5	3.2	2.8	3.3	2.3	3.2	3.0	2.5	2.8
Echocardiogram	74	22.7	22.3	22.0	19.9	22.3	22.4	19.7	19.1	19.6	19.7	6.9	7.7	4.5	3.9	6.8	3.2	5.9	5.5	4.5	5.9
Iris	150	4.2	5.6	4.0	5.0	4.8	4.3	6.2	5.8	4.6	4.7	1.5	2.1	1.4	1.8	1.9	2.1	0.9	1.7	1.5	2.1
Hepatitis	155	13.1	12.2	13.5	11.7	11.8	11.0	14.5	14.6	13.8	13.9	1.2	2.0	0.8	2.2	1.9	3.1	1.2	0.6	1.0	1.4
Wine-Recognition	178	2.4	1.7	2.2	2.0	2.0	1.7	2.1	1.4	1.1	1.3	1.0	0.7	1.0	0.6	0.9	0.7	0.7	0.4	0.8	0.7
Sonar	208	20.6	19.9	21.0	20.0	20.6	19.9	19.5	18.8	18.8	19.1	5.0	5.2	4.9	5.5	5.2	5.8	3.8	4.0	3.7	4.5
Glass-Identification	214	24.6	21.1	29.0	24.5	21.8	19.8	25.9	25.0	23.6	29.7	14.7	12.6	11.7	10.3	10.2	12.8	13.2	8.2	9.3	8.7
Heart-Disease(Cleveland)	270	15.6	14.9	13.9	15.7	16.1	15.5	15.6	15.1	14.9	14.9	2.7	2.0	1.9	1.8	1.5	2.0	1.3	1.7	2.2	1.8
Liver-Disorders	345	27.6	27.5	30.2	25.7	29.6	28.6	27.7	27.5	29.8	27.3	9.5	8.9	7.4	11.7	7.3	10.3	8.8	8.0	7.9	8.3
Ionosphere	351	8.7	9.6	8.2	10.4	10.4	9.3	8.8	9.8	10.7	8.1	0.7	0.7	0.5	0.7	0.5	1.2	1.9	0.7	0.4	2.3
Horse-Colic	368	18.8	19.6	19.3	18.9	19.2	18.5	19.1	18.8	19.5	19.4	1.7	1.2	1.5	1.7	1.6	1.8	1.5	1.6	1.1	1.3
Credit-Screening(Australia)	690	14.0	12.8	13.8	12.6	12.6	12.2	12.9	12.6	12.8	13.1	1.6	1.7	1.5	1.9	1.3	2.1	1.3	1.6	1.4	1.3
Breast-Cancer(Wisconsin)	699	2.4	2.5	2.7	2.5	2.5	2.5	2.4	2.5	2.6	2.5	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.2
Pima-Indians-Diabetes	768	21.5	22.3	23.4	21.2	22.8	21.7	23.0	22.5	23.2	21.7	3.4	3.3	1.8	4.7	2.6	4.3	3.5	3.4	2.9	3.6
Vehicle	846	31.9	31.9	36.0	32.2	32.4	31.8	32.2	32.4	32.4	31.4	6.9	7.0	6.4	6.7	5.7	6.3	6.1	6.0	6.0	6.8
Annealing	898	2.9	1.9	3.2	1.7	1.7	1.6	1.8	1.4	1.5	1.7	0.8	0.5	0.4	0.4	0.6	0.6	0.5	0.7	0.6	0.4
German	1000	21.9	22.1	22.3	21.2	22.3	21.0	21.8	21.1	20.9	21.8	3.1	3.1	2.9	3.8	2.9	3.7	3.6	3.6	3.6	3.5
Multiple-Features	2000	27.6	27.9	27.5	28.6	27.9	27.2	27.3	27.9	28.2	27.6	3.4	3.9	3.4	4.3	3.1	4.0	4.4	4.0	3.6	3.8
Hypothyroid	3163	2.7	2.5	2.5	1.5	2.2	1.5	1.5	1.4	1.4	1.8	0.8	0.3	0.2	0.3	0.2	0.3	0.3	0.2	0.2	0.3
Satimage	6435	18.0	18.3	19.4	17.0	18.0	17.1	16.9	16.8	16.8	17.0	0.8	0.6	0.7	1.1	0.4	0.7	0.8	0.8	0.8	0.8
Musk	6598	13.1	16.9	22.4	8.5	14.6	7.6	6.2	6.2	6.5	7.9	0.7	1.5	0.6	0.9	0.8	0.7	0.6	0.6	0.4	0.7
Pioneer-MobileRobot	9150	11.0	11.8	18.0	16.1	12.9	2.8	1.6	1.6	1.9	3.0	2.5	3.2	3.2	3.2	2.4	1.9	1.7	1.7	1.5	1.9
Handwritten-Digits	10992	12.0	12.3	12.9	12.1	12.1	10.7	10.5	10.5	10.5	10.5	0.5	0.9	0.4	1.4	0.6	1.4	2.0	2.1	2.1	1.4
Australian-Sign-Language	12546	35.8	36.3	36.9	34.0	35.4	34.0	34.1	34.1	34.0	34.1	2.5	1.4	1.8	2.5	1.0	1.8	2.0	1.9	1.9	1.9
Letter-Recognition	20000	23.9	26.5	30.3	26.2	24.7	22.5	22.2	22.2	22.2	22.5	5.5	3.3	4.5	4.2	3.2	3.2	3.3	3.3	3.3	3.2
Adult	48842	18.0	18.3	18.3	16.8	17.9	16.6	15.2	15.2	15.2	16.5	0.2	0.3	0.2	0.5	0.2	0.5	1.0	0.9	0.8	0.5
Ipums-la-99	88443	16.9	17.2	25.1	16.9	16.9	15.9	13.5	12.7	13.0	15.9	4.1	4.0	12.7	4.1	3.5	4.7	4.9	4.3	4.4	4.6
Census-Income	299285	24.4	24.3	24.6	23.3	24.4	23.1	18.9	18.1	18.3	23.1	0.2	0.2	0.2	0.2	0.2	0.2	1.1	1.0	1.0	0.2
Forest-Covertype	581012	32.0	32.5	31.9	31.1	32.0	30.3	29.6	29.6	29.6	30.3	0.4	0.5	0.3	1.0	0.3	1.4	2.3	2.4	2.4	1.4
ME	-	17.1	17.2	18.8	16.7	17.2	15.7	15.8	15.4	15.6	16.1	3.0	2.8	2.8	2.9	2.4	2.9	2.8	2.6	2.4	2.6
GE	-	13.5	13.2	14.6	12.7	13.2	11.4	11.4	10.7	11.0	11.7	1.7	1.6	1.4	1.7	1.3	1.7	1.8	1.7	1.6	1.7

Table 6.3: Win/lose/tie records on classification error of PD, FFD and NDD

Method	PD				FFD				NDD			
	Win	Lose	Tie	Sign Test	Win	Lose	Tie	Sign Test	Win	Lose	Tie	Sign Test
vs. EWD	22	7	0	<0.01	20	8	1	0.02	22	7	0	<0.01
vs. EFD	22	6	1	<0.01	19	8	2	0.03	24	3	2	<0.01
vs. FLD	23	6	0	<0.01	22	7	0	<0.01	23	6	0	<0.01
vs. EMD	21	5	3	<0.01	20	9	0	0.03	25	3	1	<0.01
vs. LD	20	8	1	0.02	19	8	2	0.03	21	7	1	<0.01

Table 6.4: Computational time per fold (seconds)

-	Adult	Ipums.la.99	Census-Income	Forest-Covertype
LD_FFD	547	6124	47234	56950
NDD	1	14	19	59

Table 6.5: Win/lose/tie records on classification bias of previous primary methods

	vs. PD				*vs.* FFD			
EWD	5	24	0	< 0.01	8	19	2	0.03
EFD	3	23	3	< 0.01	10	19	0	0.07
EMD	4	22	3	< 0.01	11	16	2	0.22

Table 6.6: Win/lose/tie records on classification variance of previous primary methods

	vs. PD				*vs.* FFD			
EWD	14	12	3	0.42	10	17	2	0.12
EFD	17	6	6	0.02	14	11	4	0.35
EMD	12	12	5	0.58	13	15	1	0.43

Table 6.7: Win/lose/tie records of previous composite methods

	Classification error				Classification bias				Classification variance			
	Win	Lose	Tie	Sign test	Win	Lose	Tie	Sign test	Win	Lose	Tie	Sign test
FLD *vs.* EWD	10	18	1	0.09	9	20	0	0.03	22	2	5	< 0.01
LD *vs.* EFD	19	8	2	0.03	14	12	3	0.42	23	3	3	< 0.01
LD_FFD *vs.* LD	21	8	0	0.01	19	9	1	0.04	12	14	3	0.42

Table 6.8: Compare PD and FFD on smaller datasets

	PD		FFD		Win/lose/tie record of PD against FFD	Sign test
	ME	GM	ME	GM		
Classification error	18.2	13.1	18.6	13.7	11/5/1	0.11
Classification bias	14.5	10.4	15.3	11.3	12/5/0	0.07
Classification variance	3.7	2.3	3.4	2.1	5/11/1	0.11

Note: ME is mean, GM is geometric mean.

Table 6.9: Compare PD and FFD on larger datasets

	PD		FFD		Win/lose/tie record of PD against FFD	Sign test
	ME	GM	ME	GM		
Classification error	19.2	14.8	18.5	13.8	4/7/1	0.27
Classification bias	17.4	13.0	16.5	11.7	2/9/1	0.03
Classification variance	1.7	1.2	2.0	1.5	9/2/1	0.03

Note: ME is mean, GM is geometric mean.

Table 6.10: Win/lose/tie records on classification error of WPD

WPD *vs.*	EWD	EFD	FLD	EMD	LD	PD	FFD	NDD
Win	21	23	22	20	20	8	14	7
Lose	8	5	5	6	9	13	12	19
Tie	0	1	2	3	0	8	3	3
Sign Test	0.01	<0.01	<0.01	<0.01	0.03	0.19	0.42	0.01

Chapter 7

Conclusion

We have studied discretization for naive-Bayes learning through the previous chapters. In this chapter, we summarize what we have learned. We also discuss directions for future research. We complete this thesis by highlighting our contributions to the area of discretization and naive-Bayes learning.

7.1 Summary of thesis

The kernel issues of this thesis are to understand why discretization can be effective for naive-Bayes learning and accordingly to devise discretization techniques such that naive-Bayes classifiers with discretization achieve both classification efficacy and efficiency.

What is discretization? In Chapter 2 we have tackled how to define discretization. Discretization is a data processing procedure that transforms one type of data into another type of data. In the existing large amount of literature that address discretization, there is considerable variation in the terminology used to describe each of these two data types. However, many of the various terms have distinct definitions from the perspective of statistics. Thus confusion exists for defining discretization. We carry out a broad survey on this issue, turning to the authority of statistics text books. We make clear the differ-

ences among various terms, and suggest it most proper to define discretization as transforming *quantitative* data into *qualitative* data.

Why can discretization be effective? In Chapter 3 we explain discretization's working mechanism in naive-Bayes learning. Discretization seeks to substitute a qualitative attribute value $X_i^*=x_i^*$ for a quantitative attribute value $X_i=x_i$. We have proved in Theorem 1 that if discretization results in that $p(C=c|X_i^*=x_i^*)$ is an accurate estimate of $p(C=c|X_i=x_i)$ for each quantitative x_i given a test instance $\mathbf{x}$, $p(C=c|\mathbf{X}^*=\mathbf{x}^*)$ will be an accurate estimate of $p(C=c|\mathbf{X}=\mathbf{x})$.

What affects discretization effectiveness? Having a clear answer to this question helps us to devise proper discretization techniques for naive-Bayes learning. In Chapter 3, according to Theorem 1, we have argued that it is sensible for discretization to focus on the accuracy of $p(C=c|X_i^*=x_i^*)$ as an estimate of $p(C=c|X_i=x_i)$. Two factors, the *decision boundary* and the *error tolerance of probability estimation*, can affect this estimation. Different discretization methods can have different approaches to dealing with these two factors, and hence have different effects on classification bias and variance of the generated naive-Bayes classifiers. We name these effects *discretization bias* and *variance*. We argue that while discretization is desirable when the quantitative data's true underlying probability density functions are not available, practical discretization techniques are necessarily heuristic in nature. In particular, we provide insights into managing discretization bias and variance by tuning interval frequency and interval number.

What is the situation of current research on discretization? In Chapter 4, we have conducted a comprehensive literature review of discretization meth-

ods in the research area of machine learning. We are particularly interested in those methods that were originally developed or are often used for naive-Bayes classifiers; and in analyzing their behaviors on discretization bias and variance, which we think illuminating.

Why should we devise new discretization techniques? We believe that our work presented in this thesis is necessary. Combining Chapter 3 and Chapter 4, we have found that the majority of the existing discretization methods were developed in learning contexts other than naive-Bayes learning, and do not suit naive-Bayes classifiers' requirements of effective discretization. We have also found that there do exist a few methods that were developed in the context of naive-Bayes learning. However, their sub-optimal effectiveness of managing discretization bias and variance, or their sub-optimal computational efficiency suggests that these methods have potential problems for naive-Bayes learning. Concerning the widespread employment of naive-Bayes classifiers, we believe that there is a real and immediate need for improving discretization effectiveness for naive-Bayes classifiers.

What solutions can we offer? In Chapter 5, we have evaluated the impact of discretization bias and variance on naive-Bayes classification performance. Since we have obtained the insight that discretization bias and variance relate to discretized interval frequency and interval number, our new discretization techniques focus on adjusting interval frequency and number to manage discretization bias and variance, so as to lower the naive-Bayes classification error. Using different chains of reasoning, we propose three new techniques, *proportional discretization* (PD), *fixed frequency discretization* (FFD) and *non-disjoint discretization* (NDD). PD sets interval frequency and interval number equal, both

proportional to the size of the training data. By this means, it seeks a good trade-off between discretization bias and variance by equally weighing bias reduction and variance reduction. Also it can use increasing training data to decrease both discretization bias and variance. FFD sets interval frequency as small as possible under the condition that there are sufficient training instances per interval for the naive-Bayes probability estimation. The interval number increases with the training data size increasing. By this means, FFD can control discretization variance from being high, and can use additional training data to lower discretization bias. NDD is proposed when we further realize that discretization resulting in *favorable* intervals for quantitative values can be expected to lower the naive-Bayes classification error. However, disjoint discretization cannot be effective to find favorable intervals. Accordingly, NDD forms overlapping (non-disjoint) intervals for a quantitative attribute, and always tries to choose a favorable interval for the value of the current instance to be classified. Two key components of NDD strategy are locating a value towards the middle of its interval, and selecting proper interval frequency. In our current research, we employ for NDD the frequency strategy of FFD due to its simplicity, efficiency and efficacy.

How can we prove the efficiency and efficacy of our new techniques for naive-Bayes classifiers? Although our theoretical analysis leads to an optimistic expectation of our new methods' effectiveness of reducing the naive-Bayes classification error, theory needs to be verified by practice. Consequently, in Chapter 6, we conduct the empirical evaluation for our new techniques against previous key discretization methods in naive-Bayes learning. We run our experiments on a large suite of natural datasets from UCI machine

learning repository [Blake and Merz 1998] and KDD archive [Bay 1999]. These datasets represent applications with hundreds to hundreds of thousands of instances. This differs from most previous work that was confined to experimental datasets of relatively small size. The experimental results agree with our predictions. If compared with each previous method, each of PD, FFD and NDD achieves lower naive-Bayes classification error with frequency significant at the 0.05 level. If comparing their own effectiveness, NDD is more effective than both PD and FFD. We attribute this to the facts that NDD always seeks favorable intervals by locating a quantitative attribute value towards the middle of its interval and that NDD employs the frequency strategy of FFD. As for PD and FFD, there is no statistically significant difference between their effectiveness. However, we analyze that PD and EFD have different effectiveness of managing discretization bias and variance according to different training data sizes. Thus we propose *weighted proportional discretization* (WPD) that combines PD and FFD. According to our experimental results, WPD also achieves lower naive-Bayes classification error than every previous method with significant frequency, while PD, FFD and WPD obtain equivalent effectiveness. These observations support our suggestion that the optimal interval frequency (interval number) strategy varies from case to case. Thus although our new techniques are desirable, an optimal universal discretization strategy for naive-Bayes learning is unobtainable. Hence, the best we can hope for is heuristic approaches that work well with certain types of real-world applications.

7.2 Future work

While it has shed light on a new understanding of discretization in naive-Bayes learning, our research also brings before us several issues worth further exploration.

- Although discretization is an approach parallel to probability density estimation, these two are not exclusive. Actually the key factors affecting discretization's effectiveness, the decision boundary and the error tolerance of probability estimation, both depend on the probability distribution of the training data. Thus the more we know about the data distribution, the better our discretization can be. Although the conventional 'normal distribution' assumption has been shown ineffective, there do exist many more sophisticated approaches to modelling the probability density function [Silverman 1986]. Further investigation of these approaches might benefit discretization by offering more information about the nature of the data.

- We have demonstrated that our techniques generally perform better than alternatives on the datasets in our study. However, in practice one is more interested in the best technique for the particular data to be studied, not the average performance across many domains [Kohavi 1995b]. It is desirable that an algorithm is devised such that it can utilize background knowledge of the data. For example, although we set the sufficient interval frequency m as 30 for fixed frequency discretization (FFD) in our experiments, an extension of FFD may prove more efficacious, which is able to dynamically adjust m corresponding to the individuality

of applications.

- Although there exist a large number of discretization methods, we have suggested that there is no universally optimal discretization strategy for naive-Bayes learning, and the best for which one can hope is approaches that work well with certain types of real-world applications. Thus an interesting piece of work is to analyze what types of discretization methods (for example, according to our taxonomies in Chapter 2) are effective for what types of applications. Finding out this relationship is very likely to help choose most appropriate discretization methods for particular applications in practice. However, a challenge in this work is how to categorize applications into certain types so that the match between discretization techniques and applications can be properly built up.

- One appealing merit of naive-Bayes classifiers, as we have argued in Chapter 3, is that their learning is incremental. Thus it is desirable if discretization methods employed by naive-Bayes learning can also be incremental. However, at the current stage, our new discretization techniques are not incremental. It will be of great utility to improve our techniques' incrementality.

- Although all of our proposed discretization methods are originally oriented to naive-Bayes learning, a natural question is: can they be applied with other learning algorithms? For example, Bayes network learning relaxes naive-Bayes' attribute independence assumption. Can our new techniques facilitate Bayes network learning as well? If yes, to what extent? If no, why?

- Another inspiring issue arising from our study is that unsupervised discretization methods are able to outperform supervised ones (in our experiments the entropy minimization discretization [Fayyad and Irani 1993]) in the context of naive-Bayes learning. This is contrary to the previous understanding that supervised methodologies tend to have an advantage over unsupervised methodologies [Dougherty, Kohavi, and Sahami 1995; Hsu, Huang, and Wong 2000; Hsu, Huang, and Wong 2003]. Further investigation of this issue may prove illuminating.

7.3 Concluding remarks

In this thesis, we have studied discretization in naive-Bayes learning. This is of special significance since naive-Bayes classifiers are popular with classification tasks, and the quantitative data are usually discretized to train naive-Bayes classifiers. New understandings in both theory and practice have come into being as a result of our study. In theory, we clarify the confusion of defining discretization in existing literature and conclude that discretization usually refers to a process of converting quantitative data to qualitative data. We prove a theorem that explains why discretization can be effective in naive-Bayes leaning and why discretization is desirable when the true underlying probability density functions of the quantitative data are not available. We analyze two factors, the decision boundary and the error tolerance of probability estimation, which can affect discretization effectiveness. A discretization method's effectiveness is reflected by its effects on the bias and variance of the generated naive-Bayes classifiers. we name these effects discretization bias

and variance. We argue that discretization methods that are able to well manage discretization bias and variance are of great utility in naive-Bayes learning. We further suggest that one effective approach to managing discretization bias and variance is to adjust interval frequency and interval number. However, we obtain the understanding that the optimal interval frequency (interval number) strategy varies depending on individual applications. Thus an universally optimal discretization strategy for naive-Bayes learning is unobtainable. Hence, the best for which one can hope is to develop heuristic approaches that work well with certain types of real-world applications.

In practice, four new discretization techniques have been developed and evaluated. To the best of our knowledge, these techniques are the first to explicitly deal with discretization bias and variance. Although each technique differs in their specific strategy, they have some common merits. First, all of them can achieve lower naive-Bayes classification error than every previous key discretization method in our study with frequency significant at the 0.05 level. Second, all of them are able to actively take advantage of increasing information in large data to reduce discretization bias and variance. Thus they are expected to demonstrate greater advantage than previous methods especially when learning from large data. It is desirable that a machine learning algorithm maximizes the information that it derives from large datasets, since increasing the size of a dataset can provide a *domain-independent* way of achieving higher accuracy [Freitas and Lavington 1996; Provost and Aronis 1996]. Third, all of our new techniques have computational time and space complexity as low as the simplest discretization methods such as equal frequency discretization [Catlett 1991; Kerber 1992; Dougherty, Kohavi, and Sa-

hami 1995]. This is especially important since large datasets with high dimensional attribute spaces and huge numbers of instances are increasingly used in real-world applications; and naive-Bayes classifiers are particularly attractive for these applications because of their time and space efficiency. These merits together with the outstanding effectiveness on lowering the naive-Bayes classification error give us grounds for being positive that our new discretization techniques are of great utility for naive-Bayes learning.

References

AN, A. AND CERCONE, N. 1999. Discretization of continuous attributes for learning classification rules. In *Proceedings of the 3rd Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining* (1999), pp. 509–514. (pp. 53, 79, 81, 93)

ANDROUTSOPOULOS, I., KOUTSIAS, J., CHANDRINOS, K., AND SPYROPOULOS, C. 2000. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with encrypted personal e-mail messages. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2000), pp. 160–167. (p. 3)

AUER, P., HOLTE, R., AND MAASS, W. 1995. Theory and application of agnostic PAC-learning with small decision trees. In *Proceedings of the 12th International Conference on Machine Learning* (1995), pp. 21–29. (p. 60)

BAUER, E. AND KOHAVI, R. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning 36*, 1-2, 105–139. (p. 3)

BAY, S. D. 1999. The UCI KDD archive [http://kdd.ics.uci.edu]. Department of Information and Computer Science, University of California, Irvine. (pp. 109, 137)

BAY, S. D. 2000. Multivariate discretization of continuous variables for set

mining. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2000), pp. 315–319. (pp. 17, 85, 93)

BAY, S. D. AND PAZZANI, M. J. 1999. Detecting change in categorical data: Mining contrast sets. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (1999), pp. 302–306. (p. 85)

BERKA, P. AND BRUHA, I. 1998. Discretization and grouping: Preprocessing steps for data mining. In *Posters of Principles of Data Mining and Knowledge Discovery, Second European Symposium* (1998), pp. 239–245. (p. 71)

BLAKE, C. L. AND MERZ, C. J. 1998. UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/mlrepository.html]. Department of Information and Computer Science, University of California, Irvine. (pp. 109, 137)

BLUMAN, A. G. 1992. *Elementary Statistics, A Step By Step Approach*. Wm. C. Brown Publishers. (pp. 12, 22, 23)

BREIMAN, L. 1996. Bias, variance and arcing classifiers, technical report 460, statistics department, university of california, berkerley. (pp. 34, 111)

BRIJS, T. AND VANHOOF, K. 1998. Cost sensitive discretization of numeric attributes. In *Proceedings of 2nd European Symposium on Principles of Data Mining and Knowledge Discovery* (1998), pp. 102–110. (p. 74)

CASELLA, G. AND BERGER, R. L. 1990. *Statistical Inference*. Pacific Grove, Calif. (p. 23)

CATLETT, J. 1991. On changing continuous attributes into ordered discrete

attributes. In *Proceedings of the European Working Session on Learning* (1991), pp. 164–178. (pp. 29, 49, 50, 58, 63, 66, 86, 93, 99, 103, 108, 141)

CERQUIDES, J. AND LOPEZ DE MANTARAS, R. 1997. Proposal and empirical comparison of a parallelizable distance-based discretization method. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (1997), pp. 139–142. (p. 66)

CESTNIK, B. 1990. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the 9th European Conference on Artificial Intelligence* (1990), pp. 147–149. (pp. 3, 23)

CESTNIK, B., KONONENKO, I., AND BRATKO, I. 1987. Assistant 86: A knowledge-elicitation tool for sophisticated users. In *Proceedings of the 2nd European Working Session on Learning* (1987), pp. 31–45. (p. 3)

CHLEBUS, B. S. AND NGUYEN, S. H. 1998. On finding optimal discretizations for two attributes. In *Proceedings of the 1st International Conference on Rough Sets and Current Trends in Computing* (1998), pp. 537–544. (p. 72)

CHMIELEWSKI, M. R. AND GRZYMALA-BUSSE, J. W. 1996. Global discretization of continuous attributes as preprocessing for machine learning. *International Journal of Approximate Reasoning 15*, 319–331. (pp. 60, 93)

CLARK, P. AND BOSWELL, R. 1991. Rule induction with CN2: Some recent improvements. In *Proceedings of the 5th European Working Session on Learning* (1991), pp. 151–163. (p. 87)

CLARK, P. AND NIBLETT, T. 1989. The CN2 induction algorithm. *Machine Learning 3*, 261–283. (p. 3)

CRAWFORD, E., KAY, J., AND ERIC, M. 2002. IEMS - the intelligent email sorter. In *Proceedings of the 19th International Conference on Machine Learning* (2002), pp. 83–90. (p. 3)

DILLON, W. AND GOLDSTEIN, M. 1984. *Multivariate Analysis*. John Weily and Sons, Inc. (p. 65)

DOMINGOS, P. AND PAZZANI, M. 1996. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the 13th International Conference on Machine Learning* (1996), pp. 105–112. Morgan Kaufmann Publishers. (pp. 3, 109)

DOMINGOS, P. AND PAZZANI, M. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning 29*, 103–130. (pp. 3, 21, 27)

DOUGHERTY, J., KOHAVI, R., AND SAHAMI, M. 1995. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning* (1995), pp. 194–202. (pp. 5, 17, 18, 25, 28, 29, 49, 50, 53, 66, 86, 99, 103, 108, 140, 141)

DUDA, R. AND HART, P. 1973. *Pattern Classification and Scene Analysis*. John Wiley & Sons. (pp. 21, 27, 41)

FAYYAD, U. M. AND IRANI, K. B. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence* (1993), pp. 1022–1027. (pp. 29, 53, 60, 67, 70, 74, 77, 78, 79, 80, 81, 87, 93, 108, 109, 140)

FRANK, E. AND WITTEN, I. H. 1999. Making better use of global discretization. In *Proceedings of the 16th International Conference on Machine Learning*

(1999), pp. 115–123. Morgan Kaufmann Publishers. (pp. 49, 87, 93)

FRASCONI, P., SODA, G., AND VULLO, A. 2001. Text categorization for multi-page documents: a hybrid naive Bayes HMM approach. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries* (2001), pp. 11–20. (p. 2)

FREITAS, A. A. AND LAVINGTON, S. H. 1996. Speeding up knowledge discovery in large relational databases by means of a new discretization algorithm. In *Advances in Databases, Proceedings of the 14th British National Conference on Databases* (1996), pp. 124–133. (pp. 6, 64, 93, 141)

FRIEDMAN, J. H. 1997. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery 1*, 1, 55–77. (pp. 34, 54)

GAMA, J. 2000. Iterative Bayes. *Intelligent Data Analysis 4*, 475–488. (p. 3)

GAMA, J., TORGO, L., AND SOARES, C. 1998. Dynamic discretization of continuous attributes. In *Proceedings of the 6th Ibero-American Conference on AI* (1998), pp. 160–169. (pp. 69, 93, 95)

GRZYMALA-BUSSE, J. W. AND STEFANOWSKI, J. 2001. Three discretization methods for rule induction. *International Journal of Intelligent Systems 16*, 29–38. (p. 87)

GUTHRIE, L. AND WALKER, E. 1994. Document classification by machine: Theory and practice. In *Proceedings of the 15th International Conference on Computational Linguistics* (1994), pp. 1059–1063. (p. 2)

HO, K. M. AND SCOTT, P. D. 1997. Zeta: A global method for discretization

of continuous variables. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (1997), pp. 191–194. (pp. 68, 93)

HOLTE, R., ACKER, L., AND PORTER, B. 1989. Concept learning and the problem of small disjuncts. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence* (1989), pp. 813–818. (p. 59)

HOLTE, R. C. 1993. Very simple classification rules perform well on most commonly used datasets. *Machine Learning 11*, 63–91. (pp. 59, 93)

HSU, C.-N., HUANG, H.-J., AND WONG, T.-T. 2000. Why discretization works for naive Bayesian classifiers. In *Proceedings of the 17th International Conference on Machine Learning* (2000), pp. 309–406. (pp. 18, 27, 29, 35, 37, 39, 47, 51, 56, 57, 96, 104, 108, 140)

HSU, C.-N., HUANG, H.-J., AND WONG, T.-T. 2003. Implications of the Dirichlet assumption for discretization of continuous variables in naive Bayesian classifiers. *Machine Learning*. in press. (pp. 18, 27, 29, 35, 47, 51, 56, 57, 96, 104, 108, 140)

HUSSAIN, F., LIU, H., TAN, C. L., AND DASH, M. 1999. Discretization: An enabling technique. Technical Report, TRC6/99, School of Computing, National University of Singapore. (p. 95)

ISHIBUCHI, H., YAMAMOTO, T., AND NAKASHIMA, T. 2001. Fuzzy data mining: Effect of fuzzy discretization. In *The 2001 IEEE International Conference on Data Mining* (2001). (p. 86)

JOACHIMS, T. 1997. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *Proceedings of the 14th International Conference on Machine Learning* (1997), pp. 143–151. (p. 2)

JOHN, G. H., KOHAVI, R., AND PFLEGER, K. 1994. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning* (1994), pp. 121–129. (p. 66)

JOHN, G. H. AND LANGLEY, P. 1995. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence* (1995), pp. 338–345. (pp. 4, 23, 24, 25, 28)

KALT, T. 1996. A new probabilistic model of text classification and retrieval. Technical Report IR-78, Center for Intelligent Information Retrieval, University of Massachusetts. (p. 2)

KERBER, R. 1992. Chimerge: Discretization for numeric attributes. In *National Conference on Artificial Intelligence* (1992), pp. 123–128. AAAI Press. (pp. 17, 29, 49, 50, 58, 61, 63, 64, 66, 86, 93, 99, 103, 108, 141)

KIM, Y.-H., HAHN, S.-Y., AND ZHANG, B.-T. 2000. Text filtering by boosting naive Bayes classifiers. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2000), pp. 168–175. (p. 3)

KOHAVI, R. 1995a. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (1995), pp. 1137–1145. (p. 110)

KOHAVI, R. 1995b. *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Department of Computer Science, Stanford University, USA. (pp. 66, 138)

KOHAVI, R. 1996. Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the 2nd International Conference on*

Knowledge Discovery and Data Mining (1996), pp. 202–207. (p.3)

KOHAVI, R. AND PROVOST, F. 1998. Glossary of terms, special issue on applications of machine learning and the knowledge discovery process. *Machine Learning 30*, 271–274. (p.110)

KOHAVI, R. AND SAHAMI, M. 1996. Error-based and entropy-based discretization of continuous features. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (1996), pp. 114–119. (pp.49, 60, 74)

KOHAVI, R., SOMMERFIELD, D., AND DOUGHERTY, J. 1997. Data mining using MLC++: A machine learning library in C++. *International Journal on Artificial Intelligence Tools 6*, 4, 537–566. (p.3)

KOHAVI, R. AND WOLPERT, D. 1996. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the 13th International Conference on Machine Learning* (1996), pp. 275–283. (p.34)

KOHONEN, T. 1995. *Self-Organization Maps*. Springer-Verlag, Berlin Heidelberg. (p.75)

KOLLER, D. AND SAHAMI, M. 1997. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning* (1997), pp. 170–178. (p.2)

KONG, E. B. AND DIETTERICH, T. G. 1995. Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th International Conference on Machine Learning* (1995), pp. 313–321. (p.34)

KONONENKO, I. 1990. Comparison of inductive and naive Bayesian learn-

ing approaches to automatic knowledge acquisition. (p.3)

KONONENKO, I. 1992. Naive Bayesian classifier and continuous attributes. *Informatica 16*, 1, 1–8. (pp.32, 51, 52, 108)

KONONENKO, I. 1993. Inductive and Bayesian learning in medical diagnosis. *Applied Artificial Intelligence 7*, 317–337. (pp.3, 51, 108)

KONONENKO, I. 2001. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine 23*, 1, 89–109. (p.3)

KUKAR, M., GROSELJ, C., KONONENKO, I., AND FETTICH, J. 1997. An application of machine learning in the diagnosis of ischaemic heart disease. In *Proceedings of the 6th Conference on Artificial Intelligence in Medicine Europe* (1997), pp. 461–464. (p.3)

KWEDLO, W. AND KRETOWSKI, M. 1999. An evolutionary algorithm using multivariate discretization for decision rule induction. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery* (1999), pp. 392–397. (pp.77, 93)

LANGLEY, P., IBA, W., AND THOMPSON, K. 1992. An analysis of Bayesian classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence* (1992), pp. 223–228. (p.3)

LARKEY, L. S. AND CROFT, W. B. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval* (1996), pp. 289–297. (p.2)

LAVRAC, N. 1998. Data mining in medicine: Selected techniques and ap-

plications. In *Proceedings of the 2nd International Conference on The Practical Applications of Knowledge Discovery and Data Mining* (1998), pp. 11–31. (p. 3)

LAVRAC, N., KERAVNOU, E., AND ZUPAN, B. 2000. Intelligent data analysis in medicine. *Encyclopedia of Computer Science and Technology 42*, 9, 113–157. (p. 3)

LEWIS, D. D. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1992), pp. 37–50. (p. 2)

LEWIS, D. D. 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning* (1998), pp. 4–15. (p. 2)

LEWIS, D. D. AND GALE, W. A. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1994), pp. 3–12. (p. 2)

LI, H. AND YAMANISHI, K. 1997. Document classification using a finite mixture model. In *Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics* (1997), pp. 39–47. (p. 2)

LOPEZ DE MANTARAS, R. 1991. A distance-based attribute selection measure for decision tree induction. *Machine Learning 6*, 81–92. (pp. 66, 67)

LUDL, M.-C. AND WIDMER, G. 2000a. Relative unsupervised discretization for association rule mining. In *Proceedings of the 4th European Conference*

on Principles and Practice of Knowledge Discovery in Databases (2000). (pp. 84, 93)

LUDL, M.-C. AND WIDMER, G. 2000b. Relative unsupervised discretization for regression problems. In *Proceedings of the 11th European Conference on Machine Learning* (2000), pp. 246–253. (p. 84)

MAASS, W. 1994. Efficient agnostic PAC-learning with simple hypothese. In *Proceedings of the 7th Annual ACM Conference on Computational Learning Theory* (1994), pp. 67–75. (p. 60)

MACSKASSY, S. A., HIRSH, H., BANERJEE, A., AND DAYANIK, A. A. 2001. Using text classifiers for numerical classification. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (2001), pp. 885–890. (pp. 87, 88, 93)

MARON, M. 1961. Automatic indexing: An experimental inquiry. *Journal of the Association for Computing Machinery 8*, 404–417. (p. 2)

MARON, M. AND KUHNS, J. 1960. On relevance, probabilistic indexing, and information retrieval. *Journal of the Association for Computing Machinery 7*, 3, 216–244. (p. 2)

MCCALLUM, A. AND NIGAM, K. 1998. A comparison of event models for naive Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization* (1998), pp. 41–48. (p. 2)

MCCALLUM, A., ROSENFELD, R., MITCHELL, T. M., AND NG, A. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the 15th International Conference on Machine Learning* (1998), pp. 359–367. (p. 2)

McSherry, D. 1997a. Avoiding premature closure in sequential diagnosis. *Artificial Intelligence in Medicine 10*, 3, 269–283. (p. 3)

McSherry, D. 1997b. Hypothesist: A development environment for intelligent diagnostic systems. In *Proceedings of the 6th Conference on Artificial Intelligence in Medicine in Europe* (1997), pp. 223–234. (p. 3)

Michalewicz, Z. 1996. *Genetic Algorithm + Data Structures = Evolutionary Programs, Third Edition*. Springer. (p. 78)

Mitchell, T. M. 1997. *Machine Learning*. McGraw-Hill Companies. (pp. 2, 4, 25)

Miyahara, K. and Pazzani, M. J. 2000. Collaborative filtering with the simple Bayesian classifier. In *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence* (2000), pp. 679–689. (p. 3)

Montani, S., Bellazzi, R., Portinale, L., Fiocchi, S., and Stefanelli, M. 1998. A case-based retrieval system for diabetic patient therapy. In *IDAMAP 98 working notes* (1998), pp. 64–70. (p. 3)

Monti, S. and Cooper, G. 1998. A multivariate discretization method for learning Bayesian networks from mixed data. In *Proceedings of the 14th Conference of Uncertainty in AI* (1998), pp. 404–413. (p. 93)

Mooney, R. J. and Roy, L. 2000. Content-based book recommending using learning for text categorization. In *Proceedings of DL-00, 5th ACM Conference on Digital Libraries* (2000), pp. 195–204. ACM Press, New York, US. (p. 3)

Moore, D. S. and McCabe, G. P. 2002. *Introduction to the Practice of Statis-*

tics, Fourth Edition. Michelle Julet. (pp. 35, 96)

MORA, L., FORTES, I., MORALES, R., AND TRIGUERO, F. 2000. Dynamic discretization of continuous values from time series. In *Proceedings of the 11th European Conference on Machine Learning* (2000), pp. 280–291. (pp. 82, 95)

NIGAM, K., MCCALLUM, A., THRUN, S., AND MITCHELL, T. M. 1998. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the 15th National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98* (1998), pp. 792–799. (p. 2)

PANTEL, P. AND LIN, D. 1998. Spamcop: A spam classification & organization program. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization* (1998), pp. 95–98. (p. 3)

PAZZANI, M. AND BILLSUS, D. 1997. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning 27*, 3, 313–331. (p. 2)

PAZZANI, M., MURAMATSU, J., AND BILLSUS, D. 1996. Syskill & webert: Identifying interesting web sites. In *Proceedings of the National Conference on Artificial Intelligence* (1996), pp. 54–61. (p. 2)

PAZZANI, M. J. 1995. An iterative improvement approach for the discretization of numeric attributes in Bayesian classifiers. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining* (1995), pp. 228–233. (pp. 5, 25, 54, 95, 108)

PERNER, P. AND TRAUTZSCH, S. 1998. Multi-interval discretization meth-

ods for decision tree learning. In *Proceedings of Advances in Pattern Recognition, Joint IAPR International Workshops SSPR'98 and SPR'98* (1998), pp. 475–482. (pp. 53, 75, 77, 93)

PFAHRINGER, B. 1995. Compression-based discretization of continuous attributes. In *Proceedings of the 12th International Conference on Machine Learning* (1995). (pp. 62, 63)

PROVOST, F. J. AND ARONIS, J. M. 1996. Scaling up machine learning with massive parallelism. *Machine Learning* 23, 1, 33–46. (pp. 6, 141)

PROVOST, J. 1999. Naive-Bayes vs. rule-learning in classification of email. Technical Report AI-TR-99-284, Artificial Intelligence Lab, The University of Texas at Austin. (p. 3)

QUINLAN, J. R. 1986. Induction of decision trees. *Machine Learning 1*, 81–106. (p. 58)

QUINLAN, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers. (pp. 18, 53)

RENNIE, J. 2000. IFILE: An application of machine learning to mail filtering. In *Proceedings of the KDD-2000 Workshop on Text Mining* (2000). (pp. 3, 27)

RICHELDI, M. AND ROSSOTTO, M. 1995. Class-driven statistical discretization of continuous attributes (extended abstract). In *European Conference on Machine Learning* (1995), pp. 335–338. (pp. 49, 61, 64, 93)

RISSANEN, J. 1978. Modelling by shortest data description. *Automatica 14*, 465–471. (p. 63)

ROY, N. AND MCCALLUM, A. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning* (2001), pp. 441–448. (p. 27)

SAMUELS, M. L. AND WITMER, J. A. 1999. *Statistics For The Life Sciences, Second Edition*. Prentice-Hall. (pp. 12, 22, 23)

SCHEAFFER, R. L. AND MCCLAVE, J. T. 1995. *Probability and Statistics for Engineers* (Fourth ed.)., Chapter Continuous Probability Distributions, pp. 186. Duxbury Press. (p. 24)

SILVERMAN, B. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall Ltd. (pp. 24, 25, 138)

STARR, B., ACKERMAN, M. S., AND PAZZANI, M. 1996a. Do I care? – tell me what's changed on the web. In *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access* (1996). (p. 2)

STARR, B., ACKERMAN, M. S., AND PAZZANI, M. 1996b. Do-I-care: A collaborative web agent. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (1996), pp. 273–274. (p. 3)

STONE, M. 1974. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36, 111–147. (p. 66)

TING, K. AND ZHENG, Z. 1999. Improving the performance of boosting for naive Bayesian classification. In *Proceedings of the 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining* (1999), pp. 296–305. (p. 3)

TORGO, L. AND GAMA, J. 1997. Search-based class discretization. In *Proceedings of the 9th European Conference on Machine Learning* (1997), pp. 266–

273. (pp. 65, 66, 70, 95)

TSYMBAL, A., PUURONEN, S., AND PATTERSON, D. 2002. Feature selection for ensembles of simple Bayesian classifiers. In *Proceedings of the 13th International Symposium on Foundations of Intelligent Systems* (2002), pp. 592–600. (p. 3)

WANG, K. AND LIU, B. 1998. Concurrent discretization of multiple attributes. In *The Pacific Rim International Conference on Artificial Intelligence* (1998), pp. 250–259. (pp. 69, 93)

WEBB, G. I. 2000. Multiboosting: A technique for combining boosting and wagging. *Machine Learning 40*, 2, 159–196. (pp. 34, 111, 112)

WEISS, N. A. 2002. *Introductory Statistics, Sixth Edition.* Greg Tobin. (p. 98)

YANG, Y. AND LIU, X. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1999), pp. 42–49. (p. 26)

YANG, Y. AND WEBB, G. I. 2001. Proportional k-interval discretization for naive-Bayes classifiers. In *Proceedings of the 12th European Conference on Machine Learning* (2001), pp. 564–575. (p. x)

YANG, Y. AND WEBB, G. I. 2002a. A comparative study of discretization methods for naive-Bayes classifiers. In *Proceedings of the Pacific Rim Knowledge Acquisition Workshop* (2002), pp. 159–173. (p. x)

YANG, Y. AND WEBB, G. I. 2002b. Non-disjoint discretization for naive-Bayes classifiers. In *Proceedings of the 19th International Conference on Machine Learning* (2002), pp. 666–673. (p. x)

YANG, Y. AND WEBB, G. I. 2003. Weighted proportional k-interval discretization for naive-Bayes classifiers. In *the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (2003). (p. x)

ZAFFALON, M. AND HUTTER, M. 2002. Robust feature selection using distributions of mutual information. In *Proceedings of the 14th International Conference on Uncertainty in Artificial Intelligence* (2002), pp. 577–584. (p. 27)

ZELIC, I., KONONENKO, I., LAVRAC, N., AND VUGA, V. 1997. Induction of decision trees and Bayesian classification applied to diagnosis of sport injuries. *Journal of Medical Systems 21*, 6, 429–444. (p. 3)

ZHANG, H., LING, C. X., AND ZHAO, Z. 2000. The learnability of naive Bayes. In *Proceedings of Canadian Artificial Intelligence Conference* (2000), pp. 432–441. (pp. 3, 26)

ZHENG, Z. 1998. Naive Bayesian classifier committees. In *Proceedings of the 10th European Conference on Machine Learning* (1998), pp. 196–207. (p. 3)

ZHENG, Z. AND WEBB, G. I. 2000. Lazy learning of Bayesian rules. *Machine Learning 41*, 1, 53–84. (p. 46)

ZUPAN, B., DEMSAR, J., KATTAN, M. W., OHORI, M., GRAEFEN, M., BOHANEC, M., AND BECK, J. R. 2001. Orange and decisions-at-hand: Bridging predictive data mining and decision support. In *Proceedings of ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning* (2001), pp. 151–162. (p. 3)