

NOISE TOLERANT DATA MINING

A Dissertation Presented

by

Yan Zhang

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
Specializing in Computer Science

May, 2008

Accepted by the Faculty of the Graduate College, The University of Vermont, in partial fulfillment of the requirements for the degree of Doctor of Philosophy, specializing in Computer Science.

Thesis Examination Committee:

Xindong Wu, Ph.D. Advisor

Xingquan Zhu, Ph.D. Co-advisor

Jeffrey P. Bond, Ph.D. Co-advisor

Christian Skalka, Ph.D.

Richard Single, Ph.D. Chairperson

Frances E. Carr, Ph.D. Vice President for Research and
Dean of the Graduate College

Date: March 21, 2008

Abstract

Learning from noisy data sources is a practical and important issue in Data Mining research. As errors continuously impose difficulty on discriminant analysis, identifying and removing suspicious data items is regarded as one of the most effective data preprocessing techniques, commonly referred to as data cleansing. Despite the effectiveness of these traditional approaches, many problems still remain unsolved.

In this thesis, we study the problem of noise tolerant data mining, which addresses the problem of learning from noisy information sources. We construct a noise Modeling, Diagnosis, and Utilization (MDU) framework, which bridges up the gap between data preprocessing methods and the actual data mining step. This framework provides a conceptual working flow for a typical data mining task and outlines the importance of noise tolerant data mining.

Based on the proposed mining framework, we present an effective strategy that combines noise handling and classifier ensembling so as to build robust learning algorithms on noisy data. Our study concludes that systems that consider both accuracy and the diversity among base learners, will eventually lead to a classifier superior to other alternatives. Based on this strategy, we design two algorithms, Aggressive Classifier Ensembling (ACE) and Corrective Classification (C2). The essential idea is to appropriately take care of the diversity and accuracy among base learners for effective classifier ensembling. In our experimental results, we show that the error detection and correction module is effective in ACE and C2, C2 outperforms its two degenerates ACE and Bagging constantly, and C2 is more stable than Boosting and DECORATE.

From the prospective of systematic noise modeling, we propose an association based learning method to trace and analyze the erroneous data items. We assume the occurrence of the noise in the data is associated with certain events in other attribute values, and use a set of associative corruption (AC) rules to model this type of noise. We apply association rule mining to extract noise patterns and design an effective method to handle noisy data.

To my husband and my parents, who always give me love and support.

Acknowledgements

At this point, I would like to express my sincere appreciation to Professor Xindong Wu. I am deeply indebted to Dr. Wu for the opportunity to work on this research under his guidance. In the four years' study, I have always received his enthusiastic encouragement and invaluable advice. I have learned from him lots of things ranging from experiences of doing research to writing a scientific paper.

I am also grateful to the help from my co-advisors Dr. Xingquan Zhu and Dr. Jeffrey P. Bond. Dr. Zhu was a research assistant professor in the Computer Science department at the University of Vermont and is now an assistant professor in the Department of Computer Science & Engineering at Florida Atlantic University. He kindly gives me much guidance and important suggestions on my research topic. Dr. Bond is a research associate professor in the Department of Microbiology and Molecular Genetics at the University of Vermont. He provides me the guidance on statistical methods and illuminating discussions.

I also want to thank my committee member Dr. Christian Skalka, and my committee chair Dr. Richard Single for their advice and help.

The faculty members and graduate students in the Computer Science department at the University of Vermont are thanked for creating a very pleasant research environment and for their friendship.

Finally, I would like to acknowledge the Department of Computer Science at the University of Vermont for providing me the award of a Graduate Research Scholarship and research facilities.

Table of Contents

Acknowledgments	iii
List of Tables	vi
List of Figures	vii
Chapter 1 Introduction	1
1.1 Motivation	2
1.2 Contributions	5
1.3 Organization	7
Chapter 2 Data Quality and Noise Handling	9
2.1 Background of Data Quality	9
2.1.1 Metrics for Measuring Data Quality	10
2.1.2 Data Quality in Data Mining	13
2.1.3 Data Accuracy Assessment	13
2.2 Noisy Data Handling	15
2.2.1 Noise	15
2.2.2 Removing Noisy Instances	16
2.2.3 Erroneous Attribute Value Detection and Correction	18
2.2.4 Detections of Noise, Outliers, and Frauds	19
2.3 Robust Algorithms	20
2.3.1 Single-Model Algorithms	21
2.3.2 Classifier Ensembling	22
2.3.3 Error Awareness Data Mining	23
2.4 A Noise Modeling, Diagnosis, and Utilization Framework	23
2.5 Chapter Summary	26
Chapter 3 Classifier Ensembling – Principles and Methods	27
3.1 Classifier Ensembling Principle	27
3.1.1 Accuracy	29
3.1.2 Diversity	30
3.1.3 Accuracy & Diversity	33
3.2 Diverse Base Learners	33
3.2.1 Diversity Enhancement Methodologies	34

3.2.2	Bagging	37
3.2.3	Boosting	38
3.2.4	DECORATE	39
3.3	Bias-Variance Decomposition for Classifier Ensembles	40
3.4	Chapter Summary	45
Chapter 4	Corrective Classifier Ensembling	47
4.1	Problem Statement	48
4.1.1	Training Attribute Predictor (AP)	50
4.1.2	Filtering Noisy Instances	51
4.1.3	Constructing Solution List	51
4.1.4	Selecting Solution for Error Correction	53
4.2	ACE: An Aggressive Classifier Ensemble with Error Detection, Correction, and Cleansing	55
4.3	C2: Corrective Classification	58
4.4	Experimental Results	61
4.4.1	Experimental Data	61
4.4.2	Experiment Settings	63
4.4.3	Prediction Accuracy	63
4.4.4	Diverse and Accurate Base Learners	65
4.4.5	Noise Detection Analysis	69
4.4.6	Stability	70
4.4.7	Complexity Comparison	77
4.5	Discussions	78
4.6	Chapter Summary	79
Chapter 5	Noise Modeling with Associative Corruption Rules	81
5.1	Problem Statement	82
5.2	Method	83
5.2.1	Algorithm ACF	84
5.2.2	Algorithm ACB	85
5.3	Experiments	90
5.3.1	Experiment Settings	90
5.3.2	Experimental Results	91
5.3.3	Discussions	93
5.4	Chapter Summary	95
Chapter 6	Concluding Remarks	97
6.1	Summary of Thesis	97
6.2	Future Work	99
References		102

List of Tables

2.1	Four Task-Dependent Metrics for Data Quality	12
4.1	Training Attribute Predictors (AP)	51
4.2	Filtering Suspicious Instances	52
4.3	Solution Set Construction	54
4.4	Random Selection	56
4.5	Best Selection	57
4.6	Aggressive Classifier Ensemble (ACE) Construction	58
4.7	Pseudocode of Corrective Classification (C2)	60
4.8	Dataset Summary	62
4.9	Connections Between Diversity and Variance	76
5.1	Algorithm ACF (Associative Corruption Forward)	86
5.2	Optimization	87
5.3	Building Naive Bayes Learners	88
5.4	Algorithm ACB (Associative Corruption Backward)	89

List of Figures

2.1	Connections among Noise, Outlier, and Fraud	20
2.2	Framework for Noise Modeling, Diagnosis, and Utilization	25
3.1	Accuracy and Diversity Analysis of Classifier Ensembles	31
4.1	The Framework of ACE	56
4.2	The Framework of C2	59
4.3	Comparative Results of Prediction Accuracy on Ten Datasets from UCI Data Repository	66
4.4	Comparison of Three Methods on Dataset Monks3 with Noise Levels 10%-40%	68
4.5	Noise Detection Results	70
4.6	Multiplicative Factor Values in the Bias-variance Decomposition for Classifier Ensembles	71
4.7	Comparative Results of the Variance in Ensembling Methods on Ten Datasets from UCI Data Repository	74
4.8	Time Complexity Comparison	78
5.1	Information on the Datasets for Experiments	93
5.2	Comparative Results of Five Learning Models	94

Chapter 1

Introduction

Data mining research is dedicated to exploring and extracting implicit, previously unknown, and potentially useful information from data. Along with the continuing development of new technologies, the volume of data collections increases in almost all fields of human endeavor. Lack of data is no longer the problem – but lack of effective and efficient methods to prepare, learn and take act on the massive data has become a crucial problem. In this thesis, we investigate noise tolerant data mining, which addresses the problem of learning in the presence of noise from a high-level viewpoint. To tackle this important problem, a number of relevant issues have been studied by different research groups in the data mining community. Although some progress has been made over the years, limitations in the existing methods and related issues still remain unsolved.

In this chapter, we will first provide the motivations behind this thesis work, as well as an outline for the problems to be addressed in the following chapters. We will then state the thesis questions clearly, including reasons why they are important, followed by a chapter-by-chapter synopsis of the thesis contents.

1.1 Motivation

Existing research efforts (Maletic and Marcus 2000; Orr 1998) have suggested that the average error rate of a dataset in a data mining application is around 5%-10%. There are numerous reasons that contribute to data imperfections. For instance, faulty measuring devices, transcription errors, and transmission irregularities. In order to target the problem of learning in the presence of noise, researchers have developed two categories of methods – data quality enhancement and robust model construction. First and foremost, effective control, evaluation, and enhancement of the quality of the input data is important. It's worthwhile remembering the old saying “Garbage in and Garbage out” when carrying out data mining. Poor quality of data can jeopardize any attempt to use data analysis for decision making. From the other prospective, it is always good to develop algorithms that can tolerate a certain amount of noise. In application problems such as image recognition and signal processing, the robustness of the learning method is especially important.

The problem of data quality improvement includes many aspects. A lot of efforts have been attempted towards data quality enhancement through advanced data preprocessing, including data cleansing, data integration and transformation, data reduction, and data discretization (Han and Kamber 2000). Among these topics, this thesis focuses on the noise handling, which is a subtopic of data cleansing. Relevant research work on noise handling includes outlier detection (Rousseeuw and Leroy 1987), mislabeled data identification (Brodley and Friedl 1999) erroneous attribute value detection (Teng 1999), and suspicious instance ranking (Zhu, Wu, and Yang 2004). These research efforts have addressed important issues in learning from noisy data, and different methodologies have been proposed to handle the data imperfections. The limitations of existing noise handling techniques are from three aspects. First, the noise detection and correction methods intend to automatically and permanently filter out or change noisy data items, while in many situations, there are no objective metrics available to assess the validity of such data manipulations. As a result, these methods may result in the information loss, and the incorrectly filtered or

changed data entries make the succeeding data mining algorithms insufficient to discover the genuine knowledge models. For many content sensitive domains, such as medical, financial, or security databases, this kind of methods is simply not a good option. Second, most noise handling methods take the principle “purging the noisy data before mining”, such that they are usually performed without the consideration of the succeeding mining algorithms. As a result, these methods neglect the fact that the noise information may aid in the learning model construction. Third, existing noise handling methods pay little attention to noise modeling and understanding. Under the situation where noise sources and noise patterns are needed to be explored, especially when the noisy data conforms to certain systematic patterns, it is necessary to include a noise modeling procedure.

Instead of trying to improve the data quality, the other category of methods rely on robust model construction to handle noisy information sources, which essentially leaves the burden of noise handling to the learning algorithms. The dataset retains the noisy instances, and each algorithm must involve its own noise-handling routine to ensure the robustness of the model constructed from the data. The most representative methods include pruning (Quinlan 1986) and prototype selection (Aha, Kibler, and Albert 1991). The essential idea is to prevent the learner from overfitting to the training data. To achieve this purpose, it is important to avoid overly complicated structures that are prone to fit the noise and data exceptions in the learning model. These methods focus more on optimizing the structure of the model, rather than diagnosing possible erroneous data entries. Consequently, they may not take much effect on learning from the data that contains erroneous entries, especially when the errors in the source data follow a systematic pattern.

For the same purpose to ensure robustness, classifier ensembling methods stand out by the idea of constructing the learning model from a set of base learners. Different ensembling methods usually have a unique way to construct and combine their base learners. Classifier ensembling has been proven an effective design that generally outperforms the single model (Dietterich 2000a). Two representative methods are Bagging (Breiman 1996) and Boosting

(Freund and Schapire 1996). Although not purposely designed for learning on noisy data sources, they perform quite effectively on noisy data compared to single-structure models. However, due to the nature of these methods, they face the same problems as the robust algorithms as previously stated.

In summary, existing data mining efforts in learning from noisy data sources are effective in their own scenarios, but some problems are still open. We present these problems as follows.

- Insufficient data understanding. No mechanisms are available to assist users to gain a comprehensive understanding of their data, such as finding and ranking erroneous data entries, modeling data errors and pinpointing error sources.
- Information loss and data sensitivity. For data imperfections, existing applications rely heavily on data preprocessing to filter the data and release processed data copies for future actions. This filtering/cleansing approach, however, may eventually incur unrestorable information loss. In many situations, due to data sensitivity, its is not allowable to make any unauthorized data changes.
- Noise tolerant data mining. It will be very difficult, if not impossible, to immunize real-world data from errors, even if data preprocessing or domain experts have been fully involved. Developing generic data mining solutions which can tolerate and accommodate data errors is crucial to advancing real-world data mining systems.

Motivated by these observations, our research is devoted to constructing a high-level noise tolerant data mining framework, that builds up the connection between the noisy data and the learning algorithm. The essential idea is to properly model, diagnose, and utilize the noise information to assist the actual data mining process.

1.2 Contributions

This thesis performs a systematic study on noise tolerant data mining, which addresses the problem of learning in the presence of noise. We aim to advance noise modeling and handling in inductive learning. In particular, the main contributions of the thesis are threefold, and we state them as follows.

Firstly, this work comprehensively discusses and clarifies important issues of the noise handling problem in data mining research, including data quality, noise, and the connections among noise detection, outlier detection, and fraud detection. Motivated from these analyses, a novel noise modeling and reasoning framework is proposed. Some detailed work is described as follows.

- **Survey on Data Quality and Related Issues.** Data quality is a common issue that exists in many domains such as database systems, data mining, management information systems, etc. In this thesis, we categorize the multi-dimensional metrics that assess data quality into three classes, which applies to a more general data mining research topic than existing methods. Among various data preprocessing techniques for data quality improvement, noise detection and correction represents a category of effective and promising solutions. We also discuss the inherent connections among noise detection, outlier detection and fraud detection. (Chapter 2)
- **Noise MDU Framework.** Having discussed issues about noise handling, we propose a three-phase noise **M**odeling, **D**iagnosis, and **U**tilization framework to build up the connections between the noisy data and the actual learning algorithm. This framework is especially useful for the data that describes a systematic noise pattern. (Chapter 2)

Secondly, we will propose a novel and effective strategy to construct learning models from noisy data for classification problems. The essential idea is to take advantage of both classifier ensembling and noise handling methods for the purpose of delivering a diverse

and accurate classifier ensemble. The principle of classifier ensembling is analyzed, and two novel learning algorithms are proposed. We describe them in detail as follows.

- **Analysis of Classifier Ensembling Principle.** We introduce and analyze the principle of designing an effective classifier ensemble from the viewpoint of learning from noisy data sources. First, we point out the importance of both accuracy and diversity of the base learners in a classifier ensemble by analyzing the mechanism of classifier ensembles. Second, we analyze the rationale of bias-variance decomposition, which further demonstrates that maintaining a proper balance between the accuracy and the disagreement among base learners is important. (Chapter 3)
- **ACE Algorithm.** ACE (Aggressive Classifier Ensemble) is designed to be a classifier ensemble with accurate base learners. The training data of each base learner is a noise corrected copy of the original dataset. Because the noisy instances are corrected by randomly choosing the possible corrections, the corrected copy of the same training data differs from each other. The experimental results show that the base learners of ACE are not diverse; however, it empirically verifies the effectiveness of the noise handling method we have applied. (Chapter 4)
- **C2 Ensembling.** Following the ACE design, we propose C2 (Corrective Classification) ensembling, which leverages the advantages of error detection and correction to deliver an accurate and diverse classifier ensemble. C2 essentially relies on the bootstrap sampling principle to build diverse base training datasets, followed by an error detection and correction module on each training dataset. The resulting classifier ensemble shows its efficacy over several other popular classifier ensembling methods. (Chapter 4)

Thirdly, this thesis explores the noise modeling problem in a systematic way. In particular, an association based noise modeling problem is proposed and analyzed, which is described as follows.

- **Noise Modeling and Reasoning by AC rules.** We propose an association based learning method to trace and analyze the erroneous data items, in which the occurrence of an error on one attribute is dependent on several other attributes. We use a set of associative corruption (AC) rules to model this type of noise. Our approach consists of two algorithms, Associative Corruption Forward (ACF) and Associative Corruption Backward (ACB). Algorithm ACF is proposed for noise inference, and ACB is designed for noise elimination. The experimental results show that the ACF algorithm can infer the noise formation correctly, and ACB indeed enhances the data quality for supervised learning. (Chapter 5)

1.3 Organization

The remaining chapters of this thesis are organized as follows.

In Chapter 2, we discuss important concepts relevant to data quality and noise handling, including data quality, noise and related concepts. Along with a comprehensive review of state-of-the-art literature, we present our unique viewpoints on data quality in data mining, and review corresponding noise handling methods for data accuracy enhancement. We then review robust algorithms for learning from noisy data. Based on the analysis of possible sources of noisy data, the weakness of existing methods, and the connections among noise detection, outlier detection, and fraud detection, we present the design of the noise Modeling, Diagnosis, and Utilization (MDU) framework, so as to build up a connection between the noisy data and the learning algorithm. The essential idea is to model the noise pattern, diagnose the possible erroneous data items, and utilize the noise information to assist the actual data mining process.

In Chapter 3, we introduce the theoretical background of classifier ensembling and review popular strategies that create a diverse classifier ensemble. We present the two major principles of designing a classifier ensemble, and explore the reasons why classifier ensembles usually gain performance than single-model learners. In addition, we review and analyze the

bias-variance decomposition in order to understand the prediction behavior of the classifier ensemble. Besides the prediction accuracy, which is a major measure for evaluating the robustness of a learning algorithm, we define another criterion – stability, to evaluate a classifier ensemble.

In Chapter 4, we present an effective strategy – a combination of noise handling and classifier ensembling, to build robust learning algorithms on noisy data. In particular, we propose two algorithms – Aggressive Classifier Ensembling (ACE) and Corrective Classification (C2). C2 incorporates data cleansing, error correction, Bootstrap sampling and classifier ensembling for effective learning from noisy data sources. ACE and Bagging are two degenerate methods of C2. Two unique designs make C2 distinct from existing algorithms. On the one hand, a set of diverse base learners of C2 that constitute the ensemble are constructed via a Bootstrap sampling process; on the other hand, C2 further improves each base learner by unifying error detection, correction and data cleansing to reduce the overall noise impact. Being corrective, the classifier ensemble is built from the data that has been preprocessed/corrected by the data cleansing and correcting modules.

In Chapter 5, we present an active learning approach to the problem of systematic noise modeling, inference, and elimination, specifically the inference of Associative Corruption (AC) rules. AC rules are defined to simulate a common noise formation process in real-world data, in which the occurrence of an error on one attribute is dependent on several other attribute values. Our approach consists of two algorithms, Associative Corruption Forward (ACF) and Associative Corruption Backward (ACB). Algorithm ACF is proposed for noise inference, and ACB is designed for noise elimination. The experimental results show that the ACF algorithm can infer the noise formation correctly, and ACB indeed enhances the data quality for classification problems.

In Chapter 6, we conclude the thesis, summarizing the issues we have targeted and how this has contributed to our understanding of learning from noisy data sources in general. In addition, we discuss several future studies that are suggested by these investigations.

Chapter 2

Data Quality and Noise Handling

In this chapter, we discuss important concepts relevant to data quality and noise handling. These concepts will be frequently referred to throughout this thesis. Along with a comprehensive review of state-of-the-art literature, we present our unique viewpoints on data quality in data mining in Section 2.1, and review corresponding noise handling methods for data accuracy enhancement in Section 2.2. We then review robust algorithms for learning from noisy data in Section 2.2.3. Based on the demonstration of possible sources of noisy data in Section 2.2.1, the connections among noise detection, outlier detection, and fraud detection in Section 2.2.4, and the weakness of existing methods, we present the design of the noise Modeling, Diagnosis, and Utilization (MDU) framework in Section 2.2.3.

2.1 Background of Data Quality

Data quality problems may arise anywhere in the context of information systems, which cover the organizational processes, procedures, and roles employed in collecting, processing, distributing and using data (Strong, Lee, and Wang 1997). In the past ten to twenty years, a large number of research efforts have been put on data quality (Gertz, Ozsu, Saake, and Sattler 2004). There are several fundamental questions regarding the data quality. Among them, the most important ones include (1) how is data quality defined and developed? (2)

how is data quality measured? (3) what dimensions of data quality are of greatest concern in data mining research? and (4) what does noise refer to? We will address these questions in this section.

Data quality is an important issue in database systems, although the term was not defined in an explicit way in the earlier days (Gertz, Ozsu, Saake, and Sattler 2004). For example, a research article in 1989 talked about data integrity (Motro 1989). It stated that “Database integrity has two complementary components: validity, which guarantees that all false information is excluded from the database, and completeness, which guarantees that all true information is included in the database.” The author claimed that for a database query, the “answers have integrity if they contain the whole truth (completeness) and nothing but the truth (validity).”

This example indicates that the **truthfulness** of the data is the most important characteristic in database systems. Later on, along with the emergence of distributed systems and data warehouses, the data has to be collected and merged from different sources. Since they may contain incompatible data storage structures, **data inconsistency** becomes a new issue that needs to be considered. When the data producing and updating speed keeps growing along with the development of web technology, the **timeliness** of the data in a database system is brought into the discussion. More recently, other concerns such as **security** and **interpretability** have become important aspects of data quality in a database management system.

2.1.1 Metrics for Measuring Data Quality

By showing the definition of data quality in database systems, we notice that data quality is a multi-dimensional concept, the content of which is evolving and getting more and more enriched. In management information systems, the assessment metrics of data quality have been investigated extensively (Kahn, Strong, and Wang 2002; Pipino, Lee, and Wang 2002). According to (Kahn, Strong, and Wang 2002), high-quality data is simply described

as “data that is fit for use by data consumers”. While this description intuitively captures the essence of quality, it is difficult to apply this definition to data quality measurement in practice. In (Pipino, Lee, and Wang 2002), the authors categorize the metrics of data quality into objective metrics and subjective metrics, where the former are summarized based on the dataset in question while the latter reflect the needs and experiences of the people who consume the data products. This categorization emphasizes the importance of the data itself and the role of the data users. In this thesis, however, we want to understand the data quality in a more general context. Thus, we categorize the assessment metrics of data quality into three classes: task-independent, task-dependent, and task-specific metrics, where the task-independent metrics reflect states of the data without the contextual knowledge of the application, the task-dependent metrics address the aspects that are generally considered by any data involved applications but the assessment results to these metrics crucially depend on the task at hand, and the task-specific metrics include any specific requirement that only pertains to the target application contexts.

The task-independent metrics are objective assessment to the data itself, and they concentrate on the intrinsic characteristics of data. Thus, they could be assessed by solely considering the dataset, no matter which application the data is applied for. The most important aspects of task-independent data quality are covered by the accuracy of the data. Besides complying with the key components concerned in traditional data quality metrics, the accuracy of data has more meaning in its content. Since the accuracy represents the data correctness, it not only requires the data to be free from errors, but also the data content to be unbiased and obtained from reliable sources.

In order to better characterize data quality dimensions, it is important to recognize that data quality cannot be studied in isolation. The same dataset may be highly qualified for one task while of little use to the other. The essential component of the task is either the human user, a machine, or any learning models that consume and make use of the data. Therefore, accurately identifying what data characteristics are required for the task in-hand is crucial.

Table 2.1: Four Task-Dependent Metrics for Data Quality

Completeness:	no missing information, sufficient records /attributes that cover the whole theory
Relevancy:	the attribute of the data is relevant
Timeliness:	information is sufficiently up-to-date for the task at hand.
Sufficiency:	the volume of data is appropriate for the application.

Here, we define task-dependent metrics as the dimensions that are generally considered by any data involved applications, but the assessment of these metrics is subjective to the task that the data is applied for. For example, the relevancy of the data is always concerned in any data mining applications, but how relevant the data is largely depended on the objective of the task. Table 2.1 shows four task-dependent metrics that are always being considered, but the assessment of these metrics to the same dataset differs upon the objective of the study.

The third category of metrics of data quality, task-specific metrics, indicate the metrics that are only meaningful to certain application domains in discussion. In other words, these metrics are only concerned in a specific task or domain. For instance, in the study of management information systems, where the human users play an important role, it is natural to assess data quality from the aspect of usefulness and usability. Any factors that have potential effect on the data-user interface may be taken into the data quality consideration. In (Strong, Lee, and Wang 1997), the data accessibility, which indicates the ease with which the users can manipulate this data to suit their needs, becomes a task-specific metric of the data quality. Whether the data representation is understandable or interpretable, and whether the data can be securely accessed are also considered as components in accessibility. Another example is related to the pharmaceutical industry. From the Federal Drug Association (FDA)'s perspective, the important aspects of data quality include a valid representation of the clinical trial and other metrics pertaining to

drug safety, pharmacokinetics, and efficacy (Collins 1998).

2.1.2 Data Quality in Data Mining

As a general problem in data mining research, the task-independent metrics such as data accuracy and some task-dependent metrics are of the most concern. In particular, it is more concerned about whether the data is free from error, whether the data contains missing information, whether the data attributes could sufficiently describe the concepts they are supposed to characterize, whether the distribution of the data records sufficiently reflect the actual population of data, and whether the data contain inconsistent, or redundant instances. There could be other task-specific metrics required for the data quality when going down to a specific data mining task.

In a data mining task, the data quality is usually controlled and enhanced in the stage of data preprocessing, which includes data cleansing, data integration, data transformation, and data reduction (Han and Kamber 2000). Data cleansing is the act of detecting, removing, or correcting corrupted or inaccurate instances from a dataset. It attempts to fill in missing attribute values, handling outliers or erroneous attribute values, and correcting inconsistent data. Data integration is a procedure that aims at merging data from multiple sources into a coherent database. Data transformation involves many necessary data manipulation methods, such as normalization, concept generalization, and discretization. Data reduction techniques are applied when the original data volume is too large to be analyzed efficiently. It helps to obtain a reduced representation of the dataset that is much smaller in volume, yet closely maintains the integrity of the original data.

2.1.3 Data Accuracy Assessment

In Section 2.1.1, we categorized the data quality dimensions into three groups, namely task-independent, task-dependent, and task-specific metrics. When performing assessments on these metrics, we face two types of situations. For task-dependent and task-specific metrics,

the assessment of a quality metric is entirely subjective to the task in hand. For example, only the data consumer is able to judge the timeliness metric of a data product. Therefore, the data consumers should follow a set of principles to develop metrics specific to their needs. On the other hand, there exist objective methods to assess the task-independent metrics, the accuracy of data in particular. For example, any out-of-range value can be objectively assessed based on the predefined rules on the data. Another possible objective assessment comes from an expert, either a human expert or advanced methods, which are believed and trusted to be able to provide a very precise judgement.

In real world applications, however, it is often difficult to apply these objective measures to improve the data accuracy. While sometimes the data generator may provide a rough estimate about the error rate of a dataset, or the error rate of a particular attribute, it is unrealistic to tell which attribute value is erroneous if it is not an out-of-range value. It is even harder to know whether a particular attribute value modification is actually correct when applying some data preprocessing techniques to the data. Consequently, we may be in the position of being unable to find a reliable judgement for our data.

One viable solution to this problem is to make a task-dependent assessment on the data accuracy, rather than an objective one. In other words, ask the data consumers whether they are satisfied with the data accuracy. In the context of data mining research, we may switch to assess whether the knowledge learned from the data is robust. If the knowledge is robust we may make the judgement that the input data is accurate. To evaluate whether a particular method enhances the data accuracy, we may compare the robustness of two sets of knowledge induced from the original data and from the modified data, respectively. Although the knowledge is usually represented by different forms according to the learning model applied, we eventually could get a set of quantitative metrics for comparison. Taking the classification tree as an example, the training accuracy, prediction accuracy, the tree size, and the number of tree nodes, could be used as the metrics to assess the robustness of a decision tree.

2.2 Noisy Data Handling

As we have discussed in Section 2.1.2, one important aspect of data quality improvement in data mining research is data accuracy enhancement. Noisy data handling, a category of data preprocessing techniques, is performed in this regard. We first introduce the concept “noise” and the possible sources of noisy data in Section 2.2.1, and review two types of noise handling methods in Section 2.2.2 and Section 2.2.3. In Section 2.2.4 we illustrate the connections among the detections of noise, outliers, and frauds.

2.2.1 Noise

“Noise” is a broad term that has been used in various studies and is interpreted in different ways. In machine learning and data mining research, several definitions have been proposed. For examples, noise is defined as “a random error or variance in a measured variable” in (Han and Kamber 2000); and is defined as “any property of the sensed pattern which is not due to the true underlying model but instead to randomness in the world or the sensors” in (Duda, Hart, and Stork 2000). We may notice from these definitions that, noise is considered as random turbulence, or randomly occurred errors. In more generalized situations, however, noise is considered as erroneous data entries, either randomly or systematically generated. For example, in (Hickey 1996), noise in the data refers to anything that obscures the relationship between the feature attributes and the target class. In this thesis, *noisy data* refers to the data that contains erroneous data entries, where an erroneous value on either a feature attribute or a class label is called *noise*, and an instance that contains noise is called a noisy instance. Unless specified otherwise, this thesis will use “noise” in the data interchangeably with “error”.

A number of possible reasons can contribute to noisy data. We demonstrate some of them as follows. First, equipment problems. The noisy data may come from the malfunction or inaccuracy of the equipment. Second, large datasets collected through automated methods. These automatic methods usually produce a large volume of data, which could

be unstructured, and under loose quality control. The data collected from the Web makes one example for this case. Data inconsistency, say out-of-range values (e.g., Age: -10) or impossible data combinations (e.g., Gender: Male, Pregnant: Yes) may occur. Third, transmission errors, which are incurred from the conveyance of data from one spot to another. When the data points are signals transmitted through a certain medium before they could be collected, they are prone to noise incurred from the transmission medium. The sensor data in wireless sensor networks makes a good example in this type. Fourth, people self-identified information. In many situations people are asked to provide information by themselves. For instance, self-identified age, income, occupation, and race in a questionnaire; self-identified smoking history, disease history and other privacy items in a medication diagnosis. These self-identified information items are prone to errors from either intentionally posed false answers, or the unawareness of correct answers. Fifth, encrypted data. The data is disguised by a specific encryption method to keep the data privacy. Privacy preserving data mining is an active research direction about it (Agrawal and Srikant 2000). Sixth, incorrect class labeling due to insufficient/incomplete descriptive attributes. The set of descriptive attributes may not be complete enough to correctly distinguish all possible group memberships. As a result, misclassification could happen. The labeling errors in the class attribute are called *class noise*. For example, in medical diagnosis where there may be several possible disease categories for a given set of observations and where further observations (i.e. a more complete description) are not possible for reasons of time or cost concerns.

2.2.2 Removing Noisy Instances

The first category of noise handling methods attempt to detect and discard the instances which are subject to noise according to certain evaluation mechanisms. The learning algorithm then constructs a model using only the remaining instances. Similar ideas exist in robust regression and outlier detection techniques in statistics (Rousseeuw and Leroy 1987).

In 1995, John proposed a method by applying pruned C4.5 decision trees (John 1995). In a decision tree, the author assumed that the set of instances used to build the node and its subtree that are later discarded by pruning are uninformative or harmful to the model construction. Therefore, the algorithm discards these confusing instances and retain the remaining instances as the training data. The algorithm continues pruning and retraining until no pruning can be done. This idea is borrowed from robust statistics (Huber 1981).

In 1999 Brodley and Friedl proposed a procedure to identify mislabeled instances from the training data (Brodley and Friedl 1999). The essential idea is using m learning algorithms to filter out the instances that are prone to labeling errors. The method first splits the training data into n parts, like an n -fold cross-validation. For each of the n parts, the m filtering algorithms are trained on the other $n - 1$ parts. The m resulting classifiers are then used to predict on instances in the excluded part, and finally decide whether the instances are correctly labeled or not. All of the instances identified as mislabeled are removed, and the filtered set of training instances is provided as the input to the final learning algorithm. The authors provided two strategies for building m filtering algorithms: the single algorithm filters when $m = 1$ and ensemble filters when $m > 1$.

In instance-based learning, removing noisy instances has been a main focus of research to improve the performance of nearest neighbor classifiers, where the data volume is large while the classification accuracy is suffered from noisy data and other data exceptions. In 1972, Wilson proposed a method that attempted to identify and remove potentially noisy instances by using a k -nearest neighbor (k -NN) classifier (Wilson 1972). All instances which are incorrectly classified by their k -nearest neighbors are assumed to be noisy instances. Tomek extended this approach by carrying out the same procedure repeatedly, until no more instances are being identified as potentially noisy instances (Tomek 1976). Instead of talking about removing noisy instances, Aha *et.al.* demonstrated that selecting instances based on their contribution to classification accuracy in an instance-based classifier improves the accuracy of the resulting classifier (Aha, Kibler, and Albert 1991).

2.2.3 Erroneous Attribute Value Detection and Correction

Rather than filtering out noisy instances, another category of noise handling methods target the problem by trying to identify and correct the erroneous attribute values. This is a better strategy to use so that the resulting dataset could preserve much of the original information, but conform more to the ideal noise-free case.

One representative method, called polishing, is designed to handle the possible erroneous feature values or class labels (Teng 1999). Instead of assuming conditional independence among the feature attributes, it is assumed that there is some pattern of relationship among the feature attributes and as well the class attribute. In a ten-fold cross validation procedure, the algorithm swaps the role between one feature attribute and the class attribute in the nine-fold training data, based on which a learner is built. This learner is then used to performing the predictions on the remaining one-fold data. If the predicted value of the feature attribute differs from the original feature attribute value, this predicted attribute-value pair is put into the correction recommendation list. In the adjustment phase, firstly a ten-fold cross validation on the training dataset is performed. The instances that are classified incorrectly are treated as possible noisy instances, and the attribute-value pair in the correction recommendation list is then used to correct these instances.

Zhu *et.al.*, extended the idea of polishing and further proposed a method to rank the potentially noisy instances by their impact on the classification process (Zhu, Wu, and Yang 2004). The impact sensitive ranking takes the information-gain ratio as the evaluation criterion to calculate the impact of each suspect instance on the learning system. The assumption is that the correction on some noisy instances may bring more benefit (higher classification accuracy) to the learning model than correcting other instances. The realistic value of this problem is that given a certain amount of expenses (e.g. processing time), the data manager can maximize the system performance by putting priority on instances with higher impacts.

2.2.4 Detections of Noise, Outliers, and Frauds

The concepts noise detection, outlier detection, and fraud detection are originated from different research area, but share some similar characteristics.

As we have discussed in Section 2.2.1, the noise could be involved in either independent variables (feature attributes) or dependent variables (class attributes); some types of noise could be logically easy to be identified but other types are not; and noise could be either accidentally incurred or deliberately falsified. The task of noise detection usually includes detecting mislabeled instances, detecting noisy attribute values in some instances, and detecting the possible noise distributions.

An outlier is generally considered to be a data point that is far outside the norm for a variable or population. In other words, it is an observation that is numerically distant from the rest of the data (Judd and McClelland 1989). Because outliers can have deleterious effects on statistical analysis, researchers should always check for them. Outliers are sorted into two major categories: those arising from errors in the data, and those arising from the inherent variability of the data (Anscombe 1960). Thus, we know that not all outliers are noise, and not all noisy data points show up as outliers. We illustrate the relationship between noise and outliers in Figure 2.1.

A fraud, in the broadest sense, is an intentional deception made for personal gain and resulting in injury to another party. Some fraud detection problems, for example money laundering, credit card fraud, telecommunication fraud, and computer intrusion, have been investigated with the aid of statistical and machine learning methods. A fraud can take an unlimited variety of different forms, where data mining methods usually play roles in two categories of fraud detection: identity theft and falsified information. Identity theft is a term for crimes involving illegal usage of another individual's identity. The most common applications of identity theft detection include the detection of credit card fraud (Hand and Blunt 2001) and cellular clone fraud (Fawcett and Provost 1997). In these problems, instances in the data usually represent either regular or fraudulent activities. One

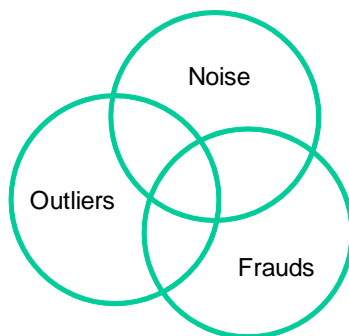


Figure 2.1: Connections among Noise, Outlier, and Fraud

typical method is to model individual customers' previous usage patterns, and a suspicious score is computed for future activities. If the score falls out of the predefined range, the corresponding instance may be considered fraudulent. This type of methods exactly follows the routine of outlier detection.

The other category of fraud detection, the detection of falsified information, aims at detecting the data entries that do not reveal the truth. The applications of this category include the detection of fabrication in clinical trials (Al-Marzouki, Evans, Marshall, and Roberts 2005), deceptive information in crime data mining (Wang, Chen, and Atabakhsh 2004; Chen, Chung, Xu, Wang, Qin, and Chau 2004), and falsified data in survey research (Johnson, Parker, and Clements 2001). In order to detect falsified information, there is an imminent need to effectively model systematic noise, and develop corresponding detection methods.

To summarize, we state that noise detection and outlier detection represent two categories of methodologies, which could be applied to various fraud detection applications.

2.3 Robust Algorithms

In this section, we first review learning methods that are attempting to build a robust model that could tolerate a certain amount of noise in the input data in Section 2.3.1. As a special type of robust models, classifier ensembling methods will be discussed in Section 2.3.2. In

the end, we review error awareness data mining in Section 2.3.3.

2.3.1 Single-Model Algorithms

The noise handling techniques reviewed in Section 2.2 tackle the problem from the input end, where they detect and eliminate the noise in preprocessing of the training set. In this section, we review inductive learning algorithms that have a mechanism for handling noise. These algorithms are called robust algorithms or noise-tolerant systems since they leave the task of dealing with imperfections to the theory learner. The rule of thumb to cope with noise is avoiding overfitting, so that the learner does not build overly complicated structures just to fit the outliers, noise, or other data exceptions. Such methods typically use statistical measures to remove the least reliable structures in the model, generally resulting in faster classification and an improvement in the ability of the learner to correctly classify independent testing data.

For example, pruning in decision trees is designed to reduce the chance of overfitting (Quinlan 1986). There are two common approaches to tree pruning: prepruning and postpruning (Han and Kamber 2000). During the construction of a decision tree, a prepruning approach prunes the tree by halting its construction early. On the other hand, postpruning approaches attempt to remove branches from a “fully grown” tree. Other noise handling mechanisms can also be incorporated in search heuristics in decision tree construction (Mingers 1989). Another example is the noise handling module in CN2 algorithm (Clark and Niblett 1989), which is a rule-based learning algorithm. When noise is present, overfitting can lead to long rules. Thus, to induce short rules, one must usually relax the requirement that the induced rules be consistent with all the training data. A particular mechanism that decides the termination in the search during rule construction is used in CN2 algorithm. There are two heuristic decisions involved during the learning process, and the CN2 employs two evaluation functions to aid in these decisions. The first function assesses the quality of a rule, and the second evaluation function tests whether a rule is significant.

Other noise handling strategies, for example, compression measures (Srinivasan, Muggleton, and Bain 1992; Gamberger, Lavrač, and Grošelj 1999), error-correcting code (Dietterich and Bakiri 1991), and a fuzzy logic based approach (Zhao and T. 1995), have greatly helped with learning from noisy data.

2.3.2 Classifier Ensembling

With the same purpose of learning robust models, classifier ensembling distinguishes itself by involving the prediction from multiple base learners instead of from a single learner. An ensemble consists of a set of individually trained classifiers, such as neural networks or decision trees, whose predictions are combined when classifying new instances. The performance of each base learner does not have to be comparable with the single learner built from the original dataset, as long as the combined learner is robust. Although classifier ensembling methods are not designed for learning from noisy data sources in general, their performance outperforms a single learner that includes a noise handling process in many cases.

Several ways to define an ensemble have been explored, such as mixtures of experts, classifier ensembles, multiple classifier systems and consensus theory (Kuncheva and Whitaker 2003; Breiman 1996; Freund and Schapire 1996; Melville and Mooney 2003; Zhou and Yu 2005). Despite these different names, their key idea is in common: a series of base learners are learned, and an ensemble which unifies all base learners is used to make the final decision, so that the combined learner is supposed to have more satisfactory results compared to any single base learner. While the base learners being constructed, one critical issue is to create a predictive error diversity among base learners. It is usually achieved by performing effective manipulations on the training data, feature attributes, class attributes, or the learning algorithms. Among various mechanisms that are claimed to be able to increase the diversity of base learners, injecting randomness is an effective, empirically proven strategy that has the support of statistical theory. Several usually used randomness producing tech-

niques include random sampling from the training data, random feature selection, injecting random noise into the class attribute or dependent variable, and randomizing parameter settings of the learning algorithm. In some method designs, other ad-hoc mechanisms are used together with the randomness injection. In the successive subsections we attempt to move towards these aspects to understand many different classifier ensembling methods.

2.3.3 Error Awareness Data Mining

In (Zhu and Wu 2006), Zhu and Wu proposed a concept “Error Awareness Data Mining”, where the authors pointed out that during the data mining process, sometimes the noise knowledge is known before the actual mining process, and such previously known error knowledge should be incorporated into the mining process for improved mining results. In other words, the data quality enhancement step should not be isolated from the model construction step. Instead, the model construction procedure could make use of the available error information, but not to try to cleanse the original data. The study presented a case for Naive Bayes, where each feature attribute has a certain probability of being corrupted. The knowledge of noise corruption is assumed to be the prior knowledge, so that the Naive Bayes classifier constitutes this information in the learning process.

2.4 A Noise Modeling, Diagnosis, and Utilization Framework

In previous sections, we have reviewed several effective noise handling methods, but there are still unsolved issues. Firstly, there is no noise modeling procedure currently available in existing noise detection methods. No mechanisms are available to assist users to gain a comprehensive understanding of their data, modeling data errors and pinpointing error sources. In many methods, although the random noise is not assumed explicitly, we may notice it from the experiment designs where the artificial noise was randomly injected for algorithm comparisons. There is little work concerning systematic noise modeling and handling. Secondly, the robust algorithms discussed in Section 2.3 focus more on optimizing

the structure of their model, rather than diagnosing possible erroneous data entries. Consequently, they may not take much effect on learning from the data that contains erroneous entries, especially when the errors in the source data follow a systematic pattern.

Motivated by these observations, we propose a noise modeling, diagnosing, and utilization framework as shown in Figure 2.2, so as to build up a connection between the noisy data and the learning algorithms. The essential idea is to model, diagnose, and utilize the noise information to assist the actual data mining process.

The noise MDU model illustrated in Figure 2.2 depicts the noise handling process as the input module together with other three major procedures: noise modeling & understanding, noise diagnosing, and noise utilization.

The input module consists of two components: noisy data (A) and noise information (B). The noisy data is the source data we want to learn from, and the noise information is any problem specific information from which a possible noise pattern can be drawn. The component B delivers very important prior information, which includes the objective of the task, the clues on the noise formation, if any, and other domain knowledge that helps. If B contains very limited noise information, the user may choose to skip the noise modeling module and go directly to noise diagnosing from the input module.

Given the input information, component G is used to perform noise modeling and understanding. In some studies where the data accurately describes a systematic noise pattern, the noise modeling may induce very valuable information. For example, in the study of stock trading data, a systematic behavior of the trading of individuals is identified (Barber, Odean, and Zhu 2004). The impact of these accumulated individual behaviors is non-trivial to the whole bulk of stock trading data, so that the study on this systematic behavior becomes valuable to economists. Another example is the fraud detection problems we have discussed in Section 2.4, in which the data entries that do not reveal truthful information are to be detected.

The modeled noise pattern, along with the input information, will assist the noise diag-

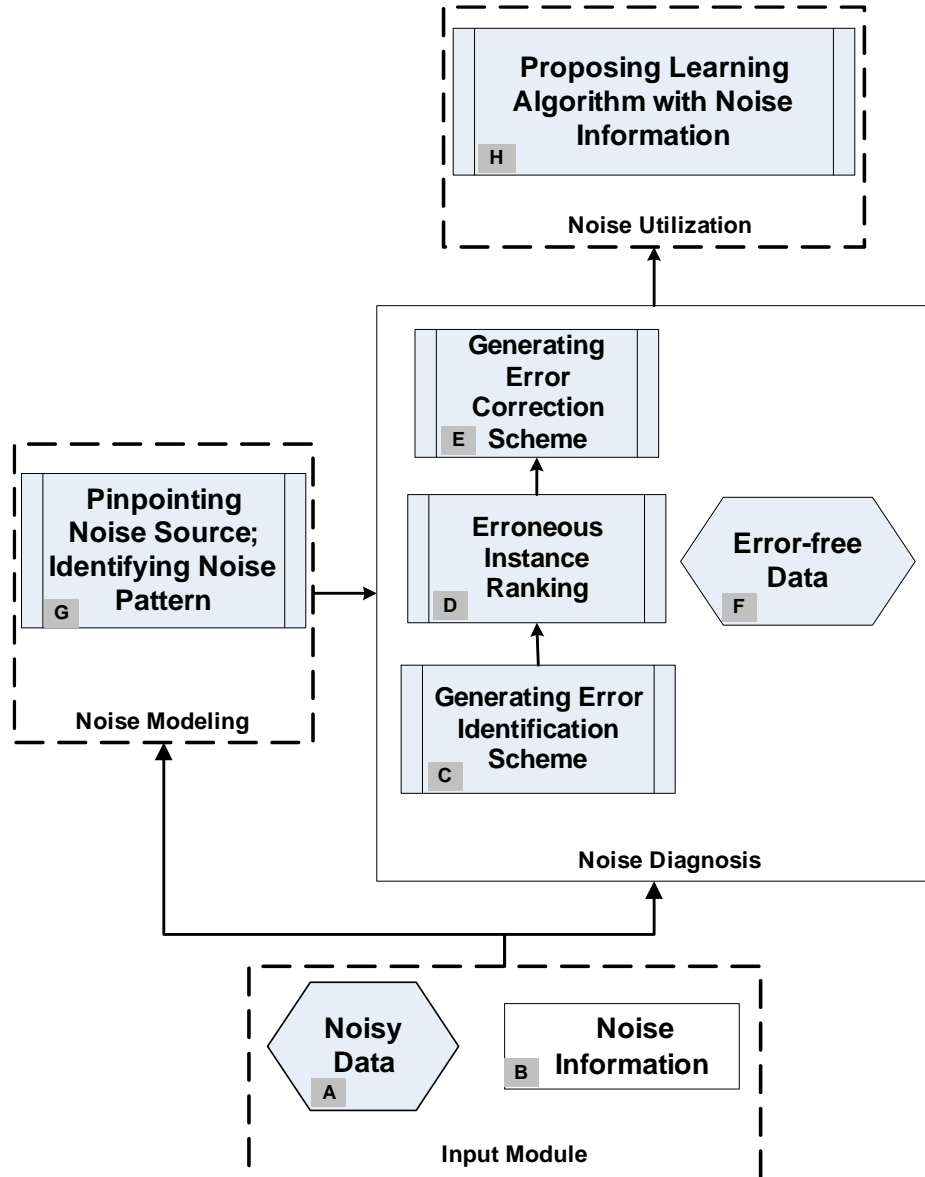


Figure 2.2: Framework for Noise Modeling, Diagnosis, and Utilization

nosing process, which includes error identification (C), erroneous instance ranking (D), and error correction scheme generation (E). The erroneous instance ranking is used to rank the potentially noisy instances, so that the instances having the greatest impact on the learning system will be handled at the first place. Assuming that we have ranked the potential noisy instances, and a subset of them are examined by some advanced techniques or additional

manual work, we will eventually obtain an error-free dataset, as denoted in Figure 2.2 by component F. Finally, a learning algorithm is proposed to take the noise information as part of it. In the problems where components B and F are unavailable, the framework depicts a traditional inductive learning method with noise detection and correction.

2.5 Chapter Summary

In this chapter, we have reviewed two categories of methods for learning from noisy data. The first category of methods aim at improving the data quality through noise handling methods. The second category of methods build robust learning models to tolerate a certain amount of noise. By discussing the limitations of existing methods, and the connections among the detections of noise, outliers, and frauds, we have pointed out the importance of noise modeling and reasoning. We have then proposed a noise MDU framework that bridges the gap between noisy data and the learning algorithm.

Based on this framework, we present two pieces of work in the remaining chapters of this thesis. We introduce a strategy of combining classifier ensembling and noise handling methods through our novel algorithms ACE and C2 in Chapter 3 and Chapter 4. In Chapter 3, we demonstrate the principles and methodologies of classifier ensembling methods. It provides theoretical background, motivation, and proper evaluation criteria for the methods ACE and C2 to be proposed in Chapter 4. In Chapter 5, we present our research on systematic noise modeling.

Chapter 3

Classifier Ensembling – Principles and Methods

In this chapter, we introduce the theoretical background of classifier ensembling and popular classifier ensembling methods. We present the principles of designing a classifier ensemble in Section 3.1, and review various strategies that create the diversity of classifier ensembles in Section 3.2. In Section 3.3, we review and analyze the bias-variance decomposition in order to understand the behavior of classifier ensembling.

3.1 Classifier Ensembling Principle

A classifier ensemble consists of a set of classifiers whose individual decisions are combined in some way, typically by weighted or unweighted voting, to classify new instances. We call the individual classifier in a classifier ensemble “base learner”. The main discovery is that a classifier ensemble is often much more accurate than the base learners that make it up. One of the most active areas of research in supervised learning has been to study methods for constructing good classifier ensembles. In order to make the classifier ensemble outperform any of its base learners, it is necessary and sufficient that the base learners are accurate and diverse. A most intuitive way to understand it is to think of the saying “three

minds are better than one”. In many situations when a decision needs to be made, it is typical that more than one expert’s opinions are collected for the decision making. When we acknowledge that the multiple-expert system is better a single expert system, certain conditions have been assumed anonymously. We clarify these assumptions as follows.

1. Any single expert can hardly always make correct decisions.
2. The panel of experts do not always have the same opinions.
3. The opinions of the experts are summarized in a reasonable way, which leads to the final decision.
4. If unweighted voting is applied to summarize the experts’ opinions, it is generally hoped that the majority opinion is correct.

On the one hand, each single expert is expected to make as few mistakes as possible, which corresponds to the base learner’s accuracy in a classifier ensemble. On the other hand, when unweighted voting is applied, the correct decision votes are expected to represent the majority opinions. In other words, when one makes a mistake, it is expected to be remedied by more than one other expert. As a result, the concept “diversity” is raised to describe such relationship among the base learners of a classifier ensemble.

It is generally accepted that an accurate base learner is a classifier that has an error rate of better than random guessing on new instances. In other words, the prediction accuracy of an accurate base learner should be at least 50% (Dietterich 2000a). However, It is apparently less obvious whether the base learners in a classifier ensemble are diverse or not, and how to measure the diversity among them.

Many researchers have proposed different ways to define “diversity” among base learners. For example, in (Dietterich 2000a) two classifiers are considered diverse if they make different errors on new data points; and in (Melville and Mooney 2003), the measure of disagreement is referred as the diversity of the ensemble. In (Kuncheva and Whitaker 2003),

Kuncheva and Whitaker carried out a study on various measures of diversity in classifier ensembles.

In the remaining of this section, we will analyze important roles of both the accuracy and diversity of base learners in a classifier ensemble. Based on the definition of diversity in (Dietterich 2000a), we assume the prediction error of base learners on some instances follows the binomial distribution. We quantify the definition of diversity as follows.

Definition 3.1 [Diversity] Diversity of a classifier ensemble, denoted as p_d , is defined as the probability of a new instance to be independently predicted by the base learners.

According to this definition, $p_d \in [0, 1]$, where $p_d = 1$ being the optimal situation. If assuming that r out of Δ test instances can be independently predicted, an estimate of p_d is $\hat{p}_d = r/\Delta$. We denote \hat{p}_d as Div in the succeeding analysis. We also denote err as the error rate of an individual base learner, and n as the number of base learners – the ensemble size of a classifier ensemble φ_A .

We will refer to, in the following subsections, the theoretical analysis between the accuracy and diversity to demonstrate why both aspects are important in the design of classifier ensembles. For the sake of simplicity and easy understanding, we assume a classifier ensemble with majority voting and equal weights for all base learners.

3.1.1 Accuracy

In this section, we analyze the importance of the prediction accuracy of base learners. We make the assumption that each base learner has independent prediction errors on every new instance. In other words, $Div = 1$. For any test instance, φ_A will produce incorrect predictions if and only if more than half of the base learners produce incorrect predictions. If we denote the random variable X as the number of base learners that make incorrect predictions, the probability for an instance to be incorrectly predicted by the classifier

ensemble φ_A can be calculated as shown in Eq. (3.1).

$$p_1 = P(X \geq \lceil n/2 \rceil) = \sum_{\kappa=\lceil n/2 \rceil}^n C_n^\kappa err^\kappa \cdot (1 - err)^{n-\kappa}. \quad (3.1)$$

So the probability for an instance to be correctly predicted by φ_A is

$$p_2 = 1 - p_1 \quad (3.2)$$

If we denote the random variable Y as the number of instances out of Δ test instances being correctly predicted by φ_A , the probability of $Y = k$ for $k = 0 \cdots \Delta$ can be calculated as shown in Eq. (3.3).

$$P(Y = k) = C_\Delta^k p_2^k (1 - p_2)^{\Delta-k} \quad (3.3)$$

When err increases from 0 to 1, the prediction accuracy of φ_A will decrease from 1 to 0. Figure 3.1 (a) shows this trend with the ensemble size n varying from 1, 10, 50 to 100. When $n = 1$, the classifier ensemble degenerates to a single learner. When $err < 0.5$ (the general assumption of the classifier ensembling), the accuracy of φ_A increases along with the increase of the ensemble size n . In reality, $Div = 1$ is hardly the case. However, we can still imagine that we can get a set of curves preserving the same relative locations and trends as shown in Figure 3.1 (a). Therefore, it is easy to conclude that **for any given Div value, the prediction accuracy of a classifier ensemble φ_A is proportional to the overall accuracies of its base learners. Increasing base learner accuracies will eventually enhance the underlying classifier ensemble.**

3.1.2 Diversity

Having analyzed the accuracy of base learners, we discuss the importance of the diversity of a classifier ensemble. We assume that r out of Δ test instances can be independently predicted, where $r \in [0, \Delta]$. As $Div = r/\Delta$, we have $Div \in [0, 1]$. We also assume that for the remaining $\Delta - r$ instances, the base learners always agree with each other when they make predictions. As a result, the probability of correctly predicting k out of Δ instances

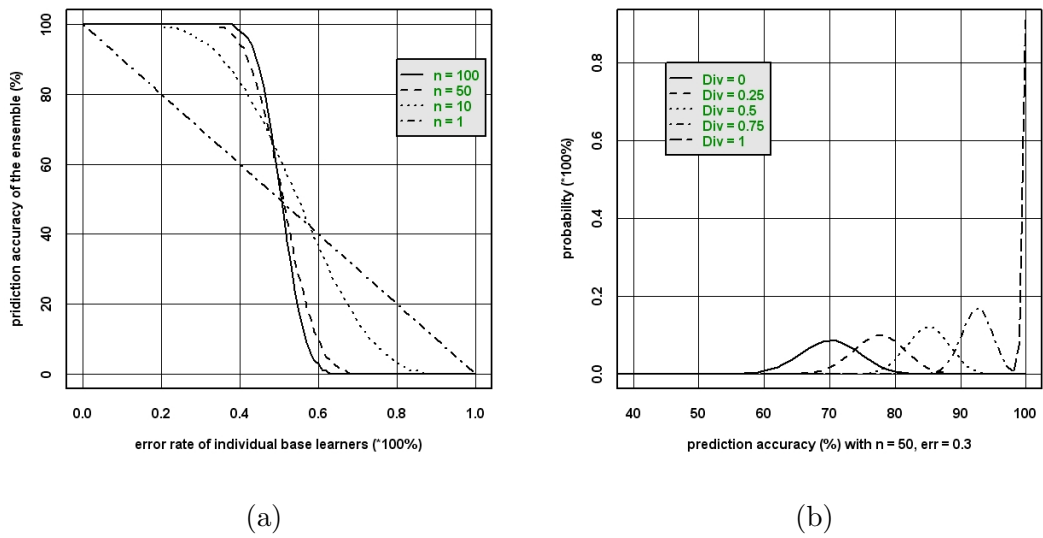


Figure 3.1: Accuracy and Diversity Analysis of Classifier Ensembles

(a): Simulation of a Classifier Ensemble with Different Ensemble sizes. The base learners in the ensemble are assumed to have independent predictions on all instances ($Div = 1$).

(b): Simulation of a Classifier Ensemble with Different Diversities. It is assumed that the base learners in the ensemble have independent predictions on r out of Δ instances, where $Div = r/\Delta$; and the base learners always agree with each other when making predictions on the remaining $\Delta - r$ instances.

by a classifier ensemble φ_A can be calculated as shown in Eq. (3.4).

$$P(Y = k) = \sum_{x=u}^v C_r^x p_2^x \cdot (1 - p_2)^{r-x} \cdot \tau_{\Delta-r}^{k-x}, \quad (3.4)$$

where $u = \max(k - (\Delta - r), 0)$, $v = \min(k, r)$ and

$$\tau_{\Delta-r}^{k-x} = C_{\Delta-r}^{k-x} (1 - err)^{k-x} \cdot err^{(\Delta-r)-(k-x)}.$$

Eq. (3.4) in fact is a convolution of two binomial distributions $Y_1 \sim \text{bin}(n_1, p_2)$ and $Y_2 \sim \text{bin}(n_2, (1 - err))$, where $Y_1 + Y_2 = Y$, $n_1 + n_2 = \Delta$. p_2 and $(1 - err)$ are the probabilities to correctly predict an instance when $Div = 1$ and $Div = 0$, respectively. Thus, Eq. (3.4) can be simplified as

$$P(Y = k) = C_{\Delta}^k p^k (1 - p)^{\Delta-k}, \quad (3.5)$$

where

$$p = Div * p_2 + (1 - Div) * (1 - err) \quad (3.6)$$

If $Div = 1$, Eq.(3.5) becomes Eq.(3.3). If $Div = 0$ the ensemble functions like a single learner, hence Eq.(3.5) becomes

$$P(Y = k) = C_{\Delta}^k (1 - err)^k \cdot err^{\Delta-k} \quad (3.7)$$

Based on Eqs.(3.1), (3.2), (3.5) and (3.6) with parameter settings $err = 0.3$ and $n = 50$, Figure 3.1 (b) shows the probability distribution of φ_A with Div varying from 0, 0.25, 0.5, 0.75 to 1. Notice that the probability distribution of $Div = 0$ is centered around 70%, which validates the situation that all base learners have the same prediction for all instances. As we can see, for a small proportion, say 25%, of the test instances being predicted independently, the accuracy can normally increase 7-9 percentage compared to the results from a single base learner. It is clear that **the prediction accuracy of the classifier ensemble will significantly increase along with the increase of its base learners' diversity.**

3.1.3 Accuracy & Diversity

The above analysis attests that both accuracy and diversity are important for a classifier ensemble to make effective predictions. Of these two factors, the accuracy of base classifiers plays an essential role, since $err < 0.5$ is a necessity for classifier ensembling to function well. Under this necessity, the diversity among base learners will crucially determine the improvements on the prediction accuracy compared to any individual base learner. Some people argue that the diversity is the major issue to construct an effective ensemble, while the accuracy of base learners does not need to be considered too much as long as $err < 0.5$. This statement makes sense, since we will definitely consider the diversity of a classifier ensemble if only one of them could be taken care of due to the limited resources. However, if we could improve the accuracy of the base learners in the ensemble as well, the classifier ensemble could be more effective than considering the diversity factor alone. This fact has been illustrated in Figure 3.1 (a), where $Div = 1$, and the prediction accuracy increases along with the accuracy of individual base learners.

Existing efforts in enhancing the diversity usually sacrifice the base learner accuracies through randomization procedures. Taking Bagging as an example, the learners built from the bootstrap sample sets are usually less accurate than the one built from the original data set D (Breiman 1996). Therefore, if we can increase the diversity among base learners and maintain their accuracies in the meantime, or vice versa, it is very likely that this framework will produce promising results.

3.2 Diverse Base Learners

Constructing diverse base learners is one of the keys to success in multiple classifier systems. In Section 3.2.1, we will summarize several categories of methods that attempt to enhance diversity among base learners. In Section 3.2.2 through Section 3.2.4, we will review three popular methods Bagging, Boosting, and DECORATE in detail.

3.2.1 Diversity Enhancement Methodologies

While researchers have developed measures of diversity from existing research effort (Kuncheva and Whitaker 2003), a complete grounded framework for enhance the diversity of a classifier ensemble is not available yet. In this section, we introduce several empirically applied strategies that attempt to create diverse base learners.

Manipulating Training Data

One common strategy to construct a classifier ensemble is to build base learners on different copies of training datasets, each of which is supposed to be sampled from the underling population. Among various sampling methods, random sampling from the original training dataset is the most straightforward way to increase the diversity among base learners. The random sampling based methods rely on resampling techniques to obtain different training sets for each of the classifiers. Additional mechanisms, such as weighted sampling and producing artificial instances, are proposed to help further increase the diversity among base training datasets. In Section 3.2.2 through Section 3.2.3, we will review three representative methods, Bagging (Breiman 1996), Boosting (Freund and Schapire 1996), and DECORATE (Melville and Mooney 2003), which are closely relevant to the successive chapters of this thesis.

Manipulating Feature Attributes

The second direction for generating multiple classifiers is to manipulate the set of input features available to the learning algorithm. A dataset is usually described with a set of attributes, while different attribute subsets might provide different views on the data. Therefore, base learners trained from different attribute subsets might be quite diverse. By varying the feature subsets used to generate the base learners, it is possible to promote diversity and produce base learners that tend to produce independent errors in different subareas of the instance space. There is a branch of study that explores the problem of en-

semble feature selection (Opitz 1999; Zenobi and Cunningham 2001; Tsymbal, Pechenizkiy, and Cunningham 2005a; Tsymbal, Pechenizkiy, and Cunningham 2005b). Among these methods, there is a category of random subsampling based algorithms that work through randomly choosing a subset of attributes to train a component learner. Ho (Ho 1998) has shown that simple random selection of feature subsets may be an effective technique for ensemble feature selection. Opitz (Opitz 1999) designed a genetic algorithm based ensemble feature selection strategy, which uses random spacing as well. Other feature attribute selection methods include genetic search, hill-climbing, ensemble forward, and backward sequential selection (Opitz 1999; Cunningham and Carney 2000; Aha and Bankert 1995).

Manipulating Class Attribute

When the first two categories of data manipulations have gained great popularity, some researchers also explored the possibility of creating the diversity by manipulating the class attribute of the training data. Breiman (Breiman 2000) proposed an output perturbation method that introduces random noise into the output in the training data. The “output” indicates either the class attribute or the dependent variable in the context of classification and regression analysis, respectively. In classification analysis, each base learner in the ensemble is generated using the original training set with randomly flipped class labels. The class label of each instance is switched according to a probability that depends on an overall switching rate, so that the class distribution of the original training set remains the same. By applying output flipping, it was shown significant improvements over Bagging on 15 natural and artificial datasets with 100 base learners in the ensemble. The statistical interpretation to output flipping is that it can produce a significant error reduction, provided that large numbers of units and high class switching rates are used. The base learners generated by this procedure have statistically uncorrelated errors in the training set. The other variant of output flipping is proposed by Martinez-Munoz and Suarez (Martinez-Munoz and Suarez 2005).

Manipulating Model Parameters

A fourth general purpose method for generating ensembles of classifiers is to effectively adjust the model parameters of the base learning models. One representative example is neural network ensembles. If the algorithm is applied to the same training examples but with different initial weights, the resulting learners can be quite different. It is because the error surface of neural network learning is full of local minima, and the learning models with different initial weights are usually trapped in different local minima. These local minima reflect partly the fitting to the regularities of the data and partly the fitting to the noise in the data. Ensemble averaging tends to cancel the noise part as it varies among the base learners, and tends to retain the fitting to the regularities of the data.

Starting each network with differing random initial weights has been shown important when using a deterministic training method such as back propagation (Kolen and Pollack 1991). Maclin and Shavlik presented an approach to initializing neural network weights that uses competitive learning to create networks that are initialized far from the origin of weight space, thereby potentially increasing the set of reachable local minima; they showed significantly improved performance over the random initialization on two real world datasets (Maclin and Shavlik 1995).

Besides neural networks, decision trees are the other commonly used learning model, the parameters of which are adjusted in a classifier ensemble. The family of random forest algorithms, which we will talk about in the next paragraph, is a good example.

Random Forest

In machine learning, a random forest is a classifier ensemble that consists of a collection of tree-structured classifiers, such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest (Breiman 2001). The concept of random forest came from the term “random decision forests” that was first proposed by Ho in 1995 (Ho 1998). A typical random forest algorithm combines

Breiman’s “bagging” idea and Ho’s “random subspace method” to construct a set of decision trees with controlled variations. Although a random forest is not parallel to the previous three types of classifier ensembling methods, we emphasize it because its efficacy and popularity.

In (Dietterich 2000b) Dietterich proposed and explored a random forest, each base learner of which is a decision tree with random internal node splitting. Random split selection is a modified version of the C4.5 decision tree in which the decision about which split to introduce at each internal node of the tree is randomized. During the decision tree construction, one feature attribute is to be selected as a tree node to split in each iteration. In C4.5 decision trees, the feature attribute that has the highest information gain ratio with the class attribute will be selected. While in this random split decision tree, each split is uniformly randomly selected from the top x feature attributes that have the highest information gain ratio with the class attribute. The implementation in this paper computes the $x = 20$ best splits and then choose one from them randomly. The results show that boosting has the best performance, while randomizing and bagging give quite similar results in low noise situations. With added classification noise, however, Bagging is clearly the best method over the other two.

3.2.2 Bagging

Bagging is a short name for “Bootstrap Aggregating”, where bootstrap indicates the training data of each base learner is obtained by sampling with replacement from the original dataset. In other words, the training set is generated by randomly drawing, with replacement, N instances - where N is the size of the original training set; some of the original examples may be repeated in the resulting training set while others may be left out. Each individual classifier in the ensemble is generated with a different random sampling of the training set. Bagging predictors are a method for generating multiple versions of a predictor and using these to get an aggregated predictor. The aggregation averages over the versions

when predicting a numerical outcome and does a plurality vote when predicting a class. The multiple versions are formed by making bootstrap replicates of the learning set and using these as new learning sets. Tests on real and simulated data sets using classification and regression trees and subset selection in linear regression show that bagging can give substantial gains in accuracy. The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy. The evidence, both experimental and theoretical, is that bagging can push a good but unstable procedure a significant step towards optimality. On the other hand, it can slightly degrade the performance of stable procedures.

3.2.3 Boosting

Bagging resamples the dataset randomly with a uniform probability distribution; an alternative would be to have a non-uniform distribution, as we have in Boosting (Freund and Schapire 1996). Boosting encompasses a family of methods (Freund and Schapire 1996; Freund and Schapire 1999; Fan, Stolfo, Zhang, and Chan 1999; Joshi, Kumar, and Agarwal 2001), the focus of which is to produce a series of base learners, where the training set used for each member of the series is chosen based on the performance of the earlier constructed base learners in the series. Instead of drawing a succession of independent bootstrap samples from the original training data, boosting maintains a weight for each instance in the training data - the higher the weight, the more the instance influences the base learner trained.

The version of boosting investigated in this dissertation is AdaBoost.M1 (Freund and Schapire 1996). Initially, the weights are set equally for all instances in the training data. A set of data is sampled from the training data where each instance is given the same weight. This sampling procedure is performed the same way as that in Bagging ensembling. But at each successive round, the vector of weights is adjusted to reflect the performance of the corresponding classifier ensemble, so the weight of misclassified instances is increased.

The purpose of the re-weighted sampling attempts to produce new base learners that are better able to predict instances for which the current ensemble’s performance is poor. In other words, instances that are incorrectly predicted by previous base learners in the series are chosen more often than instances that were correctly predicted, so that the classifier ensemble is forced to focus on the hard instances in the training set. The final classifier ensemble also aggregates the base learners by majority voting as Bagging ensembling does, but each base learner’s vote is a function of its accuracy.

3.2.4 DECORATE

DECORATE is a short name for Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples. It is an ensembling method that constructs diverse base learners using additional artificially-constructed data. It has been shown to generally outperform both boosting and bagging when training data is limited (Melville and Mooney 2003). In DECORATE an ensemble is generated iteratively, learning a base learner at each iteration and adding it to the current ensemble. Initially, the ensemble contains each base learner trained on the given training data. The base learner T_i in a successive iteration is trained on the original training data together with some artificial data. T_i is then added into the current ensemble, which is evaluated on the original training data. If the prediction error rate on the training data is greater than that from the ensemble in the previous iteration (without T_i included), T_i will be removed from current ensemble. Otherwise, T_i is kept. This process is iterated for a predefined number of times. The eventually obtained classifier ensemble will be used for future classification.

The procedure to generate the artificial data is described as follows. Firstly, a set of unlabelled artificial data with size N_a is randomly generated based on the distribution of the original training data. In constructing artificial data points, it is assumed that the feature attributes are independent with each other. For a numeric feature attribute, the mean and standard deviation from the training set is computed, and then new values are

randomly picked from the Gaussian distribution with previous computed mean and standard deviation. For a nominal feature attribute, the occurrence of each distinct value in its domain is computed, and new values are generated based on this distribution. The Laplace smoothing is used so that nominal attribute values not represented in the training set still have a non-zero probability of occurrence. Secondly, the artificially generated instances are labelled based on the prediction of the current ensemble. Given an instance, the probability of class membership is predicted by the current classifier ensemble. The instance is then labelled in a way that the probability of the selected label is inversely proportional to the current ensemble’s prediction.

From the above description of the algorithm, the training data for each base learner consists of two parts: the original training data and the artificial data. The artificial data is called “diversity data” in this method, because it serves the purpose of enhance the diversity of the base learners. These artificially constructed instances are given category labels that disagree with the current decision of the ensemble, therefore the authors believe that it easily and directly increases the diversity when a new base learner is trained on the augmented data and added to the classifier ensemble.

3.3 Bias-Variance Decomposition for Classifier Ensembles

Bias-variance analysis provides a tool to study learning algorithms (Geman, Bienenstock, and Doursat 1992; Friedman 1997; Putten and Someren 2004). In (Domingos 2000), Domingos proposed and proved a bias-variance decomposition approach, particularly for classifier ensembles. It introduces a way to interpret the correlation between the learning ability of base learners and the combined ensemble. In this section, we provide the theoretical background of the bias-variance decomposition, which quantitatively shows the expected loss of a prediction could be decomposed to three components – bias, variance and noise, for classifier ensemble learning. Of these three components, *bias* corresponds to the predictive error rate of a classifier ensemble, and *variance* can be interpreted as a metric that partly

reflects the diversity of the classifier ensemble. The definitions of bias, variance, noise and related concepts in the following analysis are borrowed from (Domingos 2000).

Given a training set $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, where x_i and y_i represent the feature attributes vector and the class attribute of an instance respectively, a learner produces a model $\varphi(D)$. Given a test instance x , this model produces a prediction for the class attribute value $y = \varphi(x, D)$. Let random variable C represent the class attribute for x , and c be the true value of C . A loss function $L(c, y)$ measures the cost of predicting y when the true value is $C = c$. The goal of supervised learning can be stated as producing a model with the smallest possible loss; i.e., a model that minimizes the average $L(c, y)$ over all instances, with each instance weighted by its probability. Commonly used loss functions are squared loss ($L(c, y) = (c - y)^2$), absolute loss ($L(c, y) = |c - y|$), and zero-one loss ($L(c, y) = 0$ if $y = c$, $L(c, y) = 1$ otherwise).

Before we get into further details, we first review several concepts from (Domingos 2000).

Definition 3.2 [Optimal Prediction] The optimal prediction y_* for an instance x is the prediction that minimizes $E_C[L(c, y_*)]$.

In general, C will be a nondeterministic function of x . It indicates that if x is sampled repeatedly, different values of C may be observed. A simple situation, as shown in Example 1, is that a particular instance x falls into different class memberships with a certain probability.

Example 1 Given a test instance x , and its class attribute $C \in \{1, 2, 3\}$, the probability of C taking 1, 2, or 3 is:

$$p(C) = \begin{cases} 0.7 & \text{if } C = 1 \\ 0.2 & \text{if } C = 2 \\ 0.1 & \text{if } C = 3 \end{cases}$$

□

In order to minimize the expected loss $E_C[L(c, y)]$ for a given instance x , we are interested in finding an optimal prediction y_* over all possible y values, so that $E_C[L(c, y_*)]$

is minimized, where the subscript C denotes that the expectation is taken with respect to all possible values of C , weighted by their probabilities given x . The optimal prediction y_* in Example 1, in terms of zero-one loss, is $y_* = 1$, which incurs the expected loss $E_C[L(c, y_*)] = 0.3$.

Definition 3.3 [Main Prediction] The main prediction for a loss function L and a set of base learners is $y_m^{L, \Delta} = \operatorname{argmin}_{y'} E_{\Delta}[L(y, y')]$.

For the purpose of simplicity we use y_m to represent $y_m^{L, \Delta}$, when there is no danger of ambiguity. The expectation $E_{\Delta}[L(y, y')]$ is taken with respect to the predictions $C = y$ produced by base learners in Δ . To make this concept more clear, we will introduce an example here.

Example 2 Supposing the number of base learners $n = 10$, and the class attribute $C \in \{1, 2, 3\}$. Let $I = \{1, 1, 1, 1, 2, 2, 2, 3, 3, 3\}$ be the multiset of the predictions from these 10 base learners. Thus, a specific prediction $C = y$ will appear more than once in I if it is produced by more than one base learner. If N_j is denoted as the count of value $j \in \{1, 2, 3\}$ in I , the probability of y taking 1 is $P(C = 1) = \frac{4}{10}$, since $N_1 = 4$. Similarly, $P(C = 2) = \frac{3}{10}$, and $P(C = 3) = \frac{3}{10}$. The main prediction y_m is calculated as follows.

$$\begin{aligned} E_{\Delta}[L(y, y')] &= \sum_y L(y, y') * P(C = y) \\ &= L(y = 1, y') * P(C = 1) + L(y = 2, y') * P(C = 2) + L(y = 3, y') * P(C = 3) \\ &= L(y = 1, y') * \frac{4}{10} + L(y = 2, y') * \frac{3}{10} + L(y = 3, y') * \frac{3}{10} \end{aligned}$$

When the zero-one loss is applied, it can be calculated that

$$E_{\Delta}[L(y, y')] = \begin{cases} 0.6 & \text{if } y' = 1 \\ 0.7 & \text{if } y' = 2 \\ 0.7 & \text{if } y' = 3 \end{cases}$$

Since $y' = 1$ makes $E_{\Delta}[L(y, y')]$ minimum, the main prediction $y_m = 1$.

□

Therefore, the main prediction y_m is the value y' whose average loss relative to all the predictions in I is minimum. In other words, the main prediction y_m is the one that “differs least” from all the other predictions in I according to L .

Definition 3.4 [Bias] The bias of a learner on an instance x is $B(x) = L(y_*, y_m)$.

This definition indicates that the bias is the loss incurred by the main prediction relative to the optimal prediction. In a classifier ensemble where the majority voting is applied, the main prediction y_m could be derived as shown in Example 2.

Definition 3.5 [Variance] The variance of base learners of an classifier ensemble on an instance x is $V(x) = E_\Delta[L(y_m, y)]$.

This definition indicates that the variance is the average loss incurred by predictions relative to the main prediction. Going back to the previous example, if the ensemble size $n = 10$, and for a test instance x the multiset of predictions $I = \{1, 1, 1, 1, 2, 2, 2, 3, 3, 3\}$, then $N_1 = 4$, $N_2 = 3$, and $N_3 = 3$. The main prediction, $y_m = \operatorname{argmax}_j\{N_1, N_2, N_3\} = 1$. The variance of the base learners on x is then calculated as follows:

$$\begin{aligned}
V(x) &= E_\Delta[L(y_m, y)] \\
&= \sum_y L(y_m, y) * p(y) \\
&= L(y_m, y = 1) * p(y = 1) + L(y_m, y = 2) * p(y = 2) + L(y_m, y = 3) * p(y = 3) \\
&= L(y_m, y = 1) * \frac{4}{10} + L(y_m, y = 2) * \frac{3}{10} + L(y_m, y = 3) * \frac{3}{10}
\end{aligned} \tag{3.8}$$

Substituting $y_m = 1$ in Equation 3.8, we get

$$V(x) = E_\Delta[L(y_m, y)] = 0 * 0.4 + 1 * 0.3 + 1 * 0.3 = 0.6.$$

Definition 3.6 [Noise¹] The noise of an instance x is $N(x) = E_C[L(c, y_*)]$.

¹This definition for noise is only valid in the context of bias-variance decomposition.

This definition indicates that the noise reflects the characteristics of a particular dataset, and it is the unavoidable component of the loss, incurred independently of the learning algorithm.

Having introduced the definitions of bias, variance, noise, and related concepts, we will show that the expected loss $E_{\Delta,C}[L(c, y)]$ for a zero-one loss function L can be decomposed into these three components $B(x)$, $V(x)$, and $N(x)$ as defined above. We quote two important claims from (Domingos 2000) as follows. The proof procedures for these claims are skipped, and interested readers may refer to the original article.

Claim 1: In two-class problems, the expected loss

$$E_{\Delta,C}[L(c, y)] = k_1 E_C[L(c, y_*)] + L(y_*, y_m) + k_2 E_{\Delta}[L(y_m, y)]$$

is valid for any real-valued loss function for which $\forall_y L(y, y) = 0$ and $\forall_{y_1 \neq y_2} L(y_1, y_2) \neq 0$, with $k_1 = P(y = y_*) - \frac{L(y_*, y)}{L(y, y_*)} P(y \neq y_*)$ and $k_2 = 1$ if $y_m = y_*$, $k_2 = -\frac{L(y_*, y_m)}{L(y_m, y_*)}$ otherwise.

Claim 2: In multi-class problems, the expected loss

$$E_{\Delta,C}[L(c, y)] = k_1 E_C[L(c, y_*)] + L(y_*, y_m) + k_2 E_{\Delta}[L(y_m, y)]$$

is valid for zero-one loss, with $k_1 = P(y = y_*) - P(y \neq y_*)P(y = c | y_* \neq c)$ and $k_2 = 1$ if $y_m = y_*$, $k_2 = -P(y = y_* | y = y_m)$ otherwise.

In this section, we have shown how Domingos's method decomposes the expected loss $E_{\Delta,C}[L(c, y)]$ into three components: bias, variance, and noise for a classifier ensemble. Each of the three components contributes a share to the expected loss. Apparently, a smaller expected loss indicates a better learning model, so that we hope the value of the three terms to be as small as possible. In particular, the average bias $E_X[B(x)]$ of a classifier ensemble on a set of test instances corresponds to the ensemble's prediction error rate, which is a major measure to evaluate the accuracy of a classification method. The average variance $E_X[k_2 V(x)]$ indicates the average prediction fluctuation of the base learners. Low

variance ($k_2 > 0$) for a correct prediction and high variance ($k_2 < 0$) for an incorrect prediction is desired to achieve low expected variance $E_X[k_2V(x)]$. This measure reflects the stability of a classifier ensemble. The average noise $E_X[k_1N(x)]$ measures the average loss incurred by the difference between the true value c and the optimal value y_* of the class attribute C on a set of test instances. $N(x)$ is an intrinsic feature of the dataset, and it is independent on which learning algorithm is used for prediction. Since $k_1 > 0$ or $k_1 < 0$ are both possible, $E_X[k_1N(x)]$ could be either positive or negative as well. As a result, it is valid that $E_X[B(x)] + E_X[k_2V(x)] > 0$ or < 0 . Similarly, $E_X[k_2V(x)]$ could also take positive/negative values. Only $E_X[B(x)]$ must take non-negative values.

3.4 Chapter Summary

In Section 3.1, we have discussed the importance of the accuracy and diversity of base learners in a classifier ensemble. We point out that it is important to consider both factors when designing a classifier ensemble. In other words, improving either factor will gain performance. It is especially important for learning from noisy data. This analysis lays the theoretical background for the motivation of our design for classifier ensemble methods ACE and C2, which will be introduced in Chapter 4.

In Section 3.2, we have discussed several empirical strategies that are applied to enhance the diversity of a classifier ensemble. We have introduced three popular classifier ensembling methods – Bagging, Boosting, and DECORATE in detail. The performance of these ensembling methods will be compared and evaluated with ACE and C2 in our experimental results (Section 4.4.6).

By analyzing the bias-variance decomposition for a classifier ensemble in Section 3.3, we aim at pointing out two issues. First, this analysis helps to explore the prediction ability of base learners, instead of treating the inside of a classifier ensemble as a “black box”. Second, it builds up a connection to interpret the relationship between the base learners and their ensemble (combined decision). Third, it provides a clear view that both the

bias ($E_X[B(x)]$) and variance ($E_X[V(x)]$), two decomposed components that relate to the learning algorithm, contribute to the expected loss when making a prediction through a classifier ensemble. In other words, both measures are important to evaluate the learning model. Guided by these facts, we will report and analyze the bias-variance decomposition results in our experiments (Section 4.4.6).

Chapter 4

Corrective Classifier Ensembling

In Chapter 2 and Chapter 3, we have reviewed several categories of methods for noise handling in classification tasks. We noticed that the noise detection and correction techniques, if appropriately applied, would effectively improve the data quality, so as to benefit the succeeding data mining tasks. On the other hand, classifier ensembling is effective and robust, as the decisions made from committees are often superior to the ones from any single learner. Nevertheless, existing classifier ensembles directly work on the original input data sources, and usually put no efforts on enhancing the underlying data quality, which eventually reduces their performances. Motivated from these facts, we propose to combine these two techniques for learning from noisy data sources. The theoretical background for this classifier ensemble design has been discussed in Chapter 3.

This chapter is organized as follows. We formalize the problem in Section 4.1, and present our basic assumptions and definitions in Section 4.2. We provide our detailed algorithms ACE and C2 in Section 4.3, and illustrate our experimental results in Section 4.4. In the end, we conclude this chapter with some discussions in Section 4.5. A portion of the materials from this chapter and chapter 3 have been published in the Proceedings of the 2005 International Conference on Tools of Artificial Intelligence, with the title “ACE: An Aggressive Classifier Ensemble with Error Detection, Correction and Cleansing” (Zhang,

Zhu, Wu, and Bond 2005); and the Proceedings of the 2006 IEEE International Conference on Data Mining, with the title “Corrective Classification: Classifier Ensembling with Corrective and Diverse Base Learners” (Zhang, Zhu, and Wu 2006).

4.1 Problem Statement

The problem of classification is a two-step process. The first step is to build a learning model $\varphi(D)$ on the basis of a set of previously collected training instances in dataset $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, where x_i and y_i represent the feature attributes vector $x_i = \langle a_{i1}, \dots, a_{im} \rangle$ and the class attribute $C = y_i$ of an instance respectively. If we use the random variable pair (X, C) to denote any training instance, (x, y) represents a particular observed training instance, where $X = \langle A_1, \dots, A_m \rangle$, $x = \langle a_1, \dots, a_m \rangle$, and $(X = x, C = y)$ is a shorthand for $((A_1 = a_1 \wedge A_2 = a_2 \wedge \dots \wedge A_m = a_m), C = y)$. We assume the training dataset D contains n instances, each of which is composed of m feature attributes and one class label. The second step of a classification problem is to assign a class label $C = y$ to a new instance defined by a feature attribute vector x . The classification task of supervised learning is to find a class label y for instance x , such that $P(C|x)$ is maximized, i.e. $y = \operatorname{argmax}_{y_i} P(C = y_i|x)$.

Under the circumstances of data imperfections, feature attribute errors in particular, the genuine feature values are occasionally contaminated and replaced with erroneous or abnormal values. To capture such errors or anomalies, we assume the existence of a hidden variable, $e \in E$, in the feature space, so that the whole state space becomes $C \times X \times E$. In reality, the hidden variable e might refer to categorical events that result in errors in features (such as the truthfulness of someone taking a survey or the erroneous values of given attributes). We can factor features $X = X_E \times X_{-E}$ based on a link between events and features (for example, truthfulness and questions that might be answered untruthfully). X_{-E} can be empty if the feature set contains no errors.

From the definition of conditional probability, we have

$$\begin{aligned}
P(y, x_E, x_{-E}) &= P(y|x_E, x_{-E})P(x_E|x_{-E})P(x_{-E}) \\
&= P(x_E|c, x_{-E})P(c|x_{-E})P(x_{-E}) .
\end{aligned} \tag{4.1}$$

It gives Eq. (4.2) if we rearrange Eq. (4.1).

$$P(y|x_E, x_{-E}) = \frac{P(x_E|y, x_{-E})P(y|x_{-E})}{P(x_E|x_{-E})} . \tag{4.2}$$

Note that for a specified x , $P(y|x_E, x_{-E}) \propto P(x_E|y, x_{-E})P(y|x_{-E}) \cdot P(x_E|y, x_{-E})$ explicitly introduces the idea of using a classifier obtained after switching the role of class label with that of certain random feature variables. If we make the hidden variable explicit we obtain Eq. (4.3).

$$\begin{aligned}
P(x_E|y, x_{-E}) &= \sum_e P(x_E, e|y, x_{-E}) \\
&= \sum_e P(x_E|e, y, x_{-E})P(e|y, x_{-E})
\end{aligned} \tag{4.3}$$

Thus Equation (4.2) can be rewritten as in Equation (4.4):

$$P(y|x_E, x_{-E}) \propto P(y|x_{-E}) \sum_e P(e|y, x_{-E})P(x_E|e, y, x_{-E}) , \tag{4.4}$$

where

- $P(y|x_{-E})$ is a classifier obtained by neglecting X_E ($P(y)$ when X_{-E} is empty);
- $P(e|y, x_{-E})$ is the prior (with respect to x_E) probability of event e , which will serve to limit the number of errors inferred;
- $P(x_E|y, e, x_{-E})$ is a classifier, obtained after switching the roles of class c and the feature variables x_E , that can explicitly account for erroneous features.

Eq. (4.4) indicates that in order to enhance the classifier learned from the imperfect information sources $P(C|x)$, we can switch the roles of class and feature variables to build special classifiers, $P(x_E|y, e, x_{-E})$. Because $P(y|x)$ and $P(x_E|y, e, x_{-E})$ are proportional, the efforts, which enhance the classifiers $P(x_E|y, e, x_{-E})$, are equivalent to the enhancement

on the learner $P(y|x)$.¹ In Sections 4.1.1 through 4.1.4, we will introduce our four-step feature attribute error detection and correction scheme.

4.1.1 Training Attribute Predictor (AP)

In this section, we introduce how to construct training attribute predictors (APs) by adopting the idea of switching the roles of the class attribute and the individual feature attribute. These attribute predictors will be applied to detect possible erroneous feature attribute values in the followed steps.

The objective of this step is to train a set of learners, called Attribute Predictors (AP), each of which is for one feature attribute in the input training dataset D . Attribute predictors are used for predicting the attribute values of a possible noise infected instance. The detailed procedure to construct attribute predictors are described in Table 4.1. We denote the random variable $F = (X, C)$, where the random feature attribute vector $X = \langle A_1, A_2, \dots, A_m \rangle$ and C denotes the class attribute. To simply the presentation, we denote $A_{m+1} = C$, such that $F = \langle A_1, A_2, \dots, A_{m+1} \rangle$, which contains m feature attributes and one class label. In dataset D the $(m+1)^{th}$ attribute always represents the class attribute. Let us define $D^{(i)}$ to be the same dataset as D , except that the i^{th} attribute is regarded as the class label, and $F_{-i} = \langle A_1, A_2, \dots, A_{i-1}, A_{i+1}, \dots, A_{m+1} \rangle$. So, $D^{(i)} = D$ when $i = (m+1)$. With all instances in D , AP_i regards A_i as the class label and uses the attribute set F_{-i} to train a classifier $\varphi(D^{(i)})$ for attribute A_i . Given an instance $F = f_k$ that is prone to contain feature attribute errors, $\varphi(f_k, D^{(i)})$ predicts the value of attribute A_i .

Assume each feature attribute $A_i \in F$, where $i = 1 \dots m$, has V_i possible values. Given an instance $F = f_k$ in S , an Attribute Prediction (AP) mechanism is applied to locate suspicious attributes in f_k .

¹This analysis is derived with great help from Dr. Jeffrey P. Bond, who is one of my co-advisors.

Table 4.1: Training Attribute Predictors (AP)

Procedure: Training-Attribute-PredictorsInput: dataset D Output: N attribute predictors, where N denotes the number of feature attributes of D

- 1 Let $N \leftarrow \#$ of feature attributes of D ;
 - 2 **For** $i \leftarrow 1$ to N
 - 3 Swap the role between the i^{th} attribute and the class attribute, i.e., set the i^{th} attribute as the class attribute and the original class attribute as one feature attribute;
 - 4 The resulting dataset is denoted as $D^{(i)}$;
 - 5 A learner AP_i is trained on $D^{(i)}$;
 - 6 **End For**
 - 7 return AP ;
-

4.1.2 Filtering Noisy Instances

In this step, the algorithm identifies a dataset $S \subset D$, where the instances in S are most likely to be noise infected. The process is achieved through a stratified ten-fold cross validation on the input dataset D . In each of the ten iterations, a learner is trained on nine-fold of the data. This learner then tests on the remaining one-fold data, where the instances that are incorrectly predicted are forwarded to suspicious instances set S . Procedure *Filtering-Suspicious-Instances* (Table 4.2) shows the pseudocode to generate S . Upon S being decided, the possible erroneous attribute values of instances in S will be detected and corrected in the followed procedures.

4.1.3 Constructing Solution List

In this step, the algorithm tries to figure out a list of possible solutions for the feature attribute values of every instance in S . A solution consists of one or more than one attribute-value pairs for an instance f_k in S . When f_k is modified according to the solution, it can be correctly predicted by the benchmark learner $\varphi(D)$. If an instance $F = f_k$ has more than

Table 4.2: Filtering Suspicious Instances

Procedure: Filtering-Suspicious-Instances

Input: dataset D

Output: a subset S , which aggregates the most suspicious instances in D

```
1  Let  $S \leftarrow \emptyset$ ;  
2  // Perform a ten-fold stratified cross validation as follows;  
3  For  $i \leftarrow 1$  to 10  
4      Denote the  $i^{th}$  fold of  $D$  as dataset  $D_i$ ;  
5      Denote the remaining nine-fold of  $D$  as dataset  $D_{-i}$ ;  
6      Train a learner  $\varphi(D_{-i})$  on  $D_{-i}$ ;  
7      For each instance  $f_k$  in  $D_i$ ;  
8          Let  $y_k \leftarrow$  the original class attribute of  $f_k$ ;  
9          If ( $y_k \neq \varphi(f_k, D_{-i})$ );  
10              $f_k$  is added into  $S$ ;  
11         End If  
12     End For  
13 End For  
14 return  $S$ ;
```

one solution, all the solutions form a solution list for f_k . The detailed procedure is shown in Table 4.3.

Assume f_k 's current value of A_i is a_i^k , and the predicted value from $AP_i(f_k)$ is $AP_i(f_k) = \varphi(f_k, D^{(i)}) = \tilde{a}_i^k$. If a_i^k is different from \tilde{a}_i^k , it implies that A_i of f_k may possibly contain an incorrect value. Then the benchmark classifier $T = \varphi(D)$ is used to justify whether the predicted value from AP_i makes more sense: If after changing a_i^k to \tilde{a}_i^k , f_k can be correctly classified by T , it indicates that the change results in a better classification. This attribute-value pair $\langle A_i, \tilde{a}_i^k \rangle$ will be added into instance f_k 's solution list EA_k . If the change still makes f_k incorrectly classified by T , other attribute-value pairs are to be checked to explore possible errors. If the prediction from each single attribute does not conclude any errors, the algorithm will check the combinations of multiple attribute values. The combination of up to z attribute values will be checked.

4.1.4 Selecting Solution for Error Correction

Suppose we have got the solution set EA in the way shown in Table 4.3, and each instance f_k in S , its solution set EA_k takes the form of $EA_k = \{\{\langle A_i, \tilde{a}_i^k \rangle\}, \{\langle A_j, \tilde{a}_j^k \rangle\}, \{\langle A_t, \tilde{a}_t^k \rangle\}\}$ or $EA_k = \{\{\langle A_i, \tilde{a}_i^k \rangle, \langle A_j, \tilde{a}_j^k \rangle\}, \{\langle A_j, \tilde{a}_j^k \rangle, \langle A_u, \tilde{a}_u^k \rangle\}\}$, where $\langle A_x, \tilde{a}_x^k \rangle$ represents one of the feature attributes in dataset S and the suggested correction value, that is, $A_x \in F_{-i}$, where $i = m + 1$. The first solution set consists of 3 solutions, which means that applying any one of the attribute-value pair changes $\langle A_i, \tilde{a}_i^k \rangle$, $\langle A_j, \tilde{a}_j^k \rangle$ or $\langle A_t, \tilde{a}_t^k \rangle$ on f_k will lead to a correction prediction of f_k by the benchmark classifier $T = \varphi(D)$. Since the correction of a single attribute can lead to the correct prediction, the algorithm will not continue to check other possibilities, e.g. changing multiple attributes simultaneously (see line (24) of Table 4.3). The second solution set consists of 2 solutions, each of which consists of attribute-value pairs. It indicates that the change of any single attribute of instance f_k cannot result in a correct prediction by classifier T , so a change combining two attribute values is tried. The above procedure repeats until either one

Table 4.3: Solution Set Construction**Procedure: Solution-Set-Construction**

Input: dataset D , dataset S that aggregates most suspicious instances in D ,
 z : the maximum number of simultaneous attribute value changes for each instance

Output: solution set EA for erroneous attributes

- 1 Let $EA \leftarrow \emptyset$; Let $n \leftarrow |S|$;
- 2 Benchmark learner $\varphi(D)$ is trained;
- 3 Let $AP \leftarrow \mathbf{Training-Attribute-Predictors}(D)$;
- 4 Let $m \leftarrow \#$ of feature attributes in D ;
- 5 **For** $k \leftarrow 1$ to n
- 6 f_k denotes the k^{th} instance in S ;
- 7 $EA_k \leftarrow \emptyset$; $AV^k \leftarrow \emptyset$; $CV^k \leftarrow \emptyset$
- 8 **For** $i \leftarrow 1$ to m
- 9 Let $a_i^k \leftarrow f_k$'s current value of the i^{th} attribute A_i ;
- 10 Let $\tilde{a}_i^k \leftarrow AP_i(f_k)$, with prediction confidence CV_i^k ;
- 11 **If** ($a_i^k \neq \tilde{a}_i^k$)
- 12 attribute-value pair $\langle A_i, \tilde{a}_i^k \rangle$ is added into AV^k ;
- 13 CV_i^k is added into CV^k ;
- 14 **End If**
- 15 **End For**
- 16 **For** $l \leftarrow 1$ to z
- 17 **For** each combination of l attribute-value pairs SOLUTION $\subset AV^k$
- 18 $f'_k \leftarrow f_k$ after applying the l attribute-value changes;
- 19 **If** f'_k could be correctly classified by $\varphi(D)$
- 20 SOLUTION is added into EA_k ;
- 21 **End If**
- 22 **End For**
- 23 $EA \leftarrow EA \cup EA_k$
- 24 **If** ($EA \neq \emptyset$) break;
- 25 **End For**
- 26 **End For**
- 27 return EA

attribute or up to z attributes will make the changed instance f_k be correctly predicted by T . In addition, for each solution acquired, we can calculate its prediction confidence γ by using the corresponding values provided by the learning algorithm, e.g., C4.5rules generates a prediction confidence for each rule. As we have noticed, each instance f_k in S could have more than one solution. We propose two solution selection schemes: Random Selection and Best Selection, as shown in Table 4.4 and Table 4.5, respectively. Random selection means that we randomly pick either one solution for the corresponding instance f_k for the correction. Best selection means that the solution with the highest prediction confidence is picked for the correction. In Random-Selection, if the solution list EA_k for instance f_k is empty, no attribute value change will be made, as shown on line 6 of Table 4.4. In Best-Selection, if the prediction confidence does not reach the predefined threshold γ , no attribute value change will be made, as shown on line 9 of Table 4.5.

4.2 ACE: An Aggressive Classifier Ensemble with Error Detection, Correction, and Cleansing

The flowchart of aggressive classifier ensembling is shown in Figure 4.1. The main idea of our design is as follows. Assuming we can get a noise free dataset E , we artificially corrupt the dataset with a certain amount of noise.

We have mentioned that one common way to construct a classifier ensemble is to inject random noise into the base training data of each base learner. Table 4.6 describes the way of algorithm ACE to build the base training dataset. On line (4) of Table 4.6, the Random-Selection procedure randomly selects any one solution for instance f_k . As no single solution is perfect (and therefore cannot be fully trusted), we refer to this random process to generate base training sets, which are produced from different aspects of data corrections. On line (5) of Table 4.4, m_k denotes the number of solutions for instance f_k . One line (11), the value of m_k being 0 implies two possibilities: (1) the prediction result of each attribute A_i

Table 4.4: Random Selection

Procedure: Random-Selection

Input: dataset S , solution set EA for S

Output: dataset S'

- 1 Let $n \leftarrow \#$ of instances in S ;
- 2 Let $S' \leftarrow \emptyset$;
- 3 **For** $k \leftarrow 1$ to n
- 4 Let $f_k \leftarrow$ the k^{th} instance in S ;
- 5 Let $m_k \leftarrow \#$ of solutions in EA_k ;
- 6 **If** ($m_k > 0$)
- 7 SOLUTION \leftarrow randomly pick a solution from EA_k ;
- 8 $f_k \leftarrow$ Modify f_k with SOLUTION;
- 9 Add f_k into S' ;
- 10 **Else If** ($m_k = 0$)
- 11 Either add f_k into S' without change or not add f_k into S' ; (Please refer to the explanations in Section 4.2.)
- 12 **End If**
- 13 **End For**
- 14 return S' ;

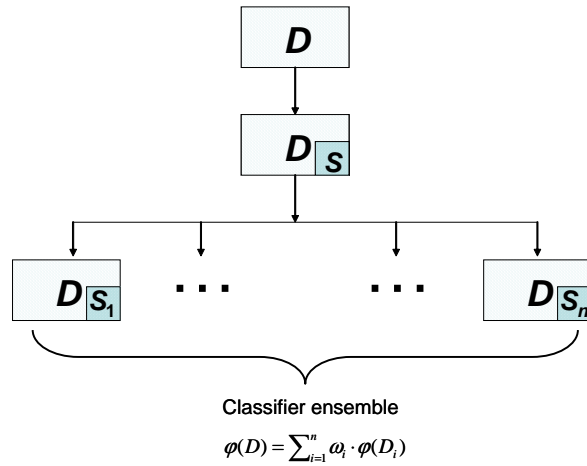


Figure 4.1: The Framework of ACE

Table 4.5: Best Selection

Procedure: Best-Selection

Input: dataset S , solution set EA for S
Threshold of prediction confidence γ

Output: dataset S'

```
1  Let  $n \leftarrow \#$  of instances in  $S$ ;  
2  Let  $S' \leftarrow \emptyset$ ;  
3  For  $k \leftarrow 1$  to  $n$   
4      Let  $f_k \leftarrow$  the  $i^{th}$  instance in  $S$ ;  
5      Let  $m_k \leftarrow \#$  of solutions in  $EA_k$ ;  
6      If ( $m_k > 0$ )  
7          SOLUTION  $\leftarrow$  the solution with the highest confidence in  $EA_k$ ;  
8          CV  $\leftarrow$  the prediction confidence of SOLUTION;  
9          If(CV  $> \gamma$ )  
10              $f_k \leftarrow$  Modify  $f_k$  with SOLUTION;  
11         End If  
12     End If  
13     Add  $f_k$  into  $S'$ ;  
14 End For  
15 return  $S'$ ;
```

Table 4.6: Aggressive Classifier Ensemble (ACE) Construction

Procedure:	ACE
Input:	dataset D ; ensemble size n ; the maximum number of simultaneous attribute value changes for each instance z .
Output:	n base datasets $\{D_1, \dots, D_n\}$.
1	$S \leftarrow \mathbf{Filtering-Suspicious-Instances}(D)$;
2	$EA \leftarrow \mathbf{Solution-Set-Construction}(D, S, z)$;
3	For $i \leftarrow 1$ to n
4	$S' \leftarrow \mathbf{Random-Selection}(S, EA)$;
5	$D_i \leftarrow D \ominus S \oplus S'$;
6	End For
7	return $\{D_1, \dots, D_n\}$

by classifier AP_i is the same as the original attribute value; or (2) after each possible change of up to z attribute values, instance f_k still cannot be correctly predicted by classifier T . In the case of possibility (1) instance f_k will be forwarded into dataset S without any change. Otherwise, if possibility (2) occurs, instance f_k will be removed eventually. Our experiments show that the value of m_k for most instances in S is bigger than 0, which guarantees that the size of dataset $D \ominus S \oplus S'$ will not reduce too much compared with the original noisy dataset D .

4.3 C2: Corrective Classification

Figure 4.2 presents the framework of C2, which consists of three major steps: diversity enhancement, accuracy enhancement, and classifier ensembling. The diversity enhancement and accuracy enhancement steps denoted in Figure 4.2 correspond to line (2), and lines (3-6) in the pseudocode described in Table 4.7, respectively. Given a noisy dataset D , for diversity enhancement purpose, a Bootstrap sampling process is first applied on D and produces Bootstrap samples D'_i with the size $\mu \times |D|$, where $\mu \in (0, 1]$. To enhance the

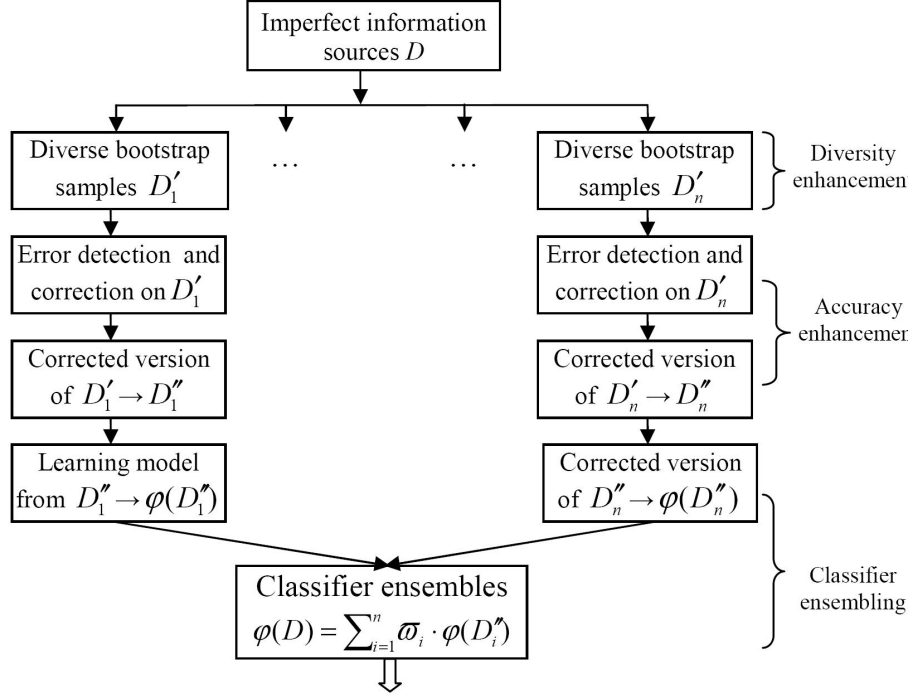


Figure 4.2: The Framework of C2

accuracy of the base learner, an error detection and correction process is performed on D'_i . During this process, a suspicious instance subset S will be constructed (Table 4.2). Following error detection and correction, a solution set EA containing error correction solutions for S will be produced (Table 4.3) and a solution for every instance in S will be recommended for error correction (Table 4.5). C2 uses the Best-Selection scheme as shown in Table 4.5.

The base learner construction of C2 is performed on line (2) of Table 4.7. The diversity enhancement of C2 is mainly achieved through Bootstrap sampling. Given data source D , with $N_0 = |D|$ instances, C2 adopts Bootstrap sampling to randomly sample $\mu \times N_0$ instances from D (with replacement) to construct one bag of Bootstrap samples. This process repeats a certain number of times and results in n bags of Bootstrap samples, on which the error detection and correction process will perform, and eventually n base learners will be built.

The proposed error detection and correction process can not ensure a 100% accuracy,

Table 4.7: Pseudocode of Corrective Classification (C2)

Input:	dataset D ; ensemble size n ;
	the maximum number of simultaneous attribute value changes for each instance z ;
	Threshold of prediction confidence γ ; the fraction of the training data μ .
Output:	n base datasets $\{D_1, \dots, D_n\}$.
1	For $i \leftarrow 1$ to n
2	$D' \leftarrow$ Bootstrap sampling $\mu \times D $ instances from D ;
3	$S \leftarrow$ Filtering-Suspicious-Instances (D');
4	$EA \leftarrow$ Solution-Set-Construction (D', S, z);
5	$S' \leftarrow$ Best-Selection (S, EA);
6	$D_i \leftarrow D' \ominus S \oplus S'$;
7	End For
8	return $\{D_1, \dots, D_n\}$;

thus new errors might be introduced to the dataset. This, however, does not necessarily mean that C2 bears the same deficiency as most existing data cleansing or error correction approaches. First of all, we will show in Section 4.4.5 that, the correctly located attributes account for the majority of all located attributes. Meanwhile, the research efforts on ACE have shown that the prediction accuracy of each base learner will most likely increase after the error correction process (Zhang, Zhu, Wu, and Bond 2005), where each base learner in the ensemble often outperforms the benchmark learner T that was built from the original noisy data. Second, the design of C2 ensures that although new errors might still be introduced to the training sets, the impact of these newly introduced errors will be smoothed out from the effect of randomly Bootstrap sampling and majority voting. With the above two pieces of evidence on (1) enhancing base learners through error correction and cleansing, and (2) reducing new errors through Bootstrap sampling and voting, we expect that C2 can perform effective learning from noisy information sources.

4.4 Experimental Results

We compare six learning algorithms: C4.5 (Quinlan 1986; Quinlan 1993), ACE (Zhang, Zhu, Wu, and Bond 2005), Bagging (Breiman 1996), C2 (Zhang, Zhu, and Wu 2006), Boosting (Freund and Schapire 1996), and DECORATE (Melville and Mooney 2004). Except C4.5, which works as the baseline algorithm, all other algorithms are classifier ensembling methods. The objective of our experiments is three-fold. Firstly, we intend to verify that C2 is superior to its two degenerates: ACE and Bagging, since they focus on enhancing the accuracy and diversity of base learners, separately. Secondly, we explore whether the error detection module in ACE and C2 is effective. Thirdly, we want to compare the performance of these ensembling methods, and try to interpret why some of them perform well and others do not.

We attempt to use two evaluation metrics – prediction accuracy and stability of classifier ensembles to show that C2 outperforms the other state-of-the-art ensembling methods we are comparing with. Of these two metrics, prediction accuracy is a widely applied criterion that indicates how accurate a learning algorithm is when making predictions on new instances; while the stability of classifier ensembling methods indicates how stable a classifier ensemble is when it is generalized to learn on different datasets. We have demonstrated the importance of these two criteria through the bias-variance analysis in Section 3.3.

4.4.1 Experimental Data

We evaluate the system performances on datasets collected from the UCI data repository, which aggregates datasets with a wide variety of data types, analysis tasks, and application areas. It provides data mining researchers with a platform to evaluate and scale their algorithms on a large set of benchmark datasets (Hettich and Bay 1999). In this study, we report the results from ten benchmark datasets, which contain categorical attributes and no missing data entries. A summary of these ten datasets is shown in Table 4.8.

Table 4.8: Dataset Summary

dataset	# inst	# attri	# class labels
car	1728	7	4
monks3	432	7	2
balance	630	5	3
nursery	12960	9	4
splice	3190	61	3
tictactoe	958	10	2
krvskp	3196	37	2
audio	226	70	24
segment	2310	20	7
soybean	683	36	19

Column 1-4 show the name, the number of instances, the number of attributes (including the class attribute), and the number of possible values in the class attribute of the dataset, respectively.

4.4.2 Experiment Settings

We have used Weka-3-4 (Witten and Frank 2000), which provides extensive machine learning algorithms written in the Java language, to implement our system. For all reported results, the learner φ is constructed by using C4.5 trees (Quinlan 1993), with $n = 50$, $z = 3$, $\mu = 1$ and $\gamma = 0.5$, unless additional explanations otherwise (refer to Table 4.7 for notations).

In order to evaluate the performance of the six learning algorithms under different noise levels, we obtain each dataset D by corrupting a dataset E with artificially generated noise. Given a real world dataset E from the UCI data repository, we first separate E into two parts: a dataset D_{base} and the corresponding testing set D_{test} . Then we manually inject attribute noise into D_{base} , where erroneous attribute values are introduced into each attribute with a level of $x*100\%$, and the error corruption for each attribute is assumed to be independent. To corrupt an attribute A_i with a noise level $x*100\%$, the value of A_i has $x*100\%$ probability of being randomly changed to another possible attribute value. After noise corruption, we denote the noise-corrupted dataset by D .

For every dataset E , the performances of the six algorithms are evaluated by averaging over 10 standard 10-fold stratified cross-validation experiments. For each 10-fold cross-validation, dataset E is randomly partitioned into 10 equal-sized subsets with the original class distribution preserved. Then each set is in turn used as the test set D_{test} while the classifier trains on the other nine sets (denoted as D_{base}). The results are averaged over 10 trials.

4.4.3 Prediction Accuracy

Prediction accuracy is the primary metric to evaluate a learning algorithm. This experiment compares the performance of C4.5, ACE, Bagging, C2, Boosting, and DECORATE, where C4.5 works as the baseline. The comparisons are made on 10 datasets and the results are reported in Figure 4.3. For each dataset, the comparative results under five noise levels (0%, 10%, 20%, 30% and 40%) are reported. There are two rows of results shown for each

experiment, in which the first row shows the average prediction accuracy, and the second row shows the standard deviation of the prediction accuracy. The shaded value in each row indicates the algorithm with the best prediction accuracy under the same noise level, by considering only C4.5, ACE, Bagging, and C2. The boxed values indicate the best performance among all the six methods. We have performed Dunnett test for a comparison of all treatments with a control, where C2 is treated as the control, and other algorithms are treated as treatments (Dunnett 1955). If the result has a significant difference with C2 under the significance level $\alpha = 0.05$, it is denoted by a “*”. The results turn out to be what we have expected. In general, the performance of C4.5 is the lower bound of all algorithms, with a few exceptions; Bagging and C2 outperform ACE consistently; Bagging, C2 and Boosting are competitive with each other, while C2 has better accuracy and stability; DECORATE has exceptional performance occasionally, but its performance varies a lot on different datasets.

Without any noise correction and cleansing, the learner voted from the Bagging ensemble almost always produces better results than C4.5. With noise correction and data cleansing, ACE can achieve better performances than C4.5, but the improvement from ACE is normally less significant than that from Bagging. Note that ACE and Bagging represent two fundamental approaches to enhance the classifier ensemble: enhancing base learner accuracies (ACE) and diversities (Bagging). It is convincing that either of them may lead to an improved classifier. C2, which enhances the base learner accuracies and diversities at the same time, is expected to outperform both ACE and C2. It can be observed that, by only considering the performances of C4.5, ACE, Bagging, and C2, C2 has won 30 and tied 1 out of 50 experiments; and Bagging has won 14 and tied 1. Among these experiments, C2 significantly outperforms Bagging for 5 experiments, and Bagging significantly outperforms C2 for 1 experiment. From these experiments we conclude that the correction module in the C2 framework plays an positive role in many, but not all cases. If we consider the three most competitive algorithms C2, Bagging, and Boosting, C2 has won 24 and tied 1; Bagging

has won 6 and tied 1; and Boosting has won 14 and tied 1. In the 16 cases that Boosting and C2 have significantly different results, C2 outperforms Boosting in 15 experiments, and Boosting outperforms C2 in 1 experiment. For other experiments, none of the ensemble methods are significantly better than any other ensemble approach at the 95% confidence level.

We also notice that Boosting performs very well on some datasets, “splice”, “segmentation”, and “soybean”, for examples. However, its performance is even inferior than C4.5 in some other cases (“monks3” and “car”), while the performances of Bagging and C2 are more stable. The performance of Decorate is worse than other ensembling methods. Its performance is worse than C4.5 in all the experiments for 4 datasets: “nursery”, “monks3”, “car”, and “balance”. One interpretation of why Boosting has unstable performance was provided by Quinlan (Quinlan 1996). He pointed out that the unstable performance might be caused by overfitting the training dataset. Although Boosting generally increases accuracy, it leads to a deterioration on some datasets. Although this interpretation makes intuitive sense, it is difficult to qualitatively verify its validity. From another prospective, we intend to interpret the prediction results of ACE, Bagging, C2, Boosting, and DECORATE by exploring the relationship between their base learners and the combined ensemble. In Section 4.4.5, we will show the bias-variance decomposition results, which provide us a closer view into the characteristics of the building blocks of a classifier ensemble.

4.4.4 Diverse and Accurate Base Learners

As introduced in Section 3.1, an effective classifier ensemble should consist of base learners with high-accuracy and high-diversity in predictions. In Figure 4.4, we report the prediction accuracies of individual base learners, in comparison with the results of classifier ensembling. Two plots on each row of Figure 4.4 correspond to the algorithm performances at the noise level 10%, 20%, 30% and 40%, respectively (on the Monks-3 dataset). The box plot on the left column in Figure 4.4 shows the prediction accuracy of individual base learners

		Prediction Accuracy (%)									Prediction Accuracy (%)						
dataset	noise (%)	C4.5	ACE	Bagging	C2	AdaBoost	DECORATE		dataset	noise (%)	C4.5	ACE	Bagging	C2	AdaBoost	DECORATE	
audio	0	77.83	81.74	81.74	82.61	81.74	81.74	7.33	nursery	0	*98.87	96.98	*98.70	97.39	*99.58	*96.29	
		9.04	5.34	6.42	6.15	8.15					0.44	0.36	0.39	0.34	0.17	0.87	
	10	62.17	70.87	68.70	69.13	70.43	70.00	5.96	10	*89.04	*92.68	*90.52	93.83	*83.71	*77.39		
		5.44	8.95	7.04	8.06	5.34					1.05	0.54	0.79	0.63	0.82	1.00	
	20	50.43	59.13	56.52	58.26	54.35	50.00	10.89	20	*77.95	*84.58	*80.93	87.50	*75.54	*68.3		
	10.69	6.22	6.48	5.50	8.51					1.03	0.75	0.65	0.68	0.99	0.98		
splice	30	32.61	33.04	41.30	43.91	41.30	42.61	8.15	30	*68.32	*76.32	*71.65	80.31	*66.18	*60.26		
		13.16	14.52	8.51	10.74	9.67					1.22	1.93	1.58	1.43	2.04	1.80	
	40	21.30	30.87	30.87	30.43	26.96	31.74	8.46	40	*59.54	*67.06	*60.79	70.05	*57.19	*51.64		
		8.31	11.49	6.94	8.45	6.74					1.63	1.63	1.29	1.74	1.40	1.62	
	0	92.76	92.79	94.33	94.42	94.36	*86.9	1.89	monks3	0	100.00	100.00	100.00	100.00	100.00	*98.14	
	1.81	1.53	1.50	1.42	1.70					0.00	0.00	0.00	0.00	0.00	2.14		
10	*88.56	*88.15	92.19	92.19	92.95	*85.58	1.80	10	97.91	99.30	97.67	99.07	*93.72	*84.88			
	1.32	2.01	1.59	2.22	1.81					2.31	1.12	1.90	1.20	3.48	3.15		
20	*82.60	*83.67	89.84	89.91	91.50	*83.54	3.30	20	*90.47	96.05	94.88	96.28	*86.98	*83.49			
	1.47	2.52	1.81	2.04	2.15					4.02	4.11	3.43	2.73	4.41	6.62		
30	*77.49	*79.91	89.15	89.12	89.31	*82.07	1.84	30	*81.86	89.30	85.35	89.77	*78.60	*77.44			
	2.36	3.45	2.72	2.12	1.87					4.36	6.86	5.03	5.61	5.00	5.38		
40	*71.88	*73.95	84.48	85.05	85.14	*76.14	1.55	40	75.81	80.70	76.74	83.49	*71.63	*70.47			
	4.02	3.11	1.87	2.31	2.08					6.60	6.94	8.56	4.02	6.74	9.75		
segment	0	94.03	94.20	94.85	94.85	95.58	-		car	0	94.34	91.73	94.22	93.24	95.78	*87.34	
		1.85	1.73	1.46	1.63	1.48					2.64	3.43	2.50	2.70	1.97	2.82	
	10	91.39	92.03	92.86	92.90	94.46	-		10	*85.14	88.50	86.99	88.96	*82.14	*76.82		
		1.41	2.11	2.30	2.11	1.79					2.02	2.86	1.55	2.01	2.70	3.00	
	20	89.00	89.22	91.00	90.87	91.73	-		20	*76.88	78.44	*76.71	80.87	*74.57	*68.90		
	1.96	2.18	1.92	1.39	1.93					2.72	2.75	2.78	3.90	2.50	2.61		
soybean	30	*85.11	*86.36	88.18	89.05	89.57	-		30	72.72	74.86	74.22	76.42	*70.40	*67.34		
		2.03	1.61	1.67	1.54	1.28					3.43	2.98	3.34	3.74	3.53	4.86	
	40	*80.91	*81.77	85.41	85.19	85.76	-		40	67.57	69.08	68.21	70.98	*66.30	*60.17		
		2.94	3.09	3.08	2.63	3.18					3.10	2.59	3.98	2.12	3.21	3.04	
	0	90.59	90.29	92.35	92.50	92.06	92.50	2.72	balance	0	68.57	*63.65	75.40	73.02	71.75	*57.30	
	2.96	3.81	3.01	3.13	3.75					3.49	4.64	4.38	5.34	4.84	6.37		
10	85.88	86.76	89.71	90.15	90.74	90.59	3.19	10	65.87	*65.56	72.70	71.75	69.37	*54.76			
	4.40	3.92	3.18	3.33	4.39					5.71	5.60	4.78	5.69	5.70	4.31		
20	*73.97	*77.50	80.44	83.97	83.38	81.32	3.92	20	62.70	67.14	68.73	68.73	62.54	*52.22			
	5.84	4.50	3.92	4.62	2.41					6.18	3.43	5.19	5.35	6.09	11.65		
30	*59.71	66.76	71.32	73.24	71.47	71.47	5.77	30	60.79	59.37	62.86	64.76	59.21	*56.67			
	7.18	7.11	4.71	6.11	6.55					6.78	5.56	7.41	8.12	7.37	4.79		
40	*47.50	*50.44	55.44	58.24	55.74	55.00	5.15	40	57.62	62.86	60.32	63.02	58.73	*54.13			
	4.16	3.92	5.76	5.81	5.12					7.33	2.92	5.18	5.99	6.03	6.63		
tictactoe	0	*86.98	*89.48	94.79	94.58	95.73	*90.42	3.17	krvskp	0	99.47	99.31	99.56	99.41	99.56	98.78	
		3.00	4.54	3.44	2.85	2.22					0.51	0.60	0.56	0.48	0.42	0.76	
	10	80.73	82.19	84.38	85.10	82.08	79.90	4.91	10	*90.63	94.50	94.88	96.22	*92.16	*92.19		
		5.13	3.59	4.74	4.23	5.46					2.63	1.66	1.41	1.36	1.58	1.88	
	20	*69.69	76.46	78.44	78.33	75.63	*73.33	3.68	20	*79.03	*85.03	88.66	88.41	*80.47	*79.75		
	6.12	4.53	3.93	3.83	3.37					4.42	2.49	2.35	2.87	3.24	3.28		
30	69.06	67.92	71.35	70.83	69.58	65.10	2.75	30	*64.44	74.94	76.69	78.75	*69.00	*68.97			
	6.59	6.38	5.63	7.22	4.41					4.29	6.10	5.27	4.26	4.29	4.15		
40	*60.52	66.15	67.50	68.23	*62.50	*60.31	4.85	40	58.16	63.19	62.25	63.19	60.75	*55.66			
	4.40	3.84	5.83	5.29	3.90					4.36	4.96	5.40	5.62	4.35	3.68		

Figure 4.3: Comparative Results of Prediction Accuracy on Ten Datasets from UCI Data Repository

There are two rows of results shown for each experiment, in which the first row shows the average prediction accuracy, and the second row shows the standard deviation of the prediction accuracy. The shaded value in each row indicates the algorithm with the best prediction accuracy under the same noise level, by considering only C4.5, ACE, Bagging, and C2. The boxed values indicate the best performance among all the methods. We have performed Dunnett test for a comparison of all treatments with a control, where C2 is treated as the control, and other algorithms are treated as treatments. If the result has a significant difference with C2 under the significance level $\alpha = 0.05$, it is denoted by a “*”. When dataset “segment” was used, the execution of algorithm DECORATE terminated in an unexpected way for unknown reasons, so that there is no prediction result reported in this table.

of three ensembling methods with the ensemble size $n = 50$. This result is acquired by summarizing the prediction accuracy of all individual base learners in each of the three classifier ensembles. The right column shows the performance of three classifier ensembles, ACE, Bagging and C2 with the ensemble size 10, 25 and 50, respectively. For a more comprehensive comparison, we also report the prediction accuracy of the benchmark learner T in Figure 4.4. In the box plot, the average performance of base learners is centered in the middle of each box. The box represents the middle 50% of the base learners in each ensemble. The lines in each box denote the performances of medians. The smaller the box is, the less variations of the base learners' performances.

There are a couple of points to note in Figure 4.4. (1) The base learners of ACE perform much better than those of the other two classifier ensembles. The probable reason for it may be that the sampling process in Bagging and C2 makes the data in each bag less representative of the true population than the original training data, while each bag in ACE has a better or the same ability as the original data to represent the true population. Hence, the learners built upon the bags in ACE are more precise than the other two. (2) The average of the base learners of ACE performs better than the benchmark classifier C4.5, which verifies that the error detection and data cleansing procedure improves the data quality. (3) The performances of both Bagging and C2 ensembles have a quite clear increasing trend along with the increase of the ensemble size while ACE does not, which reflects that the base learners in ACE lack diversity. (4) Compared to (1), Bagging and C2 ensembles outperform ACE, which shows that the diversity of a classifier ensemble plays a more important role than the accuracy of its base learners. (5) C2 outperforms Bagging in terms of both the base learners and the ensemble for most of the time, which confirms that the incorporation of the error correction component and the sampling procedure to produce each data bag is a more effective strategy. If the data cleansing procedure could be further optimized the results are expected to be more stable.

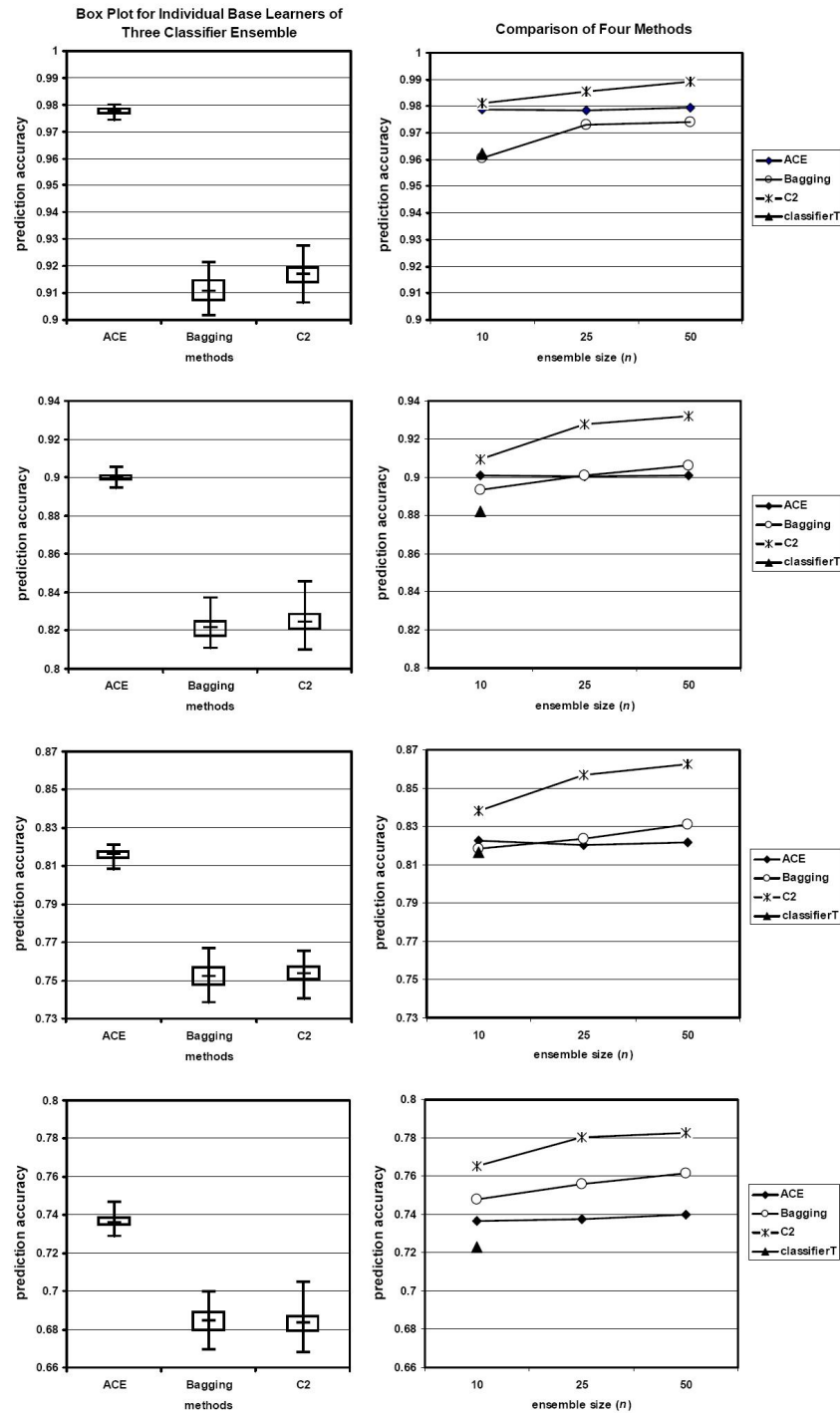


Figure 4.4: Comparison of Three Methods on Dataset Monks3 with Noise Levels 10%-40%

Firstly, the box plots on the left column show the prediction accuracy of individual base learners of an classifier ensemble (ACE, Bagging, or C2) with ensemble size $n = 50$. Secondly, the plots on the right column report the comparison results of ACE, Bagging, C2 with different ensemble size ($n= 10, 25, 50$). For the comparison purpose, the result of benchmark classifier T (C4.5) is also plotted.

4.4.5 Noise Detection Analysis

Before going into the analysis, we define the concept of “Importance of Attributes” here.

Definition 4.1 [The Importance of Attributes] The importance of an attribute A_i is proportional to the Information Gain Ratio (IGR) between A_i and the class attribute C , which indicates how informative the attribute is towards the classification. The higher IGR, the more important the attribute.

We have claimed that instances in S are the most suspicious instances. The heuristic to acquire S is: instances that do not comply with the model built from the dataset D' are more likely being noisy than others. To show this heuristic is reasonable, we report the following statistics summarized in Figure 4.5: (1) the proportion of erroneous attributes in S , denoted as P_{EA} (column 3); (2) the proportion of erroneous important attributes, denoted as P_{EIA} (column 4); (Please refer to Def. 4.1 for the definition of “the importance of attributes”) and (3) The proportion of correctly located important attributes, denoted as P_{LOC} (column 5). We order the feature attributes according to their Information Gain Ratio (IGA). The higher half of the attributes are regarded as the most important attributes. Here P_{EIA} represents the number of erroneous values in the most important attributes divided by the number of erroneous values in all attributes in S .

In Figure 4.5, the second column shows the noise level as well as the proportion of noise infected attribution values in D' . The third column P_{EA} is bigger than the corresponding noise level, which indicates that the erroneous attribute values in S is denser than that in D' . In the fourth column, P_{EIA} is above 0.5 in general, especially for low noise levels, which shows S is more likely to contain errors in more important attributes. When the noise level gets bigger, such a trend is less remarkable. The fifth column P_{LOC} is much greater than 50%, which indicates that the error locating module is able to locate the majority of important attribute noise.

dataset	noise level (%)	P_EA	P_EIA	P_LOC	dataset	noise level (%)	P_EA	P_EIA	P_LOC
audio	10	0.13	0.44	0.71	nursery	10	0.21	0.68	0.86
	20	0.21	0.45	0.67		20	0.30	0.64	0.80
	30	0.32	0.49	0.56		30	0.38	0.57	0.79
	40	0.41	0.51	0.55		40	0.48	0.52	0.81
splice	10	0.12	0.53	0.70	monks3	10	0.24	0.78	0.87
	20	0.21	0.52	0.69		20	0.34	0.72	0.86
	30	0.31	0.51	0.65		30	0.42	0.60	0.86
	40	0.41	0.50	0.58		40	0.50	0.56	0.80
segment	10	0.15	0.64	0.86	car	10	0.25	0.60	0.79
	20	0.23	0.62	0.80		20	0.34	0.55	0.76
	30	0.32	0.58	0.81		30	0.42	0.53	0.73
	40	0.41	0.53	0.78		40	0.50	0.51	0.73
soybean	10	0.15	0.52	0.78	balance	10	0.34	0.51	0.71
	20	0.24	0.51	0.76		20	0.43	0.50	0.69
	30	0.33	0.51	0.76		30	0.49	0.50	0.66
	40	0.43	0.51	0.72		40	0.56	0.50	0.56
tictactoe	10	0.19	0.61	0.81	krvskp	10	0.12	0.55	0.68
	20	0.28	0.58	0.75		20	0.22	0.53	0.65
	30	0.38	0.57	0.77		30	0.32	0.51	0.64
	40	0.47	0.52	0.74		40	0.42	0.50	0.60

Figure 4.5: Noise Detection Results

4.4.6 Stability

In Section 3.3, we introduced how bias-variance decomposition could be applied in analyzing the classifier ensembling methods in detail. We have shown that

$$\begin{aligned}
E_{\Delta,C}[L(c,y)] &= k_1 E_C[L(c,y_*)] + L(y_*,y_m) + k_2 E_{\Delta}[L(y_m,y)] \\
&= k_1 N(x) + B(x) + k_2 V(x),
\end{aligned}
\tag{4.5}$$

where k_1 and k_2 take the values on the basis of problem type as shown in Figure 4.6. Assuming $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ is a set of training data, $L(c, y)$ represents the loss of a single learner $\varphi(D)$ that makes a prediction given a test instance x . The loss is incurred by making the prediction $C = y$ while the class variable is $C = c$. $E_{\Delta,C}[L(c, y)]$ denotes the expected loss when considering all the base learners in the classifier ensemble, and considering all possible C values. $E_C[L(c, y_*)]$, defined as noise ($N(x)$), denotes the expected loss incurred by the difference between the true class variable $C = c$ and the optimal prediction $C = y_*$. $L(y_*, y_m)$, defined as bias ($L(y_*, y_m)$), denotes the loss incurred by the difference between the optimal prediction y_* and the main prediction y_m from the

	Two-class problem		Multi-class problem	
	$y_m = y_*$	$y_m \neq y_*$	$y_m = y_*$	$y_m \neq y_*$
k_1	$2P(y = y_*) - 1$		$P(y = y_*) - P(y \neq y_*)P(y = c y_* \neq c)$	
k_2	1	-1	1	$-P(y = y_* y \neq y_m)$

Figure 4.6: Multiplicative Factor Values in the Bias-variance Decomposition for Classifier Ensembles

To obtain the k_1, k_2 values shown in the table, we have following assumptions: (1) the learning algorithm is a classifier ensemble; (2) $\forall_y L(y, y) = 0$ and $\forall_{y_1 \neq y_2} L(y_1, y_2) \neq 0$; (3) the loss function is symmetric: $\forall_{y_1, y_2} L(y_1, y_2) = L(y_2, y_1)$; (4) zero-one loss is used in the loss function.

classifier ensemble. $E_\Delta[L(y_m, y)]$, defined as variance ($V(x)$), denotes the loss incurred by the difference between the main prediction y_m of the classifier ensemble and the predictions of the base learners.

As $N(x)$ reflects the intrinsic nature of the dataset, no matter which learning algorithm used for training the learner, $N(x)$ is always the same. Therefore, we are only interested in estimating $B(x)$ and $V(x)$, which are algorithm dependent components. To minimize the expected loss $E_{\Delta, C}[L(c, y)]$, smaller $B(x)$ and $k_2 V(x)$ are preferred. Fortunately, $B(x)$ and $V(x)$ could be calculated in an obvious way according to their definitions. Since we are usually interested in the performance of a learning algorithm on a set of test instances, we will calculate $E_X[B(x)]$ and $E_X[k_2 V(x)]$, the bias and variance over all instances in the test dataset. Before we show the way to calculate $E_X[B(x)]$ and $E_X[k_2 V(x)]$, we first define the concept “biased/unbiased instance” based on (Domingos 2000).

Definition 4.2 [biased/unbiased instance] A biased instance is a test instance x that is incorrectly predicted by the learning algorithm, i.e. $B(x) = 1$. On the contrary, an unbiased instance is a test instance x that is correctly predicted, and $B(x) = 0$.

The way to calculate $E_X[B(x)]$ is shown in Equation 4.6

$$E_X[B(x)] = \frac{1}{N_0} \sum_{i=1}^{N_0} B(x_i), \quad (4.6)$$

where x_i denotes a test instance, N_0 denotes the number of test instances. It is clear that $E_X[B(x)]$ is in fact the prediction error rate of a learning algorithm on a set of test instances. The prediction accuracy we reported in section 4.4.3 is $1 - E_X[B(x)]$.

$$V_u = \frac{1}{N_0} \left[\sum_{i=1}^{N_0} (1 - B(x_i)) V(x_i) \right] \quad (4.7)$$

$$V_b = \frac{1}{N_0} \left[\sum_{i=1}^{N_0} k B(x_i) V(x_i) \right] \quad (4.8)$$

Then we will show how to calculate $E_X[k_2 V(x)]$, where k_2 take the value as shown in Figure 4.6. We denote V_u and V_b as the variance on correctly predicted and incorrectly predicted instances, respectively. They are calculated in the way shown in Equation 4.7 and Equation 4.8, where $k = 1$ for two-class problems, and $k = P(y = y_* | y \neq y_m)$ for multi-class problems. In other words, k only count in the contribution from those base learners who predict correctly. The variance is additive in correctly predicted instances but subtractive in incorrectly predicted ones. Thus, the overall variance on the test instances from a particular ensembling method is V_u subtracting off V_b as shown in Equation 4.9:

$$E_X[k_2 V(x)] = V_u - V_b \quad (4.9)$$

The variance $E_X[k_2 V(x)]$ measures the stability of an ensembling method, which is also very important besides the prediction accuracy. We have mentioned that to minimize the expected loss $E_{\Delta, C}[L(c, y)]$, smaller $B(x)$ and $k_2 V(x)$ are preferred. We can interpret it in an intuitive way: a well designed classifier ensemble is the one with low error rate (high prediction accuracy) and high stability (low variance), so that the expected loss for performing a prediction is minimized. In order to achieve low bias and low variance, the ideal classifier ensemble should have a 0 bias ($E_X[B(x)] = 0$) and 0 variance ($E_X[V(x)] = 0$). In other words, it requires not only the voted result to be unbiased for all test instances, but also every base learner to have the same prediction, i.e. all the predictions are correct. However, this situation is not realistically achievable. Otherwise, since any single learner

from the base learners is perfect, no classifier ensemble is needed at all. The situations we could possibly encounter are: (1) low bias + low variance; (2) low bias + high variance; (3) high bias + low variance; and (4) high bias + high variance. It is obvious that (1) is more preferred than (2),(3), and (4); (2) is more preferred than (4); and (3) is more preferred than (4). The only difficult comparison is (2) versus (3), where we have to decide whether the bias or the variance is more important for a classifier ensemble. As we know, the key measure to evaluate a classification method is the prediction accuracy, or $(1 - \text{error rate})$, which is $(1 - E_X[B(x)])$ in our context. The variance measures the stability of a voted result. For example, for the same correctly voted prediction, a 10 : 0 voting is more stable than a 6 : 4 voting. Thus, the method with low bias and high variance is more preferred than the method with high bias and low variance. We consider bias as the major measure, and variance as the secondary measure.

In order to further discuss this problem, we show the bias-variance decomposition results from our experiments in Figure 4.7.

In Figure 4.7, we report the results of the variance of ACE, Bagging, C2, Boosting, and DECORATE. There are two rows of results provided for each trial. The prediction accuracy in the first row is the same as that reported in Table 4.3. It is for the comparison purpose by showing the prediction accuracy here. The boxed values indicate the best performance among all the methods. The second row of the results indicate the variance of the corresponding ensembling method. Overall, ACE has the lowest variance, Bagging and C2 are in the middle, Boosting and Decorate have much higher variance than the others. A lower variance indicates a more stable learning method, so that ACE has the best stability, Boosting and DECORATE are the most unstable ones.

Why does ACE have the lowest variance? We have shown in Section 4.4.4 that the base learners of ACE lacks diversity. In other words, the base learners have very similar structures and prone to agree with each other. It is less likely to have much fluctuation among the prediction results in the base learners. Thus $V(x) = E_{\Delta}[L(y_m, y)] \rightarrow 0$. When the base

		prediction accuracy (%) and Variance of Ensemble							prediction accuracy (%) and Variance of Ensemble					
dataset	noise (%)	ACE	Bagging	C2	AdaBoost	DECORATE	dataset	noise (%)	ACE	Bagging	C2	AdaBoost	DECORATE	
audio	0	81.74	81.74	82.61	81.74	81.74	nursery	0	96.98	*98.70	97.39	*99.58	*96.29	
		0.01	0.05	0.05	0.12	0.12			0.00	0.01	0.01	0.07	0.41	
	10	70.87	68.70	69.13	70.43	70.00		10	*92.68	*90.52	93.83	*83.71	*77.39	
		0.01	0.06	0.05	0.12	0.13			0.00	0.06	0.06	0.13	0.25	
	20	59.13	56.52	58.26	54.35	50.00	20	*84.58	*80.93	87.50	*75.54	*68.30		
		0.04	0.03	0.06	0.12	0.10			0.01	0.07	0.10	0.13	0.19	
splice	0	92.79	94.33	94.42	94.36	*86.90	monks3	0	100.00	100.00	100.00	100.00	*98.14	
		0.00	0.03	0.03	0.10	0.19				0.00	0.00	0.00	0.00	0.29
	10	*88.15	92.19	92.19	92.95	*85.58		10	99.30	97.67	99.07	*93.72	*84.88	
		0.00	0.05	0.05	0.15	0.19			0.00	0.04	0.04	0.13	0.16	
	20	*83.67	89.84	89.91	91.50	*83.54	20	96.05	94.88	96.28	*86.98	*83.49		
		0.00	0.08	0.08	0.18	0.21			0.00	0.07	0.06	0.14	0.14	
segment	0	94.20	94.85	94.85	95.58	-	car	0	91.73	94.22	93.24	95.78	*87.34	
		0.00	0.02	0.02	0.07	-				0.00	0.02	0.03	0.10	0.31
	10	92.03	92.86	92.90	94.46	-		10	88.50	86.99	88.96	*82.14	*76.82	
		0.00	0.02	0.02	0.07	-			0.01	0.03	0.05	0.11	0.23	
	20	89.22	91.00	90.87	91.73	-	20	78.44	*76.71	80.87	*74.57	*68.90		
		0.00	0.04	0.04	0.07	-			0.00	0.03	0.05	0.09	0.17	
soybean	0	90.29	92.35	92.50	92.06	92.50	balance	0	*63.65	75.40	73.02	71.75	*57.30	
		0.00	0.03	0.03	0.15	0.03				0.00	0.08	0.08	0.11	0.14
	10	86.76	89.71	90.15	90.74	90.59		10	*65.56	72.70	71.75	69.37	*54.76	
		0.02	0.07	0.07	0.13	0.09			0.01	0.08	0.07	0.11	0.11	
	20	*77.50	80.44	83.97	83.38	81.32	20	67.14	68.73	68.73	62.54	*52.22		
		0.02	0.10	0.11	0.16	0.13			0.01	0.07	0.07	0.08	0.08	
tictactoe	0	*89.48	94.79	94.58	95.73	*90.42	krvskp	0	99.31	99.56	99.41	99.56	98.78	
		0.05	0.09	0.10	0.17	0.19				0.00	0.00	0.00	0.05	0.03
	10	82.19	84.38	85.10	82.08	79.90		10	94.50	94.88	96.22	*92.16	*92.19	
		0.02	0.08	0.08	0.13	0.14			0.03	0.06	0.06	0.14	0.13	
	20	76.46	78.44	78.33	75.63	*73.33	20	*85.03	88.66	88.41	*80.47	*79.75		
		0.01	0.06	0.07	0.12	0.11			0.07	0.12	0.11	0.13	0.14	

Figure 4.7: Comparative Results of the Variance in Ensembling Methods on Ten Datasets from UCI Data Repository

There are two rows of results provided for each trial. The prediction accuracy in the first row is the same as that reported in Table 4.3. It is for the comparison purpose by showing the prediction accuracy here. The boxed values indicates the best performance among all the methods. The second row of the results indicate the variance of the corresponding ensembling method. The concept “variance” is defined in Section 3.3. When dataset “segment” was used, the execution of algorithm DECORATE terminated in an unexpected way for unknown reasons, so that there is no prediction result reported in this table.

learners maintain a certain level of diversity, like Bagging and C2, $V(x)$ grows larger. The trend of variance increase is even more noticeable in Boosting and DECORATE.

It is a common consent that the diversity among base learners and the variance of the classifier ensemble have certain connections. Does diversity = variance? We have defined “diversity”, denoted by p_d in Section 3.1. *Div*, which is an unbiased estimate of p_d , is defined as the proportion of test instances that can be predicted independently by the base learners in a classifier ensemble. However, it is difficult to measure which instances are independently predicted, and which are not. On the other hand, “variance” in the context of bias-variance decomposition is a concept that could be quantified. It reflects the prediction fluctuation of the base learners.

Most researchers agree that

Combining the output of several classifiers is useful only if there is disagreement among them, combining several identical classifiers produces no gain (Opitz and Maclin 1999).

In fact, the “disagreement” has the same meaning as “variance” in our context. If there is no variance among the base learners, there is no gain from combining their results. Following this claim, another argument says that

..... an ideal ensemble consists of highly correct classifiers that disagree as much as possible (Opitz and Maclin 1999).

And the authors have empirically verified that such ensembles generalize well (Opitz and Maclin 1999). However, this statement may mislead people if they do not clearly distinguish diversity and variance (disagreement). In a general setting, if we regard the accuracy of base learners being above 50% as “highly correct”, “disagree as much as possible” does not mean to achieve “higher and higher diversity”. In other words, it is not true that the greater the disagreement (variance) is, the more gain will be obtained. From our experiments we can observe that, algorithms with a moderate level of variance, such as

Table 4.9: Connections Between Diversity and Variance

Diversity		Variance
low	→	low
low	←--	low
high	--→	high
high	←--?	high

The arrows represent the relationship of diversity and variance. Whether one side can derive the other side depending on the direction of the arrows. Specifically, \leftarrow indicates that the left hand side can derive the right hand side; dashed arrows $\leftarrow--$ and $--\rightarrow$ indicate it is very likely that one side of the arrow could derive the other side of the arrow; $\leftarrow--?$ indicates that maybe the right hand side can derive the left hand side.

Bagging and C2, have the best performance. Algorithms with the highest variance, like Boosting and DECORATE however, are prone to be unstable. They work extremely well on some datasets, but have performances even worse than a single learner for others. Some researchers give the explanation that the the unstable performance of Boosting is due to the overfitting (Freund and Schapire 1997; Quinlan 1996). If it is the case, it becomes a good example that the high variance of the boosting base learners is not always the sign of the high diversity, but rather because of other reasons. So we have to understand that there could be many causes that result in the high variance, and the high diversity is only one of the possible reasons.

In summary, we would say there are some connections between diversity and variance, but diversity \neq variance. We illustrate the relation between diversity and variance in Table 4.9. If the diversity among base learners is low, the variance **has to** be small as well. On the contrary, if the variance is small, it is **very likely** that the diversity is small. If the diversity is high, it is **very likely** that the variance is high as well; however, if the variance is high, we can only say **maybe** the diversity is high.

Since it is very likely that low variance indicates low diversity, and high variance may

or may not indicate the high diversity, we consider it to be wise to maintain a middle level of variance for a classifier ensemble. Our experimental results in Table 4.7 also show that Bagging and C2, which are two ensembling methods with middle level variance, have the most accurate and stable performance.

4.4.7 Complexity Comparison

In order to analyze the time complexity of the classifier ensembling methods we are comparing with, we first define several notations as follows. We denote m , n , and l as the numbers of attributes, training instances, and base learners, respectively. In addition, we make the assumption that decision trees are used to construct the base learners of a classifier ensemble. As the time complexity of a decision tree construction needs $\Theta(mn \log n)$ (Witten and Frank 2005), the upper bound of the decision tree induction, including possible procedures such as sorting feature attributes and pruning, would be approximately $O(mn^2)$.

The running time for each classifier ensembling method can be split into two components: (1) the time spent on preparing the training data; and (2) the time spent on the model construction. We illustrate the time complexity analysis of ACE, Bagging, C2, AdaBoost, and DECORATE in Figure 4.8. Particularly, Bagging and AdaBoost take $O(n)$ and $k_1 O(n)$ respectively on the sampling of training data for learning each base learner. ACE and C2 take $O(mn)$ and $O(lmn)$ respectively on locating errors and proposing corrections on attribute values in suspicious instances for all base learners. DECORATE takes $k_2 O(n)$ on sampling artificial instances and $O(lmn^2)$ on evaluating whether to include a new base learner into the ensemble. k_1 and k_2 denote positive constants. If l base learners are constructed for each classifier ensembling method, $O(lmn^2)$ time is needed for constructing l base learners. Summarizing these two components, we can derive the overall time complexity being $O(lmn^2)$.

This analysis shows that the classifier ensembling methods ACE, C2, Bagging, AdaBoost, and DECORATE run into the same time complexity $O(lmn^2)$, when assuming

	Base Learner	ACE	Bagging	C2	AdaBoost	DECORATE
Preparing training data	-	$O(mn)$	$O(n)$	$O(lmn)$	$k_1 O(n)$	$k_2 O(n) + O(lmn^2)$
Model construction	$O(mn^2)$	$O(lmn^2)$	$O(lmn^2)$	$O(lmn^2)$	$O(lmn^2)$	$O(lmn^2)$
Total	$O(mn^2)$	$O(lmn^2)$				

Figure 4.8: Time Complexity Comparison

The running time for each classifier ensembling method can be split into two components: (1) the time spent on preparing the training data (shown in the second row); and (2) the time spent on the model construction (shown in the third row). Although the time complexity varies in component (1), the overall time complexity is the same for different classifier ensembling methods we are comparing with (shown in the fourth row).

the construction of each base learner need $O(mn^2)$ running time. In our experiments, the running time of C2 is longer than that of Bagging by a factor of 10 because of the time spending on feature attribute error detection and correction.

4.5 Discussions

Both ACE and C2 follow the guidance in Section 4.1 by switching the roles of class label and each attribute to build each predictor AP_i , which is equivalent to $P(x_E|c, e, x_{-E})$. If the predicted value from AP_i is different from the existing value, the algorithm will conclude an error as long as replacing the old value with new one can lead to a better decision with respect to T . Therefore, the algorithm essentially tries to enhance $P(x_E|c, e, x_{-E})$ and then brings improvement to $P(c|x)$. In addition to enhancing base learner accuracies, ACE and C2 apply different methods to construct the diversity among base learners. ACE injects randomness into base training data by randomly choosing correction solutions to dataset S , which is a subset of the input dataset D . By this means, each copy of the base training data becomes a different version of the corrected data. Since it is hard to decide which copy of the training data is most reliable, we train a classifier ensemble on the basis of them to get a combined decision. In our experiments shown in Section 4.4.5, we have found that

ACE does a good job in error detection and correction on the base training sets.

However, the base learners do not show clear prediction diversity compared to other algorithms, because the improvement gained from the combined effort of base learners is not significant compared to individual base learners. Considering the weak part of ACE, and noticing the success of several resampling based classifier ensemble designs, we have designed the C2 algorithm by using the sampling with replacement scheme on the input dataset D to inject random noise into the base training sets. Then the error detection and correction procedure is performed on the resampled base training set. When C2 is compared with Bagging, Boosting, and DECORATE, the prediction accuracy of C2 is higher than other methods at most of the time. Besides, according to the analysis in Section 4.4.6, C2 is more stable compared to Boosting and DECORATE. The potential problem of the noise detection procedure we have applied lies in that this procedure could introduce new noise, which mainly refer to wrongly corrected attribute values. The affect of this issue becomes non-trivial when the source data is little noise infected. Therefore, it requires the design of error correction procedure to be more precise. The respective roles of different feature attributes should be taken care of, since it has an impact on the ability our current attribute value correctors varies inherently. It also need to be investigated how adaptive the C2 algorithm is when dealing with numeric attributes.

4.6 Chapter Summary

In this chapter, we have presented an effective strategy – a combination of noise handling and classifier ensembling, to build robust learning algorithms on noisy data. Our study has concluded that systems that consider both accuracy and diversity among base learners, will eventually lead to a classifier superior to other alternatives.

We have proposed two algorithms, Aggressive Classifier Ensembling (ACE) and Corrective Classification (C2) algorithm based on this framework. The essential idea is to appropriately take care of the diversity and accuracy among base learners for effective clas-

sifier ensembling. In our experimental results, we have shown that the error detection and correction module is effective in ACE and C2, C2 outperforms its two degenerates ACE and Bagging in most trials, and C2 has more stable performance than Boosting and DECORATE.

Chapter 5

Noise Modeling with Associative Corruption Rules

As we have discussed in Section 2.2.1, the noise in the data may come from various possible sources. Many of these sources, actually, can be traced by analyzing the erroneous data items, unless they are totally random. The noise modeling module in our noise MDU framework will fulfill this objective by generating patterns from data errors. During the noise diagnosing process in the MDU framework, after the identification of erroneous instances, there are two types of possible information available for this purpose: (1) the isolated suspicious instance subset, and (2) an corresponding instance subset corrected by using other advanced tools or by domain experts. In this chapter, we propose to use an association based noise modeling approach to capture error patterns from the data. The materials from this chapter have been published in the Proceedings of the 2007 IEEE International Conference on Data Mining, with the title “Noise Modeling with Associative Corruption Rules” (Zhang and Wu 2007).

5.1 Problem Statement

Supposing that we are given a noisy dataset D_{obs} , where $D_{obs} = D_{obs1} \cup D_{obs2}$, and provided with a noise-free dataset D_{c1} , which is the dataset D_{obs1} after careful inspection and correction. Besides, assume that the mechanism of the noise formation is a set of structured knowledge in the form of Associative Corruption (AC) rules, which are similar to the first order rules defined by (Flach and Lachiche 2001), and also resemble the form of association rules (Agrawal, Imielinski, and Swami 1993). The objective of this study is two-fold: (1) discovering noise patterns by applying association rule mining, and (2) correcting the noise so as to construct a robust learner for inductive learning.

Before providing the definitions of several concepts for this study, we give some notations as follows:

- A : a set of feature attributes of D_{obs} , $A = \{A_1, A_2 \dots, A_m\}$;
- C : the class attribute of D_{obs} ;
- V : the value space of the corresponding attributes. $V = \{V_1, V_2 \dots, V_m, V_C\}$, where V_i corresponds to A_i , $A_i \in (A \cup \{C\})$;
- H : a 2-tuple structure $\langle A_i, v_i \rangle$, where
 - $A_i \in (A \cup \{C\})$, A_i is called the *Head* of H ;
 - $v_i \in V_i$, v_i is the value of *Head*, V_i corresponds to A_i .
- T : a 2-tuple structure $\langle p, v \rangle$, where
 - $p = \langle A_i, v_i \rangle$ is an instance conforming to the structure H , and A_i is called the *Head* of T ;
 - $v \in V_i$, v is the modified value of the *Head*, and V_i corresponds to A_i ;
 - $v_i \neq v$.

Facilitated with the above notations, we define several concepts on AC rules as follows:

Definition 5.1 [Associative Corruption Rule] An Associative Corruption (AC) Rule is an implication of the form $P \Rightarrow Q$, where $P = p_1 \& p_2 \& \dots \& p_k$, and each predicate p_i is represented as a structure H . There exists a p , which is a predicate of P , such that $Q = \langle p, v \rangle$ being a structure T .

With this definition, we say that D_{c1} becomes D_{obs1} after D_{c1} is corrupted by a set of AC rules. One example of an AC rule r could be: **IF** $A_1 = 0 \& A_2 = 1 \& A_3 = 1$ **THEN** $A_3 = 1$ is changed to $A_3 = 2$. In other words, $P \Rightarrow Q$, where $P = \langle A_1, 0 \rangle \& \langle A_2, 1 \rangle \& \langle A_3, 1 \rangle$ and $Q = \langle \langle A_3, 1 \rangle, 2 \rangle$.

Definition 5.2 [The Reverse of an AC Rule] For an AC rule $P \Rightarrow Q$, if we denote $Q = \langle p, v \rangle$ where $p = \langle A_i, v_i \rangle$, we switch the positions of v and v_i to acquire $Q' = \langle p', v' \rangle$, where $p' = \langle A_i, v \rangle$ and $v' = v_i$. By replacing the predicate p with p' , and with other predicates unchanged in P , we get P' . As a result, $P' \Rightarrow Q'$ is called the reverse of rule $P \Rightarrow Q$.

For example, the reverse of AC rule r in the previous example would be $P' \Rightarrow Q'$, where $P' = \langle A_1, 0 \rangle \& \langle A_2, 1 \rangle \& \langle A_3, 2 \rangle$ and $Q' = \langle \langle A_3, 2 \rangle, 1 \rangle$.

Definition 5.3 [Probabilistic AC Rule] A Probabilistic AC Rule is an AC rule r with parameter $\lambda \in (0, 1]$. It indicates that r is valid with probability $100\% * \lambda$.

5.2 Method

Intuitively, given the same learning method, the prediction ability of a learner is decided by the quality of the training data. At this point, the datasets we have include D_{obs} , D_{obs1} , D_{obs2} , and D_{c1} . If we could find out the set of AC rules R that corrupts D_{c1} , we

may be able to propose a method to correct D_{obs2} into D_{cor2} . Eventually, we will use $D_{c1cor2} = D_{c1} \cup D_{cor2}$ as the training data for the supervised learning.

The idea follows a two-step fashion. Firstly, we propose an algorithm called ACF (Associative Corruption Forward) to learn the noise formation mechanism from D_{c1} to D_{obs1} . Because of the characteristics of AC rules, ACF builds a learner for each corrupted attribute by using other noise-free attributes as feature attributes. Eventually, we will obtain a set of classification rules, which are easily converted to an AC rule set R_1 . The ideal situation would be $R_1 = R$. Secondly, we propose an algorithm called ACB (Associative Corruption Backward) that corrects D_{obs2} . Initially ACB generates R_2 , in which the AC rules are the corresponding reverse of the rules in R_1 (refer to Definition 5.2 in Section 5.1). It then employs the distribution of attribute values in D_{c1} to facilitate the correcting process, which is implemented by learning a set of Naive Bayes classifiers for each corrupted attribute based on D_{c1} . When ACB corrects an instance in D_{obs2} by using R_2 , it will be guided with those learned Naive Bayes classifiers. In the next two subsections, we will describe the implementation details of algorithms ACF and ACB.

5.2.1 Algorithm ACF

Algorithm ACF is used to infer the set of AC rules R_1 that corrupts D_{c1} . We employ the method of classification rule induction for this problem. The idea of the method is as follows. With D_{c1} being taken as the base dataset, firstly, we add one of the noise corrupted attributes in D_{obs1} to D_{c1} , denoted as A_i^* . The original attribute A_i in D_{c1} is the noise-free counterpart of A_i^* . Secondly, attribute A_i^* in D_{c1} is set as the class attribute. Thirdly, We derive a set of classification rules in D_{c1} for A_i^* , and they can be easily represented as a set of AC rules R_1 . Fourthly, we perform optimization on R_1 . Steps 1 through 4 are performed iteratively for all noise corrupted attributes in D_{obs1} , and R_1 is expanded along with the iterations. The implementation details are described in Table 5.1, in which lines 4-5 implement step 1; line 6 implements step 2; lines 7-10 implement step 3; and line 11

implements step 4.

The optimization procedure could be either simple or sophisticated according to the constraints set on AC rules. In our implementation depicted in Table 5.2, this optimization procedure is effective only to AC rules that satisfy the constraints set in the beginning of Section 4.2. For probabilistic AC rules with $\lambda = 1$, the rules with no attribute value changes will be removed, which is implemented on lines 1-4 in Table 5.2. For probabilistic AC rules with $\lambda \in (0, 1)$, additional work has to be done as shown on lines 5-10. When more than one rule have the same right hand side, we merge them into one rule r such that the left hand side of r involves the biggest common set of predicates in the original rules.

5.2.2 Algorithm ACB

Algorithm ACB (Associative Corruption Backward) is used for noise correction. For an AC rule $P \Rightarrow Q$ in R_1 , by performing the reverse operation as shown in Definition 2 to all AC rules in R_1 , ACB gets the set of correction rules R_2 .

It is obvious that even if $R_1 = R$, which is the set of exact AC rules that corrupt D_{c1} , performing R_2 on D_{obs1} does not guarantee to get D_{c1} , since it is not a strict one to one mapping. Acknowledged with this fact, ACB applies the information about the attribute distribution in D_{c1} to assist the correction process. ACB builds a Naive Bayes learner based on D_{c1} for each noise corrupted attribute. When it attempts to correct an attribute value by a correction rule in R_2 , ACB uses the corresponding Naive Bayes learner to perform a classification on that attribute. If the correction is favored by the learner, the correction will be performed; otherwise the attribute value is kept unchanged. The algorithm for building the Naive Bayes learners and the noise correction process ACB are illustrated in Table 5.3 and Table 5.4, respectively.

Table 5.1: Algorithm ACF (Associative Corruption Forward)

Input: D_{c1}, D_{obs1}

Output: A set of rules R_1 that describe the modification from D_{c1} to D_{obs1}

ACF(D_1, D_2)

- 1 Let $n = \#$ of noise corrupted attributes
 - 2 Let $ruleSet = null$
 - 3 **For** $k \leftarrow 1$ to n
 - 4 Let the attribute A_i be the corrupted attribute indicated in Rule(k)
 - 5 Add the values of A_i for all instances in D_2 to D_1 , and rename this new attribute in D_1 as A_i^*
 - 6 Set A_i^* as the class attribute in D_1
 - 7 In D_1 remove all corrupted attributes except A_i
 - 8 **For** $j \leftarrow 1$ to $\#$ of different values of attribute A_i in D_1
 - 9 Let D_{v_j} be the dataset that only contains instances with $A_i =$ its j^{th} value in D_1
 - 10 Learn a set of classification rules $rSet$ for attribute A_i^* of D_{v_j}
 - 11 $rSet = optimize(rSet)$
 - 12 Let $ruleSet = ruleSet \cup rSet$
 - 13 **End For**
 - 14 **End For**
 - 15 return $ruleSet$
-

Table 5.2: Optimization

Input: a set of AC rules $rSet$

Output: optimized $rSet$

Optimization($rSet$)

- 1 **For** $k \leftarrow 1$ to # of rules in $rSet$
 - 2 Let rule $P \leftarrow Q$ be the k^{th} rule in $rSet$, where $Q = \langle p, v \rangle$ and $p = \langle A_i, v_i \rangle$
 - 3 **If** ($v == v_i$), **Then** delete this rule from $rSet$
 - 4 **End For**
 - 5 **For** $k \leftarrow 1$ to # of rules in $rSet$
 - 6 Let rule $P \Rightarrow Q$ be the k^{th} rule in $rSet$
 - 7 **If** \exists other rules $P' \Rightarrow Q'$ in $rSet$, such that $Q = Q'$
 - 8 **Then** combine them as one rule
 - 9 **End For**
 - 10 return $ruleSet$
-

Table 5.3: Building Naive Bayes Learners

Input: D_{c1}, D_{obs1}

Output: Naive Bayes learners for every corrupted attribute in D_{c1}

LearnNBs(D_1, D_2)

```
1  Let  $n = \#$  of noise corrupted attributes
2  Let attriIndex store the corrupted attribute indexes
3  Let  $BLs = null$ 
4  For  $k \leftarrow 1$  to  $n$ 
5    Let  $i = attriIndex[k]$ 
6    Set attribute  $A_i$  as the class attribute in  $D_1$ 
7    In  $D_1$  remove all corrupted attributes except  $A_i$ 
8    In  $D_1$  learn a Naive Bayes learner  $L$  for  $A_i$ 
9    Let  $BLs = BLs \cup L$ 
10 End For
11 return  $BLs$ 
```

Table 5.4: Algorithm ACB (Associative Corruption Backward)

Input: $ruleSet, D_{c1}, D_{obs1}, D_{obs2}$

Output: the corrected dataset D_{cor2}

ACB($ruleSet, D_1, D_2, D_3$)

```
1  Let  $corRuleSet = reverse(ruleSet)$ 
2  Let  $n = \#$  of noise corrupted attributes
3  Let  $BLs = LearnNBs(D_1, D_2)$ 
4  For  $k \leftarrow 1$  to  $n$ 
5    Let the attribute  $A_i =$  the corrupted attribute indicated in  $corRuleSet(k)$ 
6    Set  $A_i$  as the class attribute in  $D_3$ 
7    For  $j \leftarrow 1$  to  $\#$  of instances in  $D_3$ 
8      If instance( $j$ ) SATISFY  $corRuleSet(k)$ 
9        Let  $V_{orig} =$  the original value of  $A_i$ 
10       Let  $V_{modi} =$  the modified value of  $A_i$ 
11       If  $p(V_{modi}) > p(V_{orig})$  by applying  $BLs(i)$ 
12         correct the value of  $A_i$  as  $V_{modi}$ 
13       End If
14     End If
15   End For
16 End For
17 return  $D_3$ 
```

5.3 Experiments

The objective of our experiments is to verify whether our noise correction procedure could produce a higher quality dataset D_{cor2} in terms of supervised learning. The evaluation procedure is as follows. We build five learning models M_1 , M_2 , M_3 , M_4 , and M_0 on training dataset D_{obs} , D_{c1} , $D_{c1obs2} = D_{c1} \cup D_{obs2}$, $D_{c1cor2} = D_{c1} \cup D_{cor2}$, and D_{clean} , respectively. Among these training datasets, D_{clean} is the originally clean dataset, which is not available in the real situation, and model M_0 serves as the benchmark model. We use the prediction accuracy on the reserved clean dataset D_{test} of M_0 through M_4 to evaluate the data quality. Two learning algorithms, C4.5 classification tree and Bagging with C4.5 as the base learners, are used for the evaluation.

We perform the experiments with two kinds of AC rules: probabilistic AC rules with $\lambda = 1$ and $\lambda = 0.8$.

5.3.1 Experiment Settings

We evaluate the system performances on datasets collected from the UCI database repository and report the results from eight benchmark datasets (Hettich and Bay 1999). In order to evaluate the performance of the proposed method, we obtain dataset D_{obs} by corrupting a dataset D_{clean} with a set of manually generated AC rules R . Given a real world dataset D_{clean} from the UCI database repository, we first separate D_{clean} into two parts: a dataset D_{base} and the corresponding testing set D_{test} . Then we corrupt D_{base} by R , which is a set of artificial AC rules. After noise corruption, we denote the noise-corrupted dataset by D_{obs} .

For every dataset D_{clean} , the performances of M_1 , M_2 , M_3 and M_4 are evaluated by using 10 times 10-fold cross-validation. For each repetition, dataset D_{clean} is randomly split into 10 equal-sized subsets with the original class distribution preserved. Each of the 10 subsets is set aside for testing purpose once, denoted as D_{test} , and the results are averaged over 10 trials.

5.3.2 Experimental Results

In the set of AC rules R that corrupts the original clean dataset, more than one AC rule are allowed. However, our current implementation forces some restrictions to R . These restrictions include:

1. Every rule in R is an AC rule;
2. For any two rules in R , the right hand side of them differs from each other;
3. If $P \Rightarrow Q \in R$, where $Q = \langle p, v \rangle$, then predicate p does not exist in both the left and right hand sides of any other rules in R .

The basic information about the datasets we use for the experiments is shown in Figure 5.1, in which “# inst” indicates the number of instances in the dataset, “#attri” indicates the number of attributes in the dataset, “AC rules” indicate the set of artificial AC rules R that corrupts the corresponding originally noise-free D_{clean} , “error rate” indicates the percentage of errors that occur in D_{obs} , and “ $\#D_{obs}/\#D_{c1}$ ” indicates the ratio of the size of D_{obs} and the size of D_{c1} . For example, the rule “IF ‘4 2’ THEN ‘5 2 1’ ” tells that if the value of the 4th attribute is the 2nd possible value of this attribute, and the value of the 5th attribute is the 2nd possible value of this attribute, then the value of the 5th attribute is changed to the 1st possible value of this attribute. If “-P 80” is added at the end of the rule above, it tells that if the value of the 4th attribute is the 2nd possible value of this attribute, and the value of the 5th attribute is the 2nd possible value of this attribute, then the value of the 5th attribute is changed to the 1st possible value of this attribute with probability 80%.

Figure 5.2 shows the comparative results of five models M_0 through M_4 . M_0 is a benchmark learner built on noise-free data D_{clean} . The unit of the results is percentage. The bold faced cells emphasize the situation when M_0 performs best over the other four models. The grey shaded cells imply the best performance of M_1 through M_4 in the same trial. For each dataset, there are four rows of results. The first two rows show the results of

C4.5, and the next two rows show the results of Bagging with C4.5 as the base learner. The row started with “cv” indicates that the results are obtained by performing cross-validation on the corresponding training data. The row started with “test” indicates the results are obtained by testing on a separate clean testing dataset D_{test} .

The results show that, without considering M_0 , M_4 is the best model, although it does not always beat others. In other words, D_{c1cor2} is the best choice as the training data for inductive learning. As a result, it supports the claim that our method for noise correction is effective. If we take M_0 into account, interestingly, when M_0 performs the best of the five models in all four trials for a single dataset, M_4 always outperforms M_1 through M_3 . According to these experiments, in general, the prediction accuracy of the cross-validation result is positively correlated with that of the results on separate testing data. In the real situation, the testing dataset may not be available for evaluation, so that we have to evaluate the robustness of learners based on the cross-validation results. However, is it appropriate that we decide the best model by simply evaluating the cross-validation results? The answer to this question would be negative. Several more factors should be taken into consideration, besides the cross-validation accuracy. Firstly, the error rate γ in D_{obs} should be an indicator that tells about the noise level. In our experiments γ varies from 5% to 25%, except one dataset “audio” whose error rate $\gamma = 68.8\%$. If the error rate is too high, the overall concept of the dataset might be corrupted. In this case, even if the model M_1 of “audio” performs the best of all models it is still suspicious. Secondly, the ratio $\tau = \#D_{obs}/\#D_{c1}$ tells about how likely D_{c1} reflects the true data distributions. If a dataset includes a lot of redundant information, maybe 1/10 of it is sufficient for the supervised learning; however, in the other extreme, if a dataset is far from complete, any subset would not be sufficient for the inductive learning. In our experiments, we set $\tau = 3$ for six out of eight datasets. The reason why $\tau = 10$ for the two datasets is that, M_2 would be the best among M_1 through M_4 if we set $\tau = 3$ for these two datasets. In that case, no noise correction is necessary, since D_{c1} could be used as the training data directly.

dataset	# inst	# attri	AC rules	error rate	$\#D_{obs}/\#D_{c1}$
car	1728	7	IF "4 2" THEN "5 2 1"	0.114	3
monks3	432	7	IF "6 1" THEN "1 1 2"	0.243	3
balance	630	5	IF "4 0" THEN "0 4 1"	0.140	3
nursery	12960	9	IF "1 3" THEN "7 2 1"	0.067	10
splice	3190	61	IF "34 0" THEN "29 2 1" -P 80	0.083	3
tictactoe	958	10	IF "9 0" THEN "4 2 1" -P 80	0.095	3
krvskp	3196	37	IF "36 0" THEN "20 1 0" -P 80	0.138	10
audio	226	70	IF "2 0" THEN "65 0 3" -P 80	0.688	3

Figure 5.1: Information on the Datasets for Experiments

For example, the rule “IF ‘4 2’ THEN ‘5 2 1’ ” tells that if the value of the 4th attribute is the 2nd possible value of this attribute, and the value of the 5th attribute is the 2nd possible value of this attribute, then the value of the 5th attribute is changed to the 1st possible value of this attribute. For more details, please refer to Section 5.3.2.

In the experimental results shown in Figure 5.2, the first four datasets are corrupted with probabilistic AC rules with $\lambda = 1$, and the remaining four are corrupted with probabilistic AC rules with $\lambda = 0.8$. For each dataset, the AC rule set R contains only one AC rule. Our method would be applicable to an AC rule set R with more than one rule as well, as long as R satisfies the constraints mentioned in the beginning of this section.

5.3.3 Discussions

In this study, the input information includes the noisy dataset D_{obs} , the clean dataset D_{c1} , which corresponds to a subset of D_{obs} , and the AC rules that cause the noise. The goal of our study is to build a robust learner for supervised learning. Our method follows the way of data preprocessing, which attempts to build a robust learner by improving the data quality. Therefore, the key issue of this problem is how to correct the remaining part of D_{obs} that is noise infected, by considering all the input information. The thinking pattern of the solution we have provided is to answer the following questions sequentially. Firstly, how is the noise formalized? Secondly, if we reverse the noise formation process, can we get the original clean data? Thirdly, what is the additional information needed to facilitate the noise correction? In our method, algorithm ACF answers the first question. Particularly, ACF is

			M_1	M_2	M_3	M_4	M_0
			D_{obs}	D_{c1}	D_{c1obs2}	D_{c1cor2}	D_{clean}
car	C4.5	cv	88.83	85.32	90.01	91.63	92.93
		test	88.94	86.98	92.07	93.17	94.33
	B	cv	88.89	87.48	90.14	92.49	93.46
		test	88.89	88.42	91.72	92.88	94.15
monks3	C4.5	cv	59.95	99.62	73.27	99.95	100.00
		test	78.70	99.30	87.02	100.00	100.00
	B	cv	60.26	99.54	72.94	99.97	100.00
		test	80.33	99.30	88.19	100.00	100.00
balance	C4.5	cv	64.89	63.59	68.22	69.82	68.67
		test	55.71	62.54	69.05	68.73	68.57
	B	cv	68.43	69.78	70.65	74.76	74.33
		test	60.48	71.75	72.86	73.81	75.08
nursery	C4.5	cv	86.75	90.06	87.63	98.10	98.15
		test	92.28	90.56	92.99	98.87	98.87
	B	cv	86.50	91.59	87.47	98.20	98.20
		test	92.25	91.63	92.71	98.64	98.73
splice	C4.5	cv	90.49	88.84	90.73	91.72	91.94
		test	90.85	89.06	91.69	92.48	92.76
	B	cv	93.60	91.93	93.68	94.12	94.22
		test	93.70	92.07	94.04	94.26	94.26
tictactoe	C4.5	cv	85.13	78.33	84.62	85.37	85.59
		test	84.04	76.92	86.85	86.54	86.75
	B	cv	92.17	83.58	92.13	92.21	93.20
		test	90.71	82.46	93.53	93.42	94.57
krvskp	C4.5	cv	91.53	94.86	92.18	94.89	99.25
		test	95.62	95.71	96.06	96.68	99.47
	B	cv	92.37	95.59	92.97	95.65	99.33
		test	96.50	96.03	96.93	97.78	99.50
audio	C4.5	cv	75.84	63.55	75.51	76.06	76.23
		test	76.54	65.91	75.65	78.28	78.22
	B	cv	80.21	67.45	80.02	79.50	80.35
		test	80.51	69.47	80.08	79.21	80.47

Figure 5.2: Comparative Results of Five Learning Models

$M_1, M_2, M_3, M_4,$ and M_0 are learned based on dataset $D_{obs}, D_{c1}, D_{c1obs2}, D_{c1cor2},$ and $D_{clean},$ respectively. M_0 is a benchmark learner built on noise-free data $D_{clean}.$ The bold faced cells emphasize the situation when M_0 performs best over other four models. The grey shaded cells imply the best performance of M_1 through M_4 in the same trial. “cv” is a short notation of “cross-validation”, which indicates that the results are obtained by performing cross-validation on the corresponding training data. “test” indicates the results are obtained by performing testing on a separate dataset.

able to infer the AC rules that corrupt the original dataset with 100% accuracy. Algorithm ACB answers the other two questions. If we reverse the noise formation process, there is

no guarantee to get the original clean data. Therefore, ACB uses the data distribution of the noise-free dataset D_{c1} as the guidance when performing the noise correction.

According to our experiments, the strategy to construct the best robustness model in the real situation would be selecting the model with the highest cross-validation accuracy among M_1 through M_4 , with the consideration of the error rate of D_{obs} , and the ratio of the sizes between D_{obs} and D_{c1} . Besides, we should be aware that the data quality is always associated with the study objective. For example, in this study, we only need to handle the errors that affect the inductive learning. Sometimes a dataset contains errors on certain feature attributes, but if these errors do not have significant impact on the target concept learning we could just ignore them.

Although our experimental results show that our method is very effective, we have to note that our method models a specific type of noise. There are at least three more questions we need to think about. Firstly, if the size of D_{c1} is very small, the Naive Bayes learners built on it would be less accurate so as to affect ACB's noise correction. Secondly, if we remove the constraints on AC rules as stated in Section 4.2, we will face a more complicated situation that might not be handled well with the current strategy. Thirdly, we should consider how to adjust our method to adapt to datasets with errors on continuous feature attributes.

5.4 Chapter Summary

In this chapter, we have modeled a particular type of associative noise with Associative Corruption (AC) rules, proposed a association based method to learn the noise patterns, and suggested a possible strategy to correct the noise for inductive learning. AC rules are defined to simulate a common noise formation process in real-world data, in which the occurrence of an error on one attribute is dependent on several other attribute values. In this problem, we have assumed that we are given an observed noisy dataset D_{obs} where $D_{obs} = D_{obs1} \cup D_{obs2}$ and a noise-free dataset D_{c1} , which is D_{obs1} after inspection and

correction. In order to propose a method to correct D_{obs2} , we have designed a two-step method that include algorithms ACF and ACB. Firstly, ACF learns the AC rules that corrupt D_{c1} by constructing classification rules for the corrupted attributes. Secondly, ACB corrects D_{obs2} by taking into account the AC rules derived from the previous step and the attribute value distribution in D_{c1} . Eventually, D_{cor2} is the corrected dataset of D_{obs2} . In our experiments, we show that our method could infer the noise formation mechanism accurately and perform a noise correction process appropriately, so as to enhance the quality of the original dataset. By using two learning methods C4.5 and Bagging with C4.5 as base learners, we show that the quality of $D_{c1cor2} = D_{c1} \cup D_{cor2}$ is better than D_{c1} , D_{obs} , or $D_{c1} \cup D_{obs2}$, taking classification accuracy as the evaluation measure. In a more general situation, the quality of the candidate datasets could be evaluated by comparing the performance of the cross-validation accuracy, with consideration of other factors such as the error rate of D_{obs} and the ratio between the sizes of D_{obs1} and D_{c1} .

Chapter 6

Concluding Remarks

This chapter summarizes the problems we have investigated in the thesis (Section 6.1), and then discusses promising directions for future research in the domain of noise tolerant data mining (Section 6.2).

6.1 Summary of Thesis

Data mining research is dedicated to exploring interesting and actionable knowledge from data. The most important and essential foundation in a data mining task is the source data, as the data quality issue is showing direct effect on the robustness of the knowledge induced. Therefore, a challenging problem, how to learn from noisy data sources has been a focus of research in the data mining community for years. Traditionally, it relies on data preprocessing techniques to improve the data quality, assisted with optimizing the structure of the learning model for smoothing out the impact of noise. During this process, data preprocessing and model construction are two isolated, sequentially performed steps. The essential idea of this scheme can be summarized as “purging the noisy data before mining”.

One inherent problem suffered from this strategy is that, there are usually no objective metrics to assess the data quality, or the effectiveness of a data cleansing manipulation. As

a result, this strategy is subject to information loss. Secondly, such a sequential problem solving pattern does not have good expendability when the noise pattern becomes complex, because we believe that learning from the data that contains accidental errors is a rather different problem from learning from deliberately falsified data or data that accurately describes a systematic pattern. In Chapter 1 we have pointed out the major limitations of current data preprocessing techniques. And this discussion was later expanded to the literature review on noise handling methods in Chapter 2.

Data quality is a term that has often been referred in information related references. In Chapter 2, we analyzed the metrics of data quality from three categories: task-independent, task-dependent, and task-specific metrics from the point of view of general data mining applications. We also clarified some data quality metrics of great concern in data mining research. The concept “noise”, and its connections to “outlier” and “fraud” are discussed in this chapter as well. Having presented the inefficiency of the current sequential noise handling strategy, we proposed a noise Modeling, Diagnosis, and Utilization (MDU) framework in Chapter 2. As suggested by its name, this framework consists of three modules, which explores and models the noise pattern, develops methods to locate and correct the noisy data entries, and designs algorithms with the aid of the modeled noise pattern, respectively. The ultimate goal of this framework is to build up a connection between the noisy data and the learning algorithm, so that the noise information could be appropriately modeled to assist the actual data mining process. This framework serves as a basis of the targeted learning problems in the remaining chapters of the thesis.

In Chapter 3 through Chapter 5, we explored two learning problems. The first problem focused more on noise diagnosis and robust model construction. We presented an effective strategy – a combination of noise handling and classifier ensembling, to build a robust learning algorithm on noisy data. Guided with this idea, we designed two methods – Aggressive Classifier Ensembling (ACE) and Corrective Classification (C2). The essential idea is to take the advantage of both traditional noise detection / correction and the classifier

committee to create multiple copies of corrected data, so as to construct a corrective classifier ensemble. In Chapter 3, we introduced the theoretical background of classifier ensembling, and extensively analyzed the important factors that may influence the performance of a classifier ensemble. In Chapter 4, we proposed the ACE and C2 algorithms, including the algorithm descriptions, implementation details, and the analysis of the experimental results.

The second problem we have investigated related to systematic noise modeling. In Chapter 5, we proposed a learning problem, where the source data was assumed to have been infected with associative noise. In other words, the errors in one attribute value is associated with several other attribute values. We modeled the noise with a set of associative corruption (AC) rules, learned the noise pattern by building classification trees, and proposed an effective method to correct the data by the aid of the inferred noise pattern.

The major contributions of this thesis to the research community of data mining were presented in Section 1.2 from three aspects. In addition, a portion of the materials from Chapter 3 and Chapter 4 have been published in the Proceedings of the 2005 IEEE International Conference on Tools with Artificial Intelligence, with the title “ACE: An Aggressive Classifier Ensemble with Error Detection, Correction and Cleansing” (honored the best paper award) (Zhang, Zhu, Wu, and Bond 2005), and in the Proceedings of the 2006 IEEE International Conference on Data Mining, with the title “Corrective Classification: Classifier Ensembling with Corrective and Diverse Base Learners” (Zhang, Zhu, and Wu 2006). The materials from Chapter 5 have been published in the Proceedings of the 2007 IEEE International Conference on Data Mining, with the title “Noise Modeling with Associative Corruption Rules” (Zhang and Wu 2007).

6.2 Future Work

This thesis attempts to provide a high-level noise tolerant data mining framework for the problem of learning from noisy data sources. Some illuminating solutions to relevant issues in learning from noisy data have been proposed. However, there are still many open

problems to consider. In this section, we provide some future work that may benefit from a more extensive investigation.

- **Impact-Sensitive Instance Ranking.** Ranking provides users an option that puts limited resources on handling a subset of noisy instances, so as to maximize the benefit to the learning problem. To make this happen, effective data utility measures, such as the impact and cost matrices must be produced. For example, Zhu *et.al.* proposed an information-gain ratio based method to rank the potentially noisy instances by their impact on the classification process (Zhu, Wu, and Yang 2004). To develop an practically viable ranking mechanism, it needs an extensive study on the underlying data and the study objective.

- **Noise Modeling and Reasoning.** Along with the continuing development of new technologies, we are in an era where the data volume keeps increasing, data quality becomes less controllable, and noise patterns become more versatile and complex. It makes modeling data errors and pinpointing error sources challenging tasks. Although we have proposed possible strategies to model noisy data as discussed in Chapter 6, there is a long way to go in order to develop practically effective modeling mechanisms for a specified problem, or for a category of problems that share similar characteristics. The study in this regard will directly benefit real applications such as the identity fraud detection in e-commerce or telecommunication, and the falsified data detection in crime data mining or survey research.

- **Mining with Noise Knowledge.** To alleviate noise impacts and avoid information loss, we need an effective system design which is capable of boosting from noisy information sources for enhanced mining results. This includes developing noise-immunization algorithms and approaches which may utilize the appropriately modeled noise patterns to benefit the mining process. Most importantly, the framework should accommodate the task-independent algorithms and the task-specific information in a

way that two parts are comparatively separated, but logically well connected, so that the task-independent algorithms could be reusable in similar problems.

Bibliography

- Agrawal, R., T. Imielinski, and A. N. Swami (1993). Mining association rules between sets of items in large databases. In *The 1993 ACM SIGMOD International Conference on Management of Data*, Washington DC, USA, pp. 207–216.
- Agrawal, R. and R. Srikant (2000). Privacy-preserving data mining. In *the ACM SIGMOD Conference on Management of Data*, Dallas, TX, pp. 439–450.
- Aha, D. W. and R. L. Bankert (1995). A comparative evaluation of sequential feature selection algorithms. In *the Fifth International Workshop on Artificial Intelligence and Statistics*, pp. 1–7.
- Aha, D. W., D. Kibler, and M. K. Albert (1991). Instance-based learning algorithm. *Machine Learning* 6(1), 37–66.
- Al-Marzouki, S., S. Evans, T. Marshall, and I. Roberts (2005). Are these data real? statistical methods for the detection of data fabrication in clinical trials. *British Medical Journal* 331(7511), 267–270.
- Anscombe, F. J. (1960). Rejection of outliers. *Technometrics* (2), 23–147.
- Barber, B. M., T. Odean, and N. Zhu (2004). Systematic noise. In *AFA*, San Diego.
- Breiman, L. (1996). Bagging predictors. *Machine Learning* 24(2), 123–140.
- Breiman, L. (2000). Randomizing outputs to increase prediction accuracy. *Machine Learning* 40(3), 229–242.
- Breiman, L. (2001). Random forest. *Machine Learning* 45(1), 5–32.

- Brodley, C. E. and M. A. Friedl (1999). Identifying mislabeled training data. *Journal of Artificial Intelligence Research* 11, 131–167.
- Chen, H., W. Chung, J. J. Xu, G. Wang, Y. Qin, and M. Chau (2004). Crime data mining: A general framework and some examples. *IEEE Computer Society* 37(4), 50–56.
- Clark, P. and T. Niblett (1989). The cn2 induction algorithm. *Machine Learning* 3, 261–283.
- Collins, C. M. (1998). Ensuring data quality in the pharmaceutical industry. *Journal of American Health Information Management Association* 69(6), 34–39.
- Cunningham, P. and J. Carney (2000). Diversity versus quality in classification ensembles based on feature selection. In *Proceedings of the Eleventh European Conference on Machine Learning (ECML-2000)*, pp. 109–116.
- Dietterich, T. G. (2000a). Ensemble methods in machine learning. *Lecture Notes in Computer Science, Proceedings of the First International Workshop on Multiple Classifier Systems 1857*, 1–15.
- Dietterich, T. G. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning* 40, 139–157.
- Dietterich, T. G. and G. Bakiri (1991). Error-correcting output codes: a general method for improving multiclass inductive learning programs. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-1991)*, Menlo Park, CA, pp. 572–577.
- Domingos, P. (2000). A unified bias-variance decomposition and its applications. pp. 231–238.
- Duda, R. O., P. E. Hart, and D. G. Stork (2000). *Pattern Classification*. Wiley-Interscience.

- Dunnett, C. W. (1955). A multiple comparison procedure for comparing several treatments with a control. *Journal of The American Statistical Association* 50, 1096–1121.
- Fan, W., S. J. Stolfo, J. Zhang, and P. K. Chan (1999). Adacost: misclassification cost-sensitive boosting. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-1999)*, pp. 97–105.
- Fawcett, T. and F. Provost (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1–28.
- Flach, P. A. and N. Lachiche (2001). Confirmation-guided discovery of first-order rules with tertius. *Machine Learning* (42), 61–95.
- Freund, Y. and R. E. Schapire (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-1996)*, 148–156.
- Freund, Y. and R. E. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* (55), 119–139.
- Freund, Y. and R. E. Schapire (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence* (14), 771–780.
- Friedman, J. (1997). On bias, variance, 0/1 loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery* (1), 55–77.
- Gamberger, D., N. Lavrač, and C. Grošelj (1999). Experiments with noise filtering in a medical domain. In *Proceedings of the sixteenth International Conference on Machine Learning (ICML-1999)*, pp. 143–151. Morgan Kaufmann, San Francisco, CA.
- Geman, S., E. Bienenstock, and R. Doursat (1992). Neural networks and the bias/variance dilemma. *Neural Computation* (4), 1–58.
- Gertz, M., M. T. Ozsu, G. Saake, and K.-U. Sattler (2004). Report on the dagstuhl seminar "data quality on the web". *SIGMOD Record* 33(1).

- Han, J. and M. Kamber (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.
- Hand, D. J. and G. Blunt (2001). Prospecting for gems in credit card data. *IMA Journal of Management Mathematics* (12), 173–200.
- Hettich, S. and S. D. Bay (1999). The uci kdd archive [<http://kdd.ics.uci.edu>] irvine, ca: University of california, department of information and computer science.
- Hickey, R. J. (1996). Noise modelling and evaluating learning from examples. *Artificial Intelligence* 82(1-2), 157–179.
- Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(8), 832–844.
- Huber, P. (1981). *Robust Statistics*. Wiley, New York.
- John, G. H. (1995). Robust decision trees: Removing outliers from databases. In *the First International Conference on Knowledge Discovery and Data Mining*, pp. 174–179.
- Johnson, T. P., V. Parker, and C. Clements (2001). Detection and prevention of data falsification in survey research. *Survey Research* 32(3), 1–15.
- Joshi, M. V., V. Kumar, and R. C. Agarwal (2001). Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Proceedings of the First International Conference on Data Mining (ICDM-2001)*, pp. 257–264.
- Judd, C. M. and G. H. McClelland (1989). *Data Analysis: A Model Comparison Approach*. New York: Harcourt Brace Jovanovich.
- Kahn, B. K., D. M. Strong, and R. Y. Wang (2002). Information quality benchmarks: Product and service performance. *Communications of the ACM* 45, 184–192.
- Kolen, J. F. and J. B. Pollack (1991). Back propagation is sensitive to initial conditions. *Advances in Neural Information Processing Systems* 3, 860–867.
- Kuncheva, L. I. and C. J. Whitaker (2003, May). Measures of diversity in classifier en-

- sembles and their relationship with the ensemble accuracy. *Machine Learning* 51(2), 181–207.
- Maclin, R. and J. W. Shavlik (1995). Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-1995)*, Montreal, Canada, pp. 524–530.
- Maletic, J. I. and A. Marcus (2000). Data cleansing: Beyond integrity analysis. *Proceedings of Information Quality*, 200–209.
- Martinez-Munoz, G. and A. Suarez (2005). Switching class labels to generate classification ensembles. *Pattern Recognition* 38(10), 1483–1494.
- Melville, P. and R. J. Mooney (2003). Constructing diverse classifier ensembles using artificial training examples. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003)*, 505–510.
- Melville, P. and R. J. Mooney (2004). Creating diversity in ensembles using artificial data. *Journal of Information Fusion (Special Issue on Diversity in Multiple Classifier Systems)* 6/1, 99–111.
- Mingers, J. (1989). An empirical comparison of selection measures for decision-tree induction. *Machine Learning* 3(4), 319–342.
- Motro, A. (1989). Integrity = validity + completeness. *ACM Transactions on Database Systems* 14(4), 480–502.
- Opitz, D. (1999). Feature selection for ensembles. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-1999)*, 379–384.
- Opitz, D. and R. Maclin (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research* 11, 169–198.
- Orr, K. (1998). Data quality and systems theory. *Communications of the ACM* 42(2), 66–71.

- Pipino, L. L., Y. W. Lee, and R. Y. Wang (2002). Data quality assessment. *Communications of the ACM*, 211–218.
- Putten, P. V. D. and M. V. Someren (2004). A bias-variance analysis of a real world learning problem: The coil challenge 2000. *Machine Learning* 57(1-2), 177–195.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning* 1(1), 81–106.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning: Book and Software Package*. Morgan Kaufmann.
- Quinlan, J. R. (1996). Bagging, boosting, and c4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-1996)*, 725–730.
- Rousseeuw, P. J. and A. M. Leroy (1987). *Robust Regression and Outlier Detection*. John Wiley.
- Srinivasan, A., S. Muggleton, and M. Bain (1992). Distinguishing exceptions from noise in non-monotonic learning. In S. Muggleton (Ed.), *Proceedings of the Second International Workshop on Inductive Logic Programming*.
- Strong, D. M., Y. W. Lee, and R. Y. Wang (1997). Data quality in context. *Communications of the ACM* 40(5), 103–110.
- Teng, C. M. (1999). Correcting noisy data. *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-1999)*, 239–248.
- Tomek, I. (1976). An experiment with edited nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics* 6(6), 448–452.
- Tsybal, A., M. Pechenizkiy, and P. Cunningham (2005a). Diversity in search strategies for ensemble feature selection. *Information Fusion* 6(1), 83–98.
- Tsybal, A., M. Pechenizkiy, and P. Cunningham (2005b). Sequential genetic search for ensemble feature selection. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2005)*, pp. 877–882.

- Wang, G., H. Chen, and H. Atabakhsh (2004). Automatically detecting deceptive criminal identities. *Communications of the ACM* 47(3), 71–76.
- Wilson, D. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics* 2, 408–421.
- Witten, I. H. and E. Frank (2000). *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann.
- Witten, I. H. and E. Frank (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.
- Zenobi, G. and P. Cunningham (2001). Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, Freiburg, Germany, pp. 576–587.
- Zhang, Y. and X. Wu (2007). Noise modeling with associative corruption rules. In *Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM-2007)*, Omaha NE, the United States, pp. 733–738.
- Zhang, Y., X. Zhu, and X. Wu (2006). Corrective classification: A classifier ensemble with corrective and diverse base learners. *Proceedings of the Sixth IEEE International Conference on Data Mining (ICDM-2006)*, 1199–1204.
- Zhang, Y., X. Zhu, X. Wu, and J. P. Bond (2005). Ace: An aggressive classifier ensemble with error detection, correction and cleansing. *Proceedings of Seventeenth International Conference on Tools with Artificial Intelligence (ICTAI-2005)*, 310–317.
- Zhao, Q. and N. T. (1995). Using qualitative hypotheses to identify inaccurate data. *Journal of Artificial Intelligence Research* (3), 119–145.
- Zhou, Z. and Y. Yu (2005). Ensembling local learners through multimodal perturbation. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics* 35(4), 725–735.

Zhu, X. and X. Wu (2006). Error awareness data mining. In *Proceedings of the IEEE International Conference on Granular Computing (GRC-2006)*, Atlanta.

Zhu, X., X. Wu, and Y. Yang (2004). Error detection and impact-sensitive instance ranking in noisy datasets. *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-2004)*, 378–384.