Terms of Use

ANT COLONY OPTIMIZATION APPROACH FOR STACKING CONFIGURATIONS

CHEN YIJUN

MPHIL

LINGNAN UNIVERSITY

2011

ANT COLONY OPTIMIZATION APPROACH FOR STACKING CONFIGURATIONS

by

CHEN YIJUN

A thesis
submitted in partial fulfillment of the
requirements for the Degree of
Master of Philosophy in Business
(Computing and Decision Sciences)

Lingnan University

2011

ABSTRACT

Ant Colony Optimization Approach for Stacking Configurations

by

Chen Yijun

Master of Philosophy

In data mining, classifiers are generated to predict the class labels of the instances. An ensemble is a decision making system which applies certain strategies to combine the predictions of different classifiers and generate a collective decision. Previous research has empirically and theoretically demonstrated that an ensemble classifier can be more accurate and stable than its component classifiers in most cases. Stacking is a well-known ensemble which adopts a two-level structure: the base-level classifiers to generate predictions and the meta-level classifier to make collective decisions. A consequential problem is: what learning algorithms should be used to generate the base-level and meta-level classifiers in the Stacking configuration? It is not easy to find a suitable configuration for a specific dataset. In some early works, the selection of a meta classifier and its training data are the major concern. Recently, researchers have tried to apply metaheuristic methods to optimize the configuration of the base classifiers and the meta classifier.

Ant Colony Optimization (ACO), which is inspired by the foraging behaviors of real ant colonies, is one of the most popular approaches among the metaheuristics. In this work, we propose a novel ACO-Stacking approach that uses ACO to tackle the Stacking configuration problem. This work is the first to apply ACO to the Stacking configuration problem. Different implementations of the ACO-Stacking approach are developed. The first version identifies the appropriate learning algorithms in generating the base-level classifiers while using a specific algorithm to create the meta-level classifier. The second version simultaneously finds the suitable learning algorithms to create the base-level classifiers and the meta-level classifier. Moreover, we study how different kinds on local information of classifiers will affect the classification results. Several pieces of local information collected from the initial phase of ACO-Stacking are considered, such as the precision, f-measure of each classifier and correlative differences of paired classifiers. A series of experiments are performed to compare the ACO-Stacking approach with other ensembles on a number of datasets of different domains and sizes. The experiments show that the new approach can achieve promising results and gain advantages over other ensembles. The correlative differences of the classifiers could be the best local information in this approach.

Under the agile ACO-Stacking framework, an application to deal with a direct marketing problem is explored. A real world database from a US-based catalog company, containing more than 100,000 customer marketing records, is used in the experiments. The results indicate that our approach can gain more

cumulative response lifts and cumulative profit lifts in the top deciles. In conclusion, it is competitive with some well-known conventional and ensemble data mining methods.

DECLARATION

I declare that this is an original work based primarily on my own research, and I warrant that all citations of previous research, published or unpublished, have been duly acknowledged.

Chen Yijun

Date

Library Rights Statement

In presenting the thesis, *Ant Colony Optimization Approach for Stacking Configurations*, in partial fulfilment of the requirements for a master of philosophy degree at Lingnan University, I agree that the Library shall make it freely available for inspection. I further agree that permission for copying, as provided for by the Copyright Law of Hong Kong, of this thesis for scholarly purposes may be granted by the Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

I hereby grant permission to the Library of Lingnan University to use my thesis for scholarly purposes.

_____

Chen Yijun

_____

Date

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
AS	Ant System
BCO	Bee Colony Optimization
BN	Bayesian Network
CART	Classification and Regression Tree
DEA	Data Envelopment Analysis
DM	Data Mining
DMU	Decision Making Unit
DT	Decision Tree
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolutionary Strategies
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
ILS	Iterated Local Search
KDD	Knowledge Discovery in Database
MDT	Meta Decision Tree
ML	Machine Learning
MLR	Multi-response Linear Regression
MRMT	Multi-Response Model Tree
NN	Neural Network
ODM	Oracle Data Mining
PAC	Probably Approximately Correct
PSO	Particle Swarm Optimization
RAI	Relative Improvement
SA	Simulated Annealing
SSE	Sum of Squared Error
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TS	Tabu Search
TSP	Traveling Salesman Problem

ACKNOWLEDGMENTS

This part should be the most pleasant part in the thesis. As I am writing the acknowledgments, the memories of my enrolment in postgraduate studies, the choice of this topic to work on and all the happiness, troubles, even sadness, during the two years of studies, all spark in my brain, as clear as if they happened just now. Two years is not a long period compared to one's life, however; meeting the right people and doing the right things in the most energetic period can greatly change a person's life. I am especially grateful for the opportunity to begin my M.Phil studies with my supervisor: Dr Wong Man-Leung at Lingnan University. Dr Wong is a gifted scholar who is always full of enthusiasm for research and ready to give me insightful advice on the difficulties I meet in my research. Besides academic support, Dr Wong also gives me suggestion on career development and living in Hong Kong culture.

I should also give my best regards to my colleagues in the Department of Computing and Decision Sciences. I would like to thank Mr Chung Chi-wai, who introduced me to his football team. I share the endless passion of football with the teammates and feel fortunate to be a member of the team. I would also like to thank Ms Eva Cheung, who is warm-hearted and patient in even trivial office affairs.

I also appreciate the care and help I have received from friends in my two years of studies as an M.Phil student. Thanks especially to Hangfei Guo, Yang Guo, Man Jin, Irene Lui, Yongsui Peng, Kailong Wang, Wei Wang ,Alex Wong, Yuanyuan Zhang and Yin Zhou (sequence does not indicate imprortance!).

Finally, I cannot forget my parents and grandparents. All my life, my parents have given me the greatest love and support. The expectations of my parents and grandparents keep me trying my best to succeed.

CHAPTER 1
Introduction

1.1 Overview

In the era of computers, a vast volume of information is created every second. The huge amount of information/data and the lack of powerful analysis tools make it hard for people to turn the massive bulk of data into useful knowledge. For example, upwards of tens of thousands of credit card transactions are processed everyday; however, a few of them are fraudulent. It is costly either to discover the fraudulent transactions manually or to ignore them. Thus stronger analysis tools should be used to help solve this problem. In order to discover the hidden knowledge (fraudulent transactions in the credit card example), Knowledge Discovery in Database (KDD) was developed. KDD is the overall process of converting raw data into useful information. This process contains a series of transformation steps: data cleaning and integration, data selection and transformation, data mining, pattern evaluation and knowledge presentation. Data Mining (DM) is a key process of KDD, which uses some models to automatically discover useful information and to extract nontrivial patterns from massive data repositories (Han and Kamber, 2000; Witten and Frank, 2005). Data mining is the integration of knowledge from statistics, artificial intelligence, machine learning, pattern recognition and database systems.

Generally speaking, data mining includes the following tasks: Association Rule Mining, Classification / Prediction, Clustering, Outlier Detection and Regression (Fayyad et al., 1996; Ngai et al., 2011). Association Rule Mining, or Association analysis, is used to discover the patterns that describe the strong association of items in a dataset. The boundary is blurred between the concepts of classification and predictive models in data mining, so we consider them as one data mining task. Classification utilizes certain learning algorithms to build classifiers on the training set thus to predict the class labels of the instances in

the testing set. Clustering is used to divide objects into conceptually meaningful groups (clusters) with the objects in a cluster being similar to one another but dissimilar to those in other clusters. Outlier Detection is used to detect the data which appears to have different characteristics than the majority. Regression is a statistical methodology which is used to reveal the relationship between independent variables and the dependent variable.

Classification is one of the most frequently used data mining tasks. A number of algorithms are developed to generate classifiers. Decision Trees (DTs), Neural Networks (NNs), and Support Vector Machines (SVMs) are well known among the algorithms. However, it becomes more and more difficult to significantly improve the performance of a single classifier when dealing with some difficult datasets. Thus, there is a growing research interest in combining different classifiers to generate a collective prediction. These combining methods are called ensembles. Ensembles are the decision making systems which apply some strategies to combine different classifiers together to achieve better performance (Dietterich, 2000). Over years of development, many ensemble schemes has been proposed. Among them, Bagging, Boosting and Stacking are three well known and widely used ensembles (Breiman, 1996; Schapire, 1990; Wolpert, 1992; Polikar, 2006). In some previous research, both empirical and theoretical, ensembles are proved to perform more accurately than any single component classifier in most cases (Dietterich, 2000; Polikar, 2006).

1.2 Research Motivation

To generate promising ensembles, two important things should be carefully considered. The first is to introduce enough diversity into the components of an ensemble. The second is to choose a suitable combining method to combine the diverse outputs into a single output (Polikar, 2006). The diversity is the foundation of an ensemble. However, as the diversity increases, the marginal effect

decreases after a certain threshold. The memories and computing cost increase steadily while the performance cannot be improved significantly. For example, the diversity of Bagging Ensemble and Boosting Ensemble are achieved by using the re-sample strategy (Breiman, 1996; Schapire, 1990). The classifiers included in Bagging are trained with the data subsets which are randomly sampled from the original dataset. A majority voting scheme is applied as the combining method to make a collective decision. Boosting uses a weighted re-sample strategy. The weights of instances are initialized equally. If the instances are misclassified, its weight will be increased thus it will be more likely to be chosen in the next training subset. The diversity generating process stops when the errors are too small. The combining scheme of Boosting is a weighted majority voting. The other well-known ensemble, Stacking, does not manipulate the training dataset, rather, it applies a two level structure: the base level and meta level (Wolpert, 1992). In the base level, the multiple classifiers are trained with different learning algorithms. The base classifiers make different errors on the same dataset because their hypotheses and prediction deductions are widely different, thus diversity is introduced. A meta classifier is applied to generate the final prediction. The meta classifier is trained with a learning algorithm using a meta dataset which combines the outputs of the base classifiers and the real class label.

One problem of Stacking is how to find the "appropriate" configuration of the base classifiers and meta classifier to each domain-specific dataset. Such configuration is still a "Black Art" (Wolpert, 1992). Although similar problems exist in Bagging and Boosting, the situation in Stacking is more complicated. The number of base classifiers and their individual performances are closely related to the diversity. In a case where the performances of the classifiers are too similar, the increase in number of classifiers cannot improve the performance significantly. Zhang et al. (Zhang et al., 2006) has demonstrated that "an ensemble of three identical classifiers with 95% accuracy is worse than an ensemble of three classifiers with 67% accuracy and least pairwise correlated error." In another situation,

if the classifiers are somewhat different from each other, then to increase the number of classifiers to improve the performance makes sense. The meta classifier is important to the fusion of the base classifiers. The selection of a suitable meta classifier is essential to a Stacking configuration. The more classifiers, the more difficult to find the optimal configuration. For example, if there are 10 learning algorithms which could be considered to generate the base classifiers, there would be $2^{10} = 1024$ combinations. If all of the ten learning algorithms are appropriate to generate the meta classifier, the total number of Stacking configurations will be $10 * 1024 = 10240$. If the number of learning algorithms increases linearly, the number of total combinations will increase exponentially. Thus an exhaustive search is not practical.

Some scholars have published their research outcomes on determining the configuration of Stacking using different methods. Ting and Witten solved two issues about the type of meta-level classifier and its kinds of input attributes (Ting and Witten, 1999). Džeroski and Ženko introduced Multi-Response Model Trees as the meta-level classifier and claimed better results (Džeroski and Ženko, 2002). Zhu (Zhu, 2010) proposed the Data Envelopment Analysis (DEA) based approach to find the Stacking configurations. Metaheuristic-based approaches which search the ensemble configurations using genetic algorithms (GAs) are studied in several research works (Ledezma et al., 2002; Ordóñez et al., 2008; Ledezma et al., 2009).

In this work, we propose an approach using Ant Colony Optimization (ACO) to optimize the Stacking configurations. ACO is a metaheuristic algorithm which is inspired by the foraging behavior in real ant colonies. Some approaches have recently been proposed to apply ACO in data mining tasks. Parpinelli et al. proposed Ant Miner to extract classification rules (Parpinelli et al., 2002). Some approaches apply ACO in feature subset selection tasks, the readers can refer to (Al-Ani, 2006; Zhang et al., 2010). However, this is the first work that applies

ACO in a Stacking configuration problem.

1.3 Organization of the Thesis

This thesis is organized as follows. In Chapter 1, the overview, the motivation and the organization of the research are given. In Chapter 2, the background of this work will be introduced, including data mining, ensemble approaches and metaheuristic optimization. Some previous works on optimizing the Stacking ensemble are reviewed. In Chapter 3, the detail of our approach is present. The algorithm framework will be introduced. Furthermore the discussion of different local information and the development of different versions of the algorithm will be given. In Chapter 4, a number of experiments are conducted to compare our approach with other ensemble methods. The experiment results will also be presented and discussed in this chapter. In Chapter 5, we use the ACO-Stacking approach in a real world direct marketing application with some modification to suit the specific cost-sensitive case. We use the cumulative response lift and cumulative profit lift instead of the accuracy as the performance evaluation criteria. The experiment results prove that the ACO-Stacking approach can tackle this problem well. In the last chapter, the conclusion is given and our contributions, limitations and the future possible extensions of this work are stated.

CHAPTER 2
Backgrounds and Literature Review

2.1 Data Mining

As we discussed in Chapter 1, data mining is a key process of KDD. Han and Kamber considered that data mining which emerged in the late 1980s, is the product of the natural evolution of information technology (Han and Kamber, 2000). The fast growing volume of data is created and stored in the cheaper and larger storage devices. Useful information and knowledge is hidden under the massive amount of data. At the same time, computers are becoming faster and algorithms are smarter and stronger. Thus more and more analysis tools have been provided for data mining. Bose and Mahapatra defined data mining as a process of identifying interesting patterns in databases that can then be used in decision making (Bose and Mahapatra, 2001). Turban et al. defined data mining as a process that uses statistical, mathematical, artificial intelligence, and machine learning techniques to extract and identify useful information and subsequently gain knowledge from a large database (Turban et al., 2010).

Nowadays, data mining is applied in many tasks for the purpose of academic research and business applications. Every year, a number of research projects on data mining algorithms or applications are proposed. Data mining is widely applied in health care (Das and Sengur, 2010), financial services (Nanni and Lumini, 2009; Yu et al., 2008), anti-spam (Ying et al., 2010) and many other fields. In industry, for example, Oracle has released their product: Oracle Data Mining (ODM). Microsoft also integrated data mining into their database systems and business intelligence solutions.

Data mining tasks can be assigned to the following classes: Association Rule Mining, Classification/Prediction, Clustering, Outliers Detection and Regression (Fayyad et al., 1996; Ngai et al., 2011).

Association Rule Mining, or Association analysis, is used to discover the patterns that describe the strong association of items in datasets. Association rule mining is usually applied to produce dependency rules which predict occurrence of an item based on occurrences of other items. An example of the application of association rule mining is the shopping basket analysis. Given a sufficient number of shopping transactions from shopping malls, association rule mining can be applied to find items that are frequently bought together by customers and further discover their dependencies. These rules of co-occurrences of items can be used by the malls to seek potential cross-selling opportunities.

Apriori, is a classic algorithm for learning association rules. The first step is frequent itemset generation. The items with co-occurrence larger than a minimum count will be considered as a frequent itemset. The Apriori principle states that if an itemset is frequent, then all of its subsets must also be frequent, and it is implemented to improve efficiency. In mathematics notation, this principle is displayed in Equation 1, where X,Y are itemsets and c(X) is the occurrence of itemset X. The second step is the rule generation. This step generates high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset. In this step, the algorithm applies a tree-like hash structure to reduce the comparisons when splitting itemsets and generating rules.

$$\forall X, Y : (X \subseteq Y) \Rightarrow c(X) \geq c(Y) \tag{1}$$

The boundary is blurred between the concepts of classification and predictive models in data mining so here we consider them as the same data mining task. Here, classification is used to represent these two notations. Classification is the task that predicts the class label or group membership of an object. Classification is perhaps the most frequently applied task among the tasks in previous financial fraud research (Ngai et al., 2011). In data mining, classification is a supervised process to identify the class labels of the new data instances when some instances with clear class labels are given. The given instances are usually

called the training set and the new instances to be predicted are usually called the testing set. Only the training set cannot generate a classifier by itself. Some learning algorithms are needed. The learning algorithms apply some hypotheses or analogs to learn the relationships from the attributes and their values of the training sets to the class labels. Thus, given the training set and the learning algorithm, a classifier can be generated. Though various learning algorithms are proposed, some of them are more welcomed by the research society and industry, for example, the Decision Trees and Neural Networks.

Decision Trees (DTs) are the tree-like decision models. We can solve a classification problem by asking a series of questions about the attributes of the dataset. When an answer is given, a follow-up question is asked until the conclusion about the class label is given. This process can be organized in a hierarchical tree structure such as the decision tree. Each non-leaf node in the decision tree represents a question on an attribute. The directed edges to different child nodes represent different answers, leading to different follow-up questions on other attributes. The leaf / terminal nodes represent the class labels.

In a decision tree construction process, the attributes need to be partitioned and tested at each node. Thus in each node, only the partition which can achieve the local optimal result will be used until there is no partition which can be conducted. When the partition of a node is terminated, this node is the leaf node. However, the local optimal results may not lead to a global optimal tree. When a large tree is generated with several attributes that are used several times in different levels, the tree is an overfitting tree. Overfitting trees can generate fairly good predictions on the training set while performing poorly in making predictions on the new instances. Some pruning strategies and others are applied to generate suboptimal, non-overfitting trees in a reasonable time. When a DT is used to predict the class labels, the instance will follow some branches from the root to the leaf and the class label of this leaf is outputted as the prediction

result of the tree.

An example of using a decision tree to classify the credit risk levels is given in Figure 1. The attributes: credit history, current debt, collateral and income are used in this DT.

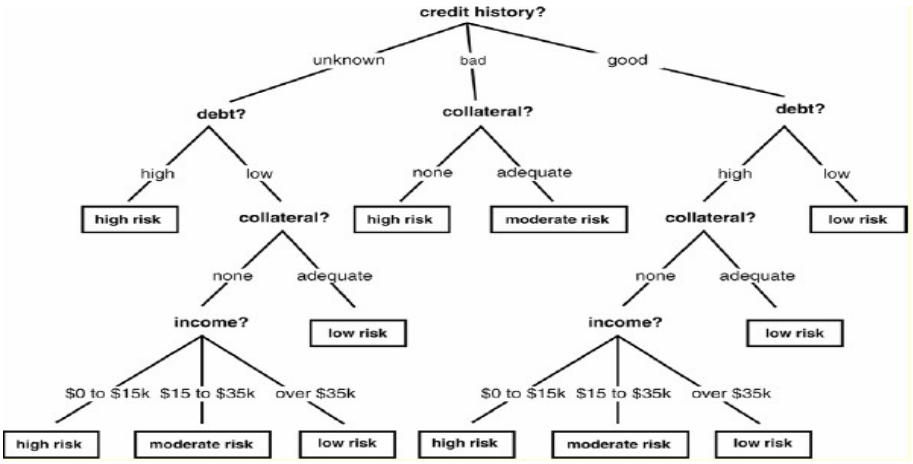


Figure 1. An Example of a Decision Tree on Credit Risk Classification

Neural Networks (NNs) generate multi-layer networks which mimic the human brain to project the input layer of the attributes to the output layer of the class labels. An example of an NN is shown in Figure 2. The neurons in the input layer represent the attributes of the dataset and the neurons in the output layer represent the group membership. One or more hidden layers are used between the input and output layers with several neurons in the layer. Between two nearby layers, the neurons are fully-connected by weighted links. However, the weights and neurons in the hidden layer are meaningless in explaining the reason of the projection from the inputs to the outputs.

The back propagation process is designed to train NNs with multiple layers. There are two phases in each iteration of the training algorithm: the forward phase and the backward phase. During the forward phase, the weights obtained from the previous iteration are used to compute the output value of each neuron from the input layer to the output layer. During the backward phase, the weights are updated in the reverse direction by using the errors from the output layer

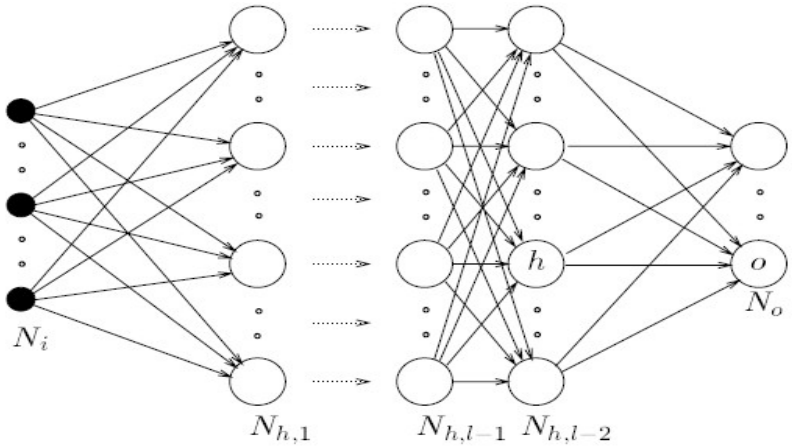


Figure 2. An Example of a Neural Network

to the input layer. When NNs are used for prediction, the values of the inputs multiple the weights and pass the products to the next layer, until the output layer is reached.

Clustering is used to divide objects into conceptually meaningful groups (clusters) with the objects in a cluster being similar to one another but dissimilar to those in other clusters. At first glance, clustering is deployed to solve a similar task to classification. However, they are actually quite different. Classification is a supervised process, while clustering is an unsupervised process. There are no given class labels for the data in a clustering process. Clustering is a discovering process which is quite different from simple segmentation, such as grouping credit card holders by their cities of residence. To group the objects in the dataset, clustering models apply different metrics to measure the "distance", which is calculated by the attributes of the data objects. The objects in the same cluster are of relatively smaller "distance" and of larger "distance" to objects in different clusters. A common distance in clustering approaches is the Euclidean distance. Equation 2 is the Euclidean distance, where p_i is the value in the i^{th} attribute of data object p and q_i is the value in the same attribute of data object q.

$$dist = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{2}$$

Some clustering techniques characterize each cluster in terms of a cluster proto-

type, which is the representative of the other objects in the cluster.

K-Means is a prototype-based clustering technique which creates a one-level partitioning of the data objects. K-Means begins by selecting K points as the initial centroids. Objects are assigned to the "nearest" clusters by calculating the distances to the centroids. The new centroids of the clusters will be reset to the "center" of the objects in the clusters after the assignment is finished. These processes will be iteratively executed several times until the centroids on longer change. The common measure to evaluate the performance of clustering is Sum of Squared Error (SSE). SSE is calculated by Equation 3, where x is a data object in cluster C_i and m_i is the centroid of cluster C_i. The performance of K-Means depends on the initialization of the number of K and the locations of the centroids. Furthermore, K-Means cannot deal with data which are density-based or have non-globular shapes.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x) \tag{3}$$

Outlier Detection is the technique used to detect data objects which appear to have different characteristics to the majority. Normal data objects often follow a certain statistical distribution process, for example, normal distribution. An outlier, is an abnormal object which deviates from the distribution of the majority of objects in the dataset. For example, a credit card holder usually has some patterns of purchasing behavior. If some abnormal purchasing behavior happens, which could be considered an outlier, the outlier detection will alart the credit card owner. Outlier detection is different from classification because the task is usually unsupervised, or semi-supervised in many scenarios. In the unsupervised scenario, the group memberships are not given in the dataset to assist and evaluate the performance. In the semi-supervised scenario, one situation is that only one group of data is given for training but the other is absent. For example, a model is trained by the normal transactions of a credit card holder to detect the suspicious transactions. Outlier detection is also different from clustering because

clustering algorithms are optimized to find clusters rather than outliers so some outliers will be grouped as a cluster. There are two main categories of approaches for outlier detection: the model-based ones and proximity-based ones.

Regression is a statistical methodology which is used to reveal the relationship between independent variables and the dependent variable. In the regression analysis, the estimation of the dependent variable is a function of the independent variables. Regression is widely used for prediction which may overlap classification and prediction. Linear regression and logistic regression are two well-known regression algorithms.

Linear regression is an extension of simple regression. Rather than using one variable as in simple regression, linear regression can use as many as all the variables in the regression function. The linear regression is shown in Equation 4 where Y denotes the dependent variable, $x_i(1 \leq i \leq n)$ represents the independent variables, β_0 is the constant and $\beta_i(1 \leq i \leq n)$ represents the coefficient of the corresponding independent variable. The value of each β_i indicates the relative importance of the corresponding independent variables for the dependent variable.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{n-1} x_{n-1} + \beta_n x_n \tag{4}$$

The logistic regression should begin with the logistic function in Equation 5.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{5}$$

Logistic function projects the input z to the range $(0, 1)$. In the Logistic function, z is calculated from Equation 6, which is the same as the Equation of linear regression but just an intermediate product.

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{n-1} x_{n-1} + \beta_n x_n \tag{6}$$

$$Y = f(z) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{n-1} x_{n-1} + \beta_n x_n)}} \tag{7}$$

The complete logistic regression is given in Equation 7, combining Equation 5 and Equation 6. Given the class labels of the instances, logistic regression can be a good classifiers. Due to its good performance, there are plenty of applications of logistic regression in the medical and social science fields.

After years of development, it becomes more and more difficult to significantly improve the performance of a single data mining model. There are growing demands for stronger analysis tools. In such cases, the intuitive idea is to combine more models together to generate collective decisions thus overcoming the weakness of a single model. For the classification problem, several different algorithms are used to generate classifiers to deal with a difficult dataset; however, the predictions created are quite different from each other in some instances. Some strategies, such as majority voting, may be applied to the results of the classifiers to generate collective predictions as the final decisions, thereby achieving a stronger model. In another scenario, concerned with the limited abilities of some models to deal with a large volume of data or the limited space of the devices, the large dataset needs to be split to generate models. These models should be combined together to give a full view of the dataset rather than the partial view of each one. Therefore the combining strategies are studied to combine the models. Sometimes there are limited instances in a dataset on which to generate a model. Some methods are used to generate different new datasets from the original dataset. The combination of the models with the new datasets can improve the performance over the models using the original dataset only. For the above reasons and other considerations, the methods of combining different models in data mining are studied. The combining methods will be discussed in the following section.

2.2 Ensembles

An ensemble was introduced into statistical mechanics and thermodynamics by J. Willard Gibbs in 1878; it is an idealization consisting of a large number of mental copies of a system, considered all at once. In data mining, ensemble refers to the strategy that combines several different models together to generate an integrated and optimal solution to specific problems. In this work, we focus on the ensembles for classification problems although there are other approaches to the ensembles for other data mining tasks (Zhou et al., 2001; Yu et al., 2007; Fern and Brodley, 2003; Strehl and Ghosh, 2002). Perhaps one of the earliest works on ensemble systems is Dasarathy and Sheela's 1979 paper, which discusses partitioning the feature space using two or more classifiers (Dasarathy and Sheela, 1979). In 1990, Hansen and Salamon showed that the generalization performance of a neural network can be improved using an ensemble of similarly configured neural networks (Hansen and Salamon, 1990). In the same year, Schapire proved that a strong classifier in probably approximately correct (PAC) sense can be generated by combining weak classifiers through boosting(Schapire, 1990). Since the seminal works, research in ensemble systems has expanded rapidly, appearing often in the literature with many creative names and ideas, including, but not limited to, multiple classifier systems (Ho et al., 1994), mixture of experts (Jacobs et al., 1991), stacked generalization (Wolpert, 1992), and combination of multiple classifiers (Xu et al., 1992). Though many of the names and ideas refer to classifiers, ensemble systems are applied to other data mining tasks, such as regression and clustering problems. For example, Yu et al. introduced a novel cocktail ensemble for regression (Yu et al., 2007). Strehl and Ghosh proposed a clustering ensemble knowledge reuse framework (Strehl and Ghosh, 2002).

These approaches usually employed different strategies to generate their individual learners and different mechanism to combine their learners. Diet-

terich generally classified the ensemble construction methods into five types (Dietterich, 2000). They are: Bayesian Voting, Manipulating the Training Examples, Manipulating the Input Features, Manipulating the Output Targets and Injecting Randomness. Polikar reviewed the ensemble based systems and classified the approaches into two categories: classifier selection and classifier fusion(Polikar, 2006). If an ensemble is generated by a set of classifiers which are trained from the same learning algorithm, this ensemble is a homogeneous ensemble. If an ensemble is generated by a set of classifiers which are trained from different learning algorithms, this ensemble is a heterogeneous ensemble (Dietterich, 2000). For example, Bagging (Breiman, 1996) and Boosting (Schapire, 1990) are homogeneous ensemble, while Stacking (Wolpert, 1992) is heterogeneous ensemble. Though there are many different ensemble schemes published, only some have been widely used and proved superior. Based on these schemes, many derivatives are introduced to further improve their capacity and performance. In the following subsection, Bagging, Boosting, Stacking and some of their variants are discussed. In the following discussions, ensembles for classification are discussed without losing generalization. Ensemble can also be applied to other data mining tasks with the same strategies as classification.

2.2.1 Bagging

Bagging, short for bootstrap aggregating, was proposed by Breiman, and is considered one of the earliest ensemble schemes (Breiman, 1996). Bagging is intuitive but powerful, especially when the data size is limited. Bagging generates a series of training subsets by random sampling an F percentage of instances from the original training set with replacement. Then the different classifiers are trained by the same classification algorithm with the training subsets. When T reaches a certain number of iterations, T classifiers are generated. These are then combined in the ensemble. The pseudo code of Bagging is given in Figure 3. Given a testing instance; different outputs are given by the trained classifiers, the

1. Iteration counter $t = 0$
2. While $t \leq T$
3.
   - Generate training subset D_t by randomly drawing F percent instances from training set
   - Train classifier C_t with the D_t
   - Add C_t to the bagging ensemble E
4. Output E as the final ensemble

Figure 3. Algorithm: Bagging

majority voting scheme is used and the prediction representing the majority will be considered as the final decision.

Random Forest: Random forests are combinations of tree predictors 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). All the decision trees are unpruned in the forest. Random Forest can be considered as a special form of Bagging.

2.2.2 Boosting

In 1990, Schapire's weak learning framework was proposed (Schapire, 1990). An elegant algorithm which boosts any given weak learners to a strong learner is also provided in this work. This algorithm is named Boosting and the pseudo code of Boosting is given in Figure 4.

The Boosting algorithm also applies re-sampling of the training data set and majority voting, however; Boosting does not treat all the instances equally, but focuses on the more informative instances which are important to the classification decision. The algorithm generates three classifiers using the same weak learner. The first learner C_1 is trained with a random subset D_1 of the training set. The second learner C_2 is trained with a more informative subset D_2 by iteratively flipping a fair coin to decide which instance should be added. If the result is

1. Generate D_1 from orinigal dataset D
2. Create C_1 with D_1
3. Do
4. - Flipping a fair coin.
   - If head, draw instances from D and present them to C_1. Add the first misclassified one to D_2
   - If tail, draw instances from D and present them to C_1. Add the first correctly classified one to D_2

   While no more instances could be added to $D2$
5. Train C_2 with D_2
6. Generate D_3 by selecting the instances which C_1 and C_2 disagree.
7. Train C_3 with D_3

Figure 4. Algorithm: Boosting

heads, some samples are selected from training set and presented to C_1 until the first one is misclassified then this instance is added to the training set of C_2. If the result is tails, the same process is conducted while the first correctly classified instance is selected. The third learner C_3 is trained with the subset D_3 with the instances which are differently classified by C_1 and C_2 by filtering the whole training set. Finally, a three-way majority voting is used to combine the three classifiers.

AdaBoost: AdaBoost was introduced by Freund and Schapire in 1997, and is a popular variation of the original Boosting scheme (Freund and Schapire, 1997). AdaBoost maintains a weighted distribution of instances, trains a series of classifiers of the same weak learning algorithm with different instances drawn according to the distribution and finally combines the classifiers through a weighted majority voting to generate the final decision. At the beginning of the process, all the instances are initialized with the same weight. For each training iteration, a training subset is drawn from the instances distribution D_t. Then the classification error of this classifier is calculated and used in changing the weight updating parameter α_t to manipulate the sample distribution. Thus the instances which are misclassified in this iteration will be given

1. Initialize the first weight distribution D_1
   for all instance in dataset S, $D_1(i) = \frac{1}{N}, i = 1, \ldots, N$
2. Iteration counter $t = 1$
3. While $t \leq T$, max iteration number
   (a) Create data subset S_t by the dirtribution D_t
   (b) Train Classifier C_t with S_t, validate C_t
   (c) Calculate error ε_t of C_t
   $\varepsilon_t = \sum_{i:C_t(\vec{x}_i) \neq y_i} D_t(i)$
   If $\varepsilon_t > \frac{1}{2}$,**abort**
   (d) Set $\alpha_t = \frac{\varepsilon_t}{1-\varepsilon_t}$
   (e) Update distribution:

$$D_{t+1}(i) = \frac{D_t(i)}{\sum_i D_t(i)} \times \begin{cases} \alpha_t & if \quad C_t(\vec{x}_i) = y_i \\ 1, & otherwise \end{cases} \tag{8}$$

4. Assign the weights $\omega_t = log\, \alpha_t$ to each classifier C_t

Figure 5. Algorithm: AdaBoost

larger probabilities of being selected in the next iteration than the correct ones. After the weight updating and normalization, the new instances distribution D_{t+1} is generated. α_t is also used as the weight of the classifier in the weighted majority voting procedure. The pseudo code of AdaBoost is given in Figure 5.

Some variations of AdaBoost, such as AdaBoost.M1 and AdaBoost.R can be referred to (Freund and Schapire, 1996; Freund and Schapire, 1997).

2.2.3 Stacking

In the above ensemble schemes, the individual weak learners are the same. A natural question is: if there are different kinds of weak learners, can we map their outputs to a single result? The Stacking approach (which is the abbreviation of Stacked Generalization) is the approach to combine classifiers from different learning algorithms (Wolpert, 1992). Since different learning algorithms apply different knowledge representations and different learning biases, thus different classifiers will be generated. Consequently, the errors from the classifiers are not closely correlated to each other, and diversity can be expected. Stacking

1. Split the training set S into N partitions, $N > 1$
2. For each learning algorithm
   (a) Use the leave-one-out training process to train the classifier C_t
   (b) Validate C_t and output the validation result R_t
3. Create the meta training set S' by joining the R_t and class label: y
4. Train the meta classifier M with S'

Figure 6. Algorithm: Stacking

has a two-level structure: the level-0 (base) classifiers and the level-1 (meta) classifier. The base level classifiers are trained with the training set and output their predictions. Then the meta classifier is trained to map the outputs of the base level classifiers to the actual class label. To generate the training set for the meta level classifier, a leave-one-out or a N-fold cross validation procedure is applied. The basic method is to train the base level classifiers with most of the training instances, such as N-1 subsets of the N subsets, and validate the classifiers with the leaving one subset. For N iterations, all the instances are trained and validated. The predictions from the validation process by different models and the real class label of the same instance will be joined together to generate a new instance in the meta data. The meta classifier is trained by the meta learning algorithm with the meta data. The meta data could be $((y_i^1, y_i^2, ..., y_i^m), y_i)$, where y_i^m means the prediction given by the m^{th} base level classifier on the i^{th} instances, and y_i is the actual class label. If there are k instances in the training set for the base classifiers, there are the same number of instances in the meta data set. During the process of testing instance, the trained base level classifiers will give their individual prediction, and the predictions will be considered as the input of the meta level classifier to generate the final decision.

2.3 Metaheuristic

Metaheuristic is a very extensive concept of a group of approximate algorithms which basically try to combine simple heuristic methods in higher level frameworks aimed at effectively solving combinatorial optimization problems. Though metaheuristic has been studied for more than a decade, there is no commonly accepted definition for it. Blum and Roli quoted some of the proposed definitions in their survey and characterized the fundamental properties of metaheuristic (Blum and Roli, 2003). The representative class of algorithms includes, but is not limited to, Ant Colony Optimization (ACO), Evolutionary Computation (EC), Iterated Local Search (ILS), Simulated Annealing (SA) and Tabu Search (TS). There are several classification systems for metaheuristic algorithms based on different considerations. The following are some classification systems: nature-inspired vs. non-nature inspired, population-based vs. single point search, dynamic vs. static objective function and one vs. various neighborhood structures. Here, we will follow the nature-inspired vs. non-nature inspired methods. ACO, EC and SA are nature inspired metaheuristics while ILS and TS are non-nature inspired. Except for the above named metaheuristics, there are several other metaheuristics such as Particle Swarm Optimization (PSO) and Bee Colony Optimization (BCO). But these will not be discussed in detail in this chapter.

2.3.1 Ant Colony Optimizations

ACO is an appealing metaheuristic inspired by the collective foraging behaviors of real ant colonies, which enable the ants to find the shortest path from their nest to the food source. Dorigo and his colleagues first proposed this idea in the early 1990s (Dorigo and Stützle, 2004; Dorigo, 1992).

In biology research, scientists have discovered that ant colonies can make collective decisions to find the shortest route to a target in their foraging behavior. Each ant is of limited intelligence in finding the best or shortest path; however, it

can use indirect methods to communicate with other ants. When an ant forages for food, it will deposit a chemical material called a *pheromone* on the ground. The ants can detect the pheromone and use it to choose their route. The ants choose their routes to walk on in a probabilistic manner, so that the paths with stronger pheromone concentrations will be chosen with larger probabilities. If the pheromone is absent, the ants will randomly choose a route. After a period, the shorter path is chosen more frequently, which means more ants walk this way and the pheromone accumulates faster. The accumulation of pheromone attracts more ants to choose this path. The double bridge experiments designed by Goss et al. have proven this behavior system (Goss et al., 1989). Figure 7 illustrates the experiment. In the first period of the experiment, the ants are equally distributed on both bridges after a few minutes because the bridges are the same. In the second period of the experiment, the ants concentrated on the shorter bridge after a few minutes. If a way is not chosen by the ants, the pheromone will evaporate. The accumulation of pheromone is a positive feedback to encourage the ants to choose the shortest way. Some ants may be "stupid" in selecting the paths with less pheromone; however, this situation is very important in allowing the ants to get rid of the local shortest path to eventually achieve the global shortest path. If the new path is shorter than the current path, the pheromone will accumulate and attract more ants to walk this way. Then the optimal path will be changed to the new path. In conclusion, although the ability of ants is limited, the optimal shortest path is likely to be achieved by the collective behaviors of ants through indirect communication.

As mentioned above, Dorigo developed the Ant System (AS) as the first approach mimicking ants' behavior (Dorigo and Stützle, 2004; Dorigo, 1992). Other ACO approaches were published because of the good performance of this idea. A non-exhaustive list of the approaches is given below. Dorigo and Gambardella proposed the Ant Colony System (Dorigo and Gambardella, 1997). Stützle and Hoos proposed the *MAX-MIN* AS (Sttzle and Hoos, 1996). Blum et

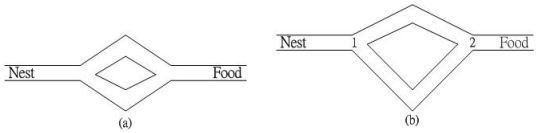


Figure 7. The Double Bridge Experiment

al. proposed the Hyper-Cube AS (Blum et al., 2001). Furthermore, some theoretical works have proved the convergence with rigid mathematical reasoning (Stützle and Dorigo, 2002; Gutjahr, 2002).

Nowadays, ACO is widely used in many applications. Traveling Salesman Problem (TSP) has been a classic optimization problem since the 1930s. In the TSP, given a list of cities and their pairwise distances, the task is to find the shortest route by which to visit all the cities once (Dorigo and Gambardella, 1997). The early works of Dorigo, Ant System, achieved remarkable performances in solving TSP (Dorigo, 1992; Dorigo and Gambardella, 1997). The interesting idea and good performance of ACO attracted more attention from other researchers, who aimed to implement this idea to solve other problems, such as Map Coloring, Vehicle Routing Problem, Project Scheduling Problems (Dorigo and Stützle, 2004). Ant-Miner is an approach which applies ACO in classification rule distraction (Parpinelli et al., 2002). Al-Ani published an approach which applies ACO in dataset feature selection (Al-Ani, 2006).

2.3.2 Other Nature-Inspired Metaheuristic

Evolutionary Computation (EC) algorithms are inspired by the Darwinian principle that living beings evolve to be adaptive to the environment by exchanging and saving genetic information. There are some basic elements in an evolutionary computing algorithm. The population represents the potential solutions of a specific problem. The fitness function, is the selection criterion to evaluate whether the individuals among the population are classed as 'elite' or 'cull'. The evolutionary operators, such as crossover or mutation, are used to

produce individuals in the next generation.

The basic EC algorithms firstly initialize a number of individuals. Then the individuals are assessed by a fitness function, in an imitation of natural selection. The individuals with higher fitness scores will inherit and be used to produce new individuals in the next generation with higher probability. The individuals with lowest fitness scores of the generation will probably be eliminated in the next generation. To produce new individuals, EC algorithms use recombination or crossover operators to two or more individuals to breed new individuals. EC algorithms also use a mutation operator to cause self-adaption of individuals. These processes iteratively execute until the finish criteria are met. Varieties of EC algorithms have been proposed for over half a century, although basically they fall into three categories which have developed independently from one another. They are Evolutionary Programming (EP) by Fogel (Fogel, 1962) and Fogel et al.(Fogel et al., 1966), Evolutionary Strategies (ES) by Rechenberg (Rechenberg, 1973) and Genetic Algorithms by Holland (Holland, 1975).

Simulated annealing (SA) is quite different from the ACO and EC. Simulated Annealing is a single point searching algorithm inspired by a physical phenomenon —- annealing of metal. In the annealing sequence, there are two main processes, the heating and controlled cooling. The heating causes the atoms in the metal to become unstuck from their initial positions (local minimum of internal energy) and move to new positions with higher energy. The controlled cooling process helps the atoms to find a position which achieves the global minimum of internal energy. Simulated Annealing is analogy of this annealing process. The origins of the algorithm are in statistical mechanics (Metropolis algorithm) and it was adapted to a search algorithm for combinatorial problems with an explicit strategy to escape from the local minimal. The pioneers in introducing this algorithm were Kirkptrick et al. in 1983 and Cerny in 1985 (Kirkpatrick et al., 1983; Černý, 1985). The algorithm starts by generating an initial solution and by ini-

tializing the temperature parameter T. At each iteration, a new solution S' is generated. If the new solution S' is better than the current solution S, it will replace the S. Else the algorithm will accept S' as a solution by the probability function $f(T, S, S')$ and then update T. Temperature T decreases during the search process. The greater T is, the more stochastic the searching process is, which means a worse solution will be accepted with higher probability. As T decreases, the searching process will converge to a simple iterative improvement.

2.3.3 Non-Nature-Inspired Metaheuristic

Tabu Search (TS) has been a very popular metaheuristic approach during the last decade. The basic idea of TS was introduced by Glover, derived from his earlier formulated ideas (Glover, 1977; Glover, 1986). The simple TS algorithm maintains a tabu list, which keeps the short term memory of the recently considered candidate solutions. The TS algorithm refuses to return to those candidate solutions until they are far enough away by tracking the tabu list. These strategies enable the algorithm to escape from local minimal and avoid cycles.

Iterated Local Search (ILS) iteratively explores the search of local minimal by some embedded heuristic Local Search. Hence, a better solution can be generated from a sequence of solutions. This simple idea has a long history. Though this idea has been rediscovered and named several times, Lorenco et al. in their handbook consider an approach as ILS by two characters: (1) there must be a single chain that is being followed; (2)the search for better solutions occurs in a reduced space defined by the output of a black-box heuristic (Lorenço et al., 2002).

2.4 Literature Review

2.4.1 Applying ACO in Data Mining

ACO approach is widely used in many aspects of data mining. In data mining tasks, feature subset selection is an important step to reduce the re-

dundant features and therefore build more precise and efficient models. Al-Ani presented a feature searching procedure based on ACO which utilizes both local importance of features and overall performance of feature subsets (Al-Ani, 2006). This approach is applied to speech segment and texture classification problems and outperformed the GA-based approaches. Sivagaminathan and Ramakrishnan proposed their approach which is a hybrid method based on ACO and NN to address feature selection (Sivagaminathan and Ramakrishnan, 2007). A heuristic value calculation is applied in the approach to reduce the set of available features.

Some usages of ACO on Clustering and Association Rule Mining are published. Here we summarize two approaches. Monmarché presented his research which applied ACO on K-Means clustering named AntClass (Monmarché, 1999). It first stochastically creates an initial partition using an improved ant-based approach without knowledge of the data (such as number of clusters) then the K-Means is called to speed up the convergence. Kuo et al. proposed a research framework which clusters the data first and then follows with association rules mining (Kuo et al., 2007). In the first stage, an *ant system based clustering algorithm* (ASCA) and *ant K-means* (AK) are used to cluster the database. In the second stage, an ant colony system based association rules mining algorithm is applied to discover useful rules for each cluster.

Some approaches have applied ACO to extract classification rules. Parpinelli et al. proposed an algorithm called Ant-Miner (Ant Colony-based Data Miner) to extract classification rules from a dataset (Parpinelli et al., 2002). Each ant in the colony represents a classification rule such as $IF < term_1 \cap term_2 \cap \ldots > \Rightarrow < class >$, where $term_i$s are generated in the preliminary test and represents the trails in the ground which the ants live. For each iteration, the pheromone of the trail which is adopted by the "ants" will increase. At the end of each iteration, the best "ant" is added to a list which contains all the classification rules discovered by the Ant-Miner. The authors claimed that Ant-Miner could discover

more rules and performed better than CN2, a well-known approach for the same task. TACO-Miner is the approach that applies Touring ACO (TACO) in rule extraction from trained NNs (Özbakir et al., 2009). The black-box nature of NNs does not allow production of understandable classification rules. TACO tries to optimize the activation function values by exploring the search space and then translates this information into a comprehensible classification rule.

Campos et al. used ACO to learn Bayesian Networks (Campos et al., 2002). A Bayesian Network (BN) is a probabilistic graphical model which is comprised of nodes and directed edges in the form of directed acyclic graphs. In the paper, ACO is used to guide a scoring-based search process, as ACO allows the searching to exploit heuristic knowledge with simple but efficient forms of cooperation between independent agents (ants). Pinto et al. proposed two ACO-based approaches, a local discovery ACO algorithm and a hybrid algorithm max-min ACO (MMACO), based on the local discovery algorithm max-min parents and children and ACO to learn the structure of a BN (Pinto et al., 2009).

Inspired by the promising performance and great potential of ACO, we tried to extend the usage of ACO to solve Stacking configuration problems.

2.4.2 Related Works on Stacking Configuration Problems

In this work, we study the configuration problems of Stacking ensembles and apply ACO in searching for optimal configurations. Some previous works focus on the specific problems: which algorithms can be used to generate the meta classifier and which measures / outputs of the base classifiers can be used to generate the meta training set. Some brief reviews of such works are presented here.

Ting and Witten used the probability distribution of the outputs from the base classifiers, instead of the simple class labels to generate the meta level data

(Ting and Witten, 1999). Thus the predictions and the correspondent probabilities are fully used. In order to use this type of data, the *multi-response linear regression* technology (MLR) is applied to generate the meta classifier.

Todorovski and Džeroski proposed another variant of Stacking that uses the *meta decision tree* (MDT) in the meta level, (Todorovski and Džeroski, 2000). The tree is named MLC4.5, indicating a modification from the C4.5 DT. The meta data set for the MDT is composed of the properties which reflect the confidence of the base classifiers instead of the probability distribution or the simple class label. Such properties are the entropy, the maximum probability and the fraction of training samples. The tree uses the class labels in the leaf nodes only. The leaves of MDT specify which base classifiers should be used instead of predicting the class label directly.

Based on Stacking with MLR, Seewald proposed an approach to reduce the number of attributes in the meta data independently of the number of classes, in order to overcome the weakness of MLR Stacking when dealing with datasets with more than two classes (Seewald, 2002). This approach is named StackingC (StC).

Džeroski and Ženko proposed two variants of Stacking (Džeroski and Ženko, 2002; Džeroski and Ženko, 2004). The first is based on Stacking with MLR, which extends the set of attributes for meta data by augmenting the entropy of the probability distributions and the products of the probability distributions multiplied the maximum probability. The second variant uses the *multi-response model tree* (MRMT) instead of the MLR to generate the meta classifier. This approach is called Stacking with MRMT.

The above approaches focus on the selection of the algorithms to generate the meta classifiers and the optimization of the attributes of the meta level data. Few approaches focus on the selection and combination of the base classifiers.

Recently, several approaches, using metaheuristic or operation research methods, have been proposed to find the optimal Stacking configuration by optimizing the base classifiers and meta classifier.

Ledezma et al. proposed the GA-Stacking approach, which applies the Genetic Algorithm (GA) to solve the Stacking configurations problem as an optimization problem without *a priori* assumptions (Ledezma et al., 2002; Ledezma et al., 2004; Ledezma et al., 2009). A binary encoding is used in the approach because it allows the use of canonical GAs. GA-Stacking aims not only to learn the combination of base classifiers and meta classifier, but also to learn the optimal parameters of the base classifiers. Thus each base classifier has more than one gene to represent it and its possible parameters in a chromosome. The accuracy of the classification result is used as the fitness evaluation function. This GA search process will iterate for several generations. For each generation, the accuracies of the validation sets are used as the fitness scores of the corresponding chromosomes and then sorted in a list. Some elite chromosomes (the top n ones in the list) will remain to the next generation and some cull ones (the last m ones in the list) will be eliminated. Mutation and crossover will occur in some of the chromosomes according to the mutation rate and crossover rate. Crossover means two chromosomes exchange their genes from a certain position on the chromosomes. Mutation means the value of a certain gene in a chromosome changes to a different value. Furthermore, the chromosomes which will crossover with the others or will mutate is selected randomly. After all generations are finished, the best chromosome will be chosen as the final configuration.

GA-Ensemble is proposed by Ordóñez et al. and is a variant of GA-Stacking (Ordóñez et al., 2008). At the beginning, a set of candidate base classifiers is trained to generate a classifiers' pool, thereby improving the efficiency without losing accuracy. The candidate sets must be encoded in a chromosome in the population, which represents a potential configuration. A binary encoding is used

Figure 8. Confusion Matrix

	True Buyer	True Non-Buyer
Predicted as Buyer	True Positive (TP)	False Positive (FP)
Predicted as non-Buyer	False Negative (FN)	True Negative (TN)

to accompany the canonical GA, where a 0 in the gene means that the classifier of this gene will not be used in the configuration and a 1 means the classifier will be used. The last gene in a chromosome represents two different Stacking combining schemes: 1 for MRMT and 0 for the majority voting. This GA search process will iterate as for that in GA-Stacking. However, unlike in GA-Stacking, each gene in the chromosome represents a base classifier. If mutation occurs to a certain gene, the base classifier represented by this gene is added or deleted. After all generations are finished, the best chromosome will be chosen as the final configuration.

Zhu proposed the DEA-Stacking approach which applies *data envelopment analysis* (DEA) to find optimal Stackings (Zhu, 2010). DEA is a linear programming methodology to measure the efficiency of multiple *decision making unit*s (DMUs) when the production process presents a structure of multiple inputs and outputs. DEA-Stacking considers the classifiers as the DMUs in DEA. In this approach, the inputs and outputs of a DMU are extracted from the confusion matrix of the model. An example of a confusion matrix is given in Figure 8. At the first stage, the classifiers are trained and evaluated. The DEA models take the FP and FN as the inputs and the TP and TN as the outputs of the DMUs to find out the efficient one(s) to be the base classifier(s). Several classifiers with an efficiency of 1 will be selected as the base classifiers in Stacking. At the second stage, the meta classifier is also selected by the DEA models. The Stackings with each learning algorithms in the set combining the selected base classifier(s) is treated as the DMUs to find the most efficient as the final configuration. Interested readers may refer to the book about DEA (Ramanathan, 2003).

CHAPTER 3
ACO-Stacking

Considering the performance of ACO in the approaches mentioned in Chapter 1 and Chapter 2, we try to extend the application of ACO in Stacking configuration optimization. In an ACO-Stacking construction task, a set of base classifier candidates and a set of meta classifier candidates are given as well as the training sets, the validation sets and the testing set. The base classifiers in the set are taken as the "paths" to be selected by the ants. For each iteration, an ant tries to select a path in its route to achieve better performance. Each ant is assigned with a certain meta classifier to combine with the selected "paths" into the "path" package of the ant. A Stacking model is configured with the base classifiers ("paths" of the ant) and the meta classifier. This Stacking is then trained with the training set(s) and validated with the validation set(s). If the new "path" package is better than the existing one, it will replace the existing package. Otherwise, the existing "path" package of this ant does not change. At the end, the configuration (the "path" package) of the best ant will be the final configuration of the approach. Finally this configuration is tested by using the test set. The above process is given in Figure 9. In the next section, the algorithm framework of ACO-Stacking will be discussed.

3.1 ACO-Stacking Algorithm Framework

Before the discussion of the algorithm framework, some notations that will be used in the algorithm description are given as follows:

- C is the base classifier candidates' pool which contains m classifiers generated from the learning algorithms, $C = \{c_1, \cdots, c_m\}$.
- k artificial ants in the colony, each ant carries a meta combining method and represents a Stacking configuration.

1. **Input:**
   - Datasets: Training Sets, Validation Sets
   - Learning algorithms for base classifiers and meta classifiers
2. **Generating Stacking by ACO-Stacking Framework:**
   - Applying ACO to search Stacking configurations
   - Training and validating the Stacking
   - Output the best Ant as the final configuration
3. **Testing:**
   - Applying the final configuration on the Testing set

Figure 9. General Process of ACO-Stacking

- μ_i: the pheromone associated to the c_i in C.
- η_i: the local information of c_i, which is a metric to evaluate the ability of c_i.
- S_j: the Stacking configuration constructed by the j^{th} ant, $j \leq k$.
- α_S: the evaluation criterion of the Stacking S. Here the classification accuracy of S is used as α_S.
- τ: the evaporation rate and $\tau \in [0, 1]$.
- L: the maximum iteration number.

At the beginning of the ACO-Stacking, a set C which contains base classifier candidates is given. Some pre-tests are conducted to gather the local information of the base-level classifiers. Here, the term "local information" is used to represent the metric to evaluate the individual classification performances of the base-level classifiers. Moreover, the pheromone μ_i of each base-level classifier c_i is initialized to a small positive number for the probability selection process. The pheromone will increase or decrease during the ACO searching process. Each ant in the colony is assigned with a learning algorithm as its meta combining scheme to generate the meta classifier. Thus an ant represents a Stacking configuration. The number

of ants is usually set to be multiples of the meta combining schemes. After all settings and configurations are prepared, the main process of the ACO heuristic begins. Like other ACO approaches, the ACO-Stacking will execute for several iterations. In the first iteration, each ant is given a base classifier randomly and the accuracy $\alpha_{S_i}$of this configuration is calculated from an independent validation set. In the following iterations, when the j^{th} ant begins its configuration searching, it selects a classifier *'c'* from the pool C which does not exist in its current configuration S_j using the roulette wheel selection. The probabilities of classifiers are normalized and mapped to the fractions of the roulette. The larger the fraction in the roulette, the larger possibility this classifier will be selected. The probability p_i of the classifier c_i to be selected by the j^{th} ant is given by Equation 9.

$$p_i = \begin{cases} \frac{q_i}{\sum_{t=1, c_i \notin S_j}^{m} q_t} & \text{if } \mathrm{c_i} \notin \mathrm{S_j}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

where q_i refers to the metric of the i^{th} classifier to be mapped in the roulette. The q_i could be generated by using the pheromone of the i^{th} classifier only or the product of its pheromone and its local information. Suppose that c_i is selected then a new configuration S'_j of this ant is generated where $S'_j = S_j \cup c_i$. Then S'_j is tested by the same validation set. If the performance of S'_j is better than S_j, it will replace S_j and then the ant continues to find another base classifier to add to the new S_j according to the same strategy to generate a new Stacking. If S'_j cannot improve the accuracy of S_j, this ant keeps its current Stacking configuration and stops its search in the iteration. Then the next ant in the colony starts its searching, until all the k ants finish their search. During the ants' searching process, once a classifier c_i is chosen to be added to any S_j to generate a new configuration S'_j, the pheromone of c_i will accumulate thus enhancing the probability of this classifier being selected by the other ants. The improvement of accuracy from S_j to S'_j is used to update the pheromone of c_i. The update rule is given in Equation 10.

$$\mu'_i = \mu_i * (1 - \tau) + CC * \mu_i * \frac{\alpha_{S'_j} - \alpha_{S_j}}{\alpha_{S_j}} \tag{10}$$

where CC refers to a constant number. The evaporation rate τ and CC are introduced to adjust the emphasis of historical knowledge and the current knowledge. The greater τ is , the less historical information will be used. If $\tau = 1$, all pheromone is evaporated, thus no historical information will be considered. The greater CC is, the more important current knowledge is considered. However, there is no evidence to suggest which number is suitable for CC.

During the ACO metaheuristic, the pheromone of the strong candidates will accumulate and the pheromone of the poor ones will vanish. After all iterations finish, the best configuration S_{best} of the k ants will be chosen as the final Stacking configuration.

3.2 Local Information Considerations

In the previous section, local information is mentioned as the metric to evaluate the abilities of the base classifiers. In this section, we focus on the discussion of the adoption of local information.

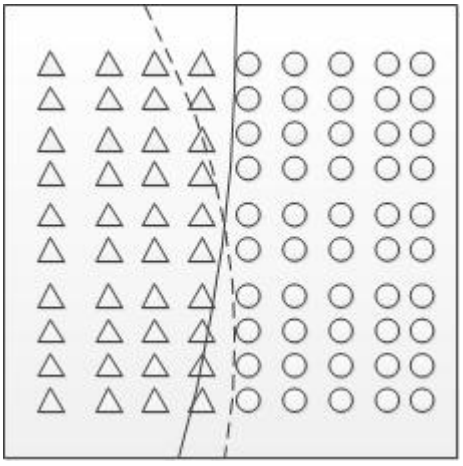
Firstly, consider the situation where an approach does not implement local information of the classifiers. In such a case, only the pheromone can affect the probabilities of selecting base classifiers. In the previous discussion, the pheromone represents how the classifier improves the global performance. However, in the early iterations of the approach, the selection of the base classifiers is quite random. Some "weak" classifiers may be selected in the early iterations and acquire pheromone accumulation. Therefore the "weak" classifiers will get larger values of pheromone and are more likely to be selected in the following iterations than some "strong" ones which have no pheromone accumulation. Such situations cause increased execution time to generate a promising configuration, as some "weak" classifiers are selected and discarded again and again. To solve this problem, selecting some "strong" classifiers in the early iterations is quite important. Local information of the classifiers is therefore used to identify the

"strong" and "weak" classifiers.

There are several different names and descriptions for this concept in ACO approaches, such as heuristic information in (Pinto et al., 2009), local importance in (Al-Ani, 2006).

The accuracy is the global performance evaluation of the Stackings constructed by ACO-Stacking in this approach. With the aim of optimizing the data fusion of ensembles which can generate better decision boundaries from the different base classifiers, one intuitive option is to use the base classifiers which already have good decision boundaries. An illustration is given in Figure 10 of two boundaries which separate two kinds of data objects. In the simple example, each decision boundary makes mistakes when separating the two categories of objects. The dotted line mistakes two triangles as the circles while the solid line mistakes one triangle as the circle. To adopt different parts of the lines will either improve or undermine the separation.

Figure 10. An Illustration of Decision Boundary



A pre-test of each c_i on the whole training set is conducted to gather measures, based on information theory or statistics, of the local information. Some measures are derived from the confusion matrix in Figure 8. The measure Recall (R) is defined in Equation 11:

$$R = \frac{TP}{TP + FN} \tag{11}$$

The measure Precision (Pr) is defined in Equation 12:

$$Pr = \frac{TP}{TP + FP} \tag{12}$$

F-measure (Equation 13) which integrates both Recall and Precision is a harmonic mean between them .

$$F = \frac{1}{1/R + 1/Pr} \tag{13}$$

From the information theory, the measure Entropy could be derived in Equation 14, where n is the number of classes in the dataset, $p(y_i)$ is the probability mass function of class y_i and b could be 2, Euler constant e or 10.

$$E = \sum_{i=1}^{n} p(y_i) I(y_i) = -\sum_{i=1}^{n} p(y_i) log_b p(y_i) \tag{14}$$

Other measures such as Cohen's Kappa (Cohen, 1960), G-Mean (Kubat et al., 1998) could also be used to measure the classifiers.

The measure Precision (Pr) could be suitable as the local information used to fuse the decision boundaries of different classifiers. Precision is the measure used to evaluate the percentage of correctly classified positive instances in the instances which are classified as positive by a classifier. We set the class which takes up the largest percentage in the dataset as the positive class in the measure of precision. The higher precision indicates fewer mistakes in the boundary of this classifier. In other words, this classifier is "stronger". So the precision of c_i is used as the local information η_i in the second version.

Although the use of Precision as the local information improves the performance of the approach, there are some limitations. Sometimes the classifiers with greater precision may have similar decision boundaries for certain difficult problems. Thus including these classifiers only overlaps their boundaries and cannot improve performance significantly. Some classifiers may have smaller precision values, but their decision boundaries are quite different from those classifiers with high precision values. In such cases, selecting these classifiers may improve the overall performance. We materialized the differences in decision boundaries

into the correlative differences of the predictions given by different classifiers on the training set. Some previous approaches inspired us to develop the measure of correlative differences of classifiers in our approach (Merz, 1999; Lu et al., 2010). In (Merz, 1999), Merz considered the usage of correspondence analysis in combining classifiers. Lu et al. proposed an ensemble pruning approach via individual diversity contribution ordering (Lu et al., 2010) .

The local information can be generated by the correlative differences. Given the pre-test set, each classifier runs a ten-fold cross validation. Both the training set and testing set are the same for each classifier in the same fold which ensures that all the classifiers are treated equally. When the pre-test is finished, all the predictions of the classifiers on the same instance in the set are collected. The difference: $D_{i,j}$ between C_i and C_j is the number of instances when they make different predictions. The difference matrix of the classifiers is:

$$\begin{vmatrix} 0 & D_{1,2} & \cdots & D_{1,n} \\ D_{2,1} & 0 & \cdots & D_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ D_{n,1} & \cdots & D_{n-1,n} & 0 \end{vmatrix}$$

In the matrix, $D_{i,j} = D_{j,i}$ and the larger $D_{i,j}$, the larger differences between C_i and C_j.

The local information η_i of the i^{th} classifier is calculated from the items in matrix by equation 15:

$$\eta_i = \frac{\sum_{t=1, c_t \in S_j}^{m} D_{i,t}}{k} \tag{15}$$

where k equals the number of classifiers in the current configuration S_j. According to the equation, the larger the average difference of the candidate classifier c_i from the classifiers in S_j, the more difference there is between the decision boundary of c_i and the decision boundaries of the current Stacking S_j. Thus if this c_i is selected, the data fusion may be improved.

Up to this point, the local information used in our approach and its important

in improving the performance of ensembles are clearly discussed. However, the time taken to apply local information still requires consideration. As discussed in Chapter 1, to generate promising ensembles, an important aspect which should be carefully considered is the introduction of enough diversity into the components of an ensemble (Polikar, 2006). Depending on local information to select base classifiers will overemphasize the "strong" classifiers and may reduce diversity in the Stacking. In the versions implementing local information in our approach, we use the pheromone alone in the roulette selection function (Equation 9) in the first half of the iterations and then use the product of pheromone and local information in the function of the following half iterations.

3.3 Different Versions of ACO-Stacking

This section describes the different versions of the ACO-Stacking approach. When exploring the usage of ACO to optimize the Stacking configuration problems, the later version not only overlaps the early versions but also uses more information and implements additional strategies. Briefly, the ACO-Stacking V1 explores many combinations of base classifiers found by ACO heuristic with *the same meta classifier*. The ACO-Stacking V2 extends the search space by using more meta classifiers and implements certain criteria as local information to guide the heuristic search process. The third version is quite similar to the second version although different local information focusing on the differences of base classifiers is used.

3.3.1 The first Version of ACO-Stacking

The first version of ACO-Stacking was proposed in 2010 (Chen and Wong, 2010). In this version, the meta learning algorithm is set as a C4.5 DT so there is only one meta classifier. Moreover, local information is not implemented in this version to guide the searching process.

1. For i from 1 to m, initialize the pheromone μ_i of C_i in C; initial L to 0
2. While the maximal iteration L does not reach
   (a) For j from 1 to k, the j^{th} ant begins its searching
      - Initialize its configuration $S_j = \emptyset$
      - Initialize the current best configuration $S' = \emptyset$
      - Set the flag of adding a new classifier to *true*
      - While the flag equals to $true$
        - Using roulette wheel technique to select a $c_i \notin S_j$ to generate a new configuration: S'_j and $S'_j = S_j \cup \{c'\}$
        - IF current best configuration $S_j = \emptyset$, THEN Set $S' = S'_j, S_j = S'_j$
        - ELSE
          * Apply S'_j to train an ensemble on the training set
          * Evaluate the accuracy of the ensemble on an independent data set
          * Compare the accuracy of S'_j to that of S'
            · IF S'_j is superior, THEN update the pheromone μ_i of c_i and,$S' = S'_j, S_j = S'_j$
            · ELSE, set the flag of adding a new classifier to $false$
   (b) Evaporation occurs when an iteration finishes.
   (c) $L = L + 1$.
3. Using the same searching process of an ant to generate the final Stacking configuration

Figure 11. The First Version of ACO-Stacking Algorithm.

Thus the update rule is given in Equation 9 with $q_i = \mu_i$. The approach is more stochastic than the other versions in exploring many possible combinations of base classifiers with the same meta classifier. In this manner, the comparisons of ants in the approach are based on the same meta classifier, only the combinations of base classifiers can affect the results. Further, more iterations are needed to continue the search to find the optimal results. The other components of the implementations are the same as those in Section 3.1. The pseudo code of ACO-Stacking is presented in Figure 11. The performance of this version is promising and was reported in (Chen and Wong, 2010).

3.3.2 The second version of ACO-Stacking

The second version differs from the first version in three main aspects. Firstly, a meta-level classifiers' set is used. In this version, the meta-level classi-

fiers of the ants can be built by assigning a learning algorithm from a set. Each learning algorithm is treated equally and is assigned to the ants by a uniform distribution. By using more learning algorithms to learn meta-level classifiers, the approach can adapt to the characteristics of the datasets in different domains.

Secondly, a classifiers' pool of the base-level classifiers is generated to accelerate the execution speed. The metaheuristic methods usually suffer from a long execution time. In the Stacking training process, the base classifiers should be trained and the outputs are used to generate the meta training set for the training of the meta-classifiers. If many Stackings will be generated and trained, the same base classifiers may be trained several times using the same training sets, which is very costly. To improve the efficiency of the approach, the classifiers' pool proposed in GA-Ensemble is generated *a priori* in our approach (Ordóñez et al., 2008). Consider the Stacking training process, where the training set is split into ten partitions. One partition is separated to be the validation partition and the other nine partitions are used to train the classifiers until all the partitions are validated. The outputs of each validation partition of this learning algorithm are joined together. For each learning algorithm in as base classifier candidate, this process is conducted. Next, all the prediction results of the classifiers on each training instance are stored in a "pool". To generate a Stacking ensemble, only the meta classifier needs to be trained. The meta training set is the conjunct of the predictions of the selected base classifiers in the "pool".

Thirdly, local information is introduced. Before the ACO-Stacking starts to search for the configurations, the base classifiers' pool is generated. Then a series of pre-tests are conducted to find the suitable metric to act as the local information. In this version, the precisions of the base classifiers are used as the local information. For each classifier c_i in the pool, its local information η_i is initialized and the pheromone μ_i is initialized with a small positive value. Once the local information of the classifier is set, it cannot be changed during

1. Generate the classifiers' pool
2. Initialize settings: τ, η, μ, L and k
3. While the maximal iteration L does not reach
   (a) For j from 1 to k, the j^{th} ant begins its searching
   - Initialize the S_j by given a meta combining method and randomly select a c from C
   - Calculate α_{S_j}
   - Set the flag $search\_next = true$
   - While $search\_next = true$
     Select a c from C according to the pheromone distribution and local information
     If no c can be selected
     set $search\_next = false$
     Else
     – Add c to generate new configuration S'_j
     – Calculate $\alpha_{S'_j}$
     If $\alpha_{S'_j} > \alpha_{S_j}$
     $S_j = S'_j$
     Update the pheromone of c
     Else
     $search\_next = false$

   (b) Evaporation works after an iteration ends.
   (c) $L++$
4. Output the S_{best} of the final iteration as the final configuration of ACO-Stacking
5. Test S_{best} in the independent testing set

Figure 12. The Second Version of ACO-Stacking Algorithm.

the searching process. Thus the possibilities of classifier c_i should be changed to Equation 16.

$$p_i = \begin{cases} \frac{\mu_i * \eta_i}{\sum_{t=1, c_i \notin S_j}^{m} \mu_t * \eta_t} & \text{if } \mathrm{c_i} \notin \mathrm{S_j}, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

This enhanced version of ACO-Stacking is proposed in (Chen and Wong, 2011). The pseudo code of ACO-Stacking is presented in Figure 12.

3.3.3 The third version of ACO-Stacking

In this work, we continuously improve the performance of this approach by using different local information. In the latest version of the ACO-Stacking approach, we adopt the correlative differences of different base classifiers on the

training set as the local information. Some strategies are needed to generate the η_i of the i^{th} classifier. The analysis of this local information and how to find it are discussed in Section 3.2. The others parts of this version are similar to those in the second version of ACO-Stacking.

3.4 Differences between ACO-Stacking and GA-Based Approaches

In Chapter 2, we briefly introduce GA-Ensemble and GA-Stacking, the GA-based Stacking configuration search approaches. Though ACO-Stacking, GA-Ensemble and GA-Stacking are all hybrids of metaheuristic with Stacking ensemble, there are some differences among them. During the ACO searching process, the ants use the pheromone as an indirect communication method, while during the GA searching process, the chromosomes cannot communicate with each other. The crossover points and the mutation points are selected randomly, so some well-performed Stackings may generate poor offspring. The searching processes in GA-based approaches are therefore more stochastic than in ACO-Stacking.

To escape from sticking in the local minimum, the weak ants in ACO-Stacking will not be eliminated but simply stop searching in this iteration. In GA-Ensemble, the last n cull chromosomes will be eliminated and the top m elite chromosomes will be kept to the next generation. The mutation and crossover operations on the elite chromosomes are used to avoid the local minimum. However, there are no strategies to stop the same weak Stackings being generated again in the following generations, which will be costly because those weak Stackings have to be evaluated again.

ACO-Stacking is more flexible than GA-Ensemble in meta classifiers selection. GA-Ensemble can only select either a multiple-response model tree or a majority voting scheme as the meta classifiers, while ACO-Stacking can select the meta classifiers from a set of learning algorithms. If the number of base clas-

sifier candidates in ACO-Stacking is the same as the number of genes representing classifier candidates in GA-Ensemble, the search space of ACO-Stacking is bigger than GA-Ensemble. Furthermore, if the best meta classifier for a certain dataset is neither the majority voting nor a model tree, GA-Ensemble is unable to find it.

CHAPTER 4
Experiments and Results

To compare the performance of ACO-Stacking approaches and the other well-known ensemble approaches, the experiments are conducted in the Waikato Environment for Knowledge Analysis - WEKA (Hall et al., 2009). This environment implements different data mining and machine learning algorithms to generate classifiers as well as some well-known ensemble methods.

To make the experiment results more robust, a ten-fold cross validation scheme is used for each data set during the experiments. A dataset is randomly split into 10 mutually exclusive and exhaustive folds. Each time, one fold is selected as the test set and the other nine folds are combined together as the training set. The learning approaches use the training set to train the models and use the test set to evaluate the models. The average of evaluation results is given.

4.1 Benchmark Datasets

To evaluate the approaches in the experiment, 18 datasets in different domains from the UCI machine learning repository (Frank and Asuncion, 2010) are used. Some descriptions of these datasets are given below and some of their properties are summarized in Table 1. During the experiment, all the datasets are kept the same as those in the repository, without any preprocessing or feature selection.

Balance-Scale This dataset was generated to model psychological experiment results. Each instance belongs to the balance scale and tips left, right or balance.

Breast-W This dataset is from the clinical cases of Breast Cancer in Wisconsin, USA.

Chess This dataset describes a chess game. The first 36 attributes describe the board and the last attribute is the class label: "White-win" or "White-no-win".

Colic This dataset is generated by the diagnosis of horse colic.

Credit-A This dataset concerns credit card applications in Australia. The attributes describe the properties of the applicants and the class label is "approval" or "not-approval".

Credit-G This dataset is about the credit risks of the clients of a financial institution in Germany.

Glass This dataset contains seven classes of glass. The attributes are the contents of chemical elements in the glass, such as Silicon and Calcium.

Heart-C The dataset identifies the presence of heart disease in patients collected in Cleveland, USA. The original contains 76 attributes, but all published experimental results refer to using a subset of 14 attributes. Here we also used the subset in our experiments.

Heart-Statlog This dataset is similar to the Heart-C, but with some differences. The number of attributes is smaller and some types of attributes are different.

Hepatitis This dataset contains the biochemical indicators of hepatitis.

Ionosphere This dataset describes the pulses in an ionosphere experiment. The targets were the free electrons in the ionosphere.

Iris This is perhaps the best known dataset which contains three types of iris plants.

Labor This dataset evaluates whether a labor contract is "good" or "bad".

Table 1. Dataset Description

Dataset	Attributes	Instances	Classes
Balance-Scale	5	625	3
Breast-w	11	699	2
Chess	37	3196	2
Colic	27	368	2
Credit-A	15	690	2
Credit-G	21	1000	2
Glass	10	214	7
Heart-C	14	303	2
Heart-Statlog	14	270	2
Hepatitis	20	155	2
Ionosphere	35	351	2
Iris	5	150	3
Labor	17	57	2
Lymphography	19	148	4
Sonar	61	208	2
Vehicle	19	846	4
Vote	17	435	2
Wine	14	178	3

Lymphography This dataset is about the biochemical indicators of Lymphography.

Sonar This dataset contains the patterns obtained by bouncing sonar signals off either a "metal" or "rock" cylinder at various angles and under various conditions. The transmitted signals are used to distinguish them.

Vehicle This dataset is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouettes of the vehicles.

Vote This dataset collects the voting results in the US Senate to identify whether the senator is a Democrat or a Republican.

Wine This dataset is generated by chemical analysis results of wines produced in the same region in Italy but derived from three different cultivars.

4.2 Experiment Settings

4.2.1 Learning Algorithms and Parameters

In order to obtain optimal configurations of Stacking, ten different learning algorithms in WEKA are used as the base classifier candidates. The ten algorithms can be categorized into several different kinds of hypotheses, thus making them as diverse as possible when generating classifiers.

- Naïve Bayes (John and Langley, 1995) learns classifiers by the naive probabilistic estimator based on the Bayes' theorem.
- Logistic (Le Cessie and Van Houwelingen, 1992) builds a multinomial logistic regression model to make predictions.
- IB1 (Aha et al., 1991) learns the instance-based nearest neighbor classifier using normalized Euclidean distance.
- IBk is similar to IB1, which uses k-nearest neighbor instead of one nearest neighbor. Here, $k = 5$ is used.
- KStar (Cleary and Trigg, 1995). KStar is an instance-based classifier. The class label of a test instance is decided by entropy-based functions.
- OneR (Holte, 1993). The classifier uses the minimum-error attribute for prediction.
- PART (Frank and Witten, 1998) builds a partial C4.5 decision tree in each iteration and turns the "best" leaf into a classification rule using the separate-and-conquer strategy.
- ZeroR. It uses 0-R classifiers for prediction.
- Decision Stump (Iba and Langley, 1992) generates a one-level decision tree classifier.

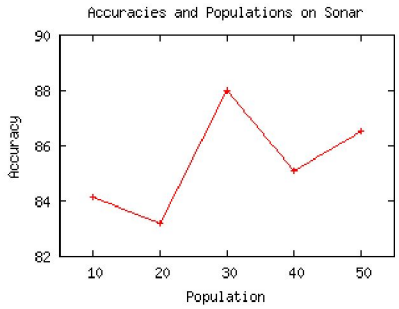


Figure 13. The Population Curve

- C4.5 DT (Quinlan, 1993) generates a decision tree classifier.

As discussed in Chapter 3, ACO-Stacking also selects an appropriate meta classifier for the Stacking. In the experiments, the learning algorithms above are also used as the meta classifier candidates.

Some necessary preliminary tests are needed to find suitable parameters of ACO and GA. The parameters of ACO, including the number of ants in the colony, the maximal iteration, the evaporation rate and the constant CC, are given in Table 2. In the preliminary test of the parameters, when the iteration number is larger than a certain threshold, the results do not change significantly. On the other hand, when the population size is increased from 10 to 50 by an increment of 10 while the other parameters are maintained, the results differ with different populations sizes. Here, an example of the results on the Sonar dataset is given in Figure 13. From the figure we can see that the best accuracy is achieved when the population size is 30. In most of the benchmark datasets, similar results can be found. When the population size and iteration number increase, the training time will increase. Considering the tradeoff of performance improvement and cost, the population size in the approach is set to 30.

4.2.2 Compared Approaches

In the experiments, the Stackings found by ACO-Stacking are compared with the following ensembles.

- AdaBoost with C4.5 DT as its base learning algorithm;
- Bagging with C4.5 DT as its base learning algorithm and $F = 0.67$;
- Random Forest;
- StackingC with Naïve Bayes, IBk and C4.5 DT as its base learning algorithms and a Multi-Response Model Tree as its meta learning algorithm (Seewald, 2002; Džeroski and Ženko, 2002);
- Ensembles from GA-Ensemble approach.

  The base classifiers' pool is set to be the same as ACO-Stacking. The meta combining method is determined by GA-Ensemble, either a Multi-Response Model Tree or a majority voting scheme can be selected. The parameters of GA are list in Table 3. The number of chromosomes in the approach is set to be the same as the number of ants in ACO-Stacking. The number of generations is set to be the same as the maximum iteration number in ACO-Stacking.

In the benchmark experiment, we does not compare with DEA-Stacking. Because the DEA-Stacking approach in (Zhu, 2010) does not clearly discuss the strategies to tackle with the multi-class datasets other than the two-class datasets while there are several multi classes datasets in our experiment. Furthermore, the experiment in the paper is not clearly discussed thus it cannot be repeated in our experiment.

Table 2. ACO parameters

Parameter	Value
Colony Size	30
Iterations	10
Evaporation Rate	0.1
CC	10

Table 3. GA parameters

Parameter	Value
Population Size	30
Generations	10
Elite Rate	0.1
Cull Rate	0.1
Cross Operation	Uniform
Mutation Rate	0.1
Crossover Rate	0.5

4.3 Results and Analysis

Table 4 summarizes the results of average accuracies of the approaches from the 18 datasets. In some datasets, such as Ionosphere, Iris and Vote, the performances of all the approaches do not significantly diverge from each other. In the simple datasets, all the approaches are promising. However, in some datasets, such as Balance-Scale and Sonar, the accuracies of Stacking based approaches (StackingC, GA-Ensemble and ACO-Stackings) are better than the non-Stacking based approaches; furthermore, the metaheuristic Stacking based approaches are better than the non-metaheuristic Stacking approaches. For example, in the Balance-Scale dataset, the accuracies of Bagging, AdaBoost and Random Forest are smaller than 80%, while the best result, 98.88%, is achieved by ACO-Stacking.

In the empirical and statistical tests, we focus on the comparison between our latest version and the other approaches. The comparisons of the different versions of ACO-Stacking are also given.

4.3.1 Empirical Analysis

The empirical $w/t/l$ test results are given at the last line of table 4, where w means ACO-Stacking V3 outperforms the relative approach, t means their

Table 4. The Classification Accuracies of the Ensembles

Dataset	Bagging	AdaBoost	Random Forest	StackingC	GA-Ensemble	ACO-Stacking V1	ACO-Stacking V2	ACO-Stacking V3
Balance-Scale	71.68^{++}	76.48^{++}	76.96^{++}	86.08^{++}	98.72	**98.88**	98.56	98.72
Breast-W	95.14^{++}	96.42	95.99	**97.28**$^{-}$	96.14^{+}	97.00	95.14^{++}	96.99
Chess	99.44	**99.50**	98.91^{++}	99.44	99.19	99.34	99.14^{++}	99.34
Colic	67.93^{++}	70.92^{++}	71.47^{+}	64.13^{++}	75.00	**82.88**$^{--}$	76.90	78.26
Credit-A	86.38	84.35	84.35	**86.81**	85.65	84.35^{++}	82.32^{++}	85.94
Credit-G	74.0	69.6^{++}	74.1	74.7	73.7^{+}	74.8^{+}	75.0	**76.1**
Glass	73.83	**79.44**$^{--}$	73.36	69.16^{++}	77.10	72.43	76.17	75.23
Heart-C	78.88	76.90	79.21	**84.16**$^{--}$	77.89	81.19	74.59^{++}	78.22
Heart-Statlog	80.0^{++}	80.37	78.15^{++}	**84.16**	80.0^{++}	81.85	75.93^{++}	82.96
Hepatitis	83.23^{+}	85.81	80.65^{++}	81.94	84.52	83.23	**87.74**	86.45
Ionosphere	**93.45**	93.16	**93.45**	90.88	92.88	92.02	89.17	92.31
Iris	95.33	93.33^{+}	95.33	95.33	95.33	94.67	**96.0**	95.33
Labor	84.21^{+}	89.47	87.72	89.47	85.96^{++}	91.29	87.72	**92.9825**
Lymphography	79.05	81.08	81.08	83.11	82.43	82.43	**85.81**$^{--}$	81.08
Sonar	74.52^{++}	77.88	80.77	81.73	86.06	81.73	**87.98**$^{--}$	83.65
Vehicle	76.60^{++}	76.24^{++}	77.07^{+}	74.11^{++}	75.53^{++}	75.2941	74.23^{++}	**79.91**
Vote	96.32	95.86	95.86	**96.78**	95.17	95.63	94.25	95.17
Wine	94.94^{++}	96.63^{++}	97.19^{+}	96.07^{++}	98.31	97.75^{+}	98.31	**98.88**
w/t/l	12/1/5	13/1/4	13/2/3	10/1/7	11/2/5	11/2/5	12/1/5	-

[1] $^{++}$ indicates ACO-Stacking V3 outperforms the corresponding approach at 0.05 level
[2] $^{+}$ indicates ACO-Stacking V3 outperforms the corresponding approach at 0.1 level
[3] $^{--}$ indicates the corresponding approach outperforms ACO-Stacking V3 at 0.05 level
[4] $^{-}$ indicates the corresponding approach outperforms ACO-Stacking V3 at 0.1 level

Table 5. RAI Test Result

	Bagging	AdaBoost	Random Forest	StackingC	GA-Ensemble	ACO-Stacking V1	ACO-Stacking V2	ACO-Stacking V3
RAI	97.05%	70.46%	71.56%	58.04%	20.98%	13.78%	29.53%	-

performances are the same and l means ACO-Stacking is not as good as the relative approach. The statistical analysis will be given in the next subsection. Looking into the $w/t/l$ test, compared with Bagging, Random Forest and GA-Ensemble, ACO-Stacking wins in ten of the eighteen datasets, ties in one of the datasets and loses in seven of the datasets respectively. Compared with StackingC, ACO-Stacking wins in ten of the datasets and loses in eight of the datasets. Compared with AdaBoost, ACO-Stacking only wins in thirteen of the datasets and loses in four of them.

Another empirical test: Relative Improvement (RAI) is also conducted to evaluate the approaches. The RAI is calculated by using the equation 17.

$$p = \sum \frac{\alpha_i - \alpha'_i}{\alpha'_i} \tag{17}$$

where α_i refers to the accuracy of ACO-Stacking in the i^{th} data set and α'_i refers to the accuracy of the approach being compared with. According to the RAI test in Table 5, ACO-Stacking V3 gains relative improvement of 97.05% with Bagging, 70.46% with AdaBoost, 71.56% with Random Forest, 58.04% with StackingC and 20.98% with GA-Ensemble.

From the two empirical tests, ACO-Stacking outperforms Bagging, AdaBoost, Random Forest and StackingC. It is slightly better than GA-Ensemble.

4.3.2 Statistical Analysis

To demonstrate the statistical significance of the experiments, pairwise T-tests are conducted. As mentioned above, the ten-fold cross-validation scheme is used on each dataset for all the approaches. Because the splits of the folds for each approach are the same, the approaches use the same datasets for each training-testing phase. Therefore the performances of the approaches on the same fold can be compared as pairs. In the paired T-test, the performances of the other approaches and those of the ACO-Stacking V3 are compared to find statistical

Table 6. Results of *w/t/l* tests and RAI tests

	Test	Result
V2-V1	*w/t/l* test	7/0/11
	RAI test	-13.10%
V3-V1	*w/t/l* test	11/2/5
	RAI test	13.78%
V3-V2	*w/t/l* test	12/1/5
	RAI test	29.63%

significance. The results of the T-test are also shown in Table 4. The detailed T-test results are given in the appendix, Table A.1.

The T-test results show that ACO-Stacking V3 significantly outperforms Bagging in seven of the 18 datasets at the 5% level and in two of them at the 10% level, Random Forest in four datasets at 5% and three at 10% and GA-Ensemble in three datasets at 5% and two at 10%. Moreover, ACO-Stacking V3 is not significantly inferior to the above three approaches in any datasets in the experiments. Compared with AdaBoost, ACO-Stacking V3 is significantly superior in five of the datasets at the 5% level and one dataset at the 10% level, while it is significantly inferior in one dataset at the 5% level. Compared with StackingC, ACO-Stacking V3 is significantly superior in five datasets at the 5% level and significantly inferior in Heart-C at 5% and in Breast-W at 10%.

Generally speaking, ACO-Stacking V3 is superior to other approaches in the T-Test results. We can therefore draw a conclusion, from both empirical and statistical analysis, that the performance of ACO-Stacking V3 is promising.

4.3.3 Comparisons of Results from Different Versions

The same tests (*w/t/l* RAI and T-Test) are also used here to compare the performance of the different versions. The results of *w/t/l* and RAI tests between the three versions are summarized in Table 6 and the T-Test results between V1 and V2 are given in Table A.6 in the appendix.

Comparing ACO-Stacking V1 and ACO-Stacking V2, in the *w/t/l* test as

Table 7. Average Numbers of Base Classifiers in Stackings

Approaches	Number of Base Classifiers
Version 1	4.9375
Version 2	3.125
Version 3	3.3333

well as the P values in T-Test (Table A.6), V1 wins in 11 of the datasets and loses in seven. ACO-Stacking V1 is significantly superior to V2 in six datasets at the 5% level and in three at the 10% level. ACO-Stacking V2 significantly outperforms V1 in only one dataset at the 5% level and in two at the 10% level. According to the RAI test, the result is -13.10%, which means ACO-Stacking V2 cannot show improvement over ACO-Stacking V1 and is worse than V1.

In the *w/t/l* test in Table 6, ACO-Stacking V3 wins in 12 of the datasets, ties in one dataset and loses in the remaining five datasets compared with ACO-Stacking V2. Furthermore, V3 outperforms V2 in six datasets at the 5% level and is inferior in Lymphography and Sonar at the 5% level. According to the RAI test in Table 6, ACO-Stacking V3 gains relative improvement of 29.53% with ACO-Stacking V2.

In the *w/t/l* test in Table 6, ACO-Stacking V3 wins in 11 of the datasets, ties in two dataset and loses in five datasets compared with ACO-Stacking V1. In the T-Test, V3 is significantly superior to V1 in one dataset at the 5% level and in two datasets at the 10% level, but inferior to V1 in one dataset at the 5% level. According to the RAI test, the relative improvement is 13.78%. V3 is slightly better than V1.

We are also interested in the number of base classifiers used in the Stackings founded by the ACO-Stacking approaches. The average numbers of base classifiers in different versions of ACO-Stacking are given in Table 7. In V1, the average number of base classifiers is much more than those in V2 and V3. This interesting phenomenon could be explained by the differences of the versions. The first version focuses on the search between the combinations with the same

meta classifiers. As discussed previously regarding the differences between the versions, given sufficient number of iterations, the first version, which is more stochastic without local information, can discover the Stackings with more base classifiers. The other versions use local information to guide the searching process and use meta classifiers set to extend the searching space. The local information in the V2 helps the construction of the combination of base classifiers to focus on the "powerful" candidates so that some less strong, but potentially useful candidates, are ignored. Thus the average number of base classifiers in V2 is smaller. The major difference between V2 and V3 is that V3 uses the correlative differences as the local information. The correlative differences focus on searching the classifiers which are not similar to the existing ones in the Stackings. This local information strategy does not ignore the classifiers with "average" performance. The optimized local information improves the performance while bringing a slightly increased average number of base classifiers. From the above analysis, the conclusion may be drawn that ACO-Stacking V3 could be the best of the three versions.

CHAPTER 5
A Real-World Direct Marketing Application

In this chapter, we modify our ACO-Stacking approach for a real-world application in direct marketing.

Direct marketing is a type of marketing that reaches its potential customers without traditional advertising, such as TV, newspapers or radio, and instead communicates directly with the consumer with advertising such as direct mail, catalogs and street advertising. Direct marketing companies often maintain massive databases of their customers' information, including (but not limited to) their contacts, their previous purchasing records, their responses to previous marketing campaigns and so on.

Not every customer in the database is interested in the catalog, so he/she will make no purchases. Other customers will only purchase occasionally, spending small amounts. Only a few customers are highly loyal to the company and purchase often. The former two kinds of customers account for a much larger percentage of the database than the loyal ones (e.g., 95% to 5%).

Furthermore, buyers contribute different profits when they respond to the campaign. Some buyers are identified as most likely to respond and make a purchase, so the company may send some gifts with the catalog. However, although they respond, they may only place a small order, thus the company can only earn a small amount of profit. In another scenario, some buyers seldom respond to the campaign but will place a big order if they respond. So in the direct marketing problem, the cost and the profit vary between customers.

Because of budget constraints and required return of marketing investment, the company cannot try to contact all the customers in the database. Therefore, it is essential to identify the customers who are more responsive to marketing activities and more profitable for the company. For a marketing campaign, typically

only the names in the top two deciles or the 90th or 80th percentile (i.e. those with the highest probabilities of responding) will receive the promotion materials from the company (Zahavi and Levin, 1997).

Direct marketing companies therefore build varieties of predictive models from the databases to narrow their target customers groups, thus realizing desirable return within the budget. Until recently, the dominant models in this field were statistically based, for example regression and discriminant analysis. Some data mining and machine learning approaches were also proposed to learn models for direct marketing applications. For instance, Zahavi and Levin applied NN to target marketing (Zahavi and Levin, 1997). Bhattacharyya proposed his approach of applying a genetic algorithm (Bhattacharyya, 1999). Cui et al. studied model selection for direct marketing (Cui et al., 2008).

Our ACO-Stacking approach, which is designed to optimize the classification problems in data mining, could be easily modified to solve this direct marketing problem.

5.1 The Direct Marketing Database

A large real-life direct marketing dataset from a U.S.-based catalog company provided by the Direct Marketing Education Foundation, is used to evaluate ACO-Stacking and other approaches. The company sells multiple product lines of merchandise, from gifts and apparel to consumer electronics. It regularly sends catalogs to its customers by mail. This dataset contains 106,284 records in a recent promotion, as well as their purchase history over a twelve-year history. The dataset also contains the demographic information from the 1995 U.S. Census and credit information from a commercial vendor. Thus there are 361 variables in each record. The most recent promotion sent a catalog to every customer in this dataset and achieved a 5.4% response rate, representing 5,740 buyers.

Table 8. Summary of the Cost / Profit (US$) of the Direct Marketing Dataset

Statistics Metric	Value
Max Profit	612.66
Min Profit	4.36
Average Profit	38.77
Standard Deviation of Profit	37.622
Max Cost	9.18
Min Cost	0.34
Average Cost	0.74
Standard Deviation of Cost	0.301
Overall Average	1.70
Overall Standard Deviation	13.342

The statistical summary of the cost / profit from the direct marketing dataset is given in Table 8 . The maximum profit is US$ 612.66, about 150 times the minimal profit and 16 times the average profit. The maximal cost is US$ 9.18, about 30 times the minimal cost and about 12.4 times the average cost. The average profit is about 52.4 times the average cost in the dataset.

In the direct marketing dataset with so many variables, it is necessary to conduct some features (variables) selection to reduce the dimension of the instances. Either a *wrapper* selection process or a *filter* selection process may be applied. In a wrapper selection process, different combinations of variables are iteratively tried and evaluated by building an actual model. In a filter selection process, certain evaluation functions, which are based on information theory or statistics, are used to score the combinations of variables to select the one with the highest score as the final combination. In this application, 17 variables are selected by the forward wrapper selection, from the total of 361 variables. For example, the lifetime total orders (TOTORD), the lifetime total sales (TOTSALE), whether the customer placed telephone orders (TELE), whether the customer paid by cash (CASH) and so on.

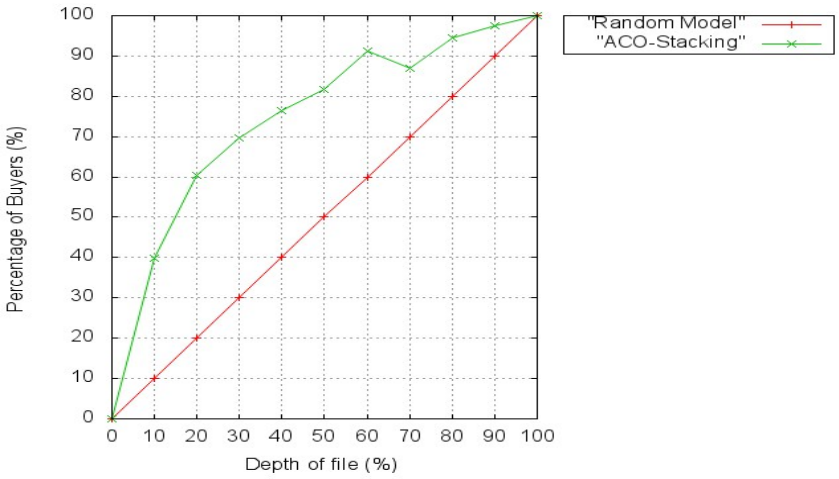
5.2 Evaluation Methods for Direct Marketing Models

In direct marketing applications, simple accuracy may not be the most appropriate method for assessing the performance of classifiers (Wong and Cui, 2010).

First, despite the larger volume of the dataset, the response rate is very small (5.4% in this case). The dataset is extremely imbalanced. If a classifier makes predictions that all the names are non-responders, the accuracy will still be 94.6%, which seems to be pretty good for conventional accuracy-based applications. However, this result is meaningless for this problem. As we mentioned, due to the budget constraints, only the names in the top deciles or top two deciles of the database are likely to be contacted in a direct marketing campaign, but the overall accuracy may not produce superior results in the top decile(s). Second, the accuracy cannot show the distinction of different misclassification errors. For direct marketing models, false negatives are more costly than false positives. The potential sale and profit of a false negative may be much larger than the mailing cost of a false positive.

The decile analysis which estimates the enhancement of the response rate and profit at different depths of the dataset is used to evaluate the classifier, rather than the overall accuracy. To use the decile analysis, the names with their response rates should be sorted into a rank list in decreasing order. The names in the first decile indicate they are most likely to respond and generate profits while the names in the last decile are unwilling to respond and purchase. The *cumulative lift*, which is usually the most important criterion for the decile analysis, will be used in this approach as well (Zahavi and Levin, 1997; Cui et al., 2008). Lift is a measure of the effectiveness of a predictive model, which is calculated as the ratio between the results obtained with the classifier and with a random model at a certain depth of the dataset. In a direct marketing essence, the response rate and the profit rate are the most important measures. Thus the cumulative response lift, cumulative profit lift and the lifted profits are used to measure the approaches. Cumulative response lift evaluates the ratio between the response received from the customers with a classifier and those with a random model at a certain depth of the dataset. Cumulative profit lift evaluates the ratio between the earning profits obtained with a classifier and those with a random model. The

Figure 14. Lift Chart of ACO-Stacking



lifted profits evaluate the actual amount of lifted profit obtained with a classifier will earn against that with a random model. A lift chart (Figure 14) and the tables (Table 10, 11, 12) are used to present the performance of different models across the ten deciles.

5.2.1 ACO-Stacking for Direct Marketing Problem

Due to the agile structure of the ACO-Stacking framework, it is not difficult to modify this approach to tackle the direct marketing problem. As we discussed in Chapter 3 and the previous section, since the optimization objective is changed from the overall accuracy to the cumulative lift, α in the approach is changed accordingly. The profits from the top two deciles of the names in the training set are used as α. However, the profit of each name (instance) is transparent to the learning algorithms in the training process. In other words, each instance in the database is treated equally with the non-cost-sensitive algorithms. The profits of the validation set are given to calculate α. We modified the algorithm to generate a ranking list of the instances in the validation set by sorting their probabilities to be positive customers in decreasing order. The profits from the names in the top two deciles of the list are calculated to be the α of this configuration. The

Table 9. ACO Parameters for Direct Marketing Application

Parameter	Value
Colony Size	20
Iterations	10
Evaporation Rate	0.1
CC	10

other things are the same as those in previous versions. Thus the ACO-Stacking is used to find the Stacking configuration which could maximize the profits of the top two deciles of the customers. In the ACO-Stacking V2 and V3, the local information η is introduced. However, in this modified approach, non specific ηs are implemented for such cost-sensitive problems.

5.3 Experiments and Results

A series of experiments and analysis of the results were conducted to evaluate the performance of ACO-Stacking in the direct marketing problem. The ten-fold cross validation scheme was used in the experiments as well.

In the experiment, the base classifiers' set is different from that in the previous approaches. Because the size of the database is much larger than that of the benchmark datasets, some instances-based learning algorithms such as KStar and IBk are replaced by other learning algorithms. In this version, the following learning algorithms: C4.5 DT, CART, Decision Stump, Logistic, NB, NB Simple, NB Updateable, OneR, PART and VFI, are used in the candidate set. NB simple is a variant of NB which models the numerical attributes by a normal distribution (Duda and Hart, 1973). NB updateable is another variant (John and Langley, 1995). VFI is an algorithm that generates a classifier that classifies an instance based on feature intervals (Demiröz and Güvenir, 1997). The other parameters are given in Table 9.

5.3.1 Compared Approaches

The ACO-Stacking approach is compared with two sets of approaches. The first set is those conventional approaches which have been applied to direct marketing problems, such as Logistic Regression, Naïve Bayes, NNs and Bayesian Networks (Zahavi and Levin, 1997; Cui et al., 2008). As discussed in Chapter 2, logistic regression can be used as a classifier. Logistic regression classifiers are a popular approach in marketing problems because they can output not only the class labels of an instance, but also a precise probability of its prediction. Neural Networks are introduced in Chapter 2 and Naïve Bayes classifier is used in Chapter 4. A Bayesian Network is a probabilistic graphical model which is composed of nodes and a directed edges in the form of directed acyclic graph. Each node in the graph represents an attribute of the dataset and the directed edges represent the conditional dependence of two nodes. Bayesian Networks are good at explanatory relationships of attributes of the dataset because they not only output the prediction of class labels but also output the probabilities of the class labels and dependencies of the other attributes (Cui et al., 2008).

The second set is those approaches which are ensemble or cost-sensitive. They are Bagging, AdaCost, AdaC2 and DEA-Stacking. Bagging is introduced in Chapter 2. The learning algorithm in this Bagging approach is Logistic Regression. The random subset fraction is 0.667.

AdaCost is a cost-sensitive version of AdaBoost (Fan et al., 1999). It uses the different costs of corresponding misclassification errors to adjust the training distribution on successive boosting rounds and thereby build a better cost-sensitive model. The different costs of corresponding misclassification errors are given by a confusion matrix. To develop a confusion matrix, one must determine which errors might be committed and their corresponding costs. In this direct marketing dataset, there are two classes: buyer and non-buyer. Therefore there are two kinds of errors: classifying buyer as non-buyer and the reverse. The con-

Figure 15. Cost Matrix of AdaCost

	True Customer	False Customer
Predicted as True	0	1
Predicted as False	10	0

fusion matrix in this experiment is given in Figure 15. The penalty for the error of mistaking non-buyer as buyer is one and the penalty for mistaking buyer as non-buyer is ten. The penalty for mistaking buyer as non-buyer is larger because the potential profit of a buyer is much larger than the cost of marketing material and mailing. In this experiment, the penalty in the matrix is estimated according to the ratio of the smallest profit by the average mailing cost, which is close to 10. Moreover, the numbers of iterations of the AdaCost is set as 10. Because of the constant cost defined by the cost matrix, AdaCost treats all instances which commit the same misclassification equally. However, the costs of different instances often vary even if the same misclassification errors are committed.

Sun et al. proposed an approach which incorporates the individual misclassification costs into the training distribution adjustment process of AdaCost (Sun et al., 2007). Three algorithms: AdaC1, AdaC2 and AdaC3 are proposed. AdaC2 performs better in their paper. So AdaC2 is compared in our experiment. The differences between AdaC2 and Adaboost is that, the update rule in Adaboost (Equation 8) is modified by adding the specific cost of each instances (C_i). The new update rule is given in Equation 18.

$$D_{t+1}(i) = \frac{C_i D_t(i)}{\sum_i C_i D_t(i)} \times \begin{cases} \alpha_t, & if \quad C_t(\vec{x}_i) = y_i \\ 1, & otherwise \end{cases} \tag{18}$$

Both AdaBoost and AdaC2 use Logistic Regression as the weak learner.

DEA-Stacking is also compared in this experiment (Zhu, 2010). The learning algorithms of ACO-Stacking are also used to generate DMUs in the experiments.

5.3.2 Results and Analysis

Table 10 and Table 13 give the results of the cumulative response lift of ACO-Stacking compared with the two sets of models. Table 11 and Table 14 give the results of the cumulative profit lift of ACO-Stacking compared with the alternative models. Moreover, Table 12 and Table 15 show the average lifted profit (US$) of ACO-Stacking and the compared methods respectively. In the tables, the number with bold font in each decile indicates that the method in this column achieves the best results in this decile compared with the other methods. The pairwise T-tests are conducted to compare the results as well. The results of the T-tests are given in the Appendix.

In the comparison of benchmark data mining methods, from Table 10, the models generated by ACO-Stacking achieve the average cumulative response lift of 401.5 and 301.3 in the top two deciles respectively. The results suggest that by mailing to the top one decile, the respondents by the model are 401.5 percent of those responding as a result of randomly mailing without a model. If mailing to the top two deciles, the ratio is 301 percent. The response lift of the models generated by ACO-Stacking is significantly higher than Bayesian Networks and Naïve Bayes in the top decile and significantly higher than Logistic Regression and Naïve Bayes in the top two deciles. From Table 11, the cumulative profit lift of the model generated by ACO-Stacking is significantly higher than the other conventional approaches in the top decile and significantly higher than Naïve Bayes in the top six deciles. According to Table 12, an average lifted profit of US$ 9198.7 will be earned if a marketing campaign is conducted to the names in the top 20% predicted by the models generated by ACO-Stacking. The results of the three tables indicate that our approach could generate the best solution when the budget is constrained to contact the customers in the top one and top two deciles compared with conventional approaches.

The comparison results of ACO-Stacking and the other ensemble and cost-

Table 10. Average Cumulative Response Lift of Ten-fold Cross-validation Compared with Conventional Data Mining Methods

Models	Logistic Regression	Bayesian Networks	Neural Networks	Naïve Bayes	ACO-Stacking
Decile	Response Lift	Response Lift	Response Lift	Response Lift	Response Lift
1	374.7 (15.9)$^+$	357.6 (17.8)$^+$	380.1 (21.7)	280.7 (19.0)$^+$	**401.5 (47.5)**
2	261.3 (8.9)$^+$	263.0 (7.8)	275.3 (9.2)	220.0 (11.4)$^+$	**301.3 (70.4)**
3	216.4 (6.4)	214.1 (7.1)	218.6 (5.7)	187.1 (6.9)$^+$	**232.3 (36.1)**
4	184.8 (3.9)	182.3 (5.3)	183.6 (4.9)	162.5 (5.3)$^+$	**192.3 (21.1)**
5	161.4 (3.7)	158.8 (2.3)	160.5 (2.9)	146.6 (3.1)$^+$	**164.2 (13.9)**
6	145.1 (2.1)	141.4 (2.4)	144.4 (2.2)	134.8 (2.2)$^+$	**145.9 (7.6)**
7	130.2 (1.4)	128.2 (1.8)	130.6 (1.5)	126.8 (1.0)$^+$	**130.9 (4.9)**
8	118.6 (1.2)	116.5 (1.4)	**118.8 (1.2)**	117.7 (1.5)	118.4 (2.9)
9	108.6 (1.2)	108.0 (0.7)	**108.9 (0.7)**	108.6 (0.5)	108.5 (1.5)
10	100.0	100.0	100.0	100.0	100.0

[1] The reported figures are the means of the lifts of the 10 experiments, with the standard deviations in parentheses.

[2] $^+$ Using paired t-test, the cumulative response lift is significantly smaller than that of ACO-Stacking at 0.05 level.

Table 11. Average Cumulative Profit Lift of Ten-fold Cross-validation Compared with Conventional Data Mining Methods

Models	Logistic Regression	Bayesian Networks	Neural Networks	Naïve Bayes	ACO-Stacking
Decile	Cum. Lift	Cum. Lift	Cum. Lift	Cum. Lift	Cum. Lift
1	589.9 (33.0)$^+$	565.6 (39.7)$^+$	597.1 (40.6)$^+$	478.2 (44.3)$^+$	**637.1 (63.4)**
2	354.8 (18.1)	365.2 (14.3)	377.5 (19.8)	326.6 (22.0)$^+$	**414.1 (92.4)**
3	274.5 (11.8)	275.0 (11.2)	278.2 (9.5)	251.1 (14.4)$^+$	**295.8 (49.1)**
4	221.3 (7.3)	220.4 (7.6)	220.7 (7.3)	203.4 (11.4)$^+$	**232.7 (30.3)**
5	184.1 (5.7)	183.5 (4.2)	183.2 (4.7)	173.5 (5.3)$^+$	**189.8 (20.1)**
6	159.3 (3.6)	156.5 (3.5)	159.3 (3.6)	151.7 (4.6)$^+$	**162.9 (11.8)**
7	139.0 (3.0)	137.9 (3.0)	139.5 (2.3)	137.2 (2.9)	**141.1 (7.5)**
8	123.3 (1.6)	122.4 (2.3)	123.6 (1.6)	123.3 (2.3)	**123.9 (4.4)**
9	110.4 (1.2)	110.5 (1.3)	111.0 (1.0)	111.1 (0.8)	**111.2 (2.0)**
10	100.0	100.0	100.0	100.0	100.0

[1] The reported figures are the means of the lifts of the 10 experiments, with the standard deviations in parentheses.

[2] $^+$ Using paired t-test, the cumulative profit lift is significantly smaller than that of ACO-Stacking at 0.05 level.

Table 12. Average Lifted Profits($) of Ten-fold Cross-validation Compared with Conventional Data Mining Methods

Deciles	Logistic Regression	Bayesian Networks	Neural Networks	Naïve Bayes	ACO-Stacking
1	7184.6	6821.0	7312.4	5553.0	**7886.2**
2	7770.5	7784.2	8155.9	6650.9	**9198.7**
3	7683.7	7707.2	7845.6	6662.6	**8600.2**
4	7120.2	7063.8	7082.6	6076.3	**7778.8**
5	6171.8	6124.9	6096.7	5392.7	**6581.9**
6	5222.6	4977.4	5227.8	4546.2	**5535.2**
7	4003.2	3886.3	4055.4	3812.4	**4209.8**
8	2732.3	2635.3	2769.3	2727.6	**2805.6**
9	1376.2	1396.6	1438.6	1464.0	**1481.9**
10	0.0	0.0	0.0	0.0	0.0

[1] The reported figures are the means of the lifts of the 10 experiments.

Table 13. Average Cumulative Response Lift of Ten-fold Cross-validation Compared with Ensemble and Cost-Sensitive Data Mining Methods

Models	Bagging	AdaCost	AdaC2	DEA-Stacking	ACO-Stacking
Decile	Response Lift	Response Lift	Response Lift	Response Lift	Response Lift
1	372.9 (17.1)$^{+}$	139.0 (46.8)$^{+}$	375.0 (17.2)$^{+}$	378.4 (20.9)$^{+}$	**401.5 (47.5)**
2	261.7 (9.2)$^{+}$	95.8 (14.2)$^{+}$	263.1 (8.5)$^{+}$	271.1 (11.1)	**301.3 (70.4)**
3	217.2 (5.0)	63.9 (9.5)$^{+}$	217.0 (6.4)	220.9 (5.3)	**232.3 (36.1)**
4	184.2 (3.7)	80.4 (56.0)$^{+}$	184.2 (4.0)	184.7 (3.8)	**192.3 (21.1)**
5	162.0 (3.7)	**194.8 (7.6)**$^{-}$	161.1 (3.4)	162.2 (3.8)	164.2 (13.9)
6	145.2 (2.6)	**162.3 (6.4)**$^{-}$	144.8 (2.0)	144.2 (2.5)	145.9 (7.6)
7	130.1 (1.5)	**139.1 (5.5)**$^{-}$	129.9 (1.5)	129.2 (1.8)	130.9 (4.9)
8	118.8 (1.1)	**121.8 (4.8)**$^{-}$	118.6 (1.0)	117.8 (1.6)	118.4 (2.8)
9	**108.7 (0.8)**	108.3 (4.0)	108.6 (0.8)	108.0 (1.0)	108.5 (1.5)
10	100.0	100.0	100.0	100.0	100.0

[1] The reported figures are the means of the lifts of the 10 experiments, with the standard deviations in parentheses.

[2] $^{+}$ Using paired t-test, the cumulative response lift is significantly smaller than that of ACO-Stacking at 0.05 level.

[3] $^{-}$ Using paired t-test, the cumulative response lift is significantly larger than that of ACO-Stacking at 0.05 level.

Table 14. Average Cumulative Profit Lift of Ten-fold Cross-validation Compared with Ensemble and Cost-Sensitive Data Mining Methods

Models	Bagging	AdaCost	AdaC2	DEA-Stacking	ACO-Stacking
Decile	Cum. Lift	Cum. Lift	Cum. Lift	Cum. Lift	Cum. Lift
1	584.9 (34.7)$^{+}$	241.3 (54.9)$^{+}$	593.6 (31.1)$^{+}$	611.7 (34.8)	**637.1 (63.4)**
2	364.1 (20.4)	145.3 (30.0)$^{+}$	367.7 (18.0)	377.0 (16.8)	**414.1 (92.4)**
3	275.2 (10.4)	86.7 (19.9)$^{+}$	275.3 (11.7)	279.4 (10.0)	**295.8 (49.1)**
4	220.5 (6.6)	100.7 (74.8)$^{+}$	220.7 (7.4)	220.4 (6.7)	**232.7 (30.3)**
5	184.5 (5.6)	**246.4 (10.1)**$^{-}$	184.0 (5.5)	184.8 (6.0)	189.8 (20.1)
6	159.6 (4.1)	**196.4 (8.3)**$^{-}$	159.2 (3.7)	158.7 (4.1)	162.9 (11.8)
7	138.9 (3.4)	**160.7 (7.0)**$^{-}$	138.5 (3.5)	138.5 (3.1)	141.1 (7.5)
8	123.5 (1.5)	**133.9 (6.1)**$^{-}$	123.2 (1.3)	123.2 (2.2)	123.9 (4.4)
9	110.8 (1.4)	**113.1 (5.2)**	110.5 (1.3)	109.9 (1.3)$^{+}$	111.2 (2.0)
10	100.0	100.0	100.0	100.0	100.0

[1] The reported figures are the means of the lifts of the 10 experiments, with the standard deviations in parentheses.

[2] $^{+}$ Using paired t-test, the cumulative profit lift is significantly smaller than that of ACO-Stacking at 0.05 level.

[3] $^{-}$ Using paired t-test, the cumulative profit lift is significantly larger than that of ACO-Stacking at 0.05 level.

Table 15. Average Lifted Profits($) of Ten-fold Cross-validation Compared with Ensemble and Cost-Sensitive Data Mining Methods

Decile	Bagging	AdaCost	AdaC2	DEA-Stacking	ACO-Stacking
1	7080.5	2047.0	7246.3	7516.6	**7886.2**
2	7707.4	1352.1	7872.8	8141.7	**9198.7**
3	7663.0	-565.2	7725.3	7912.0	**8600.2**
4	7020.0	124.8	7096.1	7087.3	**7778.8**
5	6131.5	**10713.0**	6160.1	6239.4	6581.9
6	5177.6	**8461.4**	5208.5	5177.9	5535.2
7	3921.7	**6208.1**	3962.4	3959.8	4209.8
8	2680.2	**3952.7**	2717.5	2718.1	2805.6
9	1340.2	**1708.9**	1380.4	1296.3	1481.9
10	0.0	0.0	0.0	0.0	0.0

[1] The reported figures are the means of the lifts of the 10 experiments.

sensitive approaches are more interesting than those compared with conventional approaches. As shown in Table 13, the models generated by ACO-Stacking significantly outperform those generated by Bagging, AdaCost and AdaC2 in cumulative response lift if mailing to the top two deciles. The models generated by ACO-Stacking significantly outperform those generated by DEA-Stacking in a cumulative response lift in the top decile. Compared with AdaCost, cumulative response lifts gained from the models generated by ACO-Stacking are significantly higher in the top four deciles, while significantly inferior in the following four deciles. Similar phenomena can be found in Tables 14 and 15. Due to budget constraints, the significances of cumulative lifts in the fifth and following deciles may not be attractive to marketing decision makers. As shown in Table 14, the cumulative profit lift of the model generated by ACO-Stacking is significantly higher than Bagging, AdaCost and AdaC2 in the top 10% of names. However, the models generated by ACO-Stacking do not significantly outperform those generated by DEA-Stacking. From Table 15, the direct marketers can gain most profits if they mail to the top 10%, 20%, 30% and 40% of customers according to the models generated by ACO-Stacking.

In conclusion, ACO-Stacking significantly outperforms most other approaches in the top two deciles in both cumulative response lifts and cumulative profit lifts. This suggests that our approach is the optimal solution if the bud-

get is limited to only allow the customers in the 90% or 80% percentile to be contacted.

CHAPTER 6
Conclusions

The challenge of reaching optimal Stacking configurations for specific datasets is a difficult problem in data mining and machine learning societies. Several previous works have tried to solve this problem by designing a well-performed meta classifier and optimizing the attributes of the meta training sets. The Stacking configuration problem is also a combinatorial optimization problem, which the research on Metaheuristic aims to solve. Several GA-based approaches are proposed. Inspired by the previous research and the good performance of ACO, we try to solve the Stacking configuration problem by ACO. We name our approach ACO-Stacking.

In this work, a comprehensive study is conducted to optimize the performance of the ACO-Stacking. In the study, we develop different versions of the ACO-Stacking approach by considering different ideas, such as the adoption of local information. In the first version, no local information is implemented and only one learning algorithm (C4.5 DT) is used to create the meta classifier. In this version, we focus on the effects of ACO in guiding the search and the combination of base classifiers. The first version aims to find as many as possible combinations of base classifiers with the same meta classifier. The second version is quite different from the first version. We implement the concept of a classifiers pool. The base classifiers are all trained prior to the Stacking searching process, instead of training them when they are selected by some Stackings. The classifiers pool may improve the efficiency of the training process (Ordóñez et al., 2008). The second difference is that we extend the searching space of the Stacking by introducing the meta classifiers set. The local information is introduced in this version to accelerate the convergence process to find the optimal solution. The third version is similar to the second, the major change being that the correlative differences of base classifiers are used as the local information.

A series of experiments are conducted to evaluate the performance of our approaches compared with other well-known ensemble approaches. From the experiment results, we found that our ACO-Stacking is promising in the benchmark datasets.

Furthermore, we use our approach in a real world data mining problem with some modifications. Due to the cost-sensitive nature of the direct marketing problem, the evaluation criteria of the original ACO-Stacking are changed to cost-sensitive criteria. Specifically, the cumulative profit in the top two deciles of customers is used. In the experiments, we compared our approach with several conventional approaches as well as ensemble and cost-sensitive approaches. The results indicate that our approach can gain significantly more cumulative response lift and cumulative profit lift than other approaches.

6.1 Contributions

In this work, the contributions are three-fold. Firstly, this is the first work to apply Ant Colony Optimization to a Stacking configuration problem. Stacking is a well-known ensemble; however, how to configure an optimal Stacking for a specific dataset is still regard as a "black art". Furthermore, though Ant Colony Optimization performs well in many applications, it has not been implemented in solving Stacking configuration problems. In this study, ACO is firstly integrated into the Stacking configuration searching process. Secondly, we implement the local information in the ACO-Stacking process. Several kinds of local information are studied to improve the performance of ACO. The correlative differences, which represent the variations of predictions from different classifiers, are adopted in our latest version of ACO-Stacking. Thirdly, this approach could be applied to solve real world direct marketing problems. Direct marketing data is often very imbalanced and cost-sensitive, thus making it hard to solve its problems with regular data mining models. ACO-Stacking was modified with cost-sensitive measures

to tackle this problem. It is important to emphasize that it is not necessary for the learning algorithms used to generate the base-level classifiers and meta-level classifier to be cost-sensitive. By using our ACO-Stacking method, these non-cost-sensitive learning algorithms can be employed to handle cost-sensitive data-mining problems. In comparison with other approaches, our approach performs better. In the dataset, our approach gains a higher cumulative response rate and greater profits than other approaches.

6.2 Limitations

Though our approach achieves promising performance in both benchmark experiment and real world application, there are some limitations of this research.

The metaheuristic approaches usually suffer from the problem of long execution time, which is a major limitation of such approaches. Although some strategies, such as the classifiers' pool, are applied to improve the efficiency of ACO-Stacking, the execution time is still quite long when dealing with massive datasets, compared with Bagging, Boosting and DEA-Stacking schemes.

In the real world direct marketing application, we find the standard deviation in each decile is much larger than those in other approaches, even though the average cumulative response and profits lifts of our approach are remarkable. The large standard deviation means the models learned by ACO-Stacking cannot achieve robust performances in the decile. The reason of this situation may due to there are not adequate population and iterations in the experiment.

6.3 Future Works

In this work, we limit our ACO-Stacking approach to a single performance evaluation criterion for each application. For example, only accuracy is used in the benchmark datasets and only the cumulative profit lift is used for the direct

marketing application. ACO has been proved to be strong in multi criteria optimization problems. One possible future direction is to extend ACO-Stacking to find multi-criteria ensembles. Furthermore, only two measures for local information (Precision and correlative differences) are selected and applied in the approach. However, many other criteria can be employed as local information, so the best metric for local information can be further explored.

A relative short execution time is very essential for an application. One future direction of this work is to modify the ACO-Stacking to run in parallel to improve the efficiency. Much research has been done to parallelize the ACO approach on a Graphic Processing Unit thereby to accelerate the execution efficiency without much additional resource required.

Regarding to the the standard deviation problem, more population in the colony and more training iterations are needed to allow generation of stable and well performing models.

Ensembles does not only refer to ensembles of classifiers. Nowadays, ensemble is widely used in clustering and regression tasks (Zhou et al., 2001; Fern and Brodley, 2003). In the future, we may try to use our ACO-Stacking approach in clustering and regression tasks.

APPENDIX
T-Test Results

Table A.1. T-Test Results Comparing ACO-Stacking V3 with Other Approaches

Dataset	Bagging	AdaBoost	Random Forest	StackingC	GA-Ensemble	ACO-Stacking V1	ACO-Stacking V2
Balance-Scale	0.000000	0.000000	0.0000001	0.0000001	0.500000	0.1717182	0.1717182
Breast-W	0.0111553	0.1114289	0.1209294	0.0839253	0.0834998	0.4999895	0.0194581
Chess	0.1973370	0.2201331	0.0441717	0.1974420	0.3785769	0.3785769	0.0123906
Colic	0.0049887	0.0339263	0.0542722	0.0017461	0.2608251	0.0310916	0.3244683
Credit-A	0.3235024	0.1652257	0.1069753	0.2543266	0.4641168	0.0285541	0.0018095
Credit-G	0.1120638	0.0007032	0.1364561	0.1390601	0.0626059	0.0874195	0.1322428
Glass	0.2445055	0.0094844	0.1754939	0.0477480	0.1976552	0.1782390	0.3292875
Heart-C	0.3651401	0.2570200	0.3201271	0.0120685	0.2897714	0.1043521	0.0114888
Heart-Statlog	0.0434211	0.1356231	0.0416457	0.3096503	0.0159739	0.1717182	0.0134647
Hepatitis	0.0897257	0.4234965	0.0051866	0.1958267	0.1208607	0.1297224	0.3113592
Ionosphere	0.2394984	0.3137480	0.1996802	0.1734408	0.3463169	0.4246206	0.1558482
Iris	0.4999980	0.0967120	0.5000000	0.5000000	0.5000000	0.2955275	0.3391521
Labor	0.0612270	0.3088514	0.1247536	0.3262504	0.0405631	0.4095610	0.2536493
Lymphography	0.2665084	0.4836158	0.4899213	0.2361720	0.2720702	0.2234239	0.0347614
Sonar	0.0525092	0.1965906	0.4143393	0.4899213	0.0421126	0.2545360	0.0251035
Vehicle	0.0099207	0.0153492	0.0502950	0.0001810	0.0091444	0.0944051	0.0105185
Vote	0.1237107	0.3386872	0.2915285	0.1578242	0.1941620	0.3460636	0.1678431
Wine	0.0047673	0.0184326	0.0948778	0.0256282	0.5910512	0.0839253	0.3631588

Table A.2. T-Test Results Comparing ACO-Stacking V2 with ACO-Stacking V1

Dataset	ACO-Stacking V1
Balance-Scale	0.0839330
Breast-W	0.00668037
Chess	0.0786766
Colic	0.03740137
Credit-A	0.01089679
Credit-G	0.44267226
Glass	0.0979825
Heart-C	0.00153940
Heart-Statlog	0.02943033
Hepatitis	0.12527892
Ionosphere	0.22912368
Iris	0.0645235
Labor	0.0748713
Lymphography	0.19531785
Sonar	0.00471518
Vehicle	0.04860191
Vote	0.16595619
Wine	0.47400674

Table A.3. T-Test on Cumulative Response Lifts Comparing ACO-Stacking with Conventional Methods

Decile	Logistic Regression	Bayesian Networks	Neural Networks	Naïve Bayes
1	0.036259315	0.002224982	0.057723958	1.7819E-06
2	0.043489598	0.05296566	0.117507998	0.001781568
3	0.084628573	0.059239432	0.110961909	0.001154868
4	0.123976916	0.066107841	0.090174407	0.000424558
5	0.251781095	0.121234504	0.193065827	0.00093357
6	0.371907232	0.060256015	0.268234213	0.000328569
7	0.340079488	0.081448979	0.400767859	0.013327142
8	0.417786332	0.078527228	0.319311999	0.258948255
9	0.429530714	0.213810413	0.224638725	0.376161932
10	Nil	Nil	Nil	Nil

Table A.4. T-Test on Cumulative Response Lifts Comparing ACO-Stacking with Ensemble and Cost-Sensitive Data Mining Methods

Decile	Bagging	AdaCost	AdaC2	DEA-Stacking
1	0.022937041	2.77507E-07	0.033147641	0.031149002
2	0.044883245	8.14366E-06	0.049993816	0.088338498
3	0.096070746	2.16594E-07	0.092087202	0.15179822
4	0.101273516	0.000100446	0.105308338	0.116254009
5	0.306571802	0.000244104	0.228616218	0.323837378
6	0.394514142	0.00067747	0.330965015	0.245386312
7	0.3244809	0.005984177	0.265361066	0.176449711
8	0.323312707	0.056719828	0.405915241	0.315668874
9	0.299628423	0.45680555	0.413364401	0.222432218
10	Nil	Nil	Nil	Nil

Table A.5. T-Test on Cumulative Profit Lifts Comparing ACO-Stacking with Conventional Methods

Decile	Logistic Regression	Bayesian Networks	Neural Networks	Naïve Bayes
1	0.016343997	0.000798785	0.017658141	5.12771E-06
2	0.058331533	0.054602348	0.104354778	0.006290256
3	0.099860658	0.096149711	0.134186487	0.00874408
4	0.118641855	0.08741152	0.113156542	0.003956334
5	0.187834047	0.144835398	0.160713828	0.007591255
6	0.197861484	0.049361949	0.198850356	0.003836821
7	0.252909305	0.118941717	0.271140901	0.052377315
8	0.316821908	0.208828775	0.394308082	0.307169688
9	0.084991004	0.18186134	0.288306272	0.41403787
10	Nil	Nil	Nil	Nil

Table A.6. T-Test on Cumulative Profit Lifts Comparing ACO-Stacking with Ensemble and Cost-Sensitive Data Mining Methods

Decile	Bagging	AdaCost	AdaC2	DEA-Stacking
1	0.010119174	4.28387E-08	0.019668876	0.059878445
2	0.054673752	7.60446E-06	0.068786102	0.103897403
3	0.104668761	4.16541E-07	0.10758196	0.153148792
4	0.101118203	0.000275013	0.106560416	0.108491766
5	0.198734886	4.12583E-05	0.185036423	0.231908791
6	0.20979146	8.03821E-05	0.193124733	0.177947695
7	0.233114004	0.000335489	0.210150988	0.187771763
8	0.352371464	0.003234153	0.303000095	0.327034124
9	0.199472425	0.213313693	0.099143274	0.05463849
10	Nil	Nil	Nil	Nil

BIBLIOGRAPHY

Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Al-Ani, A. (2006). Feature subset selection using ant colony optimization. *International Journal of Computational Intelligence*, 2:53–58.

Bhattacharyya, S. (1999). Direct marketing performance modeling using genetic algorithms. *INFORMS Journal on Computing*, 11(3):248–257.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

Blum, C., Roli, A., and Dorigo, M. (2001). The hyper-cube framework for ant colony optimization. In *Proceedings of the 4th Metaheuristics International Conference*, volume 2, pages 399–403.

Bose, I. and Mahapatra, R. K. (2001). Business data mining - a machine learning perspective. *Information & Management*, 39(3):211–225.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random forest. *Machine Learning*, 45(1):5–32.

Campos, L. M., Fernández-Luna, J. M., Gámez, J., and Puerta, J. M. (2002). Ant colony optimization for learning bayesian networks. *International Journal of Approximate Reasoning*, 31(3):291–311.

Chen, Y. and Wong, M. L. (2010). An ant colony optimization approach for stacking ensemble. In *Proceedings of the 2nd World Congress on Nature and Biologically Inspired Computing*, Kitakyushu, Japan.

Chen, Y. and Wong, M. L. (2011). Optimizing stacking ensemble by an ant colony optimization approach. In *Proceedings of the Genetic and Evolutionary Computing Coference (GECCO-2011)*, Dublin, Ireland.

Cleary, J. G. and Trigg, L. E. (1995). K*: An instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine Learning*, pages 108–114. Morgan Kaufmann.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

Cui, G., Wong, M. L., Zhang, G., and Li, L. (2008). Model selection for direct marketing: Performance creteria and validation methods. *Marketing Intelligence and Planning*, 26(3):275–292.

Das, R. and Sengur, A. (2010). Evaluation of ensemble methods for diagnosing of valvular heart disease. *Expert Systems with Applications*, 37:5110–5115.

Dasarathy, B. V. and Sheela, B. (1979). A composite classifier system design: Concepts and methodology. In *Proceedings of the IEEE*, volume 67, pages 708–713.

Demiröz, G. and Güvenir, A. (1997). Classification by voting feature intervals. In *Proceedings of the 9th European Conference on Machine Learning*, volume 1224, pages 85–92, London, UK. Springer-Verlag.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In Kittler, J. and Roli, F., editors, *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science*, volume 1857, pages 1–15, New York.

Dorigo, M. (1992). *Optimization, Learning and natural algorithms*. Ph.d. dissertation, Politecnico di Milano, Italy.

Dorigo, M. and Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *BioSystems*, 43:73–81.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. Wiley.

Džeroski, S. and Ženko, B. (2002). Stacking with multi-response model trees. In Roli, F. and Kittler, J., editors, *International workshop on multiple classifier systems*, volume 2364, pages 201–211. Springer.

Džeroski, S. and Ženko, B. (2004). Is combining classifiers better than selecting the best one? *Machine Learning*, 54:255–273.

Fan, W., Stolfo, S. J., Zhang, J., and Chan, P. K. (1999). Adacost: Misclassification cost-sensitive boosting. In *Proceedings of the Sixteenth International Conference on Machine Learning*, number 9, pages 97–105.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54.

Fern, X. Z. and Brodley, C. E. (2003). Random projection for high dimensional data clustering: A cluster ensemble approach. In *Machine Learning -International Workshop Then Conference*, pages 186–193.

Fogel, L. J. (1962). Toward inductive inference automata. In *Proceedings of the international federation for Information proceeding congress*, pages 395–399, Münich.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. Wiley, New York.

Frank, A. and Asuncion, A. (2010). UCI machine learning repository.

Frank, E. and Witten, I. H. (1998). Generating accurate rule sets without global optimization. In *Proceedings Of The 15th International Conference on Machine Learning*. Morgan Kaufmann.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference in Machine Learning*, pages 148–156.

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

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Science*, 8(1):156–166.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computer and Operation Research*, 13(5):533–549.

Goss, S., Aron, S., Deneubourg, J., and Pasteels, J. (1989). Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12):579–581.

Gutjahr, W. J. (2002). Aco algorithms with guaranteed convergence to the optimal solution. *Information processing letters*, 82(3):145–153.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explorations*, 11(1):10 –18.

Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 12(10):993–1001.

Ho, T. K., Hull, J. J., and Srihari, S. N. (1994). Decision combination in multiple classifier systems. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 16(1):66–75.

Holland, J. H. (1975). *Adaption in natural and artificial systems*. The University of Michigan Press, Ann Harbor, MI.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–90.

Iba, W. and Langley, P. (1992). Induction of one-level decision trees. In *Proceedings of the 9th international workshop on Machine learning*, pages 233–240, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

John, G. H. and Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kubat, M., Holte, R. C., and Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3):195–215.

Kuo, R., Lin, S., and Shih, C. (2007). Mining association rules through integration of clustering analysis and ant colony system for health insurance database in taiwan. *Expert Systems with Applications*, 33:794–808.

Le Cessie, S. and Van Houwelingen, J. C. (1992). Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201.

Ledezma, A., Aler, R., and Borrajo, D. (2002). *Heuristic Search-Based stacking of classifiers*, chapter Heuristic and Optimization for Knowledge Discovery, pages 54–67. Idea Group Publishing.

Ledezma, A., Aler, R., Sanchis, A., and Borrajo, D. (2004). Empirical evolution of optimized stacking configurations. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)'04*, pages 49–55. OLS2008.

Ledezma, A., Aler, R., Sanchis, A., and Borrajo, D. (2009). Ga-stacking: Evolutionary stacked generalization. *Intelligent Data Analysis*, 14:89–119.

Lorenço, H. R., Martin, O., and Stützle, T. (2002). *Iterated Local Search*. Handbook of Metaheuristics. Kluwer Academic Publishers, Norwell, MA.

Lu, Z., Wu, X., Zhu, X., and Bongard, J. (2010). Ensemble pruning via individual contribution ordering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 871–880, New York, NY, USA. ACM.

Merz, C. J. (1999). Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1-2):33–58.

Monmarché, N. (1999). On data clustering with artificial ants. In Freitas, A., editor, *AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions*, pages 23–26, Orlando, Florida.

Nanni, L. and Lumini, A. (2009). An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, 36:3028–3033.

Ngai, E., Hu, Y., Wong, Y., Chen, Y., and Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50(3):559–569.

Ordóñez, F. J., Ledezma, A., and Sanchis, A. (2008). Genetic approach for optimizing ensembles of classifiers. In *Proceedings of the Twenty-First International FLAIRS Conference*, pages 89–94.

Özbakir, L., Baykasoğlu, A., Kulluk, S., and Yapici, H. (2009). Taco-miner: An ant colony based algorithm for rule extraction from trained neural networks. *Expert Systems with Applications*, 36:12295–12305.

Parpinelli, R. S., Lopes, H. S., and Freitas, A. A. (2002). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 6:321–332.

Pinto, P. C., Nägele, A., Dejori, M., Runkler, T. A., and ao M.C. Sousa, J. (2009). Using a local discovery ant algorithm for bayesian network structure learning. *IEEE Transactions on Evolutionary Computation*, 13(4):767–779.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ramanathan, R. (2003). *An Introduction to Data Envelopment Analysis*. SAGE Publications.

Rechenberg, I. (1973). *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Seewald, A. K. (2002). How to make stacking better and faster while also taking care of an unknown weakness. In *Proceedings of the 19th International Conference on Machine Learning*, ICML '02, pages 554–561.

Sivagaminathan, R. K. and Ramakrishnan, S. (2007). A hybrid approach for feature subset selection using neural networks and ant colony optimization. *Expert Systems with Applications*, 33:49–60.

Strehl, A. and Ghosh, J. (2002). Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617.

Sttzle, T. and Hoos, H. (1996). Improving the ant system: A detailed report on the max-min ant system. Technical report.

Stützle, T. and Dorigo, M. (2002). A short convergence proof for a class of ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 6(4):358–365.

Sun, Y., Kamel, M. S., Wong, A. K., and Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Ting, K. M. and Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10(1):271–289.

Todorovski, L. and Džeroski, S. (2000). Combining multiple models with meta decision trees. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '00, pages 54–64. Springer Berlin / Heidelberg.

Turban, E., Sharda, R., and Delen, D. (2010). *Decision Support and Business Intelligence Systems*. Prentice Hall Press.

Černý (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

Wong, M.-L. and Cui, G. (2010). Data mining using parallel multi-objective evolutionary algorithms on graphics hardware. In *IEEE Congress on Evolutionary Computation*, pages 1–8.

Xu, L., Krzyzak, A., and Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transations on Systems, Man and Cybernetics*, 22(3):418–435.

Ying, K.-C., Lin, S.-W., and Lee, Z.-J. (2010). An ensemble approach applied to classify spam e-mails. *Expert Systems with Applications*, 37:2197–2201.

Yu, L., Wang, S., and Lai, K. K. (2008). Credit risk assessment with a multistage neural network ensemble learning approach. *Expert Systems with Applications*, 34:1434–1444.

Yu, Y., Zhou, Z.-H., and Ting, K. M. (2007). Cocktail ensemble for regression. In *Seventh IEEE International Conference on Data Mining*, pages 721–726.

Zahavi, J. and Levin, N. (1997). Applying neural computing to target marketing. *Journal of Direct Marketing*, 11(4):76–93.

Zhang, X., Chen, X., and He, Z. (2010). An aco-based algorithm for parameter optimization of support vector machines. *Expert Systems with Applications*, 37:6618–6628.

Zhang, Y., Burer, S., and Street, W. N. (2006). Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7:1315–1338.

Zhou, Z.-H., Wu, J.-X., Tang, W., and Chen, Z.-Q. (2001). Combining regression estimators: Ga-based selective neural network ensemble. *International Journal of Computational Intelligence and Applications*, 1(4):341–356.

Zhu, D. (2010). A hybrid approach for efficient ensembles. *Decision Support Systems*, 48(3):480–487.