

Diversity and Regularization in Neural Network Ensembles



Huanhuan Chen
School of Computer Science
University of Birmingham

A thesis submitted for the degree of

Doctor of Philosophy

October, 2008

Acknowledgements

First and foremost, my special thanks go to my supervisor, Prof. Xin Yao for his inspiration, enthusiasm and kindness on me. This thesis would never have been completed without his continuous help on both my study and life. I sincerely thank him for leading me into the interesting field of machine learning, and for his insightful supervision and encouragement. I have benefited greatly from his encouragement, wide knowledge, and clarity of thought in conducting this research.

The second thanks are given to Dr. Peter Tiño for kindly giving many valuable comments in numerous discussions and for clarification of ideas, which inspire me to greatly deepen my research work.

I am indebted to all those who have helped me with my work. I would like to thank my thesis group members, Dr. Peter Tiño, Dr. Jon Rowe and Dr. Russell Beale, who kept an eye on my work, took effort in reading my progress reports and provided me insightful comments.

Special acknowledgement is given to the examiners of the thesis, Prof. Richard Everson and Dr. Ata Kaban, for agreeing to be the examiners of my PhD Viva.

Aside of these, I thank the PhD student mates, for the lively discussions and the help along the way: Ping Sun, Arjun Chandra, Shuo Wang, Fernanda Minku, Conglun Yao, Siang Yew Chong, Pete Duell, Kata Praditwong and Trung Thanh Nguyen.

Finally, I would like to thank my parents and my wife Qingqing Wang. My parents always give me unconditional love, encouragement and support through all my life. My wife Qingqing Wang has provided me tremendous encouragement during my graduate study and numerous

supports in our life. There are hardly words enough to thank them for all they have done for me. This thesis is dedicated to them.

Abstract

In this thesis, we present our investigation and developments of neural network ensembles, which have attracted a lot of research interests in machine learning and have many fields of applications. More specifically, the thesis focuses on two important factors of ensembles: the diversity among ensemble members and the regularization.

Firstly, we investigate the relationship between diversity and generalization for classification problems to explain the conflicting opinions on the effect of diversity in classifier ensembles. This part proposes an ambiguity decomposition for *classifier* ensembles and introduces an ambiguity term, which is part of ambiguity decomposition, as a new measure of diversity. The empirical experiments confirm that ambiguity has the largest correlation with the generalization error in comparison with other nine most-often-used diversity measures. Then, an empirical investigation on the relationship between diversity and generalization has been conducted. The results show that diversity highly correlates with the generalization error only when diversity is low, and the correlation decreases when the diversity exceeds a threshold. These findings explain the empirical observations on whether or not diversity correlates with the generalization error of ensembles.

Secondly, this thesis investigates a special kind of diversity, error diversity, using negative correlation learning (NCL) in detail, and discovers that regularization should be used to address the overfitting problem of NCL. Although NCL has showed empirical success in creating neural network ensembles by emphasizing the error diversity, with the lack of a solid understanding of its dynamics we observe it is prone

to overfitting and we engage in a theoretical and empirical investigation to improve its performance by proposing regularized negative correlation learning (RNCL) algorithm. RNCL imposes an additional regularization term to the error function of the ensemble and then decomposes the ensemble’s training objectives into individuals’ objectives.

This thesis provides a Bayesian formulation of RNCL and implements RNCL by two techniques: gradient descent with Bayesian Inference and evolutionary multiobjective algorithm. The numerical results demonstrate the superiority of RNCL. In general, RNCL can be viewed as a *framework*, rather than an algorithm itself, meaning several other learning techniques could make use of it.

Finally, we investigate ensemble pruning as one way to balance diversity, regularization and accuracy, and we propose one probabilistic ensemble pruning algorithm in this thesis. We adopt a left-truncated Gaussian prior for this probabilistic model to obtain a set of sparse and non-negative combination weights. Due to the intractable integral by incorporating the prior, expectation propagation (EP) is employed to approximate the posterior estimation of the weight vector, where an estimate of the leave-one-out (LOO) error can be obtained without extra computation. Therefore, the LOO error is used together with Bayesian evidence for model selection. An empirical study shows that our algorithm utilizes far less component learners but performs as well as, or better than, the non-pruned ensemble.

The results are also positive when EP pruning algorithm is used to select the classifiers from the population, generated by multi-objective regularized negative correlation learning algorithm, to produce effective and efficient ensembles by balancing the diversity, regularization and accuracy.

Contents

Nomenclature	xviii
1 Introduction	1
1.1 Supervised learning	1
1.2 Ensemble of Learning Machines	4
1.3 Research Questions	5
1.3.1 Diversity in Classifier Ensembles	5
1.3.2 Regularized Negative Correlation Learning	7
1.3.3 Ensemble Pruning Methods	8
1.4 Contributions of the Thesis	9
1.5 Outline of the thesis	10
1.6 Publications resulting from the thesis	11
2 Background and Related Work	14
2.1 Ensemble of Learning Machines	14
2.1.1 Mixture of Experts	14
2.1.2 Bagging	15
2.1.3 Boosting-type Algorithms	16
2.1.4 Ensemble of Features	18
2.1.5 Random Forests	18
2.1.6 Ensemble Learning using Evolutionary Multi-objective Al- gorithm	19
2.2 Theoretical Analysis of Ensembles	20
2.2.1 Bias Variance Decomposition	20
2.2.2 Bias Variance Covariance Decomposition	21

2.2.3	Ambiguity Decomposition for Regression Ensembles	22
2.2.4	Diversity in Classifier Ensembles	23
2.3	Negative Correlation Learning Algorithm	26
2.4	Ensemble Pruning Methods	28
2.4.1	Selection based Ensemble Pruning	28
2.4.2	Weight based Ensemble Pruning	29
2.5	Summary	31
3	Diversity in Classifier Ensembles	32
3.1	Introduction	32
3.2	Ambiguity Decomposition for Classifier Ensembles	34
3.3	A New Diversity Measure	37
3.4	Correlation Between Diversity and Generalization	38
3.4.1	Visualization of Diversity Measures using Multidimensional Scaling	39
3.4.2	Correlation Analysis of Diversity Measures	45
3.5	Summary	51
4	Regularized Negative Correlation Learning	53
4.1	Introduction	53
4.2	Regularized Negative Correlation Learning	55
4.2.1	Negative Correlation Learning	55
4.2.2	Regularized Negative Correlation Learning	56
4.3	Bayesian Formulation and Regularized Parameter Optimization .	57
4.3.1	Bayesian Formulation of RNCL	58
4.3.2	Inference of Regularization Parameters	61
4.4	Numerical Experiments	63
4.4.1	Experimental Setup	63
4.4.2	Synthetic Experiments	65
4.4.3	Benchmark Results	69
4.4.4	Computational Complexity and Running Time	72
4.5	Summary	74

5	Multiobjective Regularized Negative Correlation Learning	75
5.1	Introduction	75
5.2	Multiobjective Regularized Negative Correlation Learning	77
5.2.1	Multiobjective Regularized Negative Correlation Learning	77
5.2.2	Component Network and Evolutionary Operators	79
5.2.3	Multiobjective Evaluation of Ensemble and Rank-based Fitness Assignment	81
5.2.4	Algorithm Description	83
5.3	Numerical Experiments	85
5.3.1	Experimental Setup	85
5.3.2	Synthetic Data Sets	85
5.3.3	Experimental Results on Noisy Data	93
5.3.4	Benchmark Results	93
5.3.5	Computational Complexity and Running Time	96
5.4	Summary	97
6	Predictive Ensemble Pruning by Expectation Propagation	99
6.1	Introduction	100
6.2	Sparseness-induction and Truncated Prior	102
6.3	Predictive Ensemble Pruning by Expectation Propagation	103
6.3.1	Expectation Propagation	103
6.3.2	Expectation Propagation for Regression Ensembles	104
6.3.2.1	Leave-one-out Estimation	109
6.3.3	Expectation Propagation for Classifier Ensembles	109
6.3.4	Hyperparameters Optimization for Expectation Propagation	111
6.3.5	Algorithm Description	112
6.3.6	Comparison of Expectation Propagation with Markov Chain Monte Carlo	113
6.4	Numerical Experiments	115
6.4.1	Synthetic Data Sets	116
6.4.2	Results of Regression Problems	117
6.4.3	Results of Classifier Ensembles	120
6.4.4	Computational Complexity and Running Time	128

6.5	Summary	131
7	Conclusions and future research	133
7.1	Conclusions	133
7.2	Future work	136
7.2.1	Reduce the Computational Complexity of EP Pruning . .	136
7.2.2	Theoretical Analysis of Ensemble	137
7.2.3	Semi-supervised Regularized Negative Correlation Learning	137
7.2.4	Multi-objective Ensemble Learning	138
A	Diversity Measures	140
A.1	Pairwise Diversity Measures	140
A.2	Non-pairwise Diversity Measures	142
B	Further Details of RNCL using Bayesian Inference	145
B.1	Further Details of Gaussian Posterior	145
B.2	Details of Parameter Updates	146
C	Further Details of Hyperparameters Optimization in EP	148
	References	162

List of Figures

2.1	Mixture of Experts	15
2.2	Bagging Algorithm	16
2.3	Adaboost-type Algorithm (Rätsch et al., 2001)	17
2.4	Negative Correlation Learning Algorithm	25
3.1	In MDS algorithm, residual variance vs. number of dimensions on credit card problem. The other problems yield similar plots and are omitted only to save space.	39
3.2	2D MDS plot using normalized scores (left) and standard deviation scaling (right) for credit card problem. The 10 measures of diversity are: AM-Ambiguity, Q - Q statistics, K - Kappa statistics, CC - correlation coefficient, Dis - disagreement measure, E - entropy, KW - Kohavi-Wolpert variance, Diff - measure of difficulty, GD - generalized diversity, CFD - coincident failure diversity and Err - generalization error for the six data sets. The x and y axes are coordinates of these diversity measures in 2D space.	40
3.3	2D MDS plot (10 diversity measures and generalization error) using normalized method on six data set. The results are averaged on the 100 run on each data set. The x and y axes are coordinates of these diversity measures in 2D space.	42

3.4	2D MDS plot using rank correlation coefficients. This figure is averaged on six data sets. The 10 measures of diversity are: AM-Ambiguity, Q - Q statistics, K - Kappa statistics, CC - correlation coefficient, Dis - disagreement measure, E - entropy, KW - Kohavi-Wolpert variance, Diff - measure of difficulty, GD - generalized diversity, CFD - coincident failure diversity and Err - generalization error. The x and y axes are coordinates of these diversity measures in 2D space.	45
3.5	Accuracy, Diversity, Q statistics, Entropy, Generalized Diversity (<i>GD</i>) and Generalization Error with different sampling rates (from 0.1 to 1) of Bagging for six data sets. The x-axis is the sampling rate r . The plot interval of sampling rate r from 0.1 to 0.9 is 0.05 and plot the interval between 0.9 and 1 is 0.01. The left y-axis is to record the values of Accuracy, Diversity and Generalization Error; the right y-axis is for Q statistics, Entropy and Generalized Diversity (<i>GD</i>). The results are averaged on 100 runs of 5-fold cross validation.	47
4.1	Regularized Negative Correlation Learning Algorithm	58
4.2	Comparison of NCL and RNCL on regression data sets: Sinc and Friedman test. In Figure 4.2(a) and 4.2(b), the lines in green (wide zigzag), black (dashed) and red (solid) are obtained by RNCL, NCL and the true function, respectively. Figure 4.2(c) and 4.2(d) show mean square error (MSE) of RNCL (red solid) and NCL (blue dashed) on Sinc and Friedman with different noise levels. Results are based on 100 runs.	64
4.3	Comparison of RNCL and NCL on four synthetic classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. The lines in green (light) and black (dark) are obtained by RNCL and NCL, respectively.	66

4.4	Comparison of RNCL and NCL on two classification data sets. Two classes are shown as crosses and dots. The separating lines are obtained by projecting test data over a grid. In Figure 4.4(a) and 4.4(b), the decision boundary in green (light) and black (dark) are obtained by RNCL and NCL, respectively. The randomly-selected noise points are marked with a circle. Figure 4.4(c) and 4.4(d) show the error rate of RNCL (red solid) and NCL (blue dashed) vs. the noise levels on Synth and banana data sets. The results are based on 100 runs.	68
5.1	Multiobjective Regularized Negative Correlation Learning Algorithm	83
5.2	Comparison of MRNCL and MNCL on four synthetic classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. The lines in green (thin) and black (thick) are obtained by MRNCL and MNCL, respectively.	84
5.3	Detailed information in multiobjective algorithm for two data sets, Banana and Overlap. In Figure 5.3(a) and 5.3(c), the left-y axis (red line with circles) measures the summation of the mean of three objectives, training error, regularization and correlation in different generations. The right-y axis (blue line with triangles) is the standard deviation of the summation. In Figure 5.3(b) and 5.3(d), the 3D figure records the mean value of these three objectives in different generations. The arrow points from the beginning (Generation = 1) to end (Generation = 100). The color represents different generations. Blue points stands for small generations and red points mean large generations.	86

5.4	Detailed information in multiobjective algorithm for two data sets, bumpy and relevance. In Figure 5.4(a) and 5.4(c), the left-y axis (red line with circles) measures the summation of the mean of three objectives, training error, regularization and correlation in different generations. The right-y axis (blue line with triangles) is the standard deviation of the summation. In Figure 5.4(b) and 5.4(d), the 3D figure records the mean value of these three objectives in different generations. The arrow points from the beginning (Generation = 1) to end (Generation = 100). The color represents different generations. Blue points stands for small generations and red points mean large generations.	87
5.5	Illustration the trade-off among the three objectives: training error, regularization and correlation, in the final population for four synthetic classification data sets. The color represents different correlations. Blue points stands for low correlations and red points mean large correlations.	88
5.6	2D view of the trade-off between two objectives: training error and regularization for four synthetic classification data sets. The color represents different training errors. Blue points stands for low training errors and red points mean large training errors. . . .	89
5.7	2D view of the trade-off between two objectives: training error and correlation for four synthetic classification data sets. The color represents different training errors. Blue points stands for low training errors and red points mean large training errors.	90
5.8	Comparison of MRNCL and MNCL on two classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. In Figure 5.8(a) and 5.8(b), the decision boundary in green (thin) and black (thick) are obtained by MRNCL and MNCL, respectively. The randomly-selected noise points are marked with a circle. Figure 5.8(c) and 5.8(d) show classification error of MRNCL (red solid) and MNCL (blue dashed) vs. noise levels on synth and banana data sets. The results are based on 100 runs.	92

LIST OF FIGURES

6.1	The truncated Gaussian Prior	102
6.2	The posteriors of combination weights calculated by MCMC (30000 sampling points) and EP. The color bar indicates the density (the number of overlapping points) in each place.	114
6.3	Comparison of EP-pruned ensembles and un-pruned Bagging ensembles on sinc data set. The sinc data set is generated by sampling 100 data points with 0.1 Gaussian noise from the sinc function. The Bagging ensemble consists of 100 neural networks with random selected hidden nodes (3-6 nodes).	117
6.4	Comparison of EP-pruned ensembles and un-pruned Adaboost ensembles on Synth and banana data sets. The Adaboost ensemble consists of 100 neural networks with random selected hidden nodes (3-6 nodes).	118
6.5	Comparison of evaluation time of each pruning method averaged.	131

List of Tables

3.1	Summary of Data Sets	38
3.2	Rank correlation coefficients (in %) between the diversity measures based on the average of the six data sets. The measures are: Am - Ambiguity; Q statistics; K - Kappa statistics; CC - correlation coefficient; Dis - disagreement measure; E - entropy; KW - Kohavi-Wolpert variance; Diff - measure of difficulty; GD - generalized diversity; and CFD - coincident failure diversity.	44
3.3	Rank correlation coefficients (in %) among ambiguity, nine diversity measures and Generalization Error in Different Sampling Range, where G stands for generalization error.	48
3.4	The generalization error of Bagging algorithms with different sampling rates, where $r = 0.632$ is the performance of Bagging with bootstrap. The results are averaged over 50 runs of 5 fold cross validation.	49
4.1	Summary of Regression Data Sets	69
4.2	Summary of Classification Data Sets.	70
4.3	Comparison of NCL, Bagging and RNCL on 8 Regression Data Sets, by MSE (standard deviation) and t test p value between Bagging vs. RNCL and NCL vs. RNCL. The p value with a star means the test is significant. These results are averages of 100 runs.	70
4.4	Comparison of NCL, Bagging and RNCL on 13 benchmark Data Sets, by % error (standard deviation) and t test p value between Bagging vs. RNCL and NCL vs. RNCL. The p value with a star means the test is significant. These results are averages of 100 runs.	71

LIST OF TABLES

4.5	Running Time (in seconds) of RNCL and NCL on Regression Data Sets. Results are averaged over 100 runs.	72
4.6	Running Time (in seconds) of RNCL and NCL on Classification Data Sets. Results are averaged over 100 runs.	72
4.7	Comparison of RNCL and NCL with equal time on four regression problems and four classification problems. NCL is run 10 times in 8 experiments with randomly selected regularization parameters between 0 and 1. The first row reports the best performance of NCL in the 10 runs. The results are the average results of 20 runs.	73
5.1	Comparison among the six methods on 13 benchmark Data Sets: Single RBF classifier, MRNCL, MNCL, Adaboost, Bagging, and support vector machine. Estimation of generalization error in % on 13 data sets (best method in bold face). The columns P_1 to P_4 show the results of a significance test (95% t-test) between MRNCL and MNCL, Adaboost, Bagging and SVM, respectively. The p value with a star means the test is significant. The performance is based on 100 runs (20 runs for Splice and Image). MRNCL gives the best overall performance.	94
5.2	Running Time of MRNCL and MNCL on 13 Data Sets in seconds. Results are averaged over 100 runs.	96
6.1	The pruned ensemble size, error rate and computational time of MCMC, EP and unpruned ensembles.	116
6.2	Average Test MSE, Standard Deviation for seven regression Benchmark Data sets based on 100 runs for Bagging. EP, ARD, LS, Random stand for EP pruning, ARD pruning, least square pruning and random pruning, respectively.	118
6.3	Size of Pruned Ensemble with standard deviation for Different Algorithms for Bagging. The results are based on 100 runs.	119

LIST OF TABLES

6.4	Average Test error, Standard Deviation for 13 classification Benchmark Data sets based on 100 runs for Bagging algorithm. EP, ARD, Kappa, CP, LS, Random stand for EP pruning, ARD pruning, kappa pruning, concurrency pruning, least square pruning and random pruning.	120
6.5	Size of Pruned Ensemble with standard deviation with Different Algorithms for Bagging. The results are based on 100 runs. . . .	121
6.6	Average Test error, Standard Deviation for 13 classification Benchmark Data sets based on 100 runs for Adaboost algorithm. EP, ARD, Kappa, CP, LS, Random stand for EP pruning, ARD pruning, kappa pruning, concurrency pruning, least square pruning and random pruning.	122
6.7	Size of Pruned Ensemble with standard deviation with Different Algorithms for Adaboost. The results are based on 100 runs. . . .	123
6.8	Average Test error, Standard Deviation for 13 classification Benchmark Data sets based on 100 runs for Random forests algorithm. EP, ARD, Kappa, CP, LS, Random stand for EP pruning, ARD pruning, kappa pruning, concurrency pruning, least square pruning and random pruning.	124
6.9	Size of Pruned Ensemble with standard deviation with different algorithms for random forest. The results are based on 100 runs. .	125
6.10	Average Test error, Standard Deviation for 13 classification Benchmark Data sets based on 100 runs for MRNCL algorithm. EP, ARD, Kappa, CP, LS, Random stand for EP pruning, ARD pruning, kappa pruning, concurrency pruning, lease square pruning and random pruning.	126
6.11	Size of Pruned Ensemble with standard deviation with different algorithms for MRNCL. The results are based on 100 runs. . . .	127
6.12	Running Time of EP pruning, ARD pruning and EM pruning on Regression Data Sets in seconds. Results are averaged over 100 runs.	128
6.13	Running Time of EP pruning, ARD pruning, EM pruning, Kappa pruning and concurrency pruning on Classification Data Sets in seconds. Results are averaged over 100 runs.	129

LIST OF TABLES

6.14	Summary of EP, EM, ARD, LS, Kappa, CP, random and other unpruned ensembles with poker hand problem (Train points 25010 and Test points 1 mil.). The results are averaged over ten runs.	130
A.1	A 2×2 table of the relationship between a pair of classifiers f_i and f_j .	140

Chapter 1

Introduction

This chapter introduces the problems addressed in this thesis and gives an overview of subsequent chapters. Section 1.1 describes the problem of supervised learning and some basic learning theory. In section 1.2, we introduce ensembles of learning machines and highlight their distinct advantages compared to classical machine learning techniques. Section 1.3 describes the research questions of the thesis. Section 1.4 summarizes a selection of the significant contributions made by the author. Section 1.5 concludes this chapter with an overview of the subjects addressed in each subsequent chapter.

1.1 Supervised learning

Supervised learning is a machine learning technique for learning a function from training data. The training data consist of pairs of input variables and desired outputs. Depending on the nature of the outputs, supervised learning can be classified as either regression for continuous outputs or classification when outputs are discrete.

There are many practical problems which can be effectively modeled as the learning of input-output mappings from some given examples. An example of a regression problem would be the prediction of house price in one city in which the inputs may consist of average income of residents, house age, the number of bedrooms, populations and crime rate in that area, etc. A best known classifi-

cation example is the handwritten character recognition, which has been used in many areas.

The task of supervised learning is to use the available training examples to construct a model that can be used to predict the targets of unseen data, which are assumed to follow the same probability distribution as the available training data. The predictive capability of the learned model is evaluated by the generalization ability from the training examples to unseen data. One possible definition of supervised learning (Vapnik, 1998) is presented as follows:

Definition 1 (*Vapnik, 1998*) *The problem of supervised learning is to choose from the given set of functions $f \in F : X \longrightarrow Y$ based on a training set of random independent identically distributed (i.i.d.) observations drawn from an unknown probability distribution $P(\mathbf{x}, y)$,*

$$D = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in X \times Y, \quad (1.1)$$

such that the obtained function $f(\mathbf{x})$ best predicts the supervisor's response for unseen examples (\mathbf{x}, y) , which are assumed to follow the same probability distribution $P(\mathbf{x}, y)$ as the training set.

The standard way to solve the supervised learning problem consists in defining a loss function, which measures the loss or discrepancy associated with the learning machine, and then choosing the learning machine from the given set of candidates with the lowest loss. Let $V(y, f(\mathbf{x}))$ denotes a loss function measuring the error when we predict y by $f(\mathbf{x})$, then the average error, the so called *expected risk*, is:

$$R[f] = \int_{X,Y} V(y, f(\mathbf{x})) P(\mathbf{x}, y) d\mathbf{x} dy. \quad (1.2)$$

Based on equation (1.2), the ideal model f_0 can be obtained by selecting the function with the minimal *expected risk*:

$$f_0 = \arg \min_{f \in F} R[f]. \quad (1.3)$$

However, as the probability distribution $P(\mathbf{x}, y)$ that defines the expected risk is unknown, this ideal function cannot be found in practice. To overcome this shortcoming we need to learn from the limited number of training data we

have. One popular way is to use the *empirical risk minimization* (ERM) principle (Vapnik, 1998), which approximately estimates the expected risk based on the available training data.

$$R_{erm}[f] = \frac{1}{N} \sum_{n=1}^N V(y, f(\mathbf{x}_n)). \quad (1.4)$$

However, straightforward minimization of the *empirical risk* might lead to over-fitting, meaning a function that does very well on the finite training data might not generalize well to unseen examples (Bishop, 1995; Vapnik, 1998). To address the over-fitting problem, the technique of regularization, which adds a regularization term $\Omega[f]$ to the original objective function $R_{emp}[f]$, is often employed.

$$R_{reg}[f] = R_{erm}[f] + \lambda \Omega[f]. \quad (1.5)$$

The regularization term $\Omega[f]$ controls the smoothness or simplicity of the function and the regularization parameter $\lambda > 0$ controls the trade-off between the empirical risk $R_{emp}[f]$ and the regularization term $\Omega[f]$.

The generalization ability of the learner depends crucially on the parameter λ , especially with small training sets. One approach to choose the parameter is to train several learners with different values of the parameter and estimate the generalization error for each learner and then choose the parameter λ that minimizes the estimated generalization error.

Fortunately, there is a superior alternative to estimate the regularization parameter: Bayesian inference. Bayesian inference makes it possible to efficiently estimate the regularization parameters. Compared with the traditional approaches, Bayesian approach is attractive in being logically consistent, simple, and flexible. The application of Bayesian inference to single neural network, introduced by MacKay as a statistical approach to avoid overfitting (MacKay, 1992a,b), was successful. Then, the Bayesian technique has been successfully applied in Least Squares Support Vector Machine (Gestel et al., 2002), RBF neural networks (Husmeier and Roberts, 1999), sparse Bayesian Learning, i.e., Relevance Vector Machine (Tipping, 2001).

1.2 Ensemble of Learning Machines

An ensemble of learning machines is using a set of learning machines to learn partial solutions to a given problem and then integrating these solutions in some manner to construct a final or complete solution to the original problem. Using f_1, \dots, f_M to denote M individual learning machines, a common example of ensemble for regression problem is

$$f_{ens}(x) = \sum_{i=1}^M w_i f_i(x), \quad (1.6)$$

where $w_i > 0$ is the weight of the estimator f_i in the ensemble.

This method is sometimes called *committee of learning machines*. In classification case, it is also called multiple classifier systems (Ho et al., 1994), classifier fusion (Kuncheva, 2002), combination, aggregation, etc. We will refer to an individual learning machine as the *base learner*. Note that there are some approaches using a number of base learners to accomplish a task in a style of *divide-and-conquer*. In those approaches, the base learners are in fact trained for different sub-problems instead of for the same problem, which makes those approaches usually be classified into mixture of experts (Jordan and Jacobs, 1994). However, in the ensemble, base learners are all attempting to solve the same problem.

Ensemble methods have been widely used to improve the generalization performance of the single learner. This technique originates from Hansen and Salamon's work (Hansen and Salamon, 1990), which showed that the generalization ability of a neural network can be significantly improved through ensembling a number of neural networks.

Based on the advantages of ensemble methods and increasing complexity of real-world problems, ensemble of learning machines is one of the important problem-solving techniques. Since the last decade, there have been much literature published on ensemble learning algorithms, from *Mixtures of Experts* (Jordan and Jacobs, 1994), *Bagging* (Breiman, 1996a) to various *Boosting* (Schapire, 1999), *Random Subspace* (Ho, 1998), *Random Forests* (Breiman, 2001) and *Negative Correlation Learning* (Liu and Yao, 1997, 1999b), etc.

The simplicity and effectiveness of ensemble methods take the role of key selling point in the current machine learning community. Successful applications

of ensemble methods have been reported in various fields, for instance in the context of handwritten digit recognition (Hansen et al., 1992), face recognition (Huang et al., 2000), image analysis (Cherkauer, 1996) and many more (Diaz-Uriarte and Andres, 2006; Viola and Jones, 2001).

1.3 Research Questions

This thesis focuses on two important factors of ensembles: diversity and regularization. Diversity among the ensemble members is one of the keys to the success of ensemble models. Regularization improves the generalization performance by controlling the complexity of the ensemble.

In the thesis, firstly, we investigate the relationship between diversity and generalization for classification problems and then we investigate a special kind of diversity, error diversity, using negative correlation learning (NCL) in detail, and discover that regularization should be used to address the overfitting problem of NCL. Finally, we investigate ensemble pruning as one way to balance diversity, regularization and accuracy and we propose one probabilistic ensemble pruning algorithm.

The details are presented in the following.

1.3.1 Diversity in Classifier Ensembles

It is widely believed that the success of ensemble methods greatly depends on creating diverse sets of learner in the ensemble, demonstrated by theoretical (Hansen and Salamon, 1990; Krogh and Vedelsby, 1995) and empirical studies (Chandra and Yao, 2006b; Liu and Yao, 1999a).

The empirical results reveal that the performance of an ensemble is related to the diversity among individual learners in the ensemble and better performance might be obtained with more diversity (Tang et al., 2006). For example, Bagging (Breiman, 1996a) relies on bootstrap sampling to generate diversity; Random forests (Breiman, 2001) employ both bootstrap sampling and randomization of features to produce more diverse ensembles, and thus the performance of random forests is better than that of Bagging (Breiman, 2001).

In some other empirical results (García et al., 2005; Kuncheva and Whitaker, 2003), diversity did not show much correlation with generalization when varying the diversity in the ensemble. These findings are counterintuitive since ensembles of many identical classifiers perform no better than a single classifier and ensembles should benefit from diversity.

Although diversity in an ensemble is deemed to be a key factor to the performance of ensembles (Brown et al., 2005a; Darwen and Yao, 1997; Krogh and Vedelsby, 1995) and many studies on diversity have been conducted, our understanding of diversity in classifier ensembles is still incomplete. For example, there is less clarity on how to define diversity for classifier ensembles and how diversity correlates with the generalization ability of ensemble (Kuncheva and Whitaker, 2003).

As we know, the definition of diversity for regression ensembles originates from the ambiguity decomposition (Krogh and Vedelsby, 1995), in which the error of regression ensemble is broken into two terms: the accuracy term measuring the weighted average error of the individuals and the ambiguity term measuring the difference between ensemble and individual estimators. There is *no ambiguity decomposition for classifier ensembles* with *zero-one* loss. Therefore, it is still an open question on how to define an appropriate measure of diversity for classifier ensembles (Giacinto and Roli, 2001; Kohavi and Wolpert, 1996; Partridge and Krzanowski, 1997).

Based on these problems, the thesis focuses on the following questions:

1. How to define the diversity for classifier ensembles?
2. How does diversity correlate with generalization error?

In chapter 3, we propose *ambiguity decomposition for classifier ensembles* with *zero-one* loss function, where the ambiguity term is treated as the diversity measure. The correlation between generalization and diversity (with 10 different measures including ambiguity) has been examined. The relationship between ambiguity and other diversity measures has been studied as well.

1.3.2 Regularized Negative Correlation Learning

This thesis studies one specific kind of diversity, error diversity, using Negative Correlation Learning (NCL) (Liu and Yao, 1999a,b), which emphasizes the interaction and cooperation among individual learners in the ensemble and has performed well on a number of empirical applications (Liu et al., 2000; Yao et al., 2001).

NCL explicitly manages the error diversity of an ensemble by introducing a correlation penalty term into the cost function of each individual network so that each network minimizes its MSE error together with the correlation with other ensemble members.

According to the definition of NCL, it seems that the correlation term in the cost function acts as the regularization term. However, we observe that NCL corresponds to training the ensemble as a single learning machine by considering only the empirical training error. Although NCL can use the penalty coefficient to explicitly alter the emphasis on the individual MSE and correlation portion of the ensemble, setting a zero or smaller coefficient corresponds to independently training the estimators and thus loses the advantages of NCL. In most cases NCL sets the penalty coefficient to be or close to the particular value which corresponds to training the entire ensemble as a single learning machine.

By training the entire ensemble as a single learner and minimizing the MSE error without regularization, NCL only reduces the empirical MSE error of the ensemble, but pays less attention to regularizing the complexity of the ensemble. As we know, neural network and other machine learning algorithms which only rely on the empirical MSE error are prone to overfitting the noise (Krogh and Hertz, 1992; Vapnik, 1998). Based on the above analysis, the formulation of NCL leads to over-fitting. In this thesis, we analyze this problem and provide the theoretical and empirical evidences.

Another issue of NCL is that there is no formulated approach to select the penalty parameter, though it is crucial for the performance of NCL. Optimization of the parameter usually involves cross validation, whose computation is extremely expensive.

In order to address these problems, the regularization should be used to address the overfitting problem of NCL and the thesis proposes regularized negative correlation learning (RNCL) by including an additional regularization term in the cost function of the ensemble. RNCL is implemented by two techniques, gradient descent with Bayesian inference and evolutionary multiobjective algorithm, in this thesis. Both techniques improve the performance of NCL by regularizing the NCL and automatically optimizing the parameters. The details are discussed in chapters 4 and 5.

In general, regularization is important to other ensemble methods as well. For example, the boosting algorithm, Arcing, generates a larger margin distribution than AdaBoost but performs worse (Breiman, 1999). This is because Arcing does not regularize the complexity of the base classifiers and thus degrades its performance (Reyzin and Schapire, 2006). Similarly, Bagging prefers combining *simple* or weak learners than complicated learners to succeed (Breiman, 1996a; Buhlmann and Yu, 2002).

Based on the analysis, regularization controls the complexity of ensembles and is another important factor for ensembles besides diversity.

1.3.3 Ensemble Pruning Methods

Most existing ensemble methods generate large ensembles. These large ensembles consume much more memory to store all the learning models, and they take much more computation time to get a prediction for a *fresh* data point. Although these extra costs may seem to be negligible with a small data set, they may become serious when the ensemble method is applied to a large scale real-world data set.

In addition, it is not always true that the larger the size of an ensemble, the better it is. Some theoretical and empirical evidences have shown that small ensembles can be better than large ensembles (Breiman, 1996a; Yao and Liu, 1998; Zhou et al., 2002).

For example, the boosting ensembles, Adaboost (Schapire, 1999) and Arcing (Breiman, 1998, 1999), pay more attention to those training samples that are misclassified by former classifiers in the training of next classifier and finally reduce the training error to zero. In this way, the former classifiers, with large

training error, may under-fit the data, while the latter classifiers, with low training error and weak regularization, are prone to overfitting the noise in the training data. The trade-off among diversity, regularization and accuracy in the ensemble is unbalanced and thus Boosting ensembles are prone to overfitting (Dietterich, 2000; Opitz and Maclin, 1999) . In these circumstances, it is necessary to prune some individuals to achieve better generalization.

The evolutionary ensemble learning algorithms often generate a number of learners in the population. Some are good at accuracy; some have better diversity and others pay more attention to regularization. In this setting, we had better select a subset of learners to produce an effective and efficient ensemble by balancing diversity, regularization and accuracy.

Motivated by the above reasons, this thesis investigates ensemble pruning as one way to balance diversity, regularization and accuracy and the thesis proposes one probabilistic ensemble pruning algorithm.

1.4 Contributions of the Thesis

The thesis includes a number of significant contributions to the field of neural network ensembles and machine learning.

1. The first theoretical analysis of ambiguity decomposition for *classifier* ensembles, in which a new definition of diversity measure for *classifier* ensembles has been given. The superiority of the diversity measure has been verified against nine other diversity measures (chapter 3).
2. Empirical work demonstrating the correlation between diversity (with ten different diversity measures) and generalization, which exhibits that diversity highly correlates with the generalization error only when diversity is low, but the correlation decreases when the diversity exceeds a threshold (chapter 3).
3. The first theoretical and empirical analysis demonstrating that negative correlation learning (NCL) is prone to overfitting (chapter 4).

4. A novel cooperative ensemble learning algorithm, regularized negative correlation learning (RNCL), derived from both NCL and Bayesian inference, which generalizes better than NCL. Moreover, the coefficient controlling the trade-off between empirical error and regularization in RNCL can be inferred by Bayesian inference (chapter 4).
5. An evolutionary multiobjective algorithm implementing RNCL, which searches for the best trade-off among the three objectives, to design effective ensembles. (chapter 5).
6. A new probabilistic ensemble pruning algorithm to select the component learners for more efficient and effective ensembles. Moreover, we have conducted a thorough analysis and empirical comparison of different combining strategies. (chapter 6).

1.5 Outline of the thesis

This chapter briefly introduced some preliminaries for subsequent chapters: supervised learning and ensemble of learning machines. We also described the research questions of this work and summarized the main contributions of this thesis.

Chapter 2 reviews the literature on some popular ensemble methods and their variants. Section 2.2 introduced three important theoretical results for ensemble learning, the *bias-variance* decomposition, *bias-variance-covariance* decomposition and *ambiguity* decomposition. We also review the current literature on the analysis of diversity in classifier ensembles. In section 2.3 we briefly describe negative correlation learning (NCL) algorithm and further review its board extensions and applications. After that, we move to investigate the most commonly used techniques for selecting a set of learners to generate the ensemble.

Chapter 3 concentrates on analyzing the generic classifier ensemble system from the accuracy/diversity viewpoint. We propose an ambiguity decomposition for *classifier* ensembles with *zero-one* loss and introduce the ambiguity term, which is part of ambiguity decomposition, as the definition of diversity. Then,

the proposed diversity measure together with other nine diversity measures have been employed to study the relationship between diversity and generalization.

In chapter 4, we study one specific kind of diversity, error diversity, using negative correlation learning (NCL) and discover that regularization should be used. The chapter analyzes NCL in-depth and explains the reasons why NCL is prone to overfitting the noise. Then, we propose regularized negative correlation learning (RNCL) in Bayesian framework and provide the algorithm to infer the regularization parameters based on Bayesian inference. The numerical experiments have been conducted to evaluate RNCL, NCL and other ensemble learning algorithms.

Chapter 5 extends the work in chapter 4 by incorporating evolutionary multi-objective algorithm with RNCL to design regularized and cooperative ensembles. The training of RNCL, where each individual learner needs to minimize three terms: empirical training error, correlation and regularization, is implemented by a three-objective evolutionary algorithm. The numerical studies have been conducted to evaluate this algorithm with many other approaches.

Chapter 6 investigates ensemble pruning as one way to balance diversity, regularization and accuracy, and proposes a probabilistic ensemble pruning algorithm. This chapter evaluates this algorithm with many other strategies. The corresponding training algorithms and empirical analysis have been presented.

Chapter 7 summarizes this work and describes several potential studies for future research.

1.6 Publications resulting from the thesis

The work resulting from these investigations has been reported in the following publications:

Refereed & Submitted Journal Publications

- [1] (Chen et al., 2009b) H. Chen, P. Tiño and X. Yao. *Probabilistic Classification Vector Machine*. IEEE Transactions on Neural Networks, vol.20, no.6, pp.901-914, June 2009.

1.6 Publications resulting from the thesis

- [2] (Chen et al., 2009a) H. Chen, P. Tiño and X. Yao. *Predictive Ensemble Pruning by Expectation Propagation*. IEEE Transactions on Knowledge and Data Engineering, vol.21, no.7, pp.999-1013, July 2009.
- [3] (Yu et al., 2008) L. Yu, H. Chen, S. Wang and K. K. Lai. *Evolving Least Squares Support Vector Machines for Stock Market Trend Mining*. IEEE Transactions on Evolutionary Computation. Vol 13, No. 1, Feb 2009.
- [4] (Chen and Yao, 2009b) H. Chen and X. Yao. *Regularized Negative Correlation Learning for Neural Network Ensembles*. IEEE Transactions on Neural Networks, In Press, 2009.
- [5] (Chen and Yao, 2009a) H. Chen and X. Yao. *Multiobjective Neural Network Ensembles based on Regularized Negative Correlation Learning*. IEEE Transactions on Knowledge and Data Engineering, In Press, 2009.
- [6] (Chen and Yao, 2008) H. Chen and X. Yao. *When Does Diversity in Classifier Ensembles Help Generalization?* Machine Learning Journal, 2008. In Revise.

Book chapter

- [7] (Chandra et al., 2006) A. Chandra, H. Chen and X. Yao. *Trade-off between Diversity and Accuracy in Ensemble Generation*. In Multiobjective Machine Learning, Yaochu Jin (Ed.), pp.429-464, Springer-Verlag, 2006. (ISBN: 3-540-30676-5)

Refereed conference publications

- [8] (He et al., 2009) S. He, H. Chen, X. Li and X. Yao. *Profiling of mass spectrometry data for ovarian cancer detection using negative correlation learning*. In Proceedings of the 19th International Conference on Artificial Neural Networks (ICANN'09), Cyprus, 2009.

- [9] (Chen and Yao, 2007a) H. Chen and X. Yao. *Evolutionary Random Neural Ensemble Based on Negative Correlation Learning*. In Proceedings of IEEE Congress on Evolutionary Computation (CEC'07), pages 1468-1474, 2007
- [10] (Chen et al., 2006) H. Chen, P. Tiño and X. Yao. *A Probabilistic Ensemble Pruning Algorithm*. Workshops of Sixth IEEE International Conference on Data Mining (WICDM'06), pages 878-882, 2006.
- [11] (Chen and Yao, 2007b) H. Chen and X. Yao. *Evolutionary Ensemble for In Silico Prediction of Ames Test Mutagenicity*. In Proceedings of International Conference on Intelligent Computing (ICIC'07), pages 1162-1171, 2007.
- [12] (Chen and Yao, 2006) H. Chen and X. Yao. *Evolutionary Multiobjective Ensemble Learning Based on Bayesian Feature Selection*. In Proceedings of IEEE Congress on Evolutionary Computation (CEC'06), volume 1141, pages 267-274, 2006.

Chapter 2

Background and Related Work

This chapter reviews the literature related to this thesis. In section 2.1, we summarize some major ensemble methods. Section 2.2 describes some common approaches to analyze ensemble methods. Section 2.3 investigates the existing applications and developments of negative correlation learning algorithm in the literature. This is followed by a review of techniques specifically for combining and selecting ensemble members in section 2.4.

2.1 Ensemble of Learning Machines

Neural network ensemble, which originates from Hansen and Salamon’s work (Hansen and Salamon, 1990), is a learning paradigm where a collection of neural networks is trained for the same task. There have been many ensemble methods studied in the literature. In the following, we review some popular ensemble learning methods.

2.1.1 Mixture of Experts

Mixture of Experts (MoE) is a widely investigated paradigm for creating a combination of learners (Jacobs et al., 1991). The principle of MoE is that certain experts will be able to “specialize” to particular parts of the input space by adopting a gating network who is responsible for learning the appropriate weighted combination of the specialized experts for any given input. In this way the input

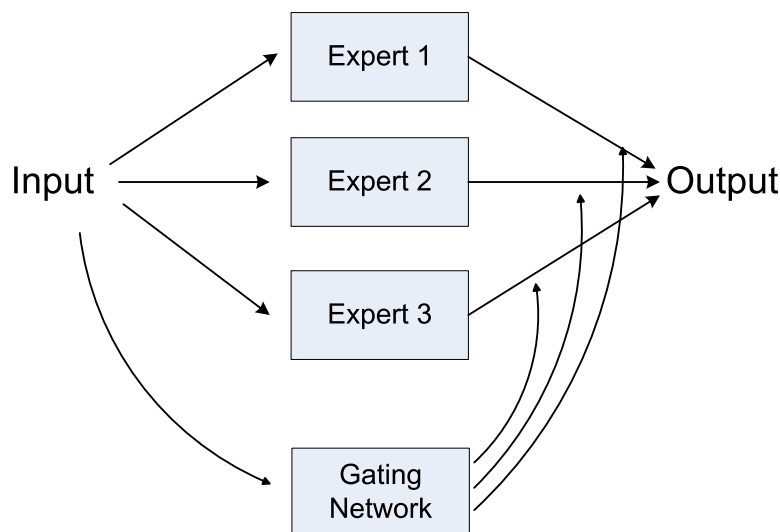


Figure 2.1: Mixture of Experts

space is *divided and conquered* by the gating network and these experts. Figure 2.1 illustrates the basic architecture of MoE.

Since the original paper on MoE (Jacobs et al., 1991), a huge number of variants of this paradigm have been developed (Ebrahimpour et al., 2007). In (Jordan and Jacobs, 1994) the idea was extended with a hierarchical mixture models. The Expectation-Maximization (EM) algorithm was employed for adjusting the parameters of MoE. Recent work on MoE included theoretical development on MoE (Ge and Jiang, 2006) and quadratically gated mixture of experts for incomplete data (Liao et al., 2007).

2.1.2 Bagging

Bagging (short for *Bootstrap Aggregation Learning*) is proposed by Breiman (Breiman, 1996a) based on bootstrap sampling. In a Bagging ensemble, each base learner is trained with a set of n training samples, drawn randomly with replacement from the original training set of size n with a uniform distribution. The resampled sets are often called bootstrap replicates of the original set. Breiman (Breiman, 1996a) showed that on average 63.2% of the original training set will

Algorithm Bagging

Input: the training set $\mathbf{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, integer M specifying size of ensemble.

For $i = 1, \dots, M$ **do:**

1. Bootstrap uniformly the original training set with replacement and generate a new training set with N items.
2. Train a learner f_i with the new training set and include it to the ensemble.

Output: Bagging ensemble

$$f(\mathbf{x}) = \frac{1}{M} \sum_i f_i(\mathbf{x}).$$

Figure 2.2: Bagging Algorithm

present in each replicate. Predictions on new samples are made by simple averaging. The algorithm of Bagging is presented in Figure 2.2.

For unstable base learners such as decision trees, Bagging works very well, but the explanation for its success remains unclear. Friedman (Friedman, 1997) suggested that Bagging succeeded by reducing the variance and leaving the bias unchanged, while Grandvalet (Grandvalet, 2004) showed experimental evidence that Bagging stabilized prediction by equalizing the influence of training examples.

2.1.3 Boosting-type Algorithms

Bagging resamples the dataset randomly with a uniform probability distribution while Boosting (Schapire, 1990) has a non-uniform distribution. There are a large family of Boosting algorithms in the literature, including cost-sensitive version (Fan et al., 1999) and Arcing-type algorithms (Breiman, 1999) that do not weigh the votes in the combination of classifiers.

We take the most widely investigated variant AdaBoost (Schapire, 1999) for an example. In the context of classification, the main idea of AdaBoost is to introduce weights on the training set. They are used to control the importance of each single sample for learning a new classifier. Those training samples that are

Algorithm AdaBoost-type

Input: the training set $\mathbf{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, integer M specifying number of iterations.

Initialize $w_1(\mathbf{x}_n) = 1/N$ for all $n = 1, \dots, N$.

For $i = 1, \dots, M$ **do**:

1. Train classifier with respect to the weighted sample set $\{\mathbf{D}, \mathbf{w}_i\}$ and obtain classifier f_i .
2. Calculate the training error e_i of f_i :

$$e_i = \sum_{n=1}^N w_i(\mathbf{x}_n) I(f_i(\mathbf{x}_n) \neq y_i),$$

abort if $e_i = 0$ or $e_i \geq \phi$, where ϕ is a constant.

3. Set

$$b_i = \log \frac{\phi(1 - e_i)}{e_i(1 - \phi)}.$$

4. Update the weights \mathbf{w}_i :

$$\mathbf{w}_{i+1} = \mathbf{w}_i \exp\{-b_i I(f_i(\mathbf{x}_n) = y_i)\} / Z_i,$$

where Z_i is a normalization factor to ensure that $\sum_{n=1}^N w_{i+1}(\mathbf{x}_n) = 1$.

Output: Booting ensemble

$$f(x) = \sum_{i=1}^M c_i f_i(x), \text{ where } c_i = \frac{b_i}{\sum_{i=1}^M |b_i|}.$$

Figure 2.3: Adaboost-type Algorithm (Rätsch et al., 2001)

misclassified by former classifiers will play a more important role in the training of next classifier. After the desired number of base classifiers has been trained, they are combined by a weighted vote, based on their training error. The Adaboost-type algorithm is presented in Figure 2.3.

It is widely believed (Breiman, 1999; Rätsch et al., 2001) that Adaboost approximately maximizes the *hard* margins of the training samples. Thus, the classical Adaboost algorithm is prone to overfitting the noise (Dietterich, 2003). To overcome this shortcoming, the *soft*-margin Adaboost (Rätsch et al., 2001) is implemented by maximizing the *soft* margins of the training samples, which al-

lows a particular level of noise in the training set and exhibits better performance than *hard* margin Adaboost.

For the regression case, relatively few papers have addressed boosting-type ensembles. The major difficulty is rigorously defining regression problem in an infinite hypothesis space. Rätsch proposed a novel approach based on a semi-infinite linear program that has an infinite number of constraints and a finite number of variables. They also provided some beautiful theoretical results and promising empirical results (Rätsch et al., 2002). The drawbacks of this algorithm come from the use of sophisticated linear programming techniques and costly computations.

2.1.4 Ensemble of Features

Apart from randomly sampling the training points, ensemble of features (Ho, 1998) samples different subsets of features to train ensemble members (Ho, 1998; Optiz, 1999).

Liao et al. built ensembles based on different feature subsets (Liao and Moody, 1999). In their approach, all input features are firstly grouped based on mutual information. Statistically similar features are assigned to the same group. Each base learner's training set is then formed by input features extracted from different groups. Some feature boosting algorithms (Song et al., 2006; Yin et al., 2005) have been proposed as well. Ensemble of features have been applied to drug design (Mamitsuka, 2003) and medical diagnosis (Tsymbal et al., 2003).

Most of the existing ensemble feature methods claim better results than traditional methods (Oliveira et al., 2005; O'Sullivan et al., 2000), especially when the data set has a large number of features and not too few samples (Ho, 1998).

2.1.5 Random Forests

Random forests (Breiman, 2001) combine Bootstrap sampling and random subspace method to generate decision forests. It consists of a number of decision trees, of which each tree is trained with the examples bootstrap sampled from the training set. In training each tree, a randomly-selected subset of features is

used to split each node. Random forests perform similarly as Adaboost in terms of error rate, and it is more robust with respect to noise.

Due to its simple characteristic and good generalization ability, random forests have a lot of applications, such as protein prediction (Chen and Liu, 2005) and classification of geographic data (Gislason et al., 2006).

2.1.6 Ensemble Learning using Evolutionary Multi-objective Algorithm

The success of ensemble methods depends on a number of factors, such as accuracy, diversity, generalization, cooperation and so on. Most of the existing ensemble algorithms implicitly encourage these terms. The recent research has demonstrated (Chandra and Yao, 2004; García et al., 2005; Oliveira et al., 2005) that the explicit encouragement of these terms by an evolutionary multiobjective algorithm is beneficial to ensemble design. The related work is reviewed as follows.

Diverse and accurate ensemble learning algorithm (DIVACE) (Chandra and Yao, 2006a,b) is an approach that emerges evolving neural network and multi-objective algorithm. In this paper, adaptive Gaussian noise on weights was used to generate the offspring and mimetic pareto neural network algorithm (Abbass, 2000) was used to evolve neural networks. Finally, diverse and accurate ensembles can be achieved through these procedures.

Then, Oliveira et al. (Oliveira et al., 2005) incorporated ensemble of feature and multi-objective algorithm. This algorithm can be divided to two levels. The first is to create a set of classifiers which have small number of features and low error rate, which is achieved by evolving these classifiers with *randomly*-chosen features. In the second level, the combination weights of ensemble are obtained by a multi-objective algorithm with two different objectives: diversity and accuracy.

Cooperative coevolution of neural network ensembles (García et al., 2005) combined both the coevolution and evolutionary multiobjective algorithm to design neural network ensembles. In this algorithm, the cooperation terms were defined as objectives. Every network was evaluated by a multi-objective method.

Thus, the algorithm encourages the collaboration among ensemble and improves the combination schemes of ensembles.

2.2 Theoretical Analysis of Ensembles

Developing theoretical foundations for ensemble learning is a key step towards understanding and applying it. Till now, there are many works that try to tackle this problem, such as *bias-variance* decomposition, *bias-variance-covariance* decomposition and *ambiguity* decomposition. The section reviews these techniques. As the error diversity, which can be directly or indirectly derived from these decompositions, is an important component in ensemble models, this section also reviews the analysis and application of diversity for classifier ensembles.

2.2.1 Bias Variance Decomposition

In the last decade, machine-learning research preferred more sophisticated representations than simple learners. However, more powerful learners do not guarantee better performance, and sometimes very simple learners outperform sophisticated ones, e.g., (Domingos and Pazzani, 1997; Holte, 1993).

The reason for this phenomenon has become clear after the *bias-variance* decomposition is proposed. In the decomposition, the predictive error consists of two components, bias and variance, and while more powerful learners reduce one (bias) they increase the other (variance). As a result of these developments, the bias-variance decomposition has become a cornerstone for our understanding of supervised learning.

The original bias-variance decomposition is proposed by Geman et al. (Geman et al., 1992). It applies to quadratic loss error functions, and states that the generalization error can be broken down into bias and variance terms. The bias-variance decomposition can be obtained as follows if we assume a noise level of zero in the testing data

$$E\{(f(\mathbf{x}) - y)^2\} = (E\{f(\mathbf{x})\} - y)^2 + E\{(f(\mathbf{x}) - E\{f(\mathbf{x})\})^2\}, \quad (2.1)$$

where the expectation $E\{\cdot\}$ is with respect to all possible training sets.

The first term, $(E\{f(\mathbf{x})\} - y)^2$, is the bias component, measuring the average distance between the output of the learner and its target. The variance term, $E\{(f(\mathbf{x}) - E\{f(\mathbf{x})\})^2\}$ is the average squared distance of its possible values from the expected values. There is a trade-off between these two terms and attempting to reduce the bias term will cause an increase in variance, and vice versa. The optimal trade-off between the bias and variance varies from application to application. Machine learning approaches are often evaluated on how well they can optimize the trade-off between these two components (Wahba et al., 1999).

However, different tasks may require different loss functions and lead several decomposition schemes. As a result, several authors have proposed bias-variance decompositions with *zero-one* loss for classification problems (Domingos, 2000; James, 2003; Kohavi and Wolpert, 1996). However, each of these decompositions has significant shortcomings. In particular, none has a clear relationship to the original decomposition for squared loss. Since in the original decomposition, the decomposition is purely additive (i.e., $loss = bias + variance$). But none has provided the similar result for *zero-one* loss using definitions of bias and variance that both have the intuitive meanings.

2.2.2 Bias Variance Covariance Decomposition

The bias-variance decomposition is mainly applicable to a single learner. In the following work (Brown et al., 2005b; Islam et al., 2003; Liu and Yao, 1999a,b; Ueda and Nakano, 1996), the decomposition is extended to take account of the possibility that the estimator could be an ensemble of estimators. The bias-variance-covariance decomposition states that the squared error of ensemble can be broken into three terms, bias, variance and covariance. The *bias-variance-covariance* decomposition is presented as follows:

$$E\{(f_{ens} - y)^2\} = \overline{bias} + \frac{1}{M}\overline{var} + (1 - \frac{1}{M})\overline{covar}, \quad (2.2)$$

where

$$\begin{aligned}\overline{bias} &= \left(\frac{1}{M} \sum_i (E_i\{f_i\} - y) \right)^2, \overline{var} = \frac{1}{M} \sum_i E_i \{ (y - E_i\{f_i\})^2 \}, \\ \overline{covar} &= \frac{1}{M(M-1)} \sum_i \sum_{j \neq i} E_{i,j} \{ (f_i - E_i\{f_i\})(f_j - E_j\{f_j\}) \},\end{aligned}$$

and $f_{ens} = \frac{1}{M} \sum_i f_i$. The expectation E_i is with respect to the training set T_i that is used to train the learner f_i .

The error of an ensemble not only depends on the bias and variance of the ensemble members, but also depends critically on the amount of error correlation among these base learners, quantified in the covariance term. This *bias-variance-covariance* decomposition also provides the theoretical grounding of negative correlation learning which takes amount of correlation together with the empirical error in training neural networks.

2.2.3 Ambiguity Decomposition for Regression Ensembles

Based on the bias-variance decomposition, Krogh and Vedelsby (Krogh and Vedelsby, 1995) gave the ambiguity decomposition which proved that at a single data point the quadratic error of the ensemble estimator is guaranteed to be less than or equal to the average quadratic error of the component estimators.

Ensemble is a variance-reduction technique. The amount of reduction in the variance term is proportional to the correlation among individual estimators' error, commonly referred to as diversity. Ambiguity decomposition is a significant technique to quantify the diversity term for regression ensembles (Brown et al., 2005b).

The ambiguity decomposition of regression ensembles proves that for a single arbitrary data point, the quadratic error of the ensemble estimator can be decomposed into two terms:

$$(f_{ens}(\mathbf{x}) - y)^2 = \sum_i^M c_i (f_i(\mathbf{x}) - y)^2 - \sum_i^M c_i (f_i(\mathbf{x}) - f_{ens}(\mathbf{x}))^2, \quad (2.3)$$

where y is the target output of a data point, c_i are the combination weights which satisfy $c_i \geq 0$, $\sum_{i=1}^M c_i = 1$, and $f_{ens}(\cdot)$ is a convex combination of the component

estimators:

$$f_{ens}(\mathbf{x}) = \sum_{i=1}^M c_i f_i(\mathbf{x}). \quad (2.4)$$

The first term, $\sum_i c_i (f_i(\mathbf{x}) - y)^2$, is the weighted average error of the individuals. The second, $\sum_i c_i (f_i(\mathbf{x}) - f_{ens}(\mathbf{x}))^2$ is the ambiguity term, measuring the amount of variability among ensemble members, which is treated as diversity. As this ambiguity term is always positive, the error of ensemble is guaranteed lower than the average individual error.

The ambiguity decomposition is an encouraging result for regression ensembles. However, it is not applicable to classifier ensembles with *zero-one* loss. In this thesis, we propose an ambiguity decomposition with *zero-one* loss for classifier ensembles and derive a new measure of diversity for classifier ensembles from the proposed ambiguity decomposition.

2.2.4 Diversity in Classifier Ensembles

The success of ensemble methods depends on generating accurate yet diverse individual learners, because ensemble of many identical learners will not perform better than a single learner.

The empirical results reveal that the performance of ensemble is related with the diversity among individual learners in the ensemble and better performance might be achieved with more diversity (Tang et al., 2006). For example, Bagging (Breiman, 1996a) relies on bootstrap sampling to generate diversity; Random forests (Breiman, 2001) employ both bootstrap sampling and randomization of feature to produce more diverse ensembles, and the performance of random forests is better than that of Bagging (Breiman, 2001).

Based on these empirical results, some researchers try to improve the performance of ensemble by increasing the diversity of ensemble (Liu and Yao, 1999a,b; Liu et al., 2000). Some research reports the positive results (Chandra and Yao, 2006a; Oliveira et al., 2005). For example, Chandra and Yao (Chandra and Yao, 2006a) reported positive results by encouraging diversity in their multi-objective evolutionary algorithms. However, some other studies cannot confirm the benefits of more diversity in the ensemble (García et al., 2005; Kuncheva and Whitaker,

2003). For example, in order to examine the relationship between diversity and generalization, Kuncheva et al. (Kuncheva and Whitaker, 2003) varied the diversity of ensemble to observe the change of the generalization of ensemble, and stated that it was not clear that the use of diversity terms had a beneficial effect on the ensemble. This observation was partially supported by (García et al., 2005), in which the experimental results showed that the performance of their algorithm was not clearly improved when the defined diversity objectives: correlation, functional diversity, mutual information and Q statistics, were considered in their evolutionary multi-objective algorithm. These contradictory results raise a lot of interest in exploring the relationship between the generalization and diversity in the ensemble.

In general, although diversity is deemed as an important factor of ensemble, there is less clarity on how to define the diversity for *classifier* ensembles and how diversity correlates with the generalization of ensembles.

In order to analyze the relationship between generalization and diversity, firstly we need to define and quantify diversity for classifier ensembles. This is straightforward for regression ensembles since ambiguity decomposition (Krogh and Vedelsby, 1995) gives the most acceptable definition of diversity for regression ensembles.

As the *zero-one* loss function employed in classifier ensembles is different from the mean-square error (MSE), there is no ambiguity decomposition for classifier ensembles. How to define an appropriate measure of diversity for classifier ensembles is still in debate. Until now, there are many proposals of diversity measures for classifier ensembles. These definitions could be grouped into two categories: pairwise diversity measures, which are based on the measurement of any pairwise classifiers, e.g. Q statistics (Yule, 1900), Kappa statistics (Dietterich, 2000), correlation coefficient (Sneath and Sokal, 1973), disagreement measure (Ho, 1998) and non-pairwise diversity measures, e.g. entropy measure (Cunningham and Carney, 2000), Kohavi-Wolpert variance (Kohavi and Wolpert, 1996), measure of difficulty (Hansen and Salamon, 1990), generalized diversity (Partridge and Krzanowski, 1997) and coincident failure diversity (Partridge and Krzanowski, 1997). These diversity measures have been detailed in appendix A.

2.3 Negative Correlation Learning Algorithm

Algorithm Negative Correlation Learning (NCL)

Input: the training set $\mathbf{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, integer M specifying size of ensemble, the learning rate η in backpropagation (BP) algorithm and integer T specifying the number of iterations.

For $t = 1, \dots, T$ **do**:

1. Calculate $f_{ens}(\mathbf{x}_n) = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}_n)$.
2. For each network from $i = 1$ to M do: for each weight $w_{i,j}$ in network i , perform a desired number of updates,

$$\begin{aligned}
 e_i &= \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n)^2 - \lambda \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2, \\
 \frac{\partial e_i}{\partial w_{i,j}} &= 2 \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n) \frac{\partial f_i}{\partial w_{i,j}} - 2\lambda \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \left(1 - \frac{1}{M}\right) \frac{\partial f_i}{\partial w_{i,j}}, \\
 \Delta w_{i,j} &= -2\eta \left\{ \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n) \frac{\partial f_i}{\partial w_{i,j}} - \lambda \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \left(1 - \frac{1}{M}\right) \frac{\partial f_i}{\partial w_{i,j}} \right\}.
 \end{aligned}$$

Output: NCL ensemble

$$f(\mathbf{x}) = \frac{1}{M} \sum_i f_i(\mathbf{x}).$$

Figure 2.4: Negative Correlation Learning Algorithm

Although these diversity measures could be used to represent the relationship among a group of classifiers, most of them do not have exact mathematical form in relation to the ensemble error, and this makes it difficult to analyze the relationship between generalization and diversity. As there are many diversity measures, it is not easy to select an appropriate one without knowing the relationship among them. Inspired by regression ensembles, this thesis proposes an *ambiguity decomposition for classifier ensembles* and takes the ambiguity term as the diversity measure for classifier ensembles. Then, ten diversity measures have been employed to study the relationship between generalization and diversity.

2.3 Negative Correlation Learning Algorithm

In this section, we briefly describe negative correlation learning (NCL) algorithm and review the related literature on the applications and developments of NCL algorithm. Finally, we present the potential problems of NCL algorithm.

Negative Correlation learning (Liu and Yao, 1997, 1999a,b; Liu et al., 2000) is a successful neural network ensemble learning algorithm, which is different from the previous work such as Bagging or Boosting. It emphasizes interaction and cooperation among the base learners in the ensemble, and uses an unsupervised penalty term in the error function to produce biased learners whose errors tend to be negatively correlated. NCL explicitly manages the *error diversity* in the ensembles.

Given the training set $\{\mathbf{x}_n, y_n\}_{n=1}^N$, NCL combines M neural networks $f_i(\mathbf{x})$ to constitute the ensemble.

$$f_{ens}(\mathbf{x}_n) = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}_n). \quad (2.5)$$

To train network f_i , the error function e_i of network i is defined by

$$e_i = \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n)^2 + \lambda p_i, \quad (2.6)$$

where λ is a weighting parameter on the penalty term p_i :

$$\begin{aligned} p_i &= \sum_{n=1}^N \left\{ (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \sum_{j \neq i} (f_j(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \right\} \\ &= - \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2. \end{aligned} \quad (2.7)$$

The first term in the right-hand side of (2.6) is the empirical training error of network i . The second term p_i is a correlation penalty function. The purpose of minimizing p_i is to negatively correlate each network's error with errors of the rest ensemble members. The λ parameter controls a trade-off between the training error term and the penalty term. With $\lambda = 0$, we would have an ensemble with each network training with plain back propagation, exactly equivalent to training

2.3 Negative Correlation Learning Algorithm

a set of networks independently of one another. If λ is increased, more and more emphasis would be placed on minimizing the penalty. NCL is implemented by a gradient descent method. The algorithm is summarized in Figure 2.4.

Since the original paper on NCL, a large number of applications and developments of this paradigm have been developed. Islam et al. (Islam et al., 2003) took a constructive approach to build the ensemble, starting from a small group of networks with minimal architecture. The networks are all partially trained using negative correlation learning.

In the following work, Brown et al. (Brown et al., 2005b) formalized this technique and provided a statistical interpretation of its success. Furthermore, for estimators that are linear combinations of other functions, they derived an upper bound on the penalty coefficient, based on properties of Hessian matrix.

In (García et al., 2005), negative correlation learning is combined with co-evolution and evolutionary multiobjective algorithm to design neural network ensembles. In this algorithm, the cooperation term with the rest of the networks were defined as objectives. Every network was evaluated in the evolutionary process using a multi-objective method. Thus, the algorithm encourages the collaboration among ensemble and improves the combination scheme of ensemble.

Chen et al. (Chen and Yao, 2007a) proposed to incorporate bootstrap of data, random feature subspace (Ho, 1998) and evolutionary algorithm with negative correlation learning to automatically design neural network ensembles. The idea promotes the diversity within the ensemble and simultaneously emphasizes the accuracy and cooperation in the ensemble.

In (Dam et al., 2008), NCL was employed in the learning classifier systems to train neural network ensembles, where NCL was shown to improve the generalization of the ensemble.

Although NCL takes the correlation of the ensemble into consideration and succeeds in the practical problems, it has potential risk of over-fitting (Chen and Yao, 2009a,b). It is observed that NCL corresponds to training the ensemble as a single learning machine by considering only the empirical training error without regularization.

The thesis analyzes this problem and proposes the theoretical and empirical evidences. In order to solve this problem, we propose regularized negative correla-

tion learning (RNCL) algorithm which incorporates an additional regularization term for the ensemble. Then we describe two techniques, gradient descent with Bayesian inference and evolutionary multiobjective algorithm, to implement the RNCL. The details of both implementations are detailed in chapter 4 and 5.

2.4 Ensemble Pruning Methods

The goal of ensemble pruning is to reduce the size of ensemble without compromising its performance. The pruning strategy for a group of learners is of fundamental importance, and can decide the performance of the whole system. As described in chapter 1, ensemble pruning can be viewed as one way to reduce the size of ensemble and balance the diversity, accuracy and generalization in the ensembles at the same time. The pruning algorithms can be classified into two categories, selection-based and weight-based pruning algorithms. In the following, we review the two kinds of strategies, respectively.

2.4.1 Selection based Ensemble Pruning

The selection-based ensemble pruning algorithms do not weigh each learner by a weighting coefficient, and they either select or reject the learner.

A straightforward method is to rank the learners according to their individual performance on a validation set and pick the best ones (Chawla et al., 2004). This simple approach may sometimes work well but is theoretically unsound. For example, an ensemble of three identical classifiers with 95% accuracy may be worse than an ensemble of three classifiers with 67% accuracy and least pairwise correlated error.

Margineantu et al. (Margineantu and Dietterich, 1997) proposed four heuristic approaches to prune ensembles generated by Adaboost. Of them, KL-divergence pruning (Margineantu and Dietterich, 1997) and Kappa pruning (Margineantu and Dietterich, 1997) aim at maximizing the pair-wise difference between the selected ensemble members. Kappa-error convex hull pruning (Margineantu and Dietterich, 1997) is a diagram-based heuristic targeting at a good accuracy-divergence trade-off among the selected subsets. Back-fitting pruning (Margineantu

and Dietterich, 1997) essentially enumerates all the possible subsets, which is computationally too costly for large ensembles. Then, Prodromidis et al. invented several pruning algorithms for their distributed data mining system (Chan et al., 1999; Prodromidis and Chan, 1998). One of the two algorithms they implemented is based on a diversity measure they defined, and the other is based on the class specialty metrics.

The major problem with the above algorithms is that they all resort to greedy search, which is usually without either theoretical or empirical quality guarantees.

2.4.2 Weight based Ensemble Pruning

The more general weight-based ensemble optimization aims at improving the generalization performance of the ensemble by tuning the weight on each ensemble member.

For regression ensembles, the optimal combination weights minimizing the mean square error (MSE) can be calculated analytically (Hashem, 1993; Krogh and Vedelsby, 1995; Perrone, 1993; Zhou et al., 2002). The study has been covered in other research areas as well, such as financial forecasting (Clemen, 1989), and operational research (Bates and Granger, 1969). According to (Hashem, 1993), the optimal weights can be obtained as:

$$w_i = \frac{\sum_{j=1}^M (\mathbf{C}^{-1})_{ij}}{\sum_{k=1}^M \sum_{j=1}^M (\mathbf{C}^{-1})_{kj}}, \quad (2.8)$$

where \mathbf{C} is the correlation matrix with elements indexed as $C_{ij} = \int p(\mathbf{x})(f_i(\mathbf{x}) - y)(f_j(\mathbf{x}) - y)dx$ that is the correlation between the i^{th} and the j^{th} component learner, wherein $p(\mathbf{x})$ is the distribution of \mathbf{x} . The correlation matrix \mathbf{C} cannot be computed analytically without knowing the distribution $p(\mathbf{x})$ but can be approximated with a training set, as follows:

$$C_{ij} \approx \frac{1}{N} \sum_{n=1}^N [(y_n - f_i(\mathbf{x}_n))(y_n - f_j(\mathbf{x}_n))]. \quad (2.9)$$

However, this approach rarely works well in real-world applications. This is because when a number of estimators are available, there are often some estimators that are quite similar in performance, which makes the correlation matrix

C ill-conditioned, hampering the least square estimation. Other issues of this formulation include (1) the optimal combination weights are computed from the training set, which often over-fits the noise and (2) in most cases the optimal solution does not reduce the ensemble size.

The least square formulation is a numerical stable algorithm to calculate these optimal combination weights. In this thesis, we use the *least square (LS) pruning* to minimize MSE in our experiments to act as a baseline algorithm. The LS pruning is applicable to binary classification problems by modeling the classification problem as a regression problem with its target as -1 or +1.

However, the LS pruning often produce negative combination weights. A strategy allowing negative combination weights is believed to be unreliable (Benediktsson et al., 1997; Ueda, 2000).

To prevent the weights from negative values, Yao et al. (Yao and Liu, 1998) proposed to use genetic algorithm (GA) to weigh the ensemble members by constraining the weighs to be positive. Then, Zhou et al. (Zhou et al., 2002) proved that small ensembles can be better than large ensembles. A similar genetic algorithm approach can be found in (Kim et al., 2002). However, these GA based algorithms try to obtain the optimal combination weights by minimizing the training error and in this way these algorithms become sensitive to noise.

Then, Demiriz et al. (Demiriz et al., 2002) employed mathematical programming to look for good weighting schemes. Those optimization approaches are effective in performance enhancement according to empirical results and are sometimes able to significantly reduce the ensemble size (Demiriz et al., 2002). However, ensemble size reduction is not explicitly built into those programs and the final size of the ensemble can still be very large in some cases.

In fact, the weighted-based ensemble pruning can be viewed as a sparse Bayesian learning problem by applying Tipping’s relevance vector machine (RVM) (Tipping, 2001). RVM is an application of Bayesian automatic relevance determination (ARD) and it prunes most of the ensemble members by employing a Gaussian prior and updating the hyperparameters in an iterative way. However, *ARD pruning* does allow negative combination weights and the solution was not optimal according to the current research (Benediktsson et al., 1997; Ueda, 2000).

To address the problem of *ARD* pruning, Chen et al. (Chen et al., 2006) modeled the ensemble pruning as a probabilistic model with truncated Gaussian prior for both regression and classification problems. The Expectation-Maximization (EM) algorithm is used to infer the combination weights and our algorithm shows good performance in both generalization error and pruned ensemble size.

2.5 Summary

This chapter provided a review on ensemble of learning machines from the following four aspects: (i) some popular ensemble learning algorithms; (ii) three generalization decompositions for analyzing ensemble models and the analysis of diversity in classifier ensembles; (iii) some developments and applications of negative correlation learning algorithm; (iv) a number of methods for ensemble pruning. For the first point, we studied the current techniques on ensemble learning and their advantages and disadvantages. The second point introduced three important theoretical results for ensemble learning, the *bias-variance* decomposition, *bias-variance-covariance* decomposition and *ambiguity* decomposition, which are fundamental to our understanding of ensemble models. We also reviewed the current literature on the analysis and application of diversity in classifier ensembles. The third point reviewed the current development and the wide applications of one specific ensemble learning algorithm, negative correlation learning and pointed out the potential problems, which ignite the explosion of regularized negative correlation learning technique in this thesis. At the last point, we summarized various selection-based and weight-based algorithms for ensemble pruning.

Chapter 3

Diversity in Classifier Ensembles

In chapter 2 we reviewed a number of decompositions for analyzing supervised learning models and ensemble models, where all of the decompositions are only applicable to regression problems. In this chapter we propose ambiguity decomposition for *classifier* ensembles and focus on two research questions: how to define the diversity for classifier ensembles and how diversity correlates with generalization error. In this chapter, section 3.2 proposes an ambiguity decomposition for *classifier* ensembles and section 3.3 derives a new diversity measure based on the proposed ambiguity decomposition. The experiments and analysis on the correlation between diversity and generalization are presented in section 3.4, followed by the summary in section 3.5. In appendix A, we detail other nine diversity measures.

3.1 Introduction

In ensemble research, it is widely believed that the success of ensemble algorithms depends on both the accuracy and diversity among individual learners in the ensemble, demonstrated by theoretical (Hansen and Salamon, 1990; Krogh and Vedelsby, 1995) and empirical studies (Liu and Yao, 1999b). In general, the component learners in an ensemble are designed to be accurate yet diverse. For example, Bagging (Breiman, 1996a) relies on bootstrap sampling to produce diverse subsets of the training set to train each individual learner. Boosting

(Schapire, 1999) employs the feedback scheme to pay more attention to the training samples that are misclassified by the former classifiers in the training of next classifier and thus promotes the diversity among these base learners; Negative correlation learning (NCL) (Liu and Yao, 1999a,b; Liu et al., 2000) optimizes the trade-off between accuracy and diversity in this ensemble.

The empirical results reveal that the performance of an ensemble is related with the diversity among individual learners in the ensemble and better performance might be achieved with more diversity (Tang et al., 2006). Many related research on analysis and applications of diversity have been conducted (Giacinto and Roli, 2001; Kohavi and Wolpert, 1996; Partridge and Krzanowski, 1997).

As we know, the definition of diversity for regression ensembles originates from ambiguity decomposition (Krogh and Vedelsby, 1995), in which the error of regression ensemble is broken into two terms: the accuracy term measuring the weighted average error of the individuals and the ambiguity term measuring the difference among ensemble and component estimators. However, there is no *ambiguity decomposition for classifier ensembles*. Therefore, how to define an appropriate measure of diversity for classifier ensembles is still an open question (Giacinto and Roli, 2001; Kohavi and Wolpert, 1996; Partridge and Krzanowski, 1997).

Although diversity among ensemble members is deemed to be a key factor to the performance of ensemble (Brown et al., 2005a; Darwen and Yao, 1997; Krogh and Vedelsby, 1995) and many studies on diversity have been conducted, there is less clarity on how to define the diversity for *classifier* ensembles and how diversity correlates with the generalization of ensemble (Kuncheva and Whitaker, 2003).

Kuncheva et al. and Garcia et al., in their empirical results (García et al., 2005; Kuncheva and Whitaker, 2003), raised some doubts about the usefulness of diversity measures in building classifier ensembles because the empirical results did not show much correlation between diversity and generalization error by varying the diversity in the ensemble. These findings are counterintuitive since ensembles of many identical classifiers perform no better than a single classifier and ensembles should benefit from diversity.

3.2 Ambiguity Decomposition for Classifier Ensembles

The focus of the chapter is to provide answers to the following questions: (i) how to define the diversity for classifier ensembles and (ii) how diversity correlates with generalization error.

In order to answer the first question, section 3.2 proposes an *ambiguity decomposition for classifier ensembles* with *zero-loss* loss function, where the error is broken into two terms: accuracy and ambiguity. We follow the definition of diversity for regression ensembles and define the diversity for classifier ensembles by the ambiguity term.

To address the second question, by taking Bagging as an example of ensemble methods, this chapter conducts empirical experiments to explore the relationship between diversity and generalization by varying the diversity of Bagging. The originality is that by varying the sampling rate r (from 0.1 to 1) of Bagging, i.e. sample $100r\%$ data from the original training set to train each component classifier, to tune the diversity, we could observe the diversity from zero, where $r = 1$, to a large value, when $r = 0.1$. This is very helpful in understanding the relationship between diversity and generalization error. The relationship among ambiguity measure and other diversity measures has been studied as well.

3.2 Ambiguity Decomposition for Classifier Ensembles

The definition of diversity for regression ensembles is derived from the ambiguity decomposition. Ambiguity decomposition of regression ensembles (Krogh and Vedelsby, 1995) proves that for a single arbitrary data point, the quadratic error of the ensemble estimator can be decomposed into two terms:

$$(f_{ens}(\mathbf{x}) - y)^2 = \sum_i^M c_i (f_i(\mathbf{x}) - y)^2 - \sum_i^M c_i (f_i(\mathbf{x}) - f_{ens}(\mathbf{x}))^2, \quad (3.1)$$

where y is the target output of a data point, c_i are the combination weights which satisfy $c_i \geq 0$, $\sum_{i=1}^M c_i = 1$, and f_{ens} is a convex combination of component estimators:

$$f_{ens}(\mathbf{x}) = \sum_{i=1}^M c_i f_i(\mathbf{x}). \quad (3.2)$$

3.2 Ambiguity Decomposition for Classifier Ensembles

The first term, $\sum_i c_i(f_i(\mathbf{x}) - y)^2$, is the weighted average error of the individuals. The second, $\sum_i c_i(f_i(\mathbf{x}) - f_{ens}(\mathbf{x}))^2$ is the ambiguity term measuring the amount of variability among ensemble members. As this ambiguity term is always positive, the ensemble error is guaranteed to be lower than the average individual error.

The ambiguity decomposition is an encouraging result for regression ensembles with quadratic loss. However, it is not applicable to classifier ensembles with *zero-one* loss. In order to define an appropriate diversity measure for classifier ensembles, we follow the idea of regression ensembles and present an ambiguity decomposition for classifier ensembles with *zero-one* loss function.

Suppose the classification task is to use an ensemble comprising M component classifiers to approximate a function $f : R^D \rightarrow Y$, where Y is the set of class labels and D is the dimension of the data, and the predictions of the component classifiers are combined through majority voting where each component classifier votes for a class and the class label receiving the most number of votes is regarded as the output of the ensemble¹.

In this chapter we only consider binary classification, i.e. $Y \in \{-1, +1\}$. Now assume there are N instances $\{\mathbf{x}_n, y_n\}_{n=1}^N$. For a single arbitrary data point \mathbf{x}_n , y_n denotes the target output of this instance. $f_i(\mathbf{x}_n)$ is the actual output of the i^{th} component classifier with data point \mathbf{x}_n . y_n and $f_i(\mathbf{x}_n)$ satisfy the relationship that $y_n \in \{-1, +1\}$ and $f_i(\mathbf{x}_n) \in \{-1, +1\}$ ($i = 1, \dots, M$), respectively. It is obvious that if the actual output of the i^{th} component classifier is correct according to the target output then $f_i(\mathbf{x}_n)y_n = +1$, otherwise $f_i(\mathbf{x}_n)y_n = -1$. The output of the ensemble is defined as:

$$f_{ens}(\mathbf{x}_n) = \text{sign} \left(\sum_{i=1}^M c_i f_i(\mathbf{x}_n) \right). \quad (3.3)$$

For classifier ensembles, the error at a single arbitrary data point is defined as:

$$\text{Err}(f_{ens}(\mathbf{x}_n) \cdot y_n) = \text{Err} \left(\text{sign} \left(\sum_{i=1}^M c_i f_i(\mathbf{x}_n) \right) \cdot y_n \right), \quad (3.4)$$

¹If there is a tie, the Error function, equation (3.5) gives zero for binary classification.

3.2 Ambiguity Decomposition for Classifier Ensembles

where $Err(x)$ is a function defined as

$$Err(x) = \begin{cases} 1 & \text{if } x = -1 \\ 0.5 & \text{if } x = 0 \\ 0 & \text{if } x = 1 \end{cases} . \quad (3.5)$$

where $x = 0$ means there is a tie, $\sum_{i=1}^M c_i f_i(\mathbf{x}_n) = 0$, in the majority voting.

Since the error function $Err(x)$ is discrete, we need to generalize it to a continuous function to facilitate the derivation. Based on the definition of $Err(x)$, a linear function $Err(x) = -\frac{1}{2}(x-1)$ can be easily obtained by fitting these three points $(-1, 1)$, $(0, 0.5)$ and $(1, 0)$. With this generalization, the following equation holds

$$Err(x) - a \cdot Err(y) = -\frac{1}{2}((x-1) - (y-1)a) = \frac{1}{2}((y-1)a - (x-1)), \quad (3.6)$$

where a is a constant.

The difference between ensemble error and the average error of component classifiers is:

$$\begin{aligned} & Err(f_{ens}(\mathbf{x}_n) \cdot y_n) - \sum_{i=1}^M c_i \cdot Err(f_i(\mathbf{x}_n) \cdot y_n) \\ &= \frac{1}{M} \sum_{i=1}^M (Err(f_{ens}(\mathbf{x}_n) \cdot y_n) - M c_i \cdot Err(f_i(\mathbf{x}_n) \cdot y_n)) \\ &= \frac{y_n}{2} \sum_{i=1}^M \left(c_i f_i(\mathbf{x}_n) - \frac{1}{M} f_{ens}(\mathbf{x}_n) \right). \end{aligned} \quad (3.7)$$

Reorganizing equation (3.7), the ambiguity decomposition can be written as follows:

$$Err(f_{ens}(\mathbf{x}_n) y_n) = \sum_{i=1}^M c_i Err(f_i(\mathbf{x}_n) y_n) - \frac{y_n}{2} \sum_{i=1}^M \left(\frac{1}{M} f_{ens}(\mathbf{x}_n) - c_i f_i(\mathbf{x}_n) \right). \quad (3.8)$$

The decomposition is made up of two terms. The first, $\sum_i c_i Err(f_i(\mathbf{x}_n) y_n)$, is the weighted average error of the individuals. The second term is the ambiguity term which measures the difference between $f_{ens}(\mathbf{x}_n)$ and component classifiers $f_i(\mathbf{x}_n)$.

3.3 A New Diversity Measure

According to equation (3.8), the proposed ambiguity decomposition for *classifier* ensembles is purely additive (i.e., $loss = accuracy - diversity$) and has a clear relationship to the original ambiguity decomposition for squared loss. Besides the purely additive similarity, the new ambiguity decomposition is related to the *margin* in classification problems.

Since

$$\sum_{i=1}^M c_i f_i(\mathbf{x}_n) = \left| \sum_{i=1}^M c_i f_i(\mathbf{x}_n) \right| \text{sign} \left(\sum_{i=1}^M c_i f_i(\mathbf{x}_n) \right) = s_n f_{ens}(\mathbf{x}_n), \quad (3.9)$$

where $s_n \in [0, 1]$ is the absolute value of $\sum_{i=1}^M c_i f_i(\mathbf{x}_n)$. The following equation can be obtained:

$$\begin{aligned} & \frac{y_n}{2} \sum_{i=1}^M \left(\frac{1}{M} f_{ens}(\mathbf{x}_n) - c_i f_i(\mathbf{x}_n) \right) \\ &= \frac{y_n}{2} (f_{ens}(\mathbf{x}_n) - s f_{ens}(\mathbf{x}_n)) = \frac{1-s}{2} y_n f_{ens}(\mathbf{x}_n). \end{aligned} \quad (3.10)$$

According to equation (3.10), when the output of ensemble is correct for the point \mathbf{x}_n , the value $y_n f_{ens}(\mathbf{x}_n) > 0$. In fact, the term $m_n = y_n f_{ens}(\mathbf{x}_n)$ in equation (3.10) is the *margin* (Rätsch et al., 2001; Schapire et al., 1998) for data point \mathbf{x}_n . The margin at \mathbf{x}_n is positive if the correct class label of the pattern is predicted. As the margin value increases, the decision stability becomes larger. Moreover, as $f(\mathbf{x}_n) \in [-1, 1]$, then $m_n \in [-1, 1]$.

Margin theory is firstly used by support vector machine. Then Breiman (Breiman, 1999) defined the margin¹ for a single point and used the concept to analyze boosting algorithms. The following work on margin includes an explanation of Adaboost as boosting the margin (Schapire et al., 1998) and construction of the soft-margin Adaboost (Rätsch et al., 2001).

Besides the margin term, the parameter $s_n = \left| \sum_{i=1}^M c_i f_i(\mathbf{x}_n) \right|$ measures the difference between the number of positive and negative votes, and a smaller s_n encourages more diversity. In order to maximize the ambiguity term, firstly we can enlarge the margin and keep the obtained margin unchanged. Secondly,

¹Note that the edge (Breiman, 1999) is just an affine transformation of the margin.

3.4 Correlation Between Diversity and Generalization

Table 3.1: Summary of Data Sets

Data Sets	Examples	Features	%Positive
Card	690	14	44.49%
Cancer	683	9	34.99%
Heart	270	13	44.44%
Sonar	208	60	53.37%
Ionosphere	351	34	64.10%
Liver	345	6	57.97%

we try to minimize the corresponding gap between the number of positive and negative votes.

Based on the above analysis, we can define the accuracy and ambiguity as follows:

$$Accuracy = \frac{1}{N} \sum_{n=1}^N \sum_{i=1}^M c_i \cdot Err(f_i(\mathbf{x}_n) \cdot y_n), \quad (3.11)$$

$$Ambiguity = \frac{1}{2N} \sum_{n=1}^N \sum_{i=1}^M \left(\frac{1}{M} f_{ens}(\mathbf{x}_n) - c_i f_i(\mathbf{x}_n) \right) y_n. \quad (3.12)$$

3.4 Correlation Between Diversity and Generalization

In this section, we conduct empirical experiments to analyze the relationship between ten diversity measures and the generalization error of Bagging. Classification and regression tree (CART) is used as the component classifier in Bagging. Six data sets have been employed in our experiments. They are Australian credit card, Wisconsin breast cancer, heart disease, sonar, ionosphere and liver disorder, which are from the UCI Machine Learning Repository (Asuncion and Newman, 2007). The characteristics of these data sets are summarized in Table 3.1.

As we know, Bagging is based on bootstrap sampling and each individual classifier is trained on almost $1 - 1/e \approx 63.2\%$ training points (Efron and Tibshirani,

3.4 Correlation Between Diversity and Generalization

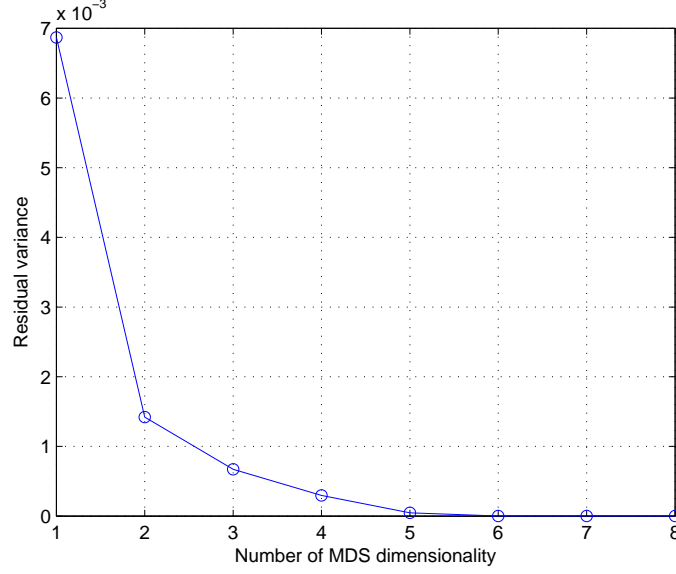


Figure 3.1: In MDS algorithm, residual variance vs. number of dimensions on credit card problem. The other problems yield similar plots and are omitted only to save space.

1993). In our experiments, we tune the sampling rate r , i.e. randomly sample $100 \times r\%$ of data to train each tree, to change the diversity. For each problem, we change the sampling rate from 0.1 to 1 with the interval 0.05 and finally we get 181 sampling rates.

For each sampling rate, an ensemble of 100 trees are built to record the accuracy, ambiguity, generalization error and other nine diversity measures. These results are based on 100 runs of 5-fold cross validation and validation data is used to record the diversity measures and generalization error.

3.4.1 Visualization of Diversity Measures using Multidimensional Scaling

This subsection uses multidimensional scaling (MDS) (Cox and Cox, 1994) as a visualization tool to investigate the relationship among these diversity measures. A MDS algorithm starts with a matrix of item-item similarities or dissimilarities, and then assigns a coordinate of each item in a low-dimensional space, aiming

3.4 Correlation Between Diversity and Generalization

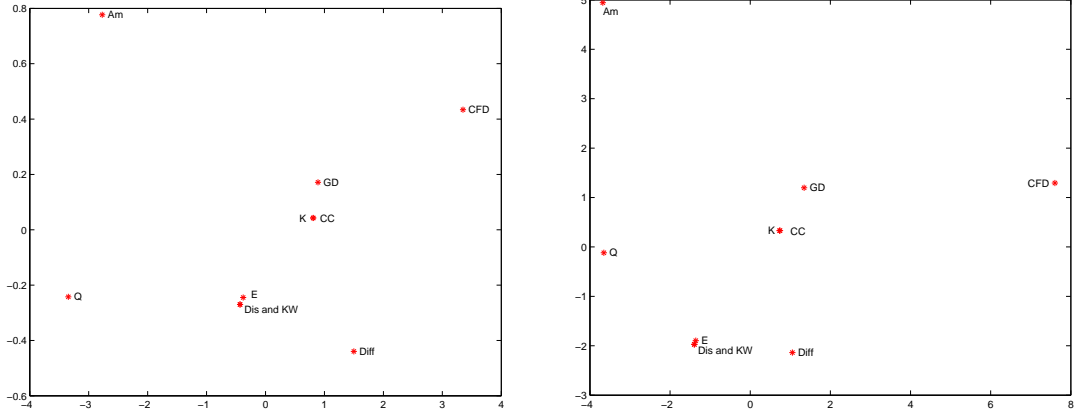


Figure 3.2: 2D MDS plot using normalized scores (left) and standard deviation scaling (right) for credit card problem. The 10 measures of diversity are: AM - Ambiguity, Q - Q statistics, K - Kappa statistics, CC - correlation coefficient, Dis - disagreement measure, E - entropy, KW - Kohavi-Wolpert variance, Diff - measure of difficulty, GD - generalized diversity, CFD - coincident failure diversity and Err - generalization error for the six data sets. The x and y axes are coordinates of these diversity measures in 2D space.

at preserving the pairwise distances of high dimensional data. As we use ten diversity measures including ambiguity for investigation, for each problem we build a 10×181 ¹ table where lines represent the diversity measures, columns represent ensembles with different sampling rates, and each entry (i, j) in the table is the diversity score with the j^{th} sampling rate using the i^{th} diversity measure.

For a MDS algorithm, each row in the table is treated as the coordinate of a point in a 181 dimensional space. The distance is calculated as the Euclidean distance between the two corresponding points in that space.

In order to calculate the pairwise Euclidean distance, we need to normalize these diversity measures. Diversity measures such as entropy and kappa statistics have range $[0, 1]$, while others (Ambiguity, Q statistics, disagreement measure, Kohavi-Wolpert variance and measure of difficulty) range from p to q where p and

¹For each problem, we change the sampling rate from 0.1 to 1 with the interval 0.05 and totally there are 181 sampling rates.

3.4 Correlation Between Diversity and Generalization

q depend on the data set and diversity measure. For some measures lower values indicate higher diversity, e.g. Q statistics; for others, higher values indicate higher diversity, e.g. ambiguity. Different diversity measures have different baseline rates that depend on the data. In order to compare diversity measures in a meaningful way, all the measures need to be placed on a similar scale. One way to do this is to scale the scores from 0 to 1, where 0 is the lowest diversity, and 1 is the highest diversity. The disadvantage of normalized scores is that recovering the raw diversity measures requires knowing the values that define the top and bottom of the scale, and as new sampling rates are employed the bottom/top of the scale may change.

As MDS is sensitive to how the performance metrics are scaled, we perform MDS two ways. The first approach uses normalized method and the second approach scales these diversity measures to mean 0.0 and standard deviation 1.0 instead of using normalized method, though scaling by standard deviation is somewhat less intuitive because scores scaled by standard deviation depend on the full distribution of models instead of just the distance that fall at the top and bottom of each scale.

To quantify the performance of a MDS algorithm, the residual variance (RV) is used as the error measure, defined in (Tenenbaum et al., 2000) as the residual square of correlation coefficient

$$RV = 1 - cov^2(\hat{D}_X, D_Y) / \sigma_{\hat{D}_X}^2 \sigma_{D_Y}^2, \quad (3.13)$$

where σ is the variance and cov is the function for computation of correlation coefficients. D_Y is the matrix of pairwise Euclidean distances in the high-dimensional embedding space and \hat{D}_X is the best estimate of the pairwise Euclidean distances in the low dimensional space. The smaller the residual variance is, the better the algorithm is.

Ten diversity measures permit $10 \times 9/2 = 45$ pairwise comparisons. We calculate Euclidean distance between each pair of measures in the high dimensional space, and then perform multidimensional scaling on these pairwise distances.

Figure 3.1 shows the residual variance as a function of the number of dimensions in the MDS. The ten diversity measures appear to span a MDS space of

3.4 Correlation Between Diversity and Generalization

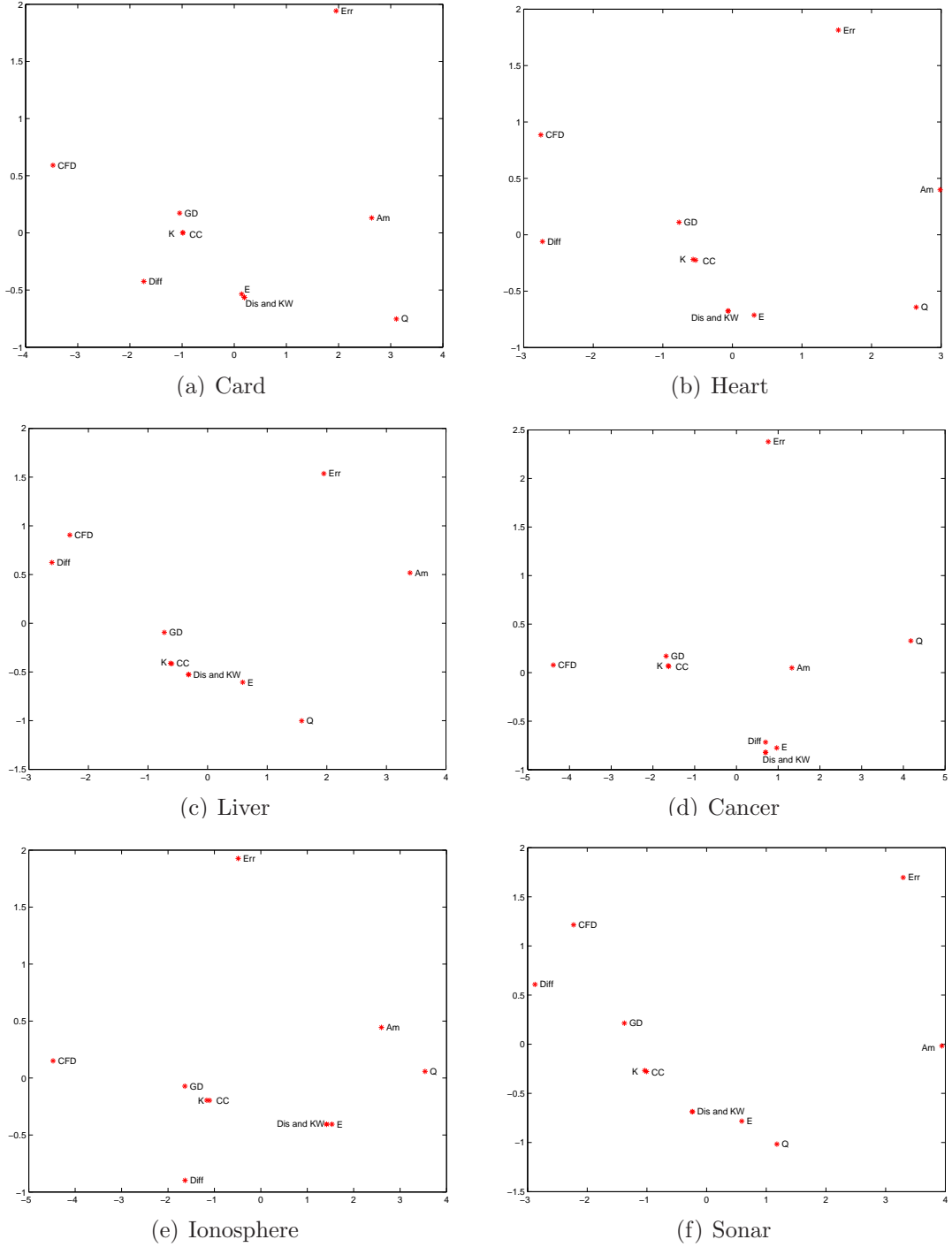


Figure 3.3: 2D MDS plot (10 diversity measures and generalization error) using normalized method on six data set. The results are averaged on the 100 run on each data set. The x and y axes are coordinates of these diversity measures in 2D space.

3.4 Correlation Between Diversity and Generalization

about 3 to 5 dimensions. In this section we examine the 2-D MDS plots for easy visualization in detail.

Figure 3.2 shows two MDS plots when dimensionality is reduced to two dimensions. The plot on the left is MDS using normalized method. The plot on the right is MDS using standard deviation scaled method.

Both MDS plots show a similar pattern. The metrics appear to form 5-6 somewhat distinct groups. In the middle is a group that includes K, CC and GD. The second group includes E, Dis and KW. The other groups are Am (by itself), Diff (by itself, or possibly with the second group), Q (by itself), and CFD (by itself).

The ambiguity does not appear to correlate strongly with any other metric. It is not surprising that disagreement measure and Kohavi-Wolpert variance lay in the same point because Kohavi-Wolpert variance differs from the averaged disagreement measure by only a coefficient (Kuncheva and Whitaker, 2003). K and CC form a cluster because the definition of Kappa statistics and correlation coefficient only differ at the denominator: the denominator of Kappa is the square of that of correlation coefficient.

It is somewhat surprising that entropy measure falls very close to disagreement measure and Kohavi-Wolpert variance. Also, GD is closer to K and CC.

Figure 3.3 shows 2-D MDS plots for six test problems. These figures also include the generalization error with 10 diversity measures. Although there are variations between the plots, the 2-D MDS plots for the six problems are remarkably consistent given that these are different test problems. The consistency in these six MDS plots suggests that we have an adequate sample size of sampling rate to reliably detect relationships between the measures.

Ambiguity consistently lies close to the generation error, which means Am exhibits a good correlation with generalization error compared with other diversity measures. For some data sets, CFD and GD also fall near the generalization error. Metrics such as Q and Diff seem to move around in these plots.

3.4 Correlation Between Diversity and Generalization

Table 3.2: Rank correlation coefficients (in %) between the diversity measures based on the average of the six data sets. The measures are: Am - Ambiguity; Q statistics; K - Kappa statistics; CC - correlation coefficient; Dis - disagreement measure; E - entropy; KW - Kohavi-Wolpert variance; Diff - measure of difficulty; GD - generalized diversity; and CFD - coincident failure diversity.

	Am	Q	K	CC	Dis	E	KW	Diff	GD	CFD	Err
Am	100	84	84	84	81	80	81	48	85	37	75
Q		100	99	99	96	95	96	52	92	27	44
K			100	100	93	92	93	49	95	32	51
CC				100	93	92	93	49	95	32	51
Dis					100	99	100	55	82	15	28
E						100	99	54	81	12	27
KW							100	55	82	15	28
Diff								100	46	47	16
GD									100	50	66
CFD										100	62
Err											100

3.4 Correlation Between Diversity and Generalization

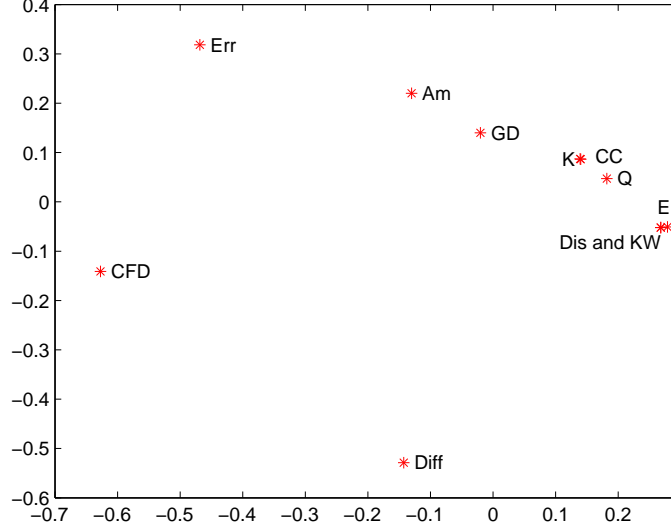


Figure 3.4: 2D MDS plot using rank correlation coefficients. This figure is averaged on six data sets. The 10 measures of diversity are: AM-Ambiguity, Q - Q statistics, K - Kappa statistics, CC - correlation coefficient, Dis - disagreement measure, E - entropy, KW - Kohavi-Wolpert variance, Diff - measure of difficulty, GD - generalized diversity, CFD - coincident failure diversity and Err - generalization error. The x and y axes are coordinates of these diversity measures in 2D space.

3.4.2 Correlation Analysis of Diversity Measures

With the MDS analysis in the previous subsection, we used ten diversity measures to investigate 181 ensembles with different sampling rates on each of the six test problems. In this section we use correlation analysis, instead of MDS, to study the relationship among ambiguity, nine other diversity measures and generalization error. Again, to make the correlation analysis easier to interpret, we firstly scale scores to the range $[0, 1]$ so that the best score is 1, and baseline score is 0.

Ten diversity measures and generalization error permit $11 \times 10 / 2 = 55$ pairwise correlations. We do these comparisons using non-parametric Spearman's rank correlation, which studies relationships between different rankings on the same set of items. We use rank correlation instead of linear correlation because rank correlation makes fewer assumptions about the relationships between the metrics,

3.4 Correlation Between Diversity and Generalization

and it is insensitive to how these diversity measures are scaled. While linear correlation assumes that the relationship of two variables is linear, which is not always correct in practice.

Table 3.2 summarizes these average pairwise correlations among ambiguity, other nine diversity measures and generalization error. Each entry in the table is the average rank correlation across the six test problems. The table is symmetric and contains 55 unique pairwise comparisons.

From this table, two pairs of diversity measures: Kappa statistics (Kappa) and correlation coefficient (CC), disagreement measure *Dis* and Kohavi-Wolpert variance *KW*, are equivalent, because their correlation coefficients are 1. In (Kuncheva and Whitaker, 2003), it is proven that *KW* differs from the averaged disagreement measure *Dis* by only a coefficient.

Metrics with pairwise rank correlations near one behave more similarly than those with smaller rank correlations. The pairs that have rank correlations above 0.90 are listed as follows:

- 1.00: Kappa to CC, Dis to KW
- 0.99: Q to Kappa (CC), Dis (KW) to E
- 0.96: Q to Dis (KW)
- 0.95: Q to E, GD to Kappa (CC)
- 0.93: Kappa to Dis (KW)
- 0.92: GD to Q, E to Kappa (CC)

The relationships revealed by rank correlation analysis are consistent with MDS analysis. As expected, Am and generalization error have higher rank correlation than other diversity measures. Q is highly correlated with the other two metrics, Kappa and CC. But the high correlation between entropy and disagreement measure is somewhat surprising and we currently do not know how to explain this.

The weakest correlations are all between coincident failure diversity (CFD) and the other metrics, even with GD, the correlation is only 0.5. However, it has second largest rank correlation with the generalization error.

Figure 3.4 shows a MDS plot for the metrics when distance between metrics is calculated as rank correlation coefficients, making MDS insensitive to how the metrics are scaled. (Distances based on rank correlation coefficients do not respect

3.4 Correlation Between Diversity and Generalization

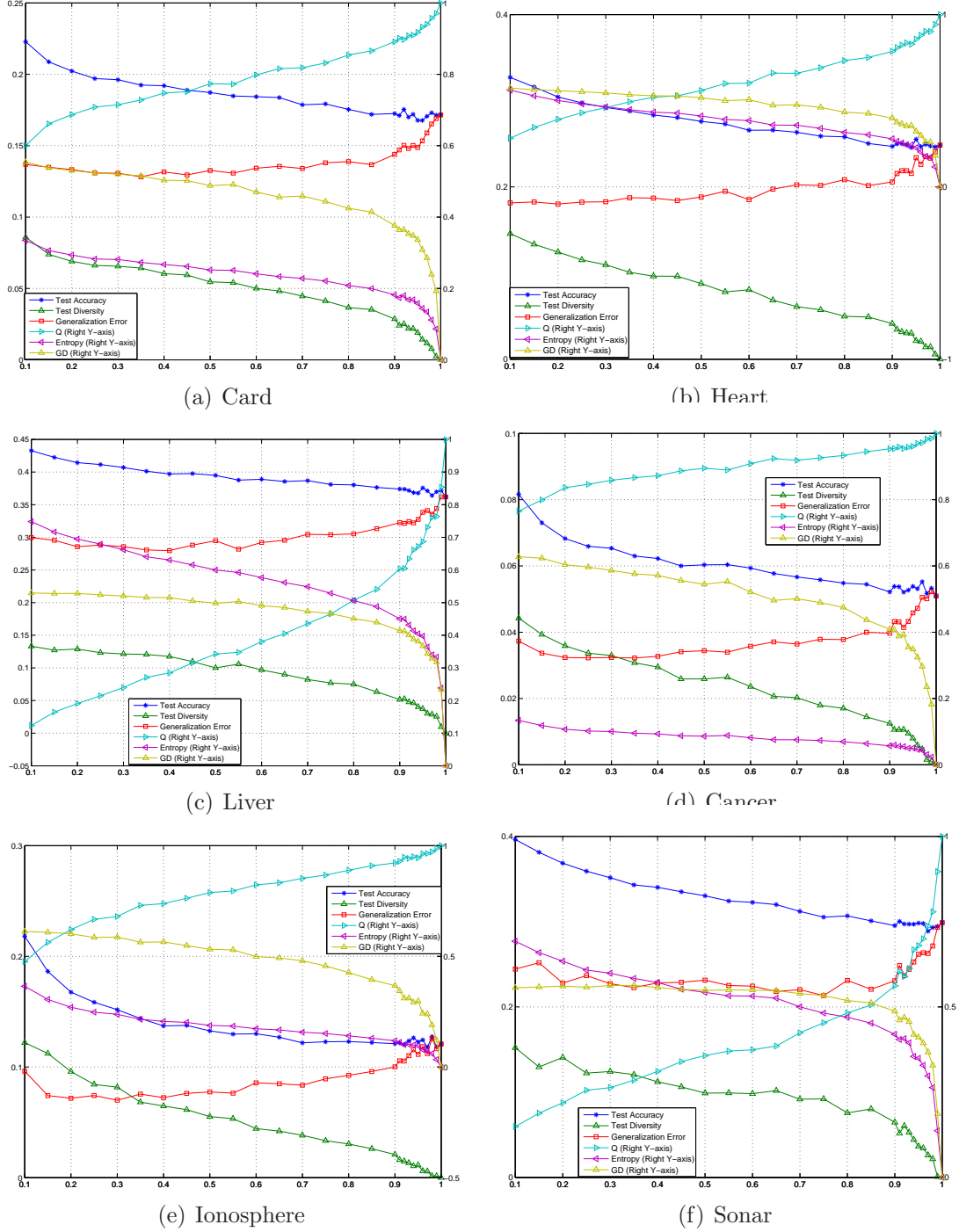


Figure 3.5: Accuracy, Diversity, Q statistics, Entropy, Generalized Diversity (GD) and Generalization Error with different sampling rates (from 0.1 to 1) of Bagging for six data sets. The x-axis is the sampling rate r . The plot interval of sampling rate r from 0.1 to 0.9 is 0.05 and plot the interval between 0.9 and 1 is 0.01. The left y-axis is to record the values of Accuracy, Diversity and Generalization Error; the right y-axis is for Q statistics, Entropy and Generalized Diversity (GD). The results are averaged on 100 runs of 5-fold cross validation.

3.4 Correlation Between Diversity and Generalization

Table 3.3: Rank correlation coefficients (in %) among ambiguity, nine diversity measures and Generalization Error in Different Sampling Range, where G stands for generalization error.

Rate	$0.9 \leq r \leq 1$						
Pairs	Card	Heart	Liver	Cancer	Iono	Sonar	Mean
Am-G	91	85	90	83	81	94	87
Q-G	80	58	73	62	65	82	70
K-G	85	58	75	70	69	85	74
CC-G	85	58	75	70	69	85	74
Dis-G	76	56	68	50	60	82	65
E-G	76	53	70	48	60	82	65
KW-G	76	56	68	50	60	82	65
Diff-G	75	62	66	59	49	83	66
GD-G	85	66	79	75	71	89	78
CFD-G	87	80	84	79	71	91	82
	$0.1 \leq r < 0.9$						
Am-G	47	66	73	58	68	20	55
Q-G	38	51	45	57	67	-9	42
K-G	42	53	46	61	71	-6	44
CC-G	42	53	46	61	71	-6	44
Dis-G	31	46	43	43	62	-16	35
E-G	31	46	42	40	61	-16	34
KW-G	31	46	43	43	62	-16	35
Diff-G	30	40	-44	46	63	22	26
GD-G	47	61	58	63	76	36	57
CFD-G	52	35	-11	63	78	40	43

3.4 Correlation Between Diversity and Generalization

Table 3.4: The generalization error of Bagging algorithms with different sampling rates, where $r = 0.632$ is the performance of Bagging with bootstrap. The results are averaged over 50 runs of 5 fold cross validation.

Rate	0.1	0.3	0.5	0.632	0.7
Card	13.69 \pm 2.12	13.07\pm1.92	13.27 \pm 2.05	13.50 \pm 1.98	13.40 \pm 1.87
Heart	17.47\pm3.39	18.48 \pm 4.48	18.93 \pm 3.69	19.12 \pm 3.36	19.89 \pm 4.12
Liver	30.33 \pm 3.68	28.52\pm3.76	28.80 \pm 3.40	29.54 \pm 3.77	30.43 \pm 4.32
Cancer	3.70 \pm 1.31	3.39 \pm 1.14	3.38\pm1.12	3.60 \pm 1.18	3.51 \pm 1.05
Iono.	9.62 \pm 2.77	7.00\pm2.26	7.74 \pm 2.57	8.51 \pm 2.61	8.36 \pm 2.77
Sonar	25.10 \pm 5.05	23.84 \pm 5.47	22.89 \pm 4.76	22.54 \pm 5.37	21.99\pm5.20

the triangle inequality so this is not a proper metric space.) The overall pattern is similar to that observed in the MDS plots in Figure 3.3. Am is closest to the generalization error, which is consistent with the rank correlation coefficients in Table 3.2. The two clusters in Figure 3.3: K, CC and GD, Dis, KW and E, seem to unite to one cluster. Q statistic is closest to K and CC, though not as close as in the other plots. CFD is at the left end of the space farthest from Am and most of other metrics. Diff is at the lower side of the space.

We also study the relationship between the generalization error and ten diversity measures with different sampling rates. Figure 3.5 illustrates the curves of the generalization vs. the sampling rate r for each data set.

From Figure 3.5, the correlation between diversity and generalization error varies with different sampling rates r . In order to quantify the difference, Table 3.3 reports the rank correlation coefficients between ten diversity measures and generalization error in two sampling zones. According to Table 3.3, nine diversity measures show similar characteristics as ambiguity: when the sampling rate r is large, i.e. ambiguity is small, ambiguity highly correlated with generalization error but when ambiguity exceeds a threshold, the correlation drops. The reason is that accuracy changes with ambiguity at the same speed.

Based on the observed relationship, three zones with different sampling rates

3.4 Correlation Between Diversity and Generalization

r are summarized in the following under the condition that there are sufficient training data.

1. Low diversity zone, i.e. large sampling rate ($0.9 \leq r \leq 1$).

When the sampling rate r converges to 1, the training of the individual classifiers in the ensemble converge. Accuracy does not change a lot with r . However, as unstable base classifiers, only a small change of training data would disturb the output and thus increase the diversity much (Buhlmann and Yu, 2002). In this way, when r changes from 1 to 0.9, accuracy term keeps almost unchanged but diversity increases a lot. The generalization error, which is the difference between accuracy and diversity, reduces a lot. The empirical results support the statement that Bagging benefits from combining unstable learners to succeed (Breiman, 1996a; Buhlmann and Yu, 2002).

2. Medium diversity zone, i.e. medium sampling rate ($0.2 \leq r < 0.9$).

In this zone, generalization error does not change much when diversity changes a lot. As diversity raises, so does accuracy, and thus the difference, i.e. generalization error, will not change much. As we know, the sampling rate of bootstrap is around $r = (1 - 1/e) \approx 0.632$, varying diversity of Bagging will not change the generalization error much. This explains the empirical results in (García et al., 2005; Kuncheva and Whitaker, 2003).

3. Large diversity zone, i.e. low sampling rate ($r < 0.2$).

When the sampling rate r is small, there is not sufficient training data available for each component classifier, so accuracy becomes larger but diversity term changes less than accuracy. Therefore, this leads an increase of the generalization error.

It is very important that whether there are sufficient training data available. If there are not enough data points available for training, the generalization will not change much in zone 1 because the accuracy term doesn't converge and will change with diversity at the same speed.

The generalization error of Bagging algorithm with different sampling rates are also reported in Table 3.4. According to the table, the ongoing sampling rate of Bagging is not the optimum. Bagging algorithm with a small sampling rate $r \leq 0.5$ outperforms that with a large sampling rate, supported by Bagging algorithm on the five data sets in our experiments. The only exception is the sonar data set because the number of data points in sonar data set is few (208). A low sampling rate results in insufficient training data for individual classifiers. This finding also confirms that Bagging algorithm with a small sampling rate performs better than the original Bagging algorithm with bootstrap (Buhlmann and Yu, 2002).

Regarding the question “whether the use of diversity measures has a beneficial effect for classifier ensembles”, our answer is partially positive. However, we should be very careful to select the diversity measure, because different diversity measures have significantly different correlation with generalization error. For example, the rank correlation between ambiguity and generalization is 0.75 but the correlation between Diff and generalization is just 0.16 in our experiments. Apart from selecting the proper diversity measure, if we want to enlarge the diversity for better performance, we had better judge whether the ensemble has adequately large diversity. If the ensemble already has a large diversity, the generalization performance of ensemble will not benefit much from more diversity. The contradictory results (García et al., 2005; Kuncheva and Whitaker, 2003) about the usefulness of diversity can be explained by the above two reasons.

3.5 Summary

This chapter investigates two fundamental questions on diversity in classifier ensembles (i) how to define the diversity for classifier ensembles and (ii) how diversity correlates with generalization error.

In order to answer the first question, we conduct the first theoretical analysis of ambiguity decomposition for *classifier* ensembles with *zero-one* loss, which breaks the error of ensemble into two terms: accuracy and ambiguity terms. As the ambiguity term measures the amount of variability among the ensemble members, the ambiguity term is adopted as a new diversity measure for classifier

ensembles. The empirical experiments confirm that ambiguity is a good measure of diversity in comparison with nine most-often-used diversity measures. This is the first contribution of the chapter.

The second contribution of the chapter is the first empirical findings that diversity highly correlates with generalization error when the diversity is small but the correlation reduces after diversity exceeds a threshold. These findings can explain the empirical results (García et al., 2005; Kuncheva and Whitaker, 2003) that varying diversity does not change the generalization error much because in their experiments the diversity was already large and did not strongly correlate with generalization error.

Our experiments also suggest that the performance of Bagging ($r \approx 0.632$) could be improved by changing the sampling rate to a small value ($0.2 \leq r \leq 0.5$) under the condition that there are sufficient training data.

Chapter 4

Regularized Negative Correlation Learning

This chapter investigates a special kind of diversity, error diversity, using negative correlation learning (NCL) (Liu and Yao, 1999a,b; Liu et al., 2000). Negative correlation learning is a neural network ensemble algorithm which considers the cooperation and interaction among the ensemble members. However, we observe that NCL is prone to overfitting the noise in the training set by training the ensemble as a single estimator and only minimizing the MSE without regularization. Therefore, regularization should be used to address the overfitting problem of NCL. To avoid overfitting, section 4.2 proposes the regularized negative correlation learning (RNCL) algorithm. In section 4.3, we formulate RNCL by Bayesian technique and propose an algorithm to infer the regularization parameters based on Bayesian inference. The numerical results on synthetic data as well as real-world data sets are presented in section 4.4. Finally, section 4.5 summarizes the chapter.

4.1 Introduction

Negative Correlation Learning (NCL) (Liu and Yao, 1999a,b; Liu et al., 2000) has shown a number of empirical applications (Chen and Yao, 2007a; Islam et al., 2003; Liu et al., 2000; Yao et al., 2001). NCL introduces a correlation penalty

term into the cost function of each individual network so that each neural network minimizes its MSE error together with the correlation with other ensemble members.

According to the definition of NCL, it seems that the correlation term in the cost function acts as the regularization term. However, we observe that the training of NCL with the penalty coefficient λ setting to 1 corresponds to treating the entire ensemble as a single estimator and considering only the empirical training error without regularization. In this case, NCL only reduces the empirical MSE of the ensemble, but it pays less attention to regularizing the complexity of the ensemble and NCL is prone to overfitting the noise in the training set. Similarly, setting a zero or small positive λ corresponds to independently training these estimators without regularization and in this case, NCL is prone to overfitting as well.

NCL can use the penalty coefficient to explicitly alter the emphasis on the individual MSE and correlation portions of the ensemble and thus alleviate the overfitting problem to some extent. However, NCL could not totally overcome the overfitting problem by tuning this parameter without regularization, especially when dealing with data with non-trivial noise, which will be evidenced by the empirical work in this paper.

The chapter analyzes the overfitting problem of NCL and proposes the theoretical and empirical evidences. In order to solve this problem, this chapter proposes regularized negative correlation learning (RNCL) algorithm which incorporates an additional regularization term for the ensemble. Then we describe that the regularization term for the ensemble can be decomposed into different parts for each network. In this paper, we describe how the training algorithm of NCL is equivalent to training a single learning machine and how RNCL controls the complexity by adding a regularization term. The regularization parameter is used to control the tradeoff between MSE and regularization and this parameter is crucial to ensemble's generalization ability.

We provide a Bayesian interpretation for RNCL, and propose an automatic algorithm for parameters optimization based on Bayesian inference. The RNCL algorithm is a generic ensemble algorithm, which is applicable to any nonlinear regression estimator minimizing the MSE, for example multilayer perceptron

(MLP) and radial basis function (RBF) neural network. In this chapter we show an example using MLP as the base estimators.

4.2 Regularized Negative Correlation Learning

This section describes negative correlation learning and its potential problem. In order to address the problem, regularized negative correlation learning (RNCL) is proposed in this section.

4.2.1 Negative Correlation Learning

Negative Correlation Learning (NCL) introduces a correlation penalty term into the error function of each individual network in the ensemble so that all the networks can be trained interactively on the same training data set.

Given the training set $\{\mathbf{x}_n, y_n\}_{n=1}^N$, NCL combines M neural networks $f_i(\mathbf{x})$ to constitute the ensemble.

$$f_{ens}(\mathbf{x}_n) = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}_n). \quad (4.1)$$

To train network f_i , the cost function e_i of network i is defined by

$$e_i = \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n)^2 + \lambda p_i, \quad (4.2)$$

where λ is a weighting parameter on the penalty term p_i :

$$\begin{aligned} p_i &= \sum_{n=1}^N \left\{ (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \sum_{j \neq i} (f_j(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \right\} \\ &= - \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2. \end{aligned} \quad (4.3)$$

The first term in the right-hand side of (4.2) is the empirical training error of network i . The second term p_i is a correlation penalty function. The purpose of minimizing p_i is to negatively correlate each network's error with errors for the rest of the ensemble. The λ parameter controls a trade-off between the training

4.2 Regularized Negative Correlation Learning

error term and the penalty term. With $\lambda = 0$, we would have an ensemble with each network training with plain back propagation, exactly equivalent to training a set of networks independently of one another. If λ is increased, more and more emphasis would be placed on minimizing the penalty.

Based on the individual error function, equation (4.2), the error function of the ensemble can be obtained by averaging these networks' errors e_i . With $\lambda = 1$, the average error E of all the networks' e_i is obtained as follows:

$$\begin{aligned} E &= \frac{1}{M} \sum_{i=1}^M e_i = \frac{1}{M} \sum_{n=1}^N \sum_{i=1}^M \{(f_i(\mathbf{x}_n) - y_n)^2 - (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2\} \\ &= \sum_{n=1}^N (f_{ens}(\mathbf{x}_n) - y_n)^2. \end{aligned} \quad (4.4)$$

From equation (4.4), NCL is equivalent to training a single estimator $f_{ens}(\mathbf{x}_n)$ instead of training each individual network separately. It is also observed that NCL only minimizes the empirical training MSE error $\sum_{n=1}^N (f_{ens}(\mathbf{x}_n) - y_n)^2$ but does not regularize the complexity of the ensemble. As discussed in section 1.1, the learning algorithm that only minimizes the empirical MSE error is prone to overfitting the noise. In section 4.4, we also present the empirical evidences that NCL is prone to overfitting.

In order to improve the generalization ability of NCL, in the next subsection we propose regularized negative correlation learning (RNCL).

4.2.2 Regularized Negative Correlation Learning

Following the traditional strategy to avoid overfitting, a regularization term is incorporated into the error function of the ensemble:

$$E_{ens} = \sum_{n=1}^N (f_{ens}(\mathbf{x}_n) - y_n)^2 + \sum_{i=1}^M \alpha_i \mathbf{w}_i^T \mathbf{w}_i, \quad (4.5)$$

where $\mathbf{w}_i = (w_{i,1}, \dots, w_{i,n_i})^T$ is the weight vector of neural network i and n_i is the total number of weights in network i . This regularization term $\sum_{i=1}^M \alpha_i \mathbf{w}_i^T \mathbf{w}_i$ is the weight decay (Krogh and Hertz, 1992) term for the ensemble.

Weight decay adds a penalty term to the error function. The usual penalty is the sum of squared weights times a decay constant. In a linear model, this form

4.3 Bayesian Formulation and Regularized Parameter Optimization

of weight decay is equivalent to ridge regression (Hoerl and Kennard, 2000). The weight decay penalty term causes the weights to converge to smaller absolute values than they otherwise would. The regularization term does help the generalization ability of neural network because large weights can hurt generalization in two different ways: a) Excessively large weights leading to hidden units can cause the output function to be too rough, possibly with near discontinuities; Excessively large weights leading to output units can cause wild outputs far beyond the range of the data if the output activation function is not bounded to the same range as the data. b) Large weights can cause excessive variance of the output (Geman et al., 1992).

In order to train each neural network with its regularization, we decompose the error function of ensemble into M parts, each part for one network. The error function for network i can be obtained as follows:

$$e_i = \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n)^2 - \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2 + \alpha_i \mathbf{w}_i^T \mathbf{w}_i. \quad (4.6)$$

Comparing this error function with the error function of NCL, equation (4.2), RNCL imposes a regularization term on every individual neural network and RNCL optimizes the regularization parameter α_i instead of the correlation parameter λ .

According to equations (4.5) and (4.6), RNCL is implemented by decomposing the error function of ensemble into a set of sub-functions, each sub-function for one network. RNCL provides one way to decompose the learning task of the ensemble with regularization into a number of subtasks for individual networks. The algorithm can be summarized in Figure 4.1.

We use scaled conjugate gradient (SCG) (Møller, 1993b) algorithm for fast RNCL training.

4.3 Bayesian Formulation and Regularized Parameter Optimization

This section formulates RNCL by Bayesian technique and proposes an algorithm for parameter optimization by Bayesian inference. We separate the section into

4.3 Bayesian Formulation and Regularized Parameter Optimization

Algorithm Regularized Negative Correlation Learning (RNCL)

Input: the training set $\mathbf{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, integer M specifying size of ensemble, the initial regularization parameter α_i , $i = 1, \dots, M$ and the learning rate η .

1. For the training set

- Calculate $f_{ens}(\mathbf{x}_n) = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}_n)$
- For each network from $i = 1$ to M do: for each weight $w_{i,j}$ in network i , perform a desired number of updates,

$$\begin{aligned}
 e_i &= \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n)^2 - \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2 + \alpha_i \mathbf{w}_i^T \mathbf{w}_i \\
 \frac{\partial e_i}{\partial w_{i,j}} &= 2 \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n) \frac{\partial f_i}{\partial w_{i,j}} - 2 \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \left(1 - \frac{1}{M}\right) \frac{\partial f_i}{\partial w_{i,j}} + 2\alpha_i w_{i,j} \\
 \Delta w_{i,j} &= -2\eta \left\{ \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n) \frac{\partial f_i}{\partial w_{i,j}} - \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \left(1 - \frac{1}{M}\right) \frac{\partial f_i}{\partial w_{i,j}} + \alpha_i w_{i,j} \right\}
 \end{aligned}$$

2. Parameter Optimization by Bayesian Inference.

3. Repeat from step 1 for a desired number of iterations.

Output: RNCL ensemble

$$f(\mathbf{x}) = \frac{1}{M} \sum_i f_i(\mathbf{x}).$$

Figure 4.1: Regularized Negative Correlation Learning Algorithm

two parts: the first part describes the model specification and the probabilistic formulation of RNCL. The second part describes the procedures to infer the regularization parameters.

4.3.1 Bayesian Formulation of RNCL

Given the training set $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, we follow the standard probabilistic formulation and assume that the targets are sampled from the model with additive noise:

$$y_n = f_{ens}(\mathbf{x}_n) + e_n = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}_n) + e_n, \quad (4.7)$$

where e_n is an independent sample from some noise process which is further assumed to be zero-mean Gaussian with variance β^{-1} .

4.3 Bayesian Formulation and Regularized Parameter Optimization

According to the Bayesian theorem, given the hyperparameters $\mu = (\mu_1, \dots, \mu_M)$ ¹ and β , we obtain the weight parameters $\mathbf{w} = (\mathbf{w}_1^T, \dots, \mathbf{w}_M^T)^T$ by maximizing the posterior $P(\mathbf{w} \mid D)$.

$$P(\mathbf{w} \mid D) = \frac{P(D \mid \mathbf{w}, \beta) P(\mathbf{w} \mid \mu)}{P(D \mid \mu, \beta)}, \quad (4.8)$$

where the probability $P(D \mid \mu, \beta)$ is a normalization factor which is independent of \mathbf{w} .

The weight vector of each network \mathbf{w}_i is assumed to have a Gaussian distribution with mean zero and variance μ_i^{-1} . The prior of the weight vector \mathbf{w} is obtained as follows.

$$P(\mathbf{w} \mid \mu) = \prod_{i=1}^M \left(\frac{\mu_i}{2\pi} \right)^{n_i/2} \exp \left(-\frac{1}{2} \mu_i \mathbf{w}_i^T \mathbf{w}_i \right), \quad (4.9)$$

where n_i is the total number of weights in network i .

Since noise e_n follows a Gaussian distribution with mean zero and variance β^{-1} , the likelihood $P(D \mid \mathbf{w}, \beta)$ can be written as

$$P(D \mid \mathbf{w}, \beta) = \prod_{n=1}^N \left(\frac{\beta}{2\pi} \right)^{1/2} \exp \left(-\frac{\beta}{2} e_n^2 \right). \quad (4.10)$$

We omit all the constants and the normalization factor, and apply Bayesian theorem:

$$P(\mathbf{w} \mid D) \propto \exp \left(-\frac{\beta}{2} \sum_{n=1}^N e_n^2 \right) \cdot \exp \left(-\sum_{i=1}^M \frac{\mu_i}{2} \mathbf{w}_i^T \mathbf{w}_i \right). \quad (4.11)$$

Taking the negative logarithm, the maximum of the posterior \mathbf{w} is obtained as the solution to the following optimization problem:

$$\min J_1(\mathbf{w}) = \frac{1}{2} \beta \sum_{n=1}^N e_n^2 + \frac{1}{2} \sum_{i=1}^M \mu_i \mathbf{w}_i^T \mathbf{w}_i. \quad (4.12)$$

The posterior $P(\mathbf{w} \mid D)$ can also be approximated by Gaussian distribution, the details are presented in the following.

¹ $\mu_i, i = 1, \dots, M$ is the inverse variance of the Gaussian distribution of weights for network i .

4.3 Bayesian Formulation and Regularized Parameter Optimization

Considering the normalization term, the posterior of weigh vector \mathbf{w} is described as

$$P(\mathbf{w} | D) = \frac{\exp(-J_1(\mathbf{w}))}{\int \exp(-J_1(\mathbf{w})) d\mathbf{w}}. \quad (4.13)$$

In order to obtain the result, the Taylor expansion of $J_1(\mathbf{w})$ is employed at point \mathbf{w}_{MP} .

$$J_1(\mathbf{w}) \approx J_1(\mathbf{w}_{MP}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP}), \quad (4.14)$$

where \mathbf{w}_{MP} is the most probable weight vector, and A is the Hessian matrix of $J_1(\mathbf{w})$.

$$A = \nabla \nabla J_1 = \nabla \nabla \left(\sum_{i=1}^M \frac{\mu_i}{2} \mathbf{w}_i^T \mathbf{w}_i + \frac{\beta}{2} \sum_{n=1}^N e_n^2 \right) = \text{diag}(\Lambda) + \beta \nabla \nabla \left(\frac{1}{2} \sum_{n=1}^N e_n^2 \right), \quad (4.15)$$

where $\Lambda = (\mu_1^{(1)}, \dots, \mu_1^{(n_1)}, \mu_2^{(1)}, \dots, \mu_2^{(n_2)}, \dots, \mu_M^{(1)}, \dots, \mu_M^{(n_M)})^T$ and the superscript indicates the number of repetitions of μ_i .

The integral can be computed as below:

$$\begin{aligned} \int \exp(-J_1(\mathbf{w})) d\mathbf{w} &= \int \exp(-J_1(\mathbf{w}_{MP}) - \frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP})) d\mathbf{w} \\ &= \exp(-J_1(\mathbf{w}_{MP})) \cdot (2\pi)^{W/2} \det A^{-\frac{1}{2}}. \end{aligned} \quad (4.16)$$

Based on these equations, the approximated posterior of \mathbf{w} is obtained as follows

$$P(\mathbf{w} | D) = \frac{\exp(-J_1(\mathbf{w}))}{\int \exp(-J_1(\mathbf{w})) d\mathbf{w}} = \frac{\exp(-\frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP}))}{(2\pi)^{W/2} |A|^{-\frac{1}{2}}}, \quad (4.17)$$

where A is the Hessian matrix of the cost function J_1 , W is the total number of weights in the ensemble and the subscript MP indicates the most probable values.

The error function J_1 is made up of two terms. The first, $\frac{1}{2}\beta \sum_{n=1}^N e_n^2$, is the sum of the empirical training error. The second, $\frac{1}{2} \sum_{i=1}^M \mu_i \mathbf{w}_i^T \mathbf{w}_i$, is the regularization term, measuring the total amount of square of weights.

Comparing equation (4.12) with (4.5), RNCL is equivalent to maximization of the posterior under Bayesian framework. The likelihood $P(D | \mathbf{w}, \beta)$ stands

4.3 Bayesian Formulation and Regularized Parameter Optimization

for the empirical training error term and the prior of the weight vector $P(\mathbf{w} \mid \mu)$ is equivalent to the regularization term.

Based on the above analysis, RNCL is an application of Bayesian framework in ensemble system. Instead of simultaneously optimizing the weight vector of ensemble, RNCL manages to train the entire ensemble by decomposing the job into a set of subtasks, which significantly reduces computational complexity compared with Bayesian framework.

Take a RBF network ensemble with linear outputs as an example. If we treat the ensemble as a single estimator, the training of the entire ensemble involves inversion of a matrix, whose computational complexity is $O(W^3) \sim O(M^3 n_i^3)$, where $W = \sum_{i=1}^M n_i$ (n_i is the number of weights in network i and M is the size of ensemble) is the total number of weights in ensemble. By decomposing the operation into a set of sub-operations, the total computational complexity is reduced to $O(M n_i^3)$. As M , the size of ensemble, is often set to be equal or greater than 25, the reduction of computational complexity is non-trivial.

Although there are two types of parameters: μ_i and β , the minimization of J_1 only depends on the ratio $\alpha_i = \mu_i/\beta$. These ratios, controlling the trade-off between the empirical training error and the regularization term, are crucial to the performance of ensemble. The next subsection presents a Bayesian approach to automatically optimize these parameters.

4.3.2 Inference of Regularization Parameters

In order to find the optimal parameters μ and β , we need to maximize the posterior of $P(\mu, \beta \mid D)$.

According to Bayesian rule, the posteriors of μ and β are obtained by

$$P(\mu, \beta \mid D) = \frac{P(D \mid \mu, \beta)P(\mu, \beta)}{P(D)} \propto P(D \mid \mu, \beta), \quad (4.18)$$

where a flat prior is assumed on the hyperparameters μ and β . According to equations (4.8) and (4.17), the marginal likelihood can be obtained in the following

4.3 Bayesian Formulation and Regularized Parameter Optimization

way (Gestel et al., 2002).

$$\begin{aligned} P(D \mid \mu, \beta) &= \frac{P(D \mid \mathbf{w}, \beta)P(\mathbf{w} \mid \mu)}{P(\mathbf{w} \mid D)} \\ &= \frac{(2\pi)^{W/2}|A|^{-\frac{1}{2}} \prod_{i=1}^M \left(\frac{\mu_i}{2\pi}\right)^{n_i/2} \left(\frac{\beta}{2\pi}\right)^{N/2} \exp(-J_1(\mathbf{w}))}{\exp(-\frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP}))}. \end{aligned} \quad (4.19)$$

Since $J_1(\mathbf{w}) \approx J_1(\mathbf{w}_{MP}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP})$ and $W = \sum n_i$ is the total number of weights in the ensemble,

$$\begin{aligned} P(D \mid \mu, \beta) &= (2\pi)^{W/2}|A|^{-\frac{1}{2}} \prod_{i=1}^M \left(\frac{\mu_i}{2\pi}\right)^{n_i/2} \left(\frac{\beta}{2\pi}\right)^{N/2} \exp(-J_1(\mathbf{w}_{MP})) \\ &= \left(\frac{1}{2\pi}\right)^{N/2} \sqrt{\frac{\prod_{i=1}^M \mu_i^{n_i} \beta^N}{\det A}} \exp(-J_1(\mathbf{w}_{MP})). \\ &\propto \sqrt{\frac{\prod_{i=1}^M \mu_i^{n_i} \beta^N}{\det A}} \exp(-J_1(\mathbf{w}_{MP})). \end{aligned} \quad (4.20)$$

In order to maximize the probability $P(D \mid \mu, \beta)$, negative logarithm is applied:

$$J_2 = \frac{1}{2} \sum_{i=1}^M \mu_i \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} + \frac{1}{2} \beta \sum_{n=1}^N e_{n,MP}^2 - \frac{1}{2} \sum_{i=1}^M n_i \log \mu_i - \frac{1}{2} N \log \beta + \frac{1}{2} \log \det A, \quad (4.21)$$

where the subscript MP indicates the most probable value.

Setting the gradient to zero and we can get the optimal $\alpha_i = \mu_i/\beta$. Please refer to appendix B for detail.

$$\alpha_i^{new} = \frac{\sum_{n=1}^N e_{n,MP}^2 \left(n_i - \sum_{j \in n_i} \frac{\alpha_i}{\lambda_j + \alpha_i} \right)}{\mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} \left(N - \sum_{j=1}^W \frac{\lambda_j}{\lambda_j + \hat{\alpha}_j} \right)}, \quad (4.22)$$

where $\hat{\alpha} = [\alpha_1^{(1)}, \dots, \alpha_1^{(n_1)}, \dots, \alpha_M^{(1)}, \dots, \alpha_M^{(n_M)}]^T$ and the superscript indicates the number of repetitions for α_i . $j \in n_i$ indicates the range $\left(\sum_{t=1}^{i-1} n_t + 1, \dots, \sum_{t=1}^i n_t \right)$, and λ_j is the eigenvalue of the Hessian matrix $\nabla \nabla \left(\frac{1}{2} \sum_{n=1}^N e_n^2 \right)$.

When the eigenvalue decomposition λ_j is calculated, the update rule of α_i^{new} involves only vector products that can be evaluated very quickly. In order to

reduce the computational complexity for large data sets, one can choose to calculate only the largest eigenvalues using the expectation maximization approach (Rosipal and Girolami, 2001).

The learning algorithm thus proceeds by repeated application of (4.22) (step 4 in Figure 4.1), concurrent with training RNCL, equivalent to updating of the posterior statistics from (4.11), until some suitable convergence criteria have been satisfied.

4.4 Numerical Experiments

In this section we present the numerical experiments of RNCL. Firstly, we present experimental results of RNCL on two synthetic regression problems and four synthetic classification problems in order to understand the behavior of the algorithm. We also design four experiments (two regressions and two classifications) with different noise levels to study the characteristics of RNCL and NCL with noise data. Secondly, we carry out extensive experiments on 8 benchmark regression data sets and 13 benchmark classification data sets to evaluate the performance of RNCL, NCL and Bagging.

4.4.1 Experimental Setup

In the experiments, three-layer feed-forward multi-layer perceptions (MLPs) are used as the base learners. The number of hidden nodes is randomly selected but restricted in the range 3 to 15. The initial connection weights of individual network are randomly chosen. We employ scaled conjugate gradient (SCG) algorithm to train MLP, NCL and RNCL. Since negative correlation learning algorithm uses MSE and correlation to train ensemble, it is not necessary to employ a large ensemble. We use 25 MLPs to constitute the ensemble of NCL and RNCL. For Bagging, we employ 100 MLPs to constitute the ensemble. The input attributes of data sets are scaled to mean zero and unit variance as the preprocessing procedure.

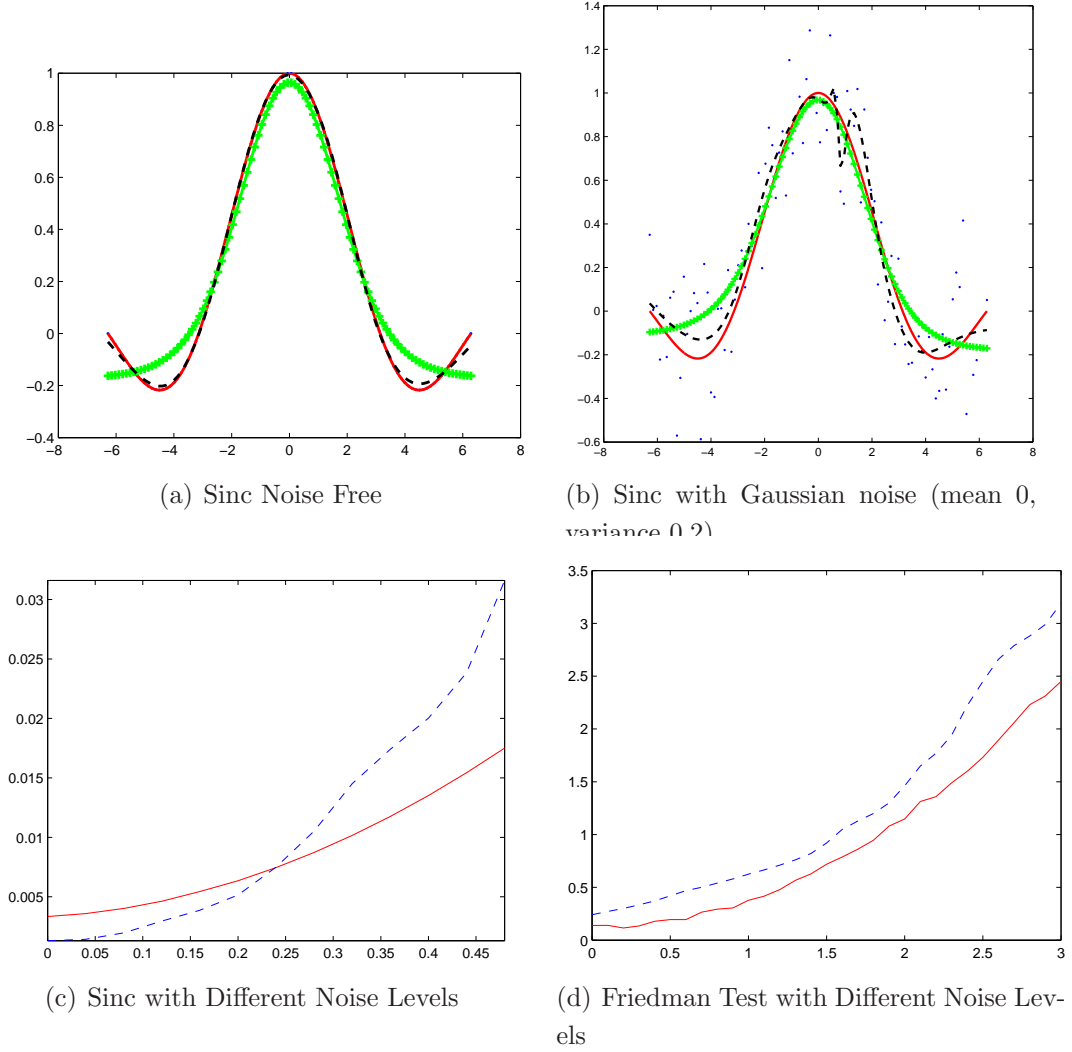


Figure 4.2: Comparison of NCL and RNCL on regression data sets: Sinc and Friedman test. In Figure 4.2(a) and 4.2(b), the lines in green (wide zigzag), black (dashed) and red (solid) are obtained by RNCL, NCL and the true function, respectively. Figure 4.2(c) and 4.2(d) show mean square error (MSE) of RNCL (red solid) and NCL (blue dashed) on Sinc and Friedman with different noise levels. Results are based on 100 runs.

4.4.2 Synthetic Experiments

As the first experiment, we compare RNCL and NCL algorithms on two synthetic regression data sets: Sinc and Friedman test. Figure 4.2(a) and 4.2(b) show the output of RNCL and NCL on sinc function with different noise levels. In the noise-free case, NCL perfectly approximates the actual function, while RNCL does not approximate the function very well near the tail. However, when the noise level increases, NCL, only minimizing the empirical error, overfits the noise in the training set while RNCL is more robust with respect to noise than NCL, refer to Figure 4.2(b).

In order to evaluate RNCL and NCL on training data with different noise levels, we add zero mean and different variance Gaussian noise to sinc and Friedman test problems. Figures 4.2(c) and 4.2(d) illustrate the average results of 100 runs. Since the standard deviations of the targets: sinc and Friedman test, are different, the range of noise levels are different in Figure 4.2(c) and 4.2(d).

For sinc data set, when the noise level (variance) is lower than 0.24, NCL outperforms RNCL. When the noise level is greater than 0.24, MSE of RNCL increases slower than that of NCL as the noise increases. For Friedman problem, RNCL outperforms NCL all the time and the difference between RNCL and NCL becomes greater when noise level passes 2.5. From these figures, RNCL is more robust with respect to noise.

In the following, we demonstrate the application of RNCL on classification problems. Firstly, we apply RNCL and NCL on four synthetic data in two dimensions in order to illustrate graphically the decision boundary.

These four data sets are (1) *synth* is generated from mixtures of two Gaussians by (Ripley, 1996). (2) *Overlap* comes from two Gaussian distributions with equal covariance, and it is expected to be separated by a linear plane. (3) *Bumpy* comes from two equal Gaussians but by being rotated by 90 degrees. Quadratic boundaries are required. (4) *Relevance* is a case where only one dimension of the data is relevant to separating the data.

In Figure 4.3, we observe a similar performance of RNCL and NCL in the case of *Relevance*. Since the data set is noise-free, both RNCL and NCL successfully separate the two classes. The situation is a little similar in the case of *Overlap*.

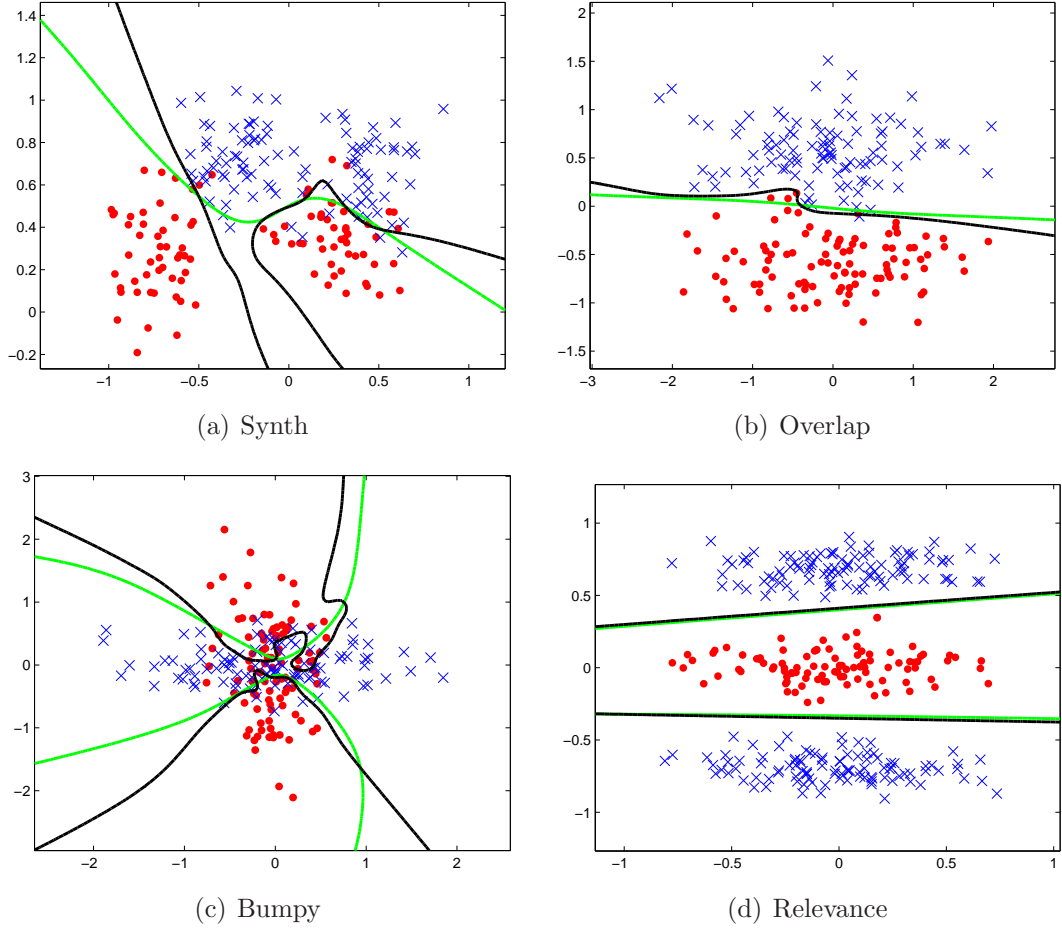


Figure 4.3: Comparison of RNCL and NCL on four synthetic classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. The lines in green (light) and black (dark) are obtained by RNCL and NCL, respectively.

RNCL produces a linear boundary according to the expectation while NCL concentrates on three overlapping points and generates a linear plane with a *corner*. Although the training error of NCL is smaller, it does not generalize well for this data set.

RNCL gives more accurate results in the other cases. In the cases of *Synth* and *Bumpy*, RNCL produces smooth boundary and disregards the outliers in the training points. In the case of *Bumpy*, the noise level is great because of these overlapping points. NCL does not generalize and produces the twisty boundary. Although the boundary line of NCL for the case of *Synth* seems smooth, it separates the decision boundary into two parts and disregards the future points between the two boundaries.

To compare RNCL and NCL on noisy classification problems, we conduct similar noise experiments as the regression problems. In the experiments, we select two data sets: *synth* and *banana*¹.

To change the noise level, we randomly select different percentages of data points and reverse their labels. We run 100 times and report the average results in Figure 4.4. Figures 4.4(a) and 4.4(b) visualize the decision boundaries of RNCL and NCL with 20% noise points.

Although the noise level is high, RNCL produces smooth boundary. NCL tries to minimize the training error and it does not generalize well. We also plot the curve, Figure 4.4(c) and 4.4(d), of the error rate vs. the noise level for these data sets. In these two figures, RNCL is a little better in the beginning, but as the noise level increases, RNCL significantly outperforms NCL.

The results of RNCL are promising on these regression and classification problems. Based on the results and analysis, RNCL inherits the advantages of NCL and can achieve a good performance with a small ensemble. The regularization term does work in RNCL and improves its ability against noise, which is especially important in practice since most of the actual data are contaminated by noise. After the analysis with synthetic data sets, the next subsection presents the results for the real-world benchmark problems.

¹<http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm>

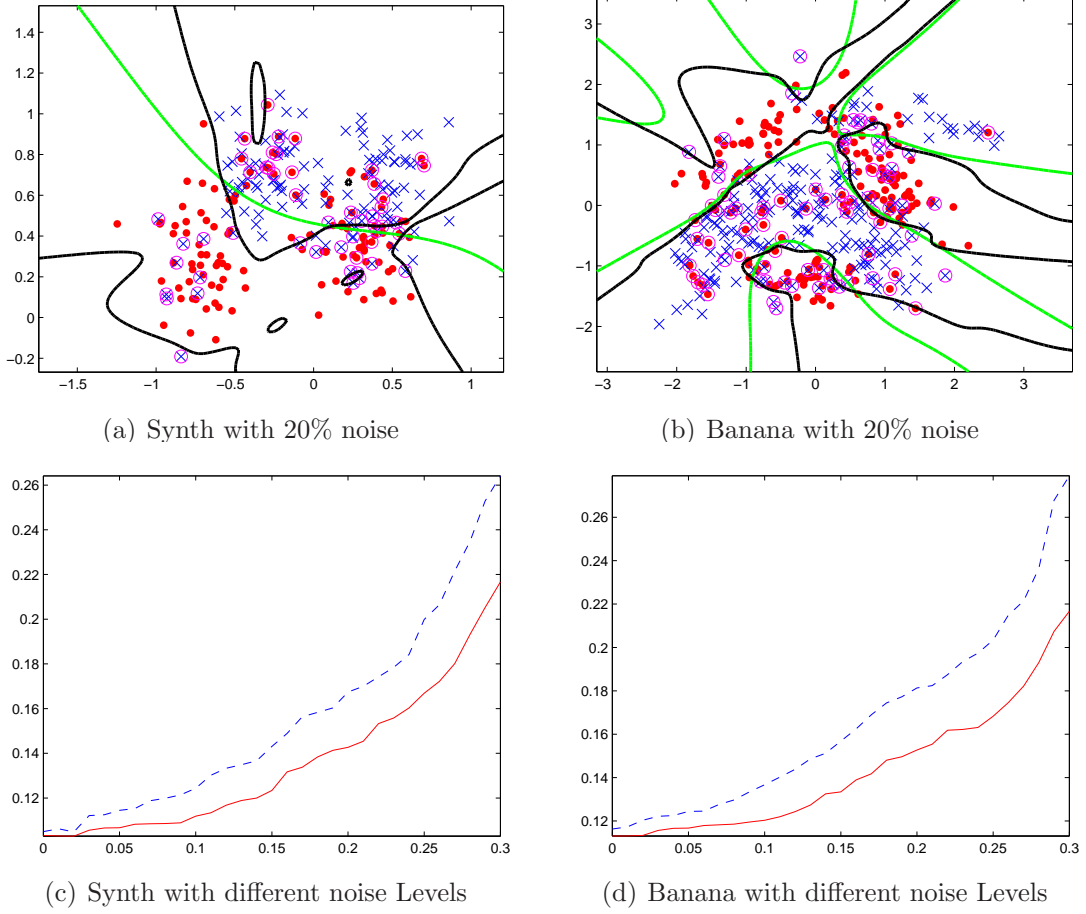


Figure 4.4: Comparison of RNCL and NCL on two classification data sets. Two classes are shown as crosses and dots. The separating lines are obtained by projecting test data over a grid. In Figure 4.4(a) and 4.4(b), the decision boundary in green (light) and black (dark) are obtained by RNCL and NCL, respectively. The randomly-selected noise points are marked with a circle. Figure 4.4(c) and 4.4(d) show the error rate of RNCL (red solid) and NCL (blue dashed) vs. the noise levels on Synth and banana data sets. The results are based on 100 runs.

Table 4.1: Summary of Regression Data Sets

Data Sets	Function	Variable	Training Points	Test Points
Mexican Hat	$y = \text{sinc} x = \frac{\sin x }{ x }$	$x \sim U[-2\pi, 2\pi]$	250	1000
Friedman 1	$y = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$	$x_i \sim U[0, 1]$	250	1000
Gabor	$y = \frac{1}{2}\pi \exp[-2(x_1^2 + x_2^2)] \cos[2\pi(x_1 + x_2)]$	$x_i \sim U[0, 1]$	250	1000
Multi	$y = 0.79 + 1.27x_1x_2 + 1.56x_1x_4 + 3.42x_2x_5 + 2.06x_3x_4x_5$	$x_i \sim U[0, 1]$	250	1000
Plane	$y = 0.6x_1 + 0.3x_2$	$x_i \sim U[0, 1]$	250	1000
Polynomial	$y = 1 + 2x + 3x^2 + 4x^3 + 5x^4$	$x \sim U[0, 1]$	250	1000
Sinc	$y = \frac{\sin x}{x}$	$x \sim U[0, 2\pi]$	250	1000
Boston House	—	-	400	106

4.4.3 Benchmark Results

In the benchmark experiments, we evaluate RNCL, NCL and Bagging with 8 regression benchmark problems and 13 classification benchmark problems. The information on the data sets used for regression is tabulated in Table 4.1. The Mexican hat was used by Weston et al. (Weston et al., 1996) in investigating the performance of support vector machines. Friedman 1 was used by Breiman (Breiman, 1996a) in testing the performance of Bagging. Gabor, Multi, and Sinc were used by Hansen (Hansen, 2000) in comparing several ensemble approaches. Plane was used by Ridgeway et al. (Ridgeway et al., 1999) in evaluating the performance of boosted naive Bayesian regressors. The constraints of the variables are also shown in Table 4.1, where $U[x, y]$ means a uniform distribution over the interval determined by x and y . Note that in our experiments additive zero-mean Gaussian noise, whose variance is one-third of the standard deviation of the target $y(x)$, is generated. The Boston House data set is obtained from UCI machine learning repository (Asuncion and Newman, 2007). In the 100 runs, we randomly select 400 data points for the training set and the rest 106 points are used for testing.

The classification data sets used in this chapter have been summarized in Table 4.2. These data sets have been preprocessed and organized by Rätsch et al.¹. These data sets include one synthetic set (banana) along with 12 other

¹<http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm>

4.4 Numerical Experiments

Table 4.2: Summary of Classification Data Sets.

Data Sets	Training Points	Test Points	Input Dimensions
Banana	400	4900	2
Cancer	200	77	9
Diabetics	468	300	8
Solar	666	400	9
German	700	300	20
Heart	170	100	13
Image	1300	1010	18
Ringnorm	400	7000	20
Splice	1000	2175	60
Thyroid	140	75	5
Titanic	150	2051	3
Twonorm	400	7000	20
Waveform	400	4600	21

Table 4.3: Comparison of NCL, Bagging and RNCL on 8 Regression Data Sets, by MSE (standard deviation) and t test p value between Bagging vs. RNCL and NCL vs. RNCL. The p value with a star means the test is significant. These results are averages of 100 runs.

Data Sets	Bagging	P value	NCL	P value	RNCL
Mexican Hat	0.0064(0.0018)	0.07	0.0069(0.0013)	0.00*	0.0060(0.0017)
Friedman	1.75(0.34)	0.00*	1.05(0.24)	0.00*	0.82(0.21)
Gabor	0.020(0.004)	0.00*	0.015(0.002)	0.00*	0.009(0.004)
Multi	0.038(0.010)	0.05	0.105(0.030)	0.00*	0.035(0.009)
Plane	0.83e-4(0.48e-4)	0.00*	1.64e-4(0.52e-4)	0.11	1.42e-4(0.48e-4)
Polynomial	0.159(0.075)	0.00*	0.128(0.044)	0.00*	0.076(0.023)
Sinc	0.0067(0.0018)	0.07	0.0070(0.0015)	0.00*	0.0066(0.0016)
Boston House	13.41(4.71)	0.02*	12.56(3.62)	0.47	12.16(3.51)

4.4 Numerical Experiments

Table 4.4: Comparison of NCL, Bagging and RNCL on 13 benchmark Data Sets, by % error (standard deviation) and t test p value between Bagging vs. RNCL and NCL vs. RNCL. The p value with a star means the test is significant. These results are averages of 100 runs.

Error	Banana	Cancer	Diabetics	Solar	German	Heart	
Bagging	11.41(0.78)	28.12(4.87)	24.23(1.78)	34.97(1.51)	24.97(2.10)	18.71(3.10)	
Bagging vs. RNCL	0.00*	0.16	0.00*	0.00*	0.10	0.00*	
NCL	11.09(0.68)	28.42(4.61)	24.57(1.96)	35.42(1.79)	25.88(2.19)	18.28(3.68)	
Bagging vs. RNCL	0.01*	0.21	0.03*	0.00*	0.00*	0.00*	
RNCL	10.42(0.65)	26.31(4.77)	23.16(1.62)	33.86(1.71)	24.01(2.23)	16.32(3.11)	
Error	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
Bagging	3.34(0.69)	1.84(0.31)	11.62(0.62)	4.48(2.36)	23.98(1.37)	3.03(0.30)	11.68(0.62)
Bagging vs. RNCL	0.00*	0.05	0.00*	0.31	0.15	0.00*	0.00*
NCL	2.65(0.46)	1.64(0.24)	11.23(0.72)	4.45(2.37)	22.71(1.32)	2.61(0.26)	12.30(0.76)
Bagging vs. RNCL	0.06	0.01*	0.03*	0.34	0.89	0.13	0.00*
RNCL	2.79(0.68)	1.79(0.19)	10.53(0.64)	4.03(2.11)	22.42(1.03)	2.79(0.21)	9.91(0.48)

real-world data sets coming from the UCI (Asuncion and Newman, 2007) and DELVE¹ repositories. The major difference between the original and Rätsch’s data is that Rätsch converted every problem into binary classes and randomly partitioned every data set into 100 training and testing folds (Splice and Image have only 20 folds in the Rätsch’s implementation and we generate additional 80 folds by random sampling to make the experiments consistent). In addition, every instance is normalized dimension-wise to have zero mean and unit standard deviation.

Table 4.3 reports the performance of these algorithms on the 8 benchmark regression data sets. According to the table, RNCL performs excellently in these data sets. For example, RNCL outperforms the other two methods in 7 out of 8 data sets, in which 6 wins are significant against NCL and 4 wins are significant against Bagging.

The performance of RNCL, NCL and Bagging on classification problems have been tabulated in Table 4.4. Based on the table, RNCL performs very well since RNCL outperforms all the other methods in 11 out of 13 data sets, comes second

¹<http://www.cs.toronto.edu/~delve/data/datasets.html>

4.4 Numerical Experiments

Table 4.5: Running Time (in seconds) of RNCL and NCL on Regression Data Sets. Results are averaged over 100 runs.

Data Sets	Mexican Hat	Friedman	Gabor	Multi	Plane	Polynomial	Sinc	House
NCL	6.4	16.1	8.4	21.2	3.3	23.6	6.2	28.6
RNCL	19.6	143.2	30.8	113.2	7.9	62.3	20.3	269.5

Table 4.6: Running Time (in seconds) of RNCL and NCL on Classification Data Sets. Results are averaged over 100 runs.

Error	Banana	Cancer	Diabetics	Solar	German	Heart	
NCL	3.6	8.2	4.6	12.1	21.6	1.6	
RNCL	12.1	29.4	17.6	36.0	168.5	16.7	
Error	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
NCL	21.3	4.8	43.6	3.1	0.8	4.0	4.2
RNCL	277.2	20.7	426.7	8.4	2.6	34.8	41.7

in 2 cases. In the results, NCL outperforms RNCL in the cases: Ringnorm and Twonorm, which are both synthetic data with little noise (see the lower error rate). This observation also validates that NCL achieves better results when noise is small and RNCL is more robust with respect to noise than NCL.

4.4.4 Computational Complexity and Running Time

Based on the algorithm in Figure 4.1, RNCL consists of two main parts: neural network training using regularized negative correlation learning and Bayesian parameter optimization.

In the first part, for each component neural network, totally M neural networks in the ensemble, one needs to train the network with an amount of epochs. Since the scaled conjugate gradient algorithm is employed in RNCL, the training can be evaluated quickly.

Table 4.7: Comparison of RNCL and NCL with equal time on four regression problems and four classification problems. NCL is run 10 times in 8 experiments with randomly selected regularization parameters between 0 and 1. The first row reports the best performance of NCL in the 10 runs. The results are the average results of 20 runs.

Data Sets	Mexican Hat	Friedman	Gabor	House	Banana	Cancer	Diabetics	Solar
(Best) NCL	0.0064	0.93	0.013	12.33	10.86	27.11	23.32	34.16
RNCL	0.0060	0.82	0.009	12.16	10.42	26.31	23.16	33.86

In the second part, the major running time is consumed in the calculation of Hessian matrix and eigen-decomposition of the Hessian matrix. This chapter employs the fast multiplication by the Hessian (Møller, 1993a; Pearlmutter, 1994) method to estimate the Hessian matrix. Although the eigenvalues have to be calculated only once, their calculation in the eigenvalue problem becomes computationally expensive for large data sets. In this case, one can choose to calculate only the largest eigenvalues using an expectation maximization approach (Rosipal and Girolami, 2001).

If the Hessian matrix is not so large, most of the computation time will be consumed in the first part. RNCL is an iterative algorithm and in most of time it will converge in less than 8 iterations. Therefore, the computation time of RNCL is 5-10 times of NCL. In Tables 4.5 and 4.6, we show the average running time of RNCL and NCL over 100 runs. The computational environment is Windows XP with Intel Core 2 Duo 1.66G CPU and 2G RAM. These algorithms including RNCL and NCL are programmed in C++.

Based on these Tables, the running time of RNCL is 5-10 times of that of NCL. In order to fairly compare RNCL with NCL, the following experiments are carried out. We run NCL 10 times on 8 data sets with randomly selected regularization parameter in the range of $[0, 1]$ and let the total time of NCL equals to the running time of RNCL.

Table 4.7 reports the *best* performance of NCL in 10 runs on 8 data sets based on the average results of 20 runs. From the table, we confirm that even

given the same amount of time to NCL, NCL cannot pick the best regularization parameter as Bayesian inference does in RNCL. The Bayesian parameter inference does improve the performance of NCL.

4.5 Summary

This chapter analyzes negative correlation learning and theoretically and empirically demonstrates that NCL is prone to overfitting the noise, which is the first contribution of the chapter.

To overcome the shortcoming of NCL, we propose the regularized negative correlation learning (RNCL) which incorporates an additional regularization term for NCL. RNCL decomposes the ensemble's training objectives, including MSE and regularization, into a set of sub-objectives, and each sub-objective is implemented by one individual neural network. Moreover, we formulate RNCL by Bayesian technique and propose an algorithm to optimize the regularization coefficients by Bayesian inference. The RNCL and the Bayesian inference algorithm constitute the second contribution of the chapter.

Several experiments have been carried out to evaluate RNCL. The experiments on two synthetic regression problems and four synthetic classification problems demonstrate the behavior of RNCL and NCL. The following experiments on two regression and two classification problems with different noise demonstrate that RNCL achieves better performance than NCL, especially when the noise is non-trivial in data sets. Secondly, we carry out extensive experiments on 8 benchmark regression and 13 benchmark classification data sets to evaluate RNCL, NCL and Bagging. RNCL has shown an excellent performance on these data sets.

This chapter analyzes the computational complexity of RNCL and carries out experiments to demonstrate that even NCL is given the same time as RNCL, NCL could not achieve the same performance as RNCL. The noise-robustness characteristic of RNCL is especially important when the training data are contaminated with noise.

In general, RNCL is a generic ensemble algorithm, which is best viewed as a framework, rather than an algorithm itself.

Chapter 5

Multiobjective Regularized Negative Correlation Learning

In section 4, we provided the theoretical and empirical evidences to explain that NCL is prone to overfitting the noise, and we proposed regularized negative correlation learning (RNCL), which was implemented by gradient descent with Bayesian inference. As discussed in chapter 4, RNCL was implemented by minimizing the three terms with weighting coefficients. Inspired by this, this chapter proposes multiobjective regularized negative correlation learning (MRNCL) algorithm, where the three terms of RNCL are treated as three objectives and multiobjective algorithm is used to search the trade-off among the three objectives. The rest of this chapter is organized as follows. After the introduction in section 5.1, the proposed algorithm is described in section 5.2. Experimental results and discussions are presented in section 5.3. Finally, section 5.4 concludes the chapter.

5.1 Introduction

Most ensemble learning algorithms train the base learners independently or sequentially, so the advantages of interaction and cooperation among the base learner are not exploited. Liu and Yao (Liu and Yao, 1999a,b; Liu et al., 2000) proposed negative correlation learning (NCL) and showed that ensemble methods benefits from considering the cooperation among the base learners. This

approach opens a new research area where the design and training of the base learners can be interdependent.

Although NCL performs well for a broad range of practical applications by considering the cooperation in the ensemble, it is not regularized, which leads to overfitting, and the weighting coefficient, which controls the trade-off between empirical error and correlation, needs to be tuned.

In order to address these problems, we proposed regularized negative correlation learning (RNCL) with an algorithm to optimize the regularization parameter by Bayesian inference in chapter 4, where RNCL was implemented by minimizing the three terms with weighting coefficients. Inspired by this, RNCL can be implemented by a multiobjective algorithm, where each minimization term is treated as an objective.

The trade-off among the three terms is crucial for the generalization performance of ensemble. Poor generalization occurs if the trade-off is unbalanced. For example, a small regularization term may lead to overfitting with noise data sets and a large regularization may seriously bias the learning outcome. The situation is applicable to the correlation term as well.

One approach to balance the trade-off is to assign coefficient parameters to these terms and choose the appropriate coefficients. The usual way to choose the coefficients is to train several networks with different values of these coefficients and estimate the generalization error for each network and then choose the coefficients that minimize the estimated generalization error. However, the computation of this approach is extremely expensive.

Evolutionary multiobjective algorithms are well suited to search the optimal trade-off among different objectives by parallelizing the searching using a population of networks and biasing toward the Pareto front and at the same time maintaining population diversity to obtain as many diverse solutions as possible. These properties are especially important in ensemble design.

This chapter proposes multiobjective regularized negative correlation learning (MRNCL) algorithm, which implements RNCL algorithm by an evolutionary multiobjective algorithm. MRNCL involves minimization of the three terms: empirical training error term, correlation penalty term and the regularization term. MRNCL algorithm not only addresses the issues concerned with NCL,

but also provides the following advantages: (1) Being a multiobjective algorithm, the approach is able to produce a diverse ensemble. Some individuals are good at minimizing the training error; some pay more attention to cooperation and the others manage to control the complexity. (2) The parameters of individual network can be effectively obtained in the evolutionary multiobjective algorithm. (3) Due to the regularization term in MRNCL, the obtained ensemble is regularized and is more robust with respect to noise. (4) There is no need to weigh the different objectives by optimizing the coefficient parameters.

5.2 Multiobjective Regularized Negative Correlation Learning

In this section, we detail multiobjective regularized negative correlation learning algorithm (MRNCL). Section 5.2.1 proposes MRNCL, followed by the description of component network and evolutionary operators in section 5.2.2. Section 5.2.3 describes multiobjective evaluation of ensemble and rank-based fitness assignment. Finally, section 5.2.4 presents the algorithm of MRNCL.

5.2.1 Multiobjective Regularized Negative Correlation Learning

Multiobjective Regularized Negative Correlation Learning (MRNCL) introduces a regularization term into the error function of NCL. Given the training set $\{\mathbf{x}_n, y_n\}_{n=1}^N$, MRNCL combines M neural networks $f_i(\mathbf{x})$ to constitute the ensemble.

$$f_{ens}(\mathbf{x}_n) = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}_n). \quad (5.1)$$

To train the ensemble, the cost function E_{ens} for the ensemble is defined as follows:

$$E_{ens} = \frac{1}{M} \sum_{n=1}^N \sum_{i=1}^M (f_i(\mathbf{x}_n) - y_n)^2 + \lambda \frac{1}{M} \sum_{i=1}^M p_i + \sum_{i=1}^M \alpha_i \mathbf{w}_i^T \mathbf{w}_i, \quad (5.2)$$

5.2 Multiobjective Regularized Negative Correlation Learning

where $\mathbf{w}_i = (w_{i,1}, \dots, w_{i,n_i})^T$ is the weight vector of neural network i , and n_i is the total number of weights in network i . α_i and λ are weighting coefficients on the regularization term and the correlation term p_i , respectively.

The correlation term p_i is presented by

$$\begin{aligned} p_i &= \sum_{n=1}^N \left\{ (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \sum_{j \neq i} (f_j(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \right\} \\ &= - \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2. \end{aligned} \quad (5.3)$$

In order to train each neural network with its regularization, we decompose the error function of ensemble into M parts, each part for a network. The error function of network i can be obtained as follows:

$$e_i = \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n)^2 - \lambda \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2 + \alpha_i \mathbf{w}_i^T \mathbf{w}_i. \quad (5.4)$$

Comparing this error function with the cost function of NCL, equation (4.2), MRNCL imposes a regularization term on every individual neural network and MRNCL needs to optimize both the correlation coefficient λ and the regularization parameters α_i .

According to equation (5.4), the training of an individual neural network in MRNCL involves minimization of the three terms: empirical training error term, correlation penalty term and the regularization term. The generalization of ensemble depends on the trade-off among the three terms and this chapter uses the evolutionary multiobjective algorithm to balance the trade-off.

The formulation of MRNCL is not a heuristic but based on the Bayesian statistical model. According to chapter 4, MRNCL is an application of Bayesian framework in ensemble system. The squared weight decay term, i.e. the regularization term, acts as the prior of weight vector in the ensemble. This is the reason why we only include the squared weight decay term as the regularization term in the multiobjective algorithm. This intrinsic Bayesian characteristic of MRNCL potentially facilitates the incorporation of Bayesian methods with evolutionary multiobjective algorithms to improve the performance of MRNCL.

According to equation (5.4), MRNCL defines the following three objectives.

5.2 Multiobjective Regularized Negative Correlation Learning

- Objective of Performance $\sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n)^2$.

This objective measures the empirical mean square error based on the training set.

- Objective of Cooperation $-\sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2$.

This cooperation term measures the amount of variability among the ensemble member and this term can also be treated as the diversity measure (Krogh and Vedelsby, 1995). From both theoretical and experimental results, the most effective combination of ensemble members occurs when the errors of these ensemble members are negatively correlated. This objective is to obtain highly correlated networks to disagree as much as possible.

- Objective of Regularization $\mathbf{w}_i^T \mathbf{w}_i = \sum_j w_{i,j}^2$.

Based on the regularization theory (Vapnik, 1995), the weight decay term (Krogh and Hertz, 1992) is employed to punish large weights. The weight decay term causes the weights to converge to smaller absolute values than they otherwise would. The regularization term helps the generalization ability of neural network and reduces the variance (Geman et al., 1992).

5.2.2 Component Network and Evolutionary Operators

A radial basis function (RBF) network is used as the component network in this chapter. The output of RBF network is computed as a linear combination of K basis functions

$$f(\mathbf{x}) = \sum_{k=1}^K w_k \phi_k(\mathbf{x}) = \Phi^T \mathbf{w}, \quad (5.5)$$

where $\mathbf{w} = (w_1, \dots, w_K)^T$ denotes the weight vector in the output layer and $\Phi = (\phi_1, \dots, \phi_K)^T$ is the vector of basis functions. The Gaussian basis functions ϕ_k are defined as

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{2\sigma_k^2}\right), \quad (5.6)$$

where μ_k and σ_k denote means and widths of the Gaussian, respectively. The training of RBF network is separated into two steps. In the first step, the means μ_k are initialized with randomly selected data points from the training set and

5.2 Multiobjective Regularized Negative Correlation Learning

the variances σ_k are determined as the Euclidean distance between μ_k and the closest $\mu_i (i \neq k, i \in \{1, \dots, K\})$. Then in the second step we perform a gradient descent with the regularized error function (weight decay)

$$\min E = \frac{1}{2} \sum_{n=1}^N (y_n - f(\mathbf{x}_n))^2 + \alpha \sum_{k=1}^K w_k^2. \quad (5.7)$$

In order to fine-tune the centers and widths, we simultaneously adjust the output weights, the RBF centers and variances. Taking the derivative of equation (5.7) with respect to RBF means μ_k and variances σ_k^2 we obtain

$$\frac{\partial E}{\partial \mu_k} = \sum_{n=1}^N (f(\mathbf{x}_n) - y_n) \frac{\partial f(\mathbf{x}_n)}{\partial \mu_k}, \quad (5.8)$$

with $\frac{\partial f(\mathbf{x}_n)}{\partial \mu_k} = w_k \frac{\mathbf{x}_n - \mu_k}{\sigma_k^2} \phi_k(\mathbf{x}_n)$ and

$$\frac{\partial E}{\partial \sigma_k} = \sum_{n=1}^N (f(\mathbf{x}_n) - y_n) \frac{\partial f(\mathbf{x}_n)}{\partial \sigma_k}, \quad (5.9)$$

with $\frac{\partial f(\mathbf{x}_n)}{\partial \sigma_k} = w_k \frac{\|\mathbf{x}_n - \mu_k\|^2}{\sigma_k^3} \phi_k(\mathbf{x}_n)$. These two derivatives are employed in the minimization of equation (5.7) by a scaled conjugate gradient descent, where we always compute the optimal output weights in every evaluation of the error function. The optimal output weights \mathbf{w} can be computed in closed form by

$$\mathbf{w} = (\Phi^T \Phi + \alpha I)^{-1} \Phi^T \mathbf{y}, \quad (5.10)$$

where $\mathbf{y} = (y_1, \dots, y_n)^T$ denotes the output vector, and I is an identity matrix.

We use RBF network as the base learner because of the following advantages. 1) Once the centers and the widths of the basis functions have been fixed, the optimal output weights \mathbf{w} can be efficiently computed in closed form, which means the performance mostly depends on the selection of basis functions. 2) It is reasonable to define crossover and mutation operators in structural-evolving RBF network by tuning these basis functions.

Based on the above reasons, the crossover operator and mutation operator for RBF networks are described as follows.

- Crossover Operator

As the performance of a RBF network mostly depends on the basis functions, i.e. the centers and the widths, the crossover operator is defined to exchange the basis functions of two RBF networks. Many crossover techniques exist in the literature, such as one-point crossover, two-point crossover and “cut and splice” crossover. In a RBF network ensemble, as different networks may have different numbers of basis functions, the “cut and splice” approach has been adopted by randomly choosing separate crossover points for two RBF networks and swap their basis functions beyond those points.

- Mutation Operator

This chapter defines two structural mutation operators for RBF networks.

1. Deleting one basis function. Randomly select one basis function and delete it.
2. Adding one basis function. The center of the new basis function is determined by a randomly selected data point from the training set. Then, the width of the basis function is chosen as the minimal distance from other centers in this RBF network.

As the crossover and mutation operations may not generate the optimal combination of basis functions, afterwards, we simultaneously adjust the output weights, the RBF centers and widths based on equations (5.8), (5.9) and (5.10). This procedure is also called parametric mutation, which only modifies the parameters of the network without modifying its topology. This parametric mutation is performed for a few iterations (in our experiments, only one scaled-conjugate-gradient update is employed).

5.2.3 Multiobjective Evaluation of Ensemble and Rank-based Fitness Assignment

In this subsection, we consider a population of individuals who have three objectives and a multiobjective algorithm is employed to select a set of best classifiers with respect to the three objectives.

5.2 Multiobjective Regularized Negative Correlation Learning

In this chapter, nondominated sorting with fitness sharing (Srinivas and Deb, 1995) and rank-based fitness assignment have been used. Nondominated sorting is based on layers of Pareto front, which ranks the individuals in the population by fronts that leads to fast convergence to Pareto front in the final population. The diversity of population is maintained by a niching method.

The nondominated sorting algorithm consists of two stages: One is to obtain the nondominated fronts of different layers and every individual of these fronts is assigned an equal dummy fitness. The algorithm used for obtaining the non-dominated set of solutions compares the individuals pairwise and marks these individuals, which are dominated by at least one member of the population, as dominated. The second is that the members of every front share their fitness (Darwen and Yao, 1996) with the constraint that none of the members of a front gets a higher fitness than any of the members of the previous front.

Since the dummy fitness assigned by nondominated sorting is raw, sometimes the range of the dummy fitness is too large, leading to the situation that some networks reproduce too rapidly, taking over the population too quickly, and preventing the evolutionary algorithm from searching other areas of the solution space. Fitness scaling is the process of mapping an arbitrary fitness range into an appropriate range.

This chapter employs rank-based fitness assignment to reassign the fitness to the networks because rank-based fitness assignment behaves in a more robust manner than proportional fitness assignment. In the rank-based fitness assignment, the population is sorted according to the raw fitness values. The fitness assigned to each individual depends only on its position in the individual's ranking and not on the actual raw fitness value.

We use a linear rank-based fitness assignment, where the fitness value for an individual is calculated as:

$$fitness(Pos) = 2 - SP + 2(SP - 1)\frac{Pos - 1}{M - 1}, \quad (5.11)$$

where M is the number of individuals in the population. Pos is the position of an individual in this population (least fit individual has $Pos = 1$, the fittest individual $Pos = M$) and SP is the selective pressure. Linear ranking allows

5.2 Multiobjective Regularized Negative Correlation Learning

1. Generate an initial RBF network population: Generate an initial population of M RBF Networks, the number of hidden nodes K for each network is specified randomly restricted by the maximal number of hidden nodes. The centers μ_k are initialized with randomly selected data points from the training set and the width σ_k are determined as the Euclidian distance between μ_k and the closest $\mu_j (j \neq k, j \in \{1, \dots, K\})$.
2. Train the initial RBF network population and recode the three objective values of each network.
3. Apply nondominated sorting with rank-based fitness assignment algorithm to obtain the rank-based fitness.
4. For 1 to maximal generation
 - Perform a desired number of crossover operations.
Choose parents based on roulette wheel selection algorithm and perform crossover. Then perform a few number of updates for weights, centers and widths. Compare the children with parents and keep the better ones.
 - Perform a desired number of mutation operations.
Choose parents based on roulette wheel selection algorithm and perform mutation. Then perform a few number of updates for weights, centers and widths. Compare the children with parents and keep the better ones.
 - Apply nondominated sorting algorithm and obtain the rank-based fitness for the new population.
5. Combine these classifier to form the ensemble.

Figure 5.1: Multiobjective Regularized Negative Correlation Learning Algorithm

values of the parameter SP in $[1.0, 2.0]$. Our algorithm adopts 1.5 as the selective pressure.

Note that we use all the individuals in the population to generate the ensemble.

5.2.4 Algorithm Description

The details of Multiobjective Regularized Negative Correlation Learning (MRNCL) are summarized in Figure 5.1. Note that in the crossover and mutation operation, the comparison of the child network with the parent network is conducted as follows.

1. Evaluate the three objective values of the child network.

5.2 Multiobjective Regularized Negative Correlation Learning

2. Include the child network into the population, then apply non-dominant sorting with fitness sharing algorithm to obtain the raw fitness values of the child network and the parent network.
3. Compare the raw fitness values and keep the better one.

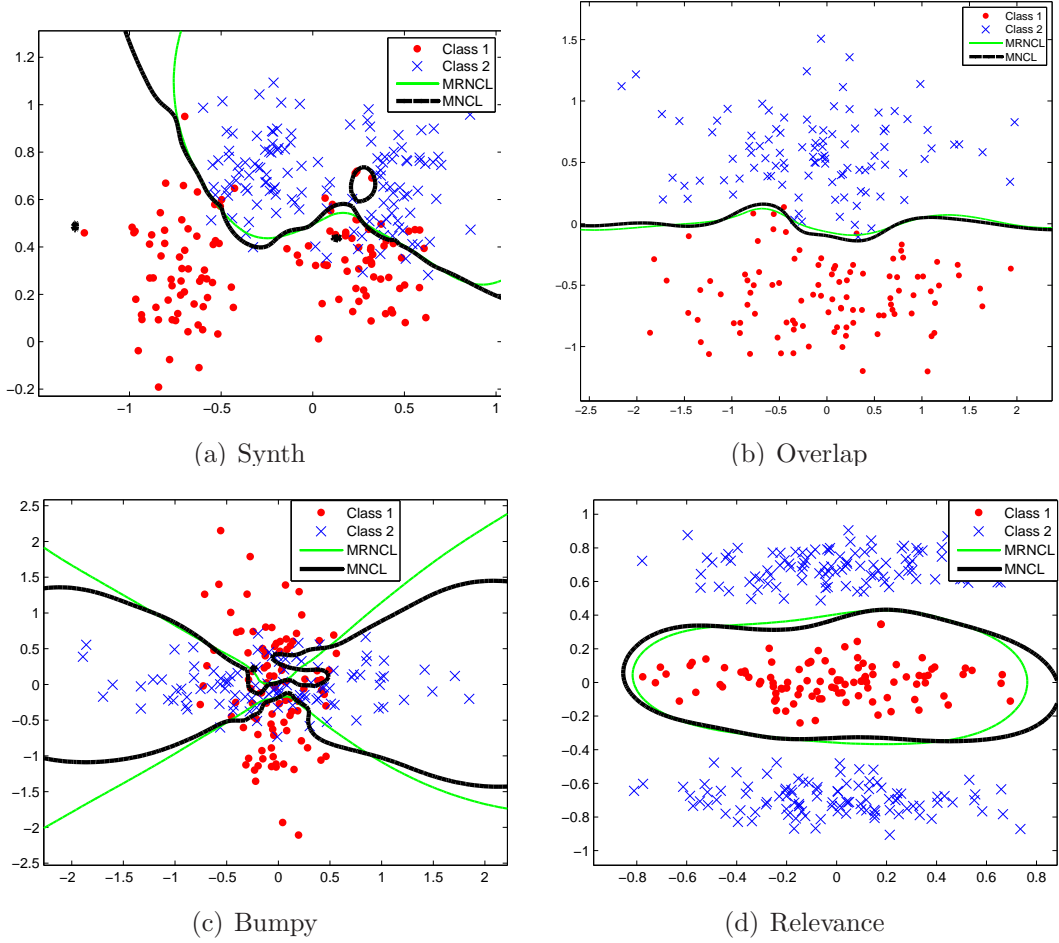


Figure 5.2: Comparison of MRNCL and MNCL on four synthetic classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. The lines in green (thin) and black (thick) are obtained by MRNCL and MNCL, respectively.

5.3 Numerical Experiments

In this section we present the experimental results of MRNCL and MNCL, which employs a multiobjective algorithm (two objectives: training error and correlation term) to train negative correlation ensembles. We use MNCL instead of gradient-based NCL because MRNCL uses a multiobjective algorithm and it is fair and natural to use multiobjective algorithm to train NCL.

In this section, firstly, we present experimental results of MRNCL on four synthetic classification problems and we design two experiments to study the characteristics of MRNCL and MNCL on data with different noise levels. Secondly, we carry out extensive experiments on 13 benchmark classification data sets to evaluate MRNCL, MNCL and other classifiers.

5.3.1 Experimental Setup

In our experiments, radial basis function (RBF) networks are used as the base classifiers. The number of hidden nodes is randomly selected but restricted in the range of 5 to 15. The parameters in the evolutionary algorithm are set to: the population size M (100), the number of crossover in one generation 20, the number of mutation in one generation 10, the number of generations (100), the parameter of fitness sharing σ_{share} (0.2). These parameters are chosen after some preliminary experiments. They are not meant to be optimal.

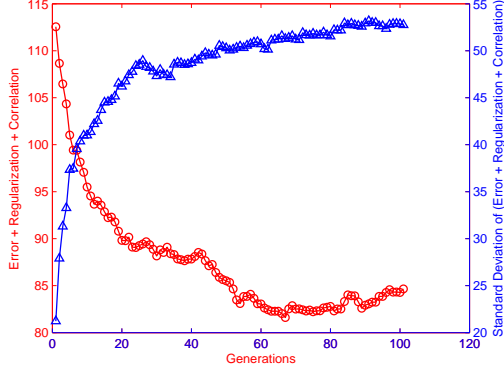
5.3.2 Synthetic Data Sets

As the first experiment, we demonstrate the results of MRNCL on four synthetic data in two dimensions in order to illustrate graphically the decision boundary.

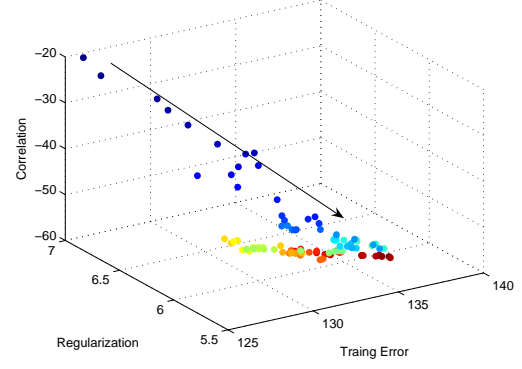
These four data sets are *synth*, *Overlap*, *Bumpy* and *Relevance*. They are used in section 4.4. The detailed information of these data sets can be referred to section 4.4.

In Figure 5.2 we present a comparison of MRNCL and MNCL. We can observe a similar performance of MRNCL and MNCL in the case of *Relevance*. Since the data set is noise-free, both MRNCL and MNCL successfully separate the two classes. The situation is similar in the case of *Overlap*. Since it is extremely

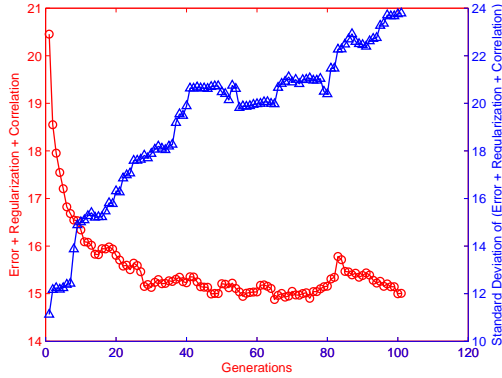
5.3 Numerical Experiments



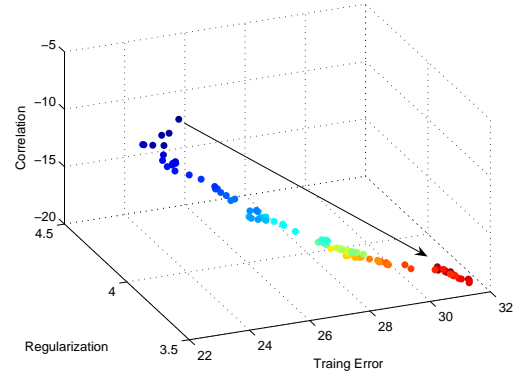
(a) Banana



(b) Banana Evolving Information in 3D

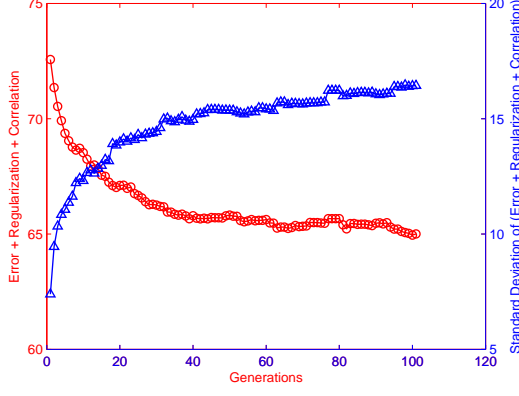


(c) Overlap

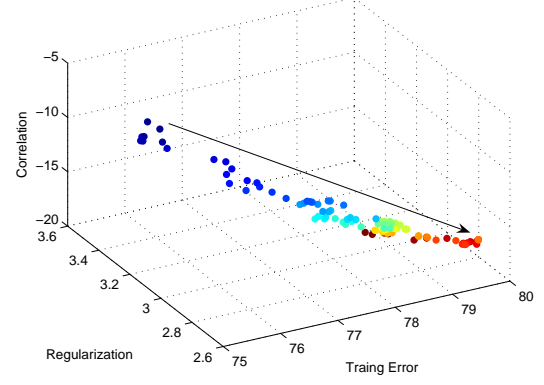


(d) Overlap Evolving Information in 3D

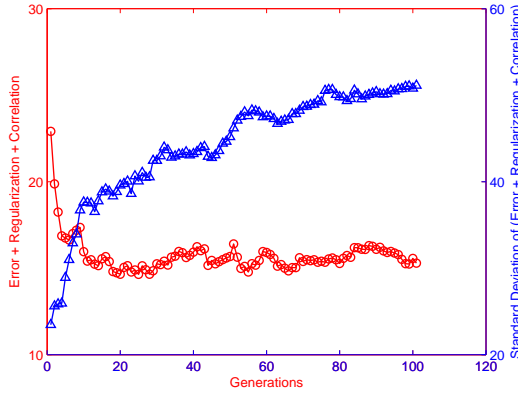
Figure 5.3: Detailed information in multiobjective algorithm for two data sets, Banana and Overlap. In Figure 5.3(a) and 5.3(c), the left-y axis (red line with circles) measures the summation of the mean of three objectives, training error, regularization and correlation in different generations. The right-y axis (blue line with triangles) is the standard deviation of the summation. In Figure 5.3(b) and 5.3(d), the 3D figure records the mean value of these three objectives in different generations. The arrow points from the beginning (Generation = 1) to end (Generation = 100). The color represents different generations. Blue points stands for small generations and red points mean large generations.



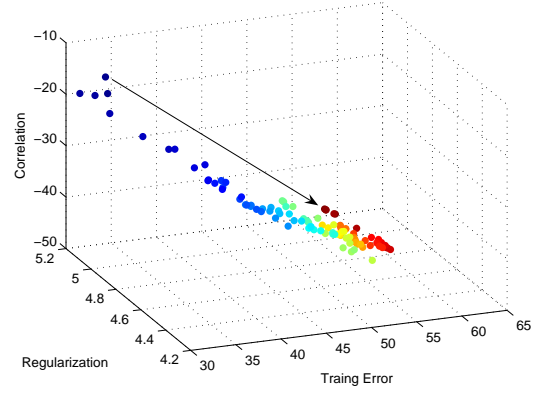
(a) Bumpy



(b) Bumpy Evolving Information in 3D



(c) Relevance



(d) Relevance Evolving Information in 3D

Figure 5.4: Detailed information in multiobjective algorithm for two data sets, bumpy and relevance. In Figure 5.4(a) and 5.4(c), the left-y axis (red line with circles) measures the summation of the mean of three objectives, training error, regularization and correlation in different generations. The right-y axis (blue line with triangles) is the standard deviation of the summation. In Figure 5.4(b) and 5.4(d), the 3D figure records the mean value of these three objectives in different generations. The arrow points from the beginning (Generation = 1) to end (Generation = 100). The color represents different generations. Blue points stands for small generations and red points mean large generations.

difficult to obtain a linear boundary from a RBF function (Gramacy and Lee, 2005), both MRNCL and MNCL produce near-linear boundary.

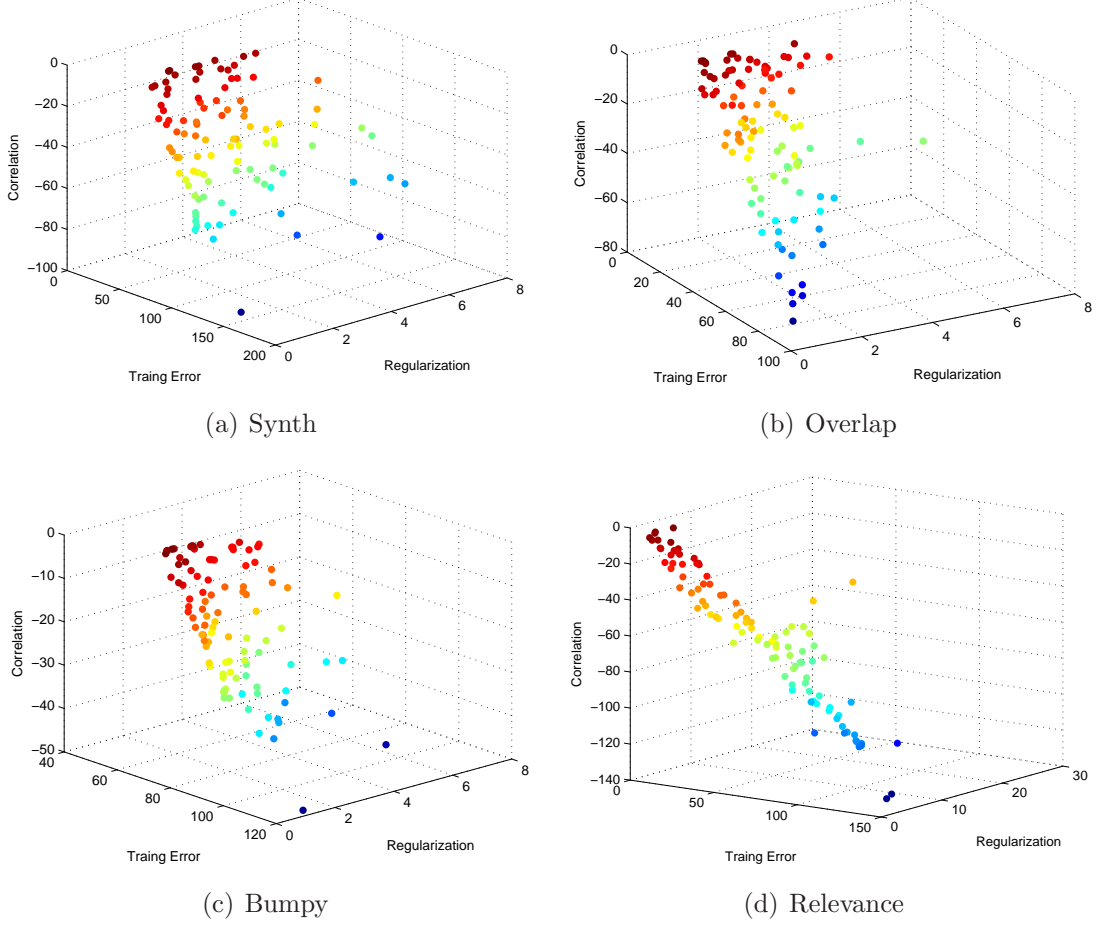


Figure 5.5: Illustration the trade-off among the three objectives: training error, regularization and correlation, in the final population for four synthetic classification data sets. The color represents different correlations. Blue points stands for low correlations and red points mean large correlations.

In other cases, MRNCL gives more accurate results. In the cases of *Synth* and *Bumpy*, MRNCL produces smooth boundary and disregards the outliers in the training points. In the case of *Bumpy*, the noise level is great because of these overlapping points. MNCL does not generalize and produces the twisty boundary. In the case of *Synth*, MNCL concentrates on several outliers and generate twisty

boundaries.

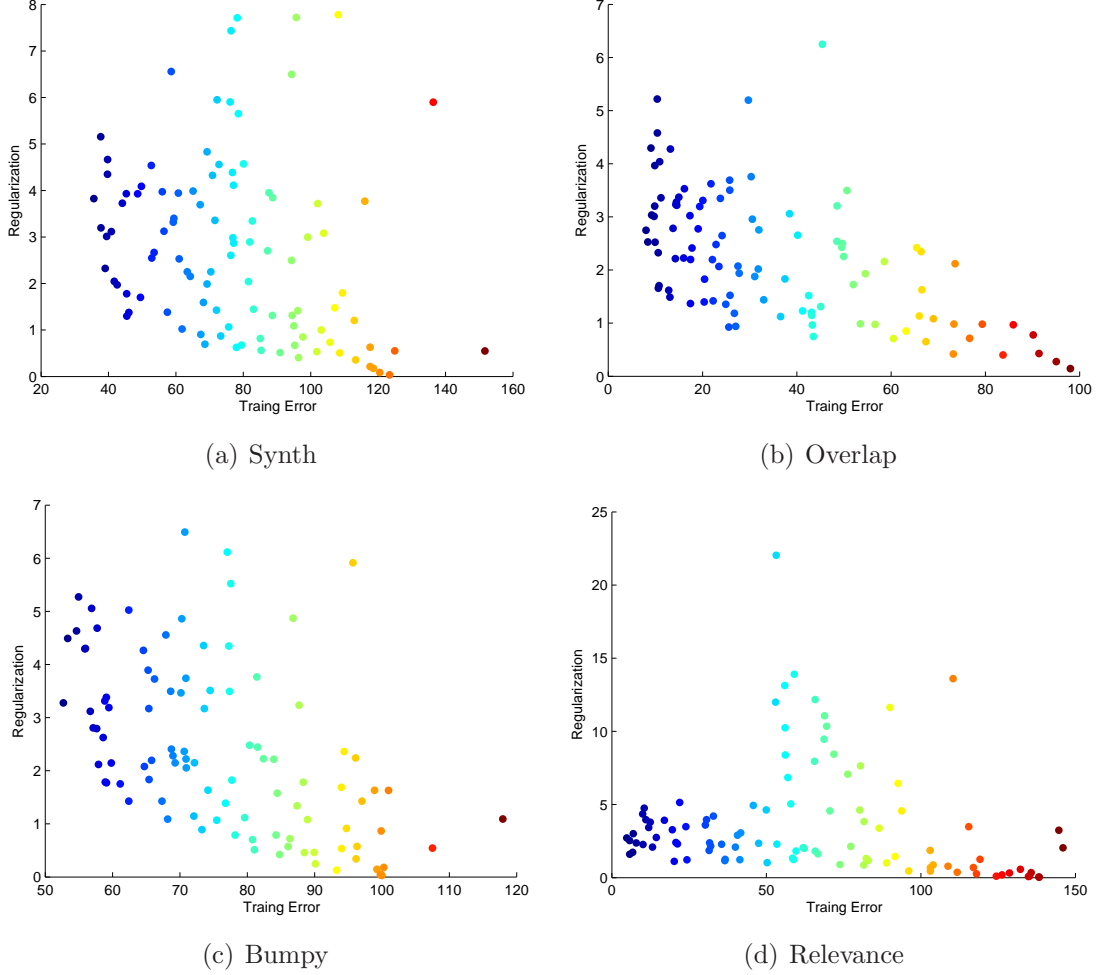


Figure 5.6: 2D view of the trade-off between two objectives: training error and regularization for four synthetic classification data sets. The color represents different training errors. Blue points stands for low training errors and red points mean large training errors.

In order to illustrate the characteristics of MRNCL, in each generation we record the mean value and standard deviation of the three objectives in the population.

Figures 5.3(b), 5.3(d), 5.4(b) and 5.4(d) illustrate the mean value of these three objectives in different generations. The arrow points from the first genera-

tion to the final generation.

According to these figures, MRNCL tries to minimize the three objectives. However, based on the analysis in section 5.2, the empirical training error is negatively correlated with the correlation term. Instead of minimizing the three objectives simultaneously, firstly, MRNCL seeks to find a good balance between the training error and the correlation term, and then MRNCL always minimizes the third objective, the regularization term, in the evolutionary algorithm.

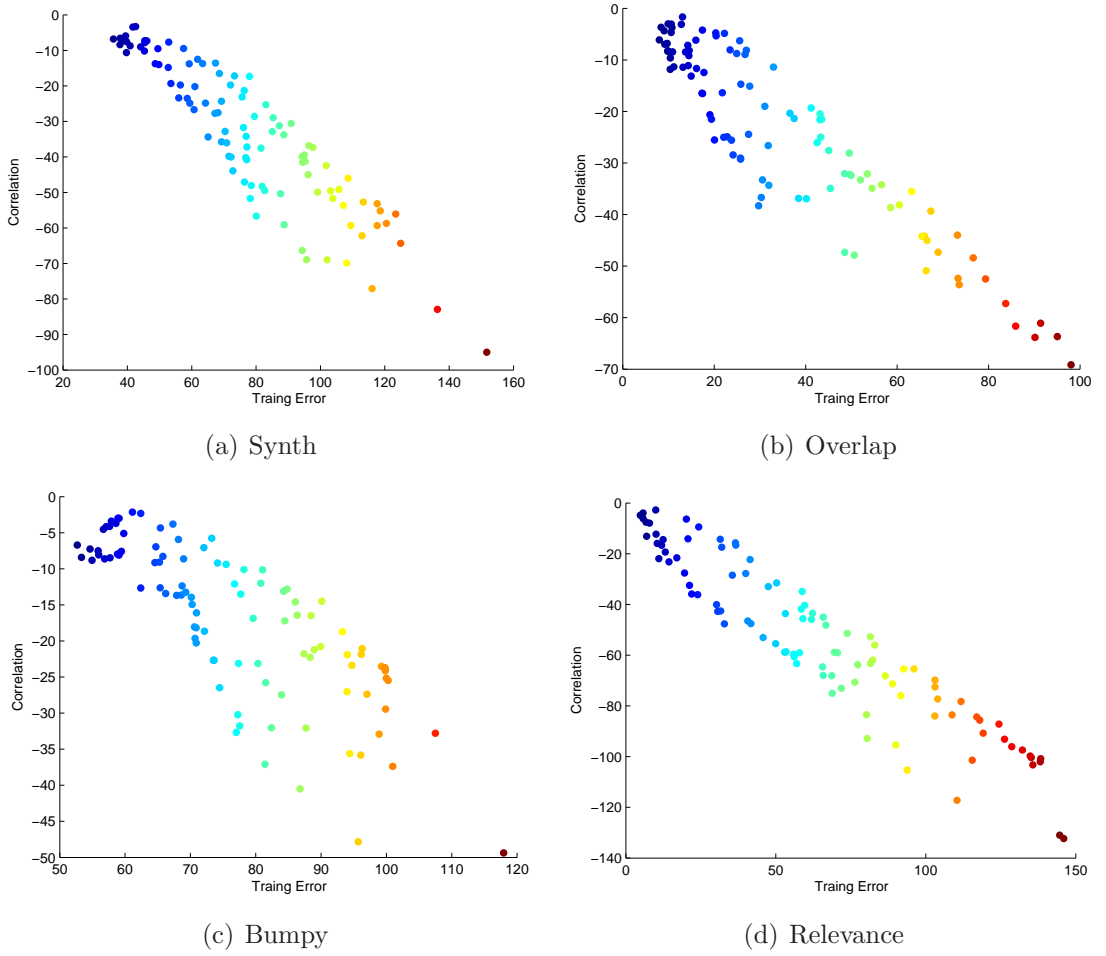


Figure 5.7: 2D view of the trade-off between two objectives: training error and correlation for four synthetic classification data sets. The color represents different training errors. Blue points stands for low training errors and red points mean large training errors.

In Figures 5.3(a), 5.3(c), 5.4(a) and 5.4(c), we show the mean value and standard deviation of the summation of the three objectives in different generations. Although the direct summation of the three objectives into one term is an inaccurate estimation, as we do not consider the weighting coefficients and we use the mean value instead of summation of the three objectives for every individual, the summation does reflect the tendency of MRNCL in the multiobjective algorithm. These figures show that MRNCL does minimize the summation of the three objectives. In these figures, the standard deviation increases from the beginning, indicating that the search space of MRNCL becomes large and the population becomes diverse from generation to generation.

To understand the trade-off among the three objectives for different problems, the 3D view of the three objectives in the final generation is illustrated in Figure 5.5. The negative correlation between the empirical error term and the correlation term has been confirmed in this figure. (The trade-off between correlation and training error is presented 2D in Figure 5.7 as well.) Figure 5.6 shows the trade-off between the empirical error term and the regularization term. The final population distributed a good trade-off between these two objectives for all the datasets except *Relevance*. As *Relevance* is a noise-free data set, most networks concentrate on the training error and correlation but not regularization. This indicates that MRNCL can choose a better trade-off among these multi-objectives for different problems.

In chapter 4, we use the same synthetic classification data and MLP network as the base learners to illustrate the decision boundaries of RNCL and NCL. As MLPs are able to produce more flexible boundaries than RBFs¹ do in MNCL, the decision boundary of NCL is smoother than that of MNCL. The advantages of MRNCL and MNCL include explicit observation of the interaction and trade-off among different objectives and automatic balance of the trade-off among these objectives.

¹A RBF network is a combination of RBF functions and thus its decision boundary is not flexible as that of MLPs.

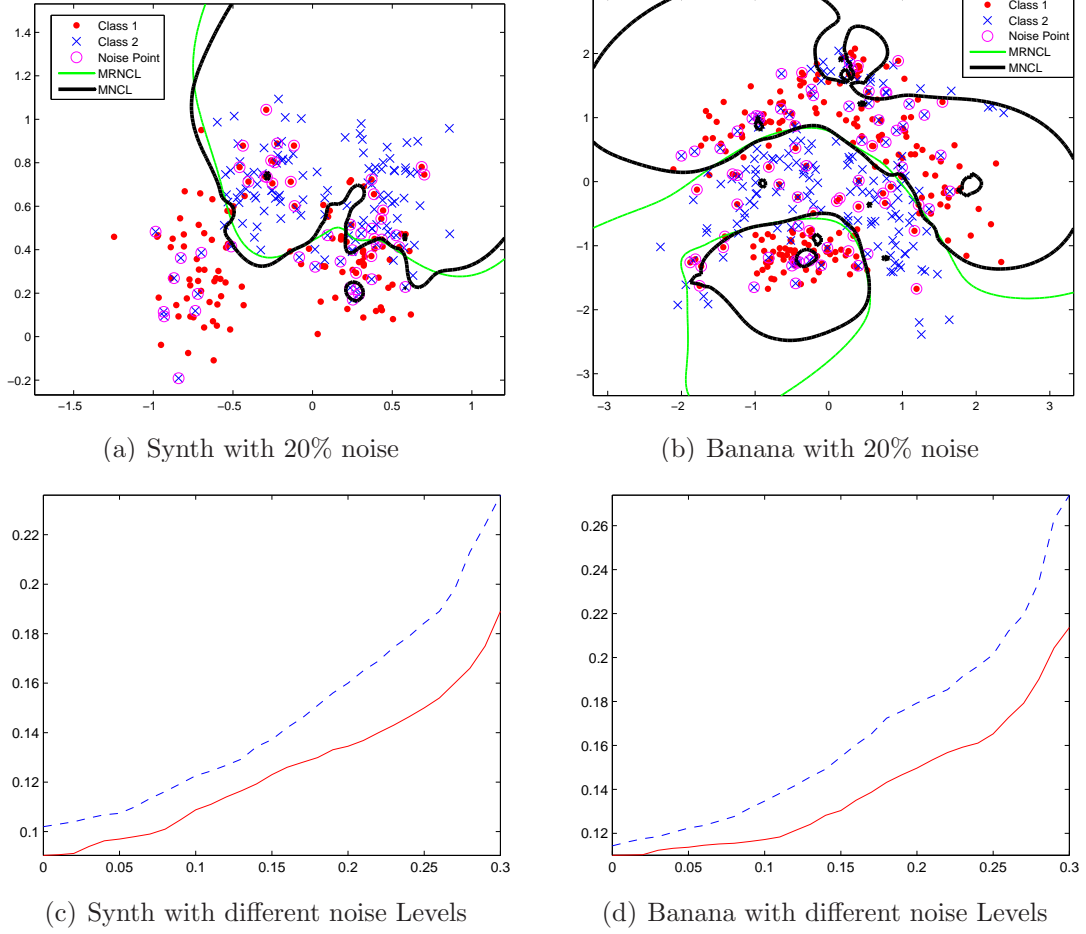


Figure 5.8: Comparison of MRNCL and MNCL on two classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. In Figure 5.8(a) and 5.8(b), the decision boundary in green (thin) and black (thick) are obtained by MRNCL and MNCL, respectively. The randomly-selected noise points are marked with a circle. Figure 5.8(c) and 5.8(d) show classification error of MRNCL (red solid) and MNCL (blue dashed) vs. noise levels on synth and banana data sets. The results are based on 100 runs.

5.3.3 Experimental Results on Noisy Data

In order to evaluate MRNCL and MNCL on data with different noise levels, we conduct two additional experiments. In the experiments, we select two data sets: synth and banana.

To change the noise level, we randomly select different percentages of data points and reverse their labels. We run 100 times and report the average results in Figure 5.8. Figure 5.8(a) and Figure 5.8(b) visualize the decision boundaries of MRNCL and MNCL with 20% noise.

Although the noise level is high, MRNCL produces smooth boundary. MNCL tries to minimize the training error and it does not generalize well. We also plot the curve, Figure 5.8(c) and 5.8(d), of the classification error rate vs. the noise level for these data sets. In these two figures, MRNCL is a little better in the beginning, but as the noise level increases, MRNCL significantly outperforms MNCL.

The results of MRNCL are promising on these classification problems. The regularization term does work in MRNCL and improves its ability against noise, which is especially important in practice since most of the actual data are contaminated by noise. After the analysis with synthetic data sets, the next section presents the results for the real-world benchmark problems.

Compared Figure 5.8 with Figure 4.4 in chapter 4, the multiobjective based algorithms have similar tendency as the gradient descent based algorithms, i.e. as noise level increases, the error rate of these algorithms increases but the difference between MRNCL and MNCL (same as RNCL and NCL) becomes large. From these two figures, multiobjective based algorithms outperform gradient descent based algorithms. The reason is that multiobjective based algorithms consider an additional weighting coefficient of the correlation term.

5.3.4 Benchmark Results

These data sets used in this chapter have been summarized in Table 4.2. The details of these datasets have been presented in chapter 4 and please refer to section 4.4 for more information.

5.3 Numerical Experiments

Table 5.1: Comparison among the six methods on 13 benchmark Data Sets: Single RBF classifier, MRNCL, MNCL, Adaboost, Bagging, and support vector machine. Estimation of generalization error in % on 13 data sets (best method in bold face). The columns P_1 to P_4 show the results of a significance test (95% t-test) between MRNCL and MNCL, Adaboost, Bagging and SVM, respectively. The p value with a star means the test is significant. The performance is based on 100 runs (20 runs for Splice and Image). MRNCL gives the best overall performance.

	RBF	MRNCL	MNCL	P_1	Adaboost	P_2	Bagging	P_3	SVM	P_4
Banana	10.8±0.6	10.7±0.7	11.2±0.7	0.00*	12.3±0.7	0.00*	11.2±0.7	0.03*	11.5±0.7	0.00*
Cancer	27.6±4.7	26.4±4.6	28.2±4.8	0.08	30.4±4.7	0.00*	27.3±4.6	0.23	26.0±4.7	0.33
Diabetics	24.3±1.9	23.2±1.7	25.3±1.9	0.00*	26.5±2.3	0.00*	24.2±1.8	0.03*	23.5±1.7	0.46
German	24.7±2.4	24.2±2.1	26.1±2.3	0.03*	27.5±2.5	0.00*	24.9±2.5	0.15	23.6±2.1	0.08
Heart	17.6±3.3	15.6±3.0	16.2±3.1	0.25	20.3±3.4	0.00*	17.2±3.4	0.00*	16.0±3.3	0.46
Image	3.3±0.6	2.6±0.7	2.6±0.7	0.42	2.7±0.7	0.31	3.0±0.6	0.04*	3.0±0.6	0.05*
Ringnorm	1.7±0.2	1.6±0.2	1.9±0.2	0.00*	1.9±0.3	0.04*	1.6±0.2	0.61	1.7±0.1	0.26
Solar	34.4±2.0	33.1±1.7	33.4±1.5	0.62	35.7±1.8	0.02*	34.1±1.9	0.07	32.4±1.8	0.21
Splice	10.0±0.8	9.9±0.6	10.2±0.5	0.08	10.1±0.5	0.06	10.0±0.5	0.34	10.9±0.7	0.00*
Thyroid	4.5±2.1	4.5±2.1	4.3±2.1	0.43	4.4±2.2	0.62	4.4±2.1	0.58	4.8±2.2	0.13
Titanic	23.3±1.3	22.3±1.1	22.2±1.3	0.12	22.6±1.2	0.08	22.8±1.2	0.04*	22.4±1.0	0.32
Twonorm	2.9±0.3	2.3±0.1	2.4±0.1	0.03*	3.0±0.3	0.00*	2.8±0.2	0.00*	3.0±0.2	0.00*
Waveform	10.7±1.1	10.4±0.6	10.6±0.7	0.13	10.8±0.6	0.04*	10.2±0.5	0.12	9.9±0.4	0.03*

The performance of MRNCL, MNCL, RBF network, Adaboost, Bagging, and SVM over 100 runs (20 runs for Splice and Image) is tabulated in Table 5.1. The performance of RBF network, Adaboost and SVM in each fold is obtained from Rättsch's implementation¹.

According to Table 5.1, MRNCL outperforms all the other methods in 7 out of 13 data sets, comes second in 4 cases and third in the remaining 2. In comparison with MNCL, MRNCL wins 10 times out of 13 and of them 5 wins are statistically significant. In the results, MNCL only performs well in the cases with little noise: Image, Thyroid and Twonorm, which are all synthetic data with little noise (see the lower error rate). The observation validates that MNCL achieves good results only when the noise level is small.

Compared MRNCL with gradient descent based RNCL in chapter 4, MRNCL achieves a little better performance than RNCL.

There are at least three reasons to explain the success of MRNCL.

1. Effective parameters of RBF ensemble, obtained by the evolutionary algorithm, improve the performance of the ensemble. The performance of RBF networks mostly depends on the number of basis functions and the selection of centers and widths in these basis functions. In a RBF network ensemble, better performance is achieved when these individuals cooperate with each other. How to select these parameters is crucial for the ensemble. In most of the existing ensemble algorithms, we have to tune these parameters manually, suffering from the tedious trial-and-error process in practice. However, our algorithm can determine these parameters automatically for different problems given that you specify some parameters for the evolutionary algorithm.
2. The multiobjective algorithm promotes the accuracy, diversity and regularization in the ensemble. The accuracy and diversity are considered as two important factors in ensemble algorithms. Our analysis reveals that besides these two factors, regularization is another important factor for ensemble performance. The regularization term controls the complexity of ensemble

¹<http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm>

and improves the performance of ensemble against noise. The existing ensemble algorithms either focus on accuracy, e.g. Adaboost and/or diversity, e.g. Bagging and NCL. In order to take all these terms into consideration, our strategy adopts a multiobjective algorithm to generate accurate, diverse and regularized ensembles.

3. A better trade-off of the three objectives can be determined automatically. Different problems require different trade-offs among the three objectives. The usual way, which assigns combination coefficients to the three terms and optimizes these coefficients by grid search, is computationally expensive. Our algorithm uses a multiobjective algorithm to choose a better trade-off for the specific problem. There is no need to weigh objectives by selecting the coefficients.

5.3.5 Computational Complexity and Running Time

Table 5.2: Running Time of MRNCL and MNCL on 13 Data Sets in seconds. Results are averaged over 100 runs.

Time(s)	Banana	Cancer	Diabetics	Solar	German	Heart	
MNCL	80.6	28.6	62.7	84.6	126.1	26.3	
MRNCL	81.3	29.4	68.6	83.0	121.5	26.7	
Time(s)	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
MNCL	220.0	192.6	283.6	19.1	46.9	186.8	132.1
MRNCL	236.2	183.7	288.7	19.4	45.6	193.8	145.7

Based on the algorithm in Figure 5.1, the major running time of MRNCL is consumed in the training of RBF networks. In the initialization step, we need to train M RBF network. In each generation, we need to train $2C+u$ RBF networks, where C is the number of crossover in one generation and u is the number of mutation in one generation. In total, we need to train $M + (2C + u)G$ RBF networks in MRNCL, where G indicates the number of generations. Given the centers and widths of a RBF network, the training of RBF is a linear optimization

problem, which can be evaluated quickly. Table 5.2 shows the average running time of MRNCL and MNCL over 100 runs. The computational environment is Windows XP with Intel Core 2 Duo 1.66G CPU and 2G RAM. The algorithms are implemented in Matlab and *C* language, where *C* language is used for the implementation of RBF network training algorithm.

Generally speaking, MRNCL needs more computational resource than RNCL. As MRNCL uses a large ensemble (100 RBF networks) than RNCL (25 MLP networks in chapter 4) and MRNCL is implemented by an evolutionary multiobjective algorithm, the running time of MRNCL is longer than that of RNCL in most cases. We are not surprised that RNCL consumes more time than MRNCL for some large datasets, such as German, Image, Splice in our experiments. It is because the training of RBF network in MRNCL is much faster than training of MLP network with the same number of hidden nodes in RNCL.

5.4 Summary

The major contribution of the chapter is to formulate the regularized negative correlation learning (RNCL) as a multiobjective optimization problem. The resulting algorithm MRNCL can effectively search the best trade-off among these three terms. To effectively evolve these networks, the crossover and mutation operators are defined to vary the structure of RBF networks. The nondominated sorting algorithm with fitness sharing and linear rank-based fitness assignment are employed to promote diversity in MRNCL.

The numerical results and visualization on both the synthetic data sets and the benchmark datasets have demonstrated that MRNCL achieves better performance than MNCL, especially when the noise is non-trivial in data sets. The comparison with other state-of-the-art algorithms also demonstrate the superiority of MRNCL.

Compared with RNCL by gradient descent with Bayesian inference in chapter 4, MRNCL often achieves a little better performance by considering an additional weighting coefficient of the correlation term. The potential advantages of the multiobjective approach include: It enables us to observe the interaction and trade-off among different objectives; and it enables us to add or remove an objective easily

without changing the overall algorithm. However, the better performance comes with the price, more computational time to train MRNCL.

Choosing one appropriate implementation from RNCL and MRNCL depends on the application and users' specification. If the users would like to observe the interaction and trade-off among different objectives and easily modify the code without changing the overall algorithm, multiobjective MRNCL is appropriate. If they pay more attention to the computational resource and prefer the explicit combination coefficients, gradient descent based RNCL with Bayesian inference is a better choice.

Chapter 6

Predictive Ensemble Pruning by Expectation Propagation

The existing ensemble learning algorithms often generate unnecessarily large ensembles, which consume extra computational resource. As these algorithms do not *explicitly* manage the trade-off among diversity, regularization and accuracy in the ensemble, these large ensembles will degrade the performance when the trade-off is unbalanced. For example, large boosting ensembles are prone to overfitting by paying more attention to accuracy but ignoring regularization. Therefore, proper ensemble pruning algorithms can improve the performance and efficiency of ensembles. In this chapter, we investigate ensemble pruning as one way to better balance diversity, regularization and accuracy in the ensemble.

This chapter extends our previous work (Chen et al., 2006) and proposes a probabilistic ensemble pruning algorithm, aiming to find a good subset of ensemble members to produce a small ensemble, which saves the computational resource and performs as well as, or better than, the non-pruned ensemble.

The rest of this chapter is organized as follows. After the introduction in section 6.1, section 6.2 describes the sparseness-induction and truncated prior. Section 6.3 presents the pruning algorithm for regression and classification problems, respectively, followed by experimental results and analysis in section 6.4. Finally, section 6.5 summarizes the chapter.

6.1 Introduction

The existing ensemble learning algorithms often generate unnecessarily large ensembles. These large ensembles are memory demanding. Obtaining a prediction for a *fresh* data point can be done expensively in large ensembles. Although these extra costs may seem to be negligible when dealing with small data sets, they may become serious when the ensemble method is applied to a large scale data set. In addition, it is not always true that the larger the size of an ensemble, the better it is. Some theoretical and empirical evidences have shown that small ensembles can be better than large ensembles (Breiman, 1996a; Yao and Liu, 1998; Zhou et al., 2002).

For example, the boosting ensembles, Adaboost (Schapire, 1999) and Arcing (Breiman, 1998, 1999), pay more attention to those training samples that are misclassified by former classifiers in the training of next classifier and finally reduce the training error to zero. In this way, the former classifiers, with large training error and large diversity, may under-fit the data, while the latter classifiers, with low training error and weak regularization, are prone to overfitting the noise in the training data. The trade-off among diversity, regularization and accuracy in the ensemble is unbalanced and thus Boosting ensembles are prone to overfitting (Dietterich, 2000; Opitz and Maclin, 1999). In these circumstances, it is necessary to prune some individuals to achieve better generalization.

The evolutionary ensemble learning algorithms often generate a number of learners in the population. Some are good at accuracy; some have larger diversity and others pay more attention to regularization. In this setting, we had better select a subset of learners to produce an effective and efficient ensemble by balancing diversity, regularization and accuracy.

In the last decades, several ensemble pruning algorithms have been proposed, such as Kappa pruning (Margineantu and Dietterich, 1997), concurrency pruning (Banfield et al., 2005). However, these algorithms all resort to greedy search, which is without either theoretical or empirical quality guarantees.

Yao et al. (Yao and Liu, 1998) adopted a global optimization approach, genetic algorithm (GA), to weigh the ensemble members by constraining the weighs to be positive. Zhou et al. (Zhou et al., 2002) proved that small ensembles

can be better than large ensembles. A similar genetic algorithm approach can be found in (Kim et al., 2002). However, these GA based algorithms try to obtain the optimal combination weights by minimizing the training error and in this way these algorithms become sensitive to noise.

Motivated by the above reasons, this thesis investigates ensemble pruning, which reduces the size of ensembles without hurting the generalization error, as one way to balance diversity, regularization and accuracy and the thesis proposes one probabilistic ensemble pruning algorithm. The algorithm treats ensemble pruning as a weight-based optimization, aiming to improve the generalization performance of the ensemble by tuning the weight of each ensemble member. By introducing a sparseness-inducing prior for each combination weight, many of the posteriors of weights are sharply distributed around zero, leading to pruning unimportant learning machines. As negative combination weights are unreliable and not intuitive (Breiman, 1996b; Hashem, 1993; Hastie et al., 2001; Leblanc and Tibshirani, 1996), we follow this constraint and employ a left-truncated prior to prevent negative values in the combination weights.

By incorporating the truncated prior, the normalization integral in Bayesian inference becomes intractable. In our previous work (Chen et al., 2006), Expectation-Maximization (EM) algorithm was used to infer the combination weights and our algorithm showed a good performance in both generalization and pruned ensemble size. However, EM algorithm is sensitive to the initialization and we need to rerun the EM algorithm from several initializations and select the best one based on the cross validation error.

The chapter extends the previous work and employs the deterministic expectation propagation (EP) (Minka, 2001) to approximate the posterior of weights. Conveniently, an estimate of the leave-one-out (LOO) error can be obtained in the training of EP. The LOO error is used together with the Bayesian model evidence for model selection in this algorithm.

6.2 Sparseness-induction and Truncated Prior

In the weight-based ensemble pruning algorithm, the ensemble is formulated as a linear combination of the individual learners:

$$f_{ens}(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^M w_i f_i(\mathbf{x}) = \mathbf{w}^T \mathbf{F}(\mathbf{x}), \quad (6.1)$$

where $\mathbf{w} = (w_1, \dots, w_M)^T$ is the weight vector of the ensemble, and $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_M(\mathbf{x}))^T$ is the vector of individual learners. The pruning algorithm is to adjust the parameters $\mathbf{w} = (w_1, \dots, w_M)^T$, setting many w_i to zeros, but not degrade the generalization performance of the ensemble. As negative weight vectors are neither intuitive nor reliable (Breiman, 1996b; Hashem, 1993; Hastie et al., 2001; Leblanc and Tibshirani, 1996), this chapter constraints the weight vector to be non-negative.

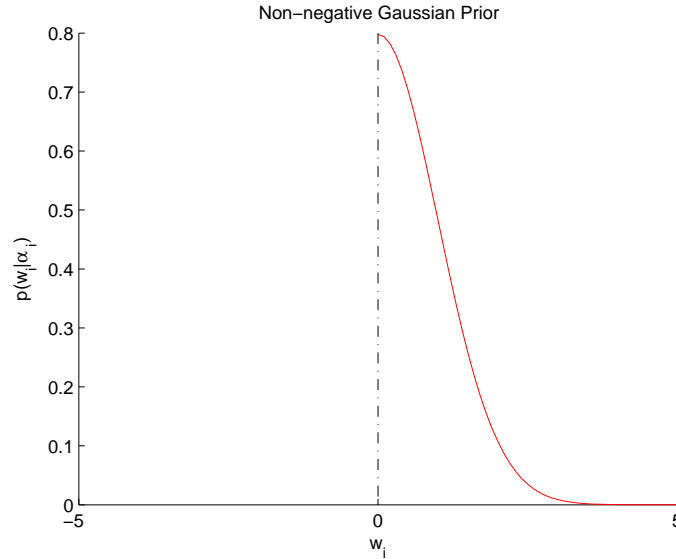


Figure 6.1: The truncated Gaussian Prior

To encourage the sparsity of weight vector \mathbf{w} and to satisfy the non-negative restriction, a left-truncated Gaussian prior is introduced for each weight w_i :

$$p(\mathbf{w}|\alpha) = \prod_{i=1}^M p(w_i|\alpha_i) = \prod_{i=1}^M N_t(w_i|0, \alpha_i^{-1}), \quad (6.2)$$

where $\alpha = (\alpha_1, \dots, \alpha_M)^T$ is the inverse variance of weight w_i and $N_t(w_i|0, \alpha_i^{-1})$ is a left-truncated Gaussian distribution. This is formalized in equation (6.3) and illustrated in Figure 6.1.

$$p(w_i|\alpha_i) = \begin{cases} 2N(w_i|0, \alpha_i^{-1}) & \text{if } w_i \geq 0 \\ 0 & \text{if } w_i < 0 \end{cases} . \quad (6.3)$$

6.3 Predictive Ensemble Pruning by Expectation Propagation

In this section, we describe our ensemble pruning algorithm and present the detailed expectation propagation procedures.

6.3.1 Expectation Propagation

Expectation propagation (Minka, 2001) is a deterministic algorithm for approximating Bayesian inference that extends assumed-density filtering (ADF) to incorporate iterative refinement of the approximations. Expectation propagation assumes that the joint distribution $p(D, \mathbf{w})$, where the data $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$ has been observed and \mathbf{w} is a parameter vector, can be factored into some simple terms:

$$p(D, \mathbf{w}) = \prod_{i=1}^M t_i \prod_{n=1}^N g_n = \prod_{i=1}^M p(w_i) \prod_{n=1}^N p(y_n|\mathbf{w}, x_n), \quad (6.4)$$

where $p(w_i)$ is the prior distribution of w_i , M is the number of weights and $\prod_{n=1}^N p(y_n|\mathbf{w}, x_n)$ is the likelihood.

Expectation propagation adopts a family of exponential functions to approximate each term by minimizing the KL-divergence (Kullback and Leibler, 1951) between the exact term and the approximation term, and then combines these approximations analytically to obtain a Gaussian posterior $q(\mathbf{w})$ on \mathbf{w} .

Expectation propagation will adopt a family of Gaussian function $\tilde{t}_i(w) = s_i \exp(-\frac{1}{2v_i}(w - m_i)^T(w - m_i))$ with different mean m_i and variances v_i to approximate each term t_i by minimizing the KL-divergence Kullback and Leibler (1951) between the exact term t_i and the approximation term \tilde{t}_i , then combine these approximations analytically to get a Gaussian posterior $q(\mathbf{w})$ on \mathbf{w} .

6.3 Predictive Ensemble Pruning by Expectation Propagation

The procedures of expectation propagation can be decomposed in the following steps:

1. Initialize the approximating term \tilde{t}_i : the prior: $t_0 = p(\mathbf{w})$ and $t_i = 1$; the posterior of w can be obtained from the product of \tilde{t}_i .

$$q(\mathbf{w}) = \frac{\Pi_i t_i(\mathbf{w})}{\int \Pi_i t_i(\mathbf{w}) d\mathbf{w}} = N(m_{\mathbf{w}}, V_{\mathbf{w}}). \quad (6.5)$$

2. Remove the approximating term \tilde{t}_i from the posterior $q(\mathbf{w})$ to get the leave-one-out posterior $q^{\setminus i}(\mathbf{w})$ which is also Gaussian.

$$q^{\setminus i}(\mathbf{w}) \propto q(\mathbf{w}) / \tilde{t}_i. \quad (6.6)$$

3. Choose the "new" posterior $q^{new}(\mathbf{w})$ to minimizing the KL-divergence Kullback and Leibler (1951) between the posterior $q^{new}(\mathbf{w})$ and the product of $q^{\setminus i}(\mathbf{w})$ with the "exact" term t_i : $q^{\setminus i}(\mathbf{w}) t_i(\mathbf{w})$.
4. Update the approximating term $\tilde{t}_i = Z_i \frac{q^{new}(\mathbf{w})}{q^{\setminus i}(\mathbf{w})}$, where Z_i is the normalizing factor.
5. As each term \tilde{t}_i approximates the exact term t_i very well, the approximating posterior of weight vector \mathbf{w} can be obtained as:

$$q(\mathbf{w}) = \frac{\Pi_i t_i(\mathbf{w})}{\int \Pi_i t_i(\mathbf{w}) d\mathbf{w}}. \quad (6.7)$$

6.3.2 Expectation Propagation for Regression Ensembles

In the regression ensemble model, we train M individual estimators using the training set $\{\mathbf{x}_n, y_n\}_{n=1}^N$, where y_n is a scalar. We assume the ensemble output is corrupted by an i.i.d. additive Gaussian noise $\epsilon_n = N(0, \sigma^2)$ with mean zero and variance σ^2 :

$$y_n = \mathbf{w}^T \mathbf{F}(\mathbf{x}_n) + \epsilon_n. \quad (6.8)$$

According to equation (6.8), the true value y_n is distributed as a Gaussian distribution with mean $\mathbf{w}^T \mathbf{F}(\mathbf{x}_n)$ and variance σ^2 . Based on the assumption of independence of training points, the likelihood can be expressed as:

$$p(\mathbf{y} | \mathbf{w}, \mathbf{x}_n, \sigma^2) = (2\pi\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{w}^T \mathbf{F}\|^2\right\}. \quad (6.9)$$

6.3 Predictive Ensemble Pruning by Expectation Propagation

where $\mathbf{y} = (y_1, \dots, y_N)^T$, $\mathbf{w} = (w_1, \dots, w_M)^T$ and $\mathbf{F} = (\mathbf{F}(\mathbf{x}_1), \dots, \mathbf{F}(\mathbf{x}_N))$ is a $M \times N$ matrix, where $\mathbf{F}(\mathbf{x}_n) = (f_1(\mathbf{x}_n), \dots, f_M(\mathbf{x}_n))^T$.

The posterior of the weight vector \mathbf{w} is denoted by

$$\begin{aligned} p(\mathbf{w}|\mathbf{x}, \mathbf{y}, \alpha) &\propto p(\mathbf{w}|\alpha) \prod_{n=1}^N p(y_n|\mathbf{x}_n, \mathbf{w}) \\ &= \prod_{i=1}^M 2N(w_i|0, \alpha_i^{-1}) \prod_{i=1}^M \Theta(w_i) \prod_{n=1}^N p(y_n|\mathbf{x}_n, \mathbf{w}), \end{aligned} \quad (6.10)$$

where $\Theta(w_i) = \Theta(\mathbf{w}^T \mathbf{e}_i) = \begin{cases} 1 & \text{if } w_i > 0 \\ 0 & \text{if } w_i \leq 0 \end{cases}$ prevents the weights from negative values and $\mathbf{e}_i = (0, \dots, 1, 0, \dots, 0)$ is used to obtain the weight w_i ($w_i = \mathbf{w}^T \mathbf{e}_i$ and $\Theta(w_i) = \Theta(\mathbf{w}^T \mathbf{e}_i)$).

According to equation (6.10), we only need to approximate the terms $\Theta(w_i)$ in calculating the posterior. Denote the exact terms $\Theta(w_i)$ by $t_i(\mathbf{w})$, and the approximate terms by $\tilde{t}_i(\mathbf{w}) = s_i \exp(-\frac{1}{2v_i}(\mathbf{w}^T \mathbf{e}_i - m_i)^2)$ which are parameterized by (m_i, v_i, s_i) . Since the likelihood terms $p(y_n|\mathbf{x}_n, \mathbf{w})$ are Gaussians, we represent these terms $p(y_n|\mathbf{x}_n, \mathbf{w})$ by $\tilde{g}_n(\mathbf{w}) = s_n \exp(-\frac{1}{2v_n}(\mathbf{w}^T \mathbf{F}(\mathbf{x}_n) - y_n)^2)$ to facilitate EP training, where $v_n = \sigma^2$ and $m_n = y_n$. After approximating every term as an exponential family distribution, the resulting distribution will be Gaussian: $p(\mathbf{w}|\mathbf{x}, \mathbf{y}, \alpha) \approx q(\mathbf{w}) = N(\mathbf{m}_w, \mathbf{V}_w)$. The EP algorithm for regression ensembles is described in the following (to simplify notations, \mathbf{F}_n stands for $\mathbf{F}(\mathbf{x}_n)$).

1. Initialization the prior term: $q(\mathbf{w}) = N(\mathbf{w}|0, \alpha^{-1})$ and initialize the approximating terms to 1, $\tilde{t}_i = 1$: $m_i = 0$, $v_i = \infty$ and $s_i = 1$.
2. Until both \tilde{g}_n and \tilde{t}_i converge: Loop $n = 1, \dots, N$, and $i = 1, \dots, M$;
 - (a) Remove the approximation term \tilde{g}_n from the posterior $q(\mathbf{w})$ to obtain the *leave-one-out* posterior $q^{\setminus n}(\mathbf{w})$: $N(\mathbf{m}_w^{\setminus n}, \mathbf{V}_w^{\setminus n})$.

$$\begin{aligned} & q^{\setminus n}(\mathbf{w}) \\ & \propto \frac{q(\mathbf{w})}{\tilde{g}_n} \\ & = \frac{\exp(-\frac{1}{2}(\mathbf{w} - \mathbf{m}_w)^T \mathbf{V}_w^{-1} (\mathbf{w} - \mathbf{m}_w))}{(2\pi)^{M/2} |\mathbf{V}_w|^{1/2} s_n \exp(-\frac{1}{2v_n}(\mathbf{w}^T \mathbf{F}(\mathbf{x}_n) - y_n)^2)} \\ & = \frac{1}{(2\pi)^{M/2} |\mathbf{V}_w|^{1/2} s_n} \exp \left\{ -\frac{1}{2} \left[\begin{aligned} & \mathbf{w}^T \left(\mathbf{V}_w^{-1} - \frac{1}{v_n} \mathbf{F}_n \mathbf{F}_n^T \right) \mathbf{w} - 2\mathbf{w}^T \right. \\ & \left. \left(\mathbf{V}_w^{-1} \mathbf{m}_w - \frac{1}{v_n} y_n \mathbf{F}_n \right) + \mathbf{m}_w^T \mathbf{V}_w^{-1} \mathbf{m}_w - \frac{1}{v_n} y_n^2 \right] \right\} \end{aligned} \right\} \end{aligned} \quad (6.11)$$

6.3 Predictive Ensemble Pruning by Expectation Propagation

Based on equation (6.11), the variance $\mathbf{V}_w^{\setminus n}$ of $q^{\setminus n}(\mathbf{w})$ can be obtained as follows:

$$\mathbf{V}_w^{\setminus n} = \left(\mathbf{V}_w^{-1} - \frac{1}{v_n} \mathbf{F}_n \mathbf{F}_n^T \right)^{-1}. \quad (6.12)$$

According to the Woodbury matrix identity (Woodbury, 1950),

$$(\mathbf{A} - \mathbf{B} \mathbf{D}^{-1} \mathbf{C})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1} \mathbf{B} (\mathbf{D} - \mathbf{C} \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{C} \mathbf{A}^{-1}. \quad (6.13)$$

$\mathbf{V}_w^{\setminus n}$ can be reformulated as:

$$\mathbf{V}_w^{\setminus n} = \mathbf{V}_w + \frac{(\mathbf{V}_w \mathbf{F}_n)(\mathbf{V}_w \mathbf{F}_n)^T}{v_n - \mathbf{F}_n^T \mathbf{V}_w \mathbf{F}_n}, \quad (6.14)$$

Based on equation (6.11), the mean $\mathbf{m}_w^{\setminus n}$ of $q^{\setminus n}(\mathbf{w})$ can be obtained as follows:

$$\mathbf{m}_w^{\setminus n} = \mathbf{V}_w^{\setminus n} \left(\mathbf{V}_w^{-1} \mathbf{m}_w - \frac{1}{v_n} y_n \mathbf{F}_n \right), \quad (6.15)$$

Since $\mathbf{V}_w^{-1} = (\mathbf{V}_w^{\setminus n})^{-1} + \frac{1}{v_n} \mathbf{F}_n \mathbf{F}_n^T$ by equation (6.12), $\mathbf{m}_w^{\setminus n}$ can be written as

$$\mathbf{m}_w^{\setminus n} = \mathbf{m}_w + (\mathbf{V}_w^{\setminus n} \mathbf{F}_n) v_n^{-1} (\mathbf{F}_n^T \mathbf{m}_w - m_n). \quad (6.16)$$

- (b) Combine $q^{\setminus n}(\mathbf{w})$ and the exact term $g_n(\mathbf{w})$ to get the new posterior $q(\mathbf{w})$.

$$q(\mathbf{w}) = \frac{g_n(\mathbf{w}) q^{\setminus n}(\mathbf{w})}{\int g_n(\mathbf{w}) q^{\setminus n}(\mathbf{w}) d\mathbf{w}}. \quad (6.17)$$

Since $q^{\setminus n}(\mathbf{w}) = N(\mathbf{w} | \mathbf{m}_w^{\setminus n}, \mathbf{V}_w^{\setminus n})$ and $g_n(\mathbf{w}) = s_n \exp(-\frac{1}{2v_n} (\mathbf{w}^T \mathbf{F}_n \mathbf{x}_n - y_n)^2)$, $q(\mathbf{w})$ is a Gaussian with the mean \mathbf{m}_w and variance \mathbf{V}_w .

$$\mathbf{V}_w = (v_n^{-1} \mathbf{F}_n \mathbf{F}_n^T + (\mathbf{V}_w^{\setminus n})^{-1})^{-1}, \quad \mathbf{m}_w = \mathbf{V}_w (v_n^{-1} \mathbf{F}_n y_n + (\mathbf{V}_w^{\setminus n})^{-1} \mathbf{m}_w^{\setminus n}). \quad (6.18)$$

- (c) Update the approximation term $\tilde{g}_n = Z_n \frac{q(\mathbf{w})}{q^{\setminus n}(\mathbf{w})}$. Since $q(\mathbf{w})$ and $q^{\setminus n}(\mathbf{w})$ are both Gaussians, \tilde{g}_n is a Gaussian (Minka, 2001).

$$\tilde{g}_n = Z_n \frac{q(\mathbf{w})}{q^{\setminus n}(\mathbf{w})} = \frac{Z_n}{N(\mathbf{m}_n; \mathbf{m}_w^{\setminus n}, \mathbf{V}_n + \mathbf{V}_w^{\setminus n})} N(\mathbf{w}; \mathbf{m}_n, \mathbf{V}_n), \quad (6.19)$$

6.3 Predictive Ensemble Pruning by Expectation Propagation

where

$$\mathbf{V}_n = (\mathbf{V}_{\mathbf{w}}^{-1} - (\mathbf{V}_{\mathbf{w}}^{\setminus n})^{-1})^{-1}, \quad (6.20)$$

$$\mathbf{m}_n = \mathbf{V}_n \mathbf{V}_{\mathbf{w}}^{-1} \mathbf{m}_{\mathbf{w}} - \mathbf{V}_n (\mathbf{V}_{\mathbf{w}}^{\setminus n})^{-1} \mathbf{m}_{\mathbf{w}}^{\setminus n}, \quad (6.21)$$

$$s_n = \frac{Z_n}{(2\pi)^{M/2} |\mathbf{V}_n|^{1/2} N(\mathbf{m}_n; \mathbf{m}_{\mathbf{w}}^{\setminus n}, \mathbf{V}_n + \mathbf{V}_{\mathbf{w}}^{\setminus n})}. \quad (6.22)$$

Based on equations (6.20) and (6.18), \mathbf{V}_n can be obtained as:

$$\mathbf{V}_n = (\mathbf{V}_{\mathbf{w}}^{-1} - (\mathbf{V}_{\mathbf{w}}^{\setminus n})^{-1})^{-1} = (v_n^{-1} \mathbf{F}_n \mathbf{F}_n^T)^{-1}. \quad (6.23)$$

The matrix $v_n^{-1} \mathbf{F}_n \mathbf{F}_n^T$ has one nonzero eigenvalue, which means \mathbf{V}_n will have one finite eigenvalue (Minka, 2001) (page 46). This special structure allows us to represent \mathbf{V}_n with a scalar, v_n

$$\mathbf{F}_n^T \mathbf{V}_n \mathbf{F}_n = v_n^{new}. \quad (6.24)$$

$v_n^{new} = v_n = \sigma^2$ will not change and $m_n = y_n$.

$$v_n = \sigma^2, \quad m_n = y_n. \quad (6.25)$$

- (d) Remove the approximation term \tilde{t}_i from the posterior $q(\mathbf{w})$ to obtain the leave-one-out posterior $q^{\setminus i}(\mathbf{w})$: $N(\mathbf{m}_{\mathbf{w}}^{\setminus i}, \mathbf{V}_{\mathbf{w}}^{\setminus i})$. Refer to the equations (6.14) and (6.16).
- (e) Combine $q^{\setminus i}(\mathbf{w})$ and the exact term $t_i(\mathbf{w})$ to get $\hat{p}(\mathbf{w}) = \frac{q^{\setminus i}(\mathbf{w}) t_i(\mathbf{w})}{\int q^{\setminus i}(\mathbf{w}) t_i(\mathbf{w}) d\mathbf{w}}$ and minimize the KL-divergence $KL(\hat{p}(\mathbf{w}) || q(\mathbf{w}))$ between $\hat{p}(\mathbf{w})$ and new posterior $q(\mathbf{w})$ subject to the constrain that $q(\mathbf{w})$ is a Gaussian distribution. Zeroing the gradient with respect to $m_{\mathbf{w}}^{\setminus i}$ and $V_{\mathbf{w}}^{\setminus i}$ gives the conditions (Minka, 2001),

$$E_{q(\mathbf{w})}[\mathbf{w}] = E_{\hat{p}(\mathbf{w})}[\mathbf{w}], \quad (6.26)$$

$$E_{q(\mathbf{w})}[\mathbf{w}^T \mathbf{w}] = E_{\hat{p}(\mathbf{w})}[\mathbf{w}^T \mathbf{w}] \quad (6.27)$$

This is the reason why the algorithm is named as expectation propagation.

$$\begin{aligned} Z_i &= \int_{\mathbf{w}} q^{\setminus i}(\mathbf{w}) t_i(\mathbf{w}) d\mathbf{w} = \int_{\mathbf{w}} q^{\setminus i}(\mathbf{w}) \Theta(\mathbf{w}^T \mathbf{e}_i) d\mathbf{w} \\ &= \Phi(z_i) = \Phi \left(\frac{(\mathbf{m}_{\mathbf{w}}^{\setminus i})^T \mathbf{e}_i}{\sqrt{\mathbf{e}_i^T \mathbf{V}_{\mathbf{w}}^{\setminus i} \mathbf{e}_i}} \right). \end{aligned} \quad (6.28)$$

6.3 Predictive Ensemble Pruning by Expectation Propagation

and

$$\frac{\partial \log Z_i}{\partial \mathbf{m}_w^{\setminus i}} = \frac{N(z_i)}{\Phi(z_i)} \frac{\mathbf{e}_i}{\sqrt{\mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i}} = \mathbf{g}_i \quad (6.29)$$

$$\frac{\partial \log Z_i}{\partial \mathbf{V}_w^{\setminus i}} = -\frac{1}{2} \rho_i \frac{(\mathbf{m}_w^{\setminus i})^T \mathbf{e}_i}{\mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i} \mathbf{e}_i \mathbf{e}_i^T = \mathbf{G}_i, \quad (6.30)$$

where

$$\rho_i = \frac{N(z_i)}{\Phi(z_i)} \frac{1}{\sqrt{\mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i}}. \quad (6.31)$$

According to the theory of expectation propagation (Minka, 2001),

$$\mathbf{m}_w = \mathbf{m}_w^{\setminus i} + \mathbf{V}_w^{\setminus i} \frac{\partial \log Z_i}{\partial \mathbf{m}_w^{\setminus i}} = \mathbf{m}_w^{\setminus i} + \mathbf{V}_w^{\setminus i} \rho_i \mathbf{e}_i, \quad (6.32)$$

$$\mathbf{V}_w = \mathbf{V}_w^{\setminus i} + \mathbf{V}_w^{\setminus i} (\mathbf{g}_i \mathbf{g}_i^T - 2\mathbf{G}_i) \mathbf{V}_w^{\setminus i} = \mathbf{V}_w^{\setminus i} + (\mathbf{V}_w^{\setminus i} \mathbf{e}_i) \left(\frac{\rho_i \mathbf{e}_i^T \mathbf{m}_w}{\mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i} \right) (\mathbf{V}_w^{\setminus i} \mathbf{e}_i)^T, \quad (6.33)$$

- (f) Update the approximation term $\tilde{t}_i = Z_i \frac{q(\mathbf{w})}{q^{\setminus i}(\mathbf{w})}$. Based on equations (6.20), (6.33) and the Woodbury formula. \mathbf{V}_i can be obtained

$$\mathbf{V}_i = [\mathbf{V}_w^{-1} - (\mathbf{V}_w^{\setminus i})^{-1}]^{-1} = \left(\frac{\rho_i \mathbf{m}_w^T \mathbf{e}_i}{\mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i} \mathbf{e}_i \mathbf{e}_i^T \right)^{-1} - \mathbf{V}_w^{\setminus i}. \quad (6.34)$$

The matrix $\frac{\rho_i \mathbf{m}_w^T \mathbf{e}_i}{\mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i} \mathbf{e}_i \mathbf{e}_i^T$ has one nonzero eigenvalue, which means \mathbf{V}_i will have one finite eigenvalue (Minka, 2001) (page 46). This special structure allows us to represent \mathbf{V}_i with a scalar, v_i

$$v_i = \mathbf{e}_i^T \mathbf{V}_i \mathbf{e}_i = \mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i \left(\frac{1}{\rho_i \mathbf{e}_i^T \mathbf{m}_w} - 1 \right). \quad (6.35)$$

Another consequence of this special structure is that instead of the full vector \mathbf{m}_i , we only need the projection $\mathbf{m}_i^T \mathbf{e}_i$, which can be stored as a scalar m_i . Based on equations (6.21) and (6.22),

$$\begin{aligned} m_i &= (\mathbf{V}_i \mathbf{V}_w^{-1} \mathbf{m}_w - \mathbf{V}_i (\mathbf{V}_w^{\setminus i})^{-1} \mathbf{m}_w^{\setminus i})^T \mathbf{e}_i \\ &= (\mathbf{m}_w^{\setminus i})^T \mathbf{e}_i + (v_i + \mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i) \rho_i, \end{aligned} \quad (6.36)$$

6.3 Predictive Ensemble Pruning by Expectation Propagation

$$\begin{aligned}
s_i &= \frac{Z_i}{(2\pi)^{M/2} |\mathbf{V}_i|^{1/2} N(\mathbf{m}_i; \mathbf{m}_w^{\setminus i}, \mathbf{V}_i + \mathbf{V}_w^{\setminus i})} \\
&= \Phi(z_i) \sqrt{\mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i v_i^{-1} + 1} \exp\left(\frac{1}{2} \frac{\mathbf{e}_i^T \mathbf{V}_w^{\setminus i} \mathbf{e}_i}{\mathbf{e}_i^T \mathbf{m}_w^{\setminus i}} \rho_i\right). \quad (6.37)
\end{aligned}$$

6.3.2.1 Leave-one-out Estimation

A nice property of EP is that it can easily obtain an estimate of the leave-one-out error without any extra computation. In each iteration, EP computes the parameters of the approximate leave-one-out posterior $q^{\setminus n}(\mathbf{w})$ (step 2(a)) that does not depend on the n^{th} data point. So we can use the mean $\mathbf{m}_w^{\setminus n}$ to approximate a classifier trained on the other $(N - 1)$ data points. Thus an estimate of the leave-one-out MSE error can be computed as

$$err_{loo} = \frac{1}{N} \sum_{n=1}^N ((\mathbf{m}_w^{\setminus n})^T \mathbf{F}(\mathbf{x}_n) - y_n)^2. \quad (6.38)$$

6.3.3 Expectation Propagation for Classifier Ensembles

For classification ensembles, the ensemble output is a linear combination of individual classifiers passed through a link function, $f_{ens}(\mathbf{x}_n) = \Phi\left(\sum_{i=1}^M w_i f_i(\mathbf{x}_n)\right)$, where $\Phi(x) = \int_{-\infty}^x N(t|0, 1)dt$ is the Gaussian cumulative distribution function.

Given the data set $D = \{\mathbf{x}_n, y\}_{n=1}^N$, the likelihood for the combination weight \mathbf{w} can be written as

$$p(y|\mathbf{x}, \mathbf{w}) = \prod_{n=1}^N p(y_n|\mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^N \Phi\left(y_n \sum_{i=1}^M w_i f_i(\mathbf{x}_n)\right). \quad (6.39)$$

By incorporating the prior with likelihood, the posterior of weight vectors \mathbf{w} is denoted by

$$\begin{aligned}
p(\mathbf{w}|\mathbf{x}, y, \alpha) &\propto p(\mathbf{w}|\alpha) \prod_{n=1}^N p(y_n|\mathbf{x}_n, \mathbf{w}) \\
&= \prod_{i=1}^M 2N(w_i|0, \alpha_i^{-1}) \prod_{i=1}^M \Theta(w_i) \prod_{n=1}^N p(y_n|\mathbf{x}_n, \mathbf{w}). \quad (6.40)
\end{aligned}$$

In EP algorithm, we need to approximate both the likelihood term $p(y_n|\mathbf{x}_n, \mathbf{w}) = \Phi\left(y_n \sum_{i=1}^M w_i f_i(\mathbf{x}_n)\right)$ and the $\Theta(w_i)$ term. Denote the exact terms $g_n(\mathbf{w}) =$

6.3 Predictive Ensemble Pruning by Expectation Propagation

$p(y_n|\mathbf{x}_n, \mathbf{w})$ and $t_i(\mathbf{w}) = \Theta(w_i) = \Theta(\mathbf{w}^T \mathbf{e}_i)$, and the approximate terms by $\tilde{g}_n(\mathbf{w}) = s_n \exp(-\frac{1}{2v_n}(y_n \mathbf{w}^T \mathbf{F}_n - m_n)^2)$ and $\tilde{t}_i(\mathbf{w}) = s_i \exp(-\frac{1}{2v_i}(\mathbf{w}^T \mathbf{e}_i - m_i)^2)$. The EP algorithm for classification ensembles is described in the following (to simplify notations, $y_n \mathbf{F}(\mathbf{x}_n)$ is written as \mathbf{F}_n).

1. Initialization the prior term: $q(\mathbf{w}) = p(\mathbf{w}|\alpha)$. Also initialize the approximating terms to 1: $\tilde{g}_n = 1$ and $\tilde{t}_i = 1$: $m = 0$, $v = \infty$ and $s = 1$.
2. Until both \tilde{g}_n and \tilde{t}_i converge: Loop $n = 1, \dots, N$, and $i = 1, \dots, M$;
 - (a) Remove the approximation term \tilde{g}_n from the posterior $q(\mathbf{w})$ to obtain the leave-one-out posterior $q^{\setminus n}(\mathbf{w})$: $N(\mathbf{m}_{\mathbf{w}}^{\setminus n}, \mathbf{V}_{\mathbf{w}}^{\setminus n})$. The Equation is exactly the same as the regression EP. (Please refer to equations (6.14) and (6.16).)
 - (b) Combine $q^{\setminus n}(\mathbf{w})$ and the exact term $g_n(\mathbf{w})$ to get $\hat{p}(\mathbf{w}) \propto q^{\setminus n}(\mathbf{w})g_n(\mathbf{w})$ and minimize the KL-divergence between $\hat{p}(\mathbf{w})$ and new posterior $q(\mathbf{w})$.

$$Z_n = \int_{\mathbf{w}} q^{\setminus n}(\mathbf{w})g_n(\mathbf{w})d\mathbf{w} = \Phi(z_n) = \Phi\left(\frac{(\mathbf{m}_{\mathbf{w}}^{\setminus n})^T \mathbf{F}_n}{\sqrt{\mathbf{F}_n^T \mathbf{V}_{\mathbf{w}}^{\setminus n} \mathbf{F}_n + 1}}\right). \quad (6.41)$$

and

$$\mathbf{m}_{\mathbf{w}} = \mathbf{m}_{\mathbf{w}}^{\setminus n} + \mathbf{V}_{\mathbf{w}}^{\setminus n} \frac{\partial \log Z_n}{\partial \mathbf{m}_{\mathbf{w}}^{\setminus n}} = \mathbf{m}_{\mathbf{w}}^{\setminus n} + \mathbf{V}_{\mathbf{w}}^{\setminus n} \mathbf{F}_n \rho_n \quad (6.42)$$

$$\begin{aligned} \mathbf{V}_{\mathbf{w}} &= \mathbf{V}_{\mathbf{w}}^{\setminus n} + \mathbf{V}_{\mathbf{w}}^{\setminus n} \left(\frac{\partial \log Z_n}{\partial \mathbf{m}_{\mathbf{w}}^{\setminus n}} \left(\frac{\partial \log Z_n}{\partial \mathbf{m}_{\mathbf{w}}^{\setminus n}} \right)^T - 2 \frac{\partial \log Z_n}{\partial \mathbf{V}_{\mathbf{w}}^{\setminus n}} \right) \mathbf{V}_{\mathbf{w}}^{\setminus n} \\ &= \mathbf{V}_{\mathbf{w}}^{\setminus n} + (\mathbf{V}_{\mathbf{w}}^{\setminus n} \mathbf{F}_n) \frac{\rho_n (\mathbf{F}_n^T \mathbf{m}_{\mathbf{w}} + \rho_n)}{\mathbf{F}_n^T \mathbf{V}_{\mathbf{w}}^{\setminus n} \mathbf{F}_n + 1} (\mathbf{V}_{\mathbf{w}}^{\setminus n} \mathbf{F}_n)^T, \end{aligned} \quad (6.43)$$

where

$$\rho_n = \frac{1}{\sqrt{\mathbf{F}_n^T \mathbf{V}_{\mathbf{w}}^{\setminus n} \mathbf{F}_n + 1}} \frac{N(z_n; 0, 1)}{\Phi(z_n)}.$$

6.3 Predictive Ensemble Pruning by Expectation Propagation

- (c) Update the approximation term $\tilde{g}_n = Z_n \frac{q(\mathbf{w})}{q(\mathbf{w})}$ according to equations (6.21), (6.20), (6.22) and the Woodbury formula. v_n , m_n and s_n are obtained

$$v_n = \mathbf{F}_n^T \mathbf{V}_n^{\setminus n} \mathbf{F}_n \left(\frac{1}{\rho_n (\mathbf{F}_n^T \mathbf{m}_w + \rho_n)} - 1 \right) + \frac{1}{\rho_n (\mathbf{F}_n^T \mathbf{m}_w + \rho_n)} \quad (6.44)$$

$$m_n = (\mathbf{m}_w^{\setminus n})^T \mathbf{F}_n + (v_n + \mathbf{F}_n^T \mathbf{V}_n^{\setminus n} \mathbf{F}_n) \rho_n, \quad (6.45)$$

$$s_n = \Phi(z_n) \sqrt{\mathbf{F}_n^T \mathbf{V}_n^{\setminus n} \mathbf{F}_n v_n^{-1} + 1} \exp\left(\frac{1}{2} \frac{\mathbf{F}_n^T \mathbf{V}_n^{\setminus n} \mathbf{F}_n + 1}{\mathbf{F}_n^T \mathbf{m}_w + \rho_n} \rho_n\right). \quad (6.46)$$

- (d) The remaining steps are the same as the regression ensemble. Please refer to EP pruning for regression ensemble steps 2(d)-(f) in section 6.3.2.

An estimate of the leave-one-out error can be obtained by

$$err_{loo} = \frac{1}{N} \sum_{n=1}^N \Theta(-y_n (\mathbf{m}_w^{\setminus n})^T \mathbf{F}(\mathbf{x}_n)), \quad (6.47)$$

6.3.4 Hyperparameters Optimization for Expectation Propagation

The previous sections present the training algorithm of EP with fixed hyperparameter α . In this subsection, we update the hyperparameter α based on the type-II marginal likelihood, also known as the evidence (Faul and Tipping, 2002). According to the updated value of α , we choose to add one learner to the ensemble, delete one ensemble member or re-estimate the hyperparameter α .

As described in previous sections, expectation propagation approximates each term as a Gaussian distribution, leading to the situation that the likelihood of every point in a classification ensemble has similar forms as a regression likelihood term. The likelihood of each data point in classifier ensemble can be obtained as

$$p(\mathbf{m} | \mathbf{w}, \mathbf{x}) = (2\pi)^{-N} |\Lambda|^{-1/2} \exp\left(-\frac{1}{2} (\mathbf{w}^T \mathbf{F} - \mathbf{m})^T \Lambda^{-1} (\mathbf{w}^T \mathbf{F} - \mathbf{m})\right), \quad (6.48)$$

where $\mathbf{m} = (m_1, \dots, m_N)$ denotes the target point vector, $\Lambda = \text{diag}(v_1, \dots, v_N)$, v_n represents the variance of the noise for the training point n . EP actually

6.3 Predictive Ensemble Pruning by Expectation Propagation

maps a classification problem into a regression problem where (m_n, v_n) defines the virtual observation data point with mean m_n and variance v_n .

Note that we can compute analytically the posterior distribution of the weights. The posterior distribution of the weight vector is thus given by:

$$p(\mathbf{w}|\mathbf{x}, \mathbf{m}, \alpha) = \frac{p(\mathbf{m}|\mathbf{w}, \mathbf{x})p(\mathbf{w}|\alpha)}{p(\mathbf{m}|\alpha, \mathbf{x})} = (2\pi)^{-N} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{w} - \mu)^T \Sigma^{-1}(\mathbf{w} - \mu)\right), \quad (6.49)$$

where the posterior covariance and mean are:

$$\Sigma = (A + \mathbf{F}\Lambda^{-1}\mathbf{F}^T)^{-1}, \quad (6.50)$$

$$\mu = \Sigma\mathbf{F}\Lambda^{-1}\mathbf{m}. \quad (6.51)$$

where $A = \text{diag}(\alpha_1, \dots, \alpha_M)$.

For regression ensemble, the posterior of weights can be easily obtained by replacing classification likelihood terms with regression likelihood terms. The posterior of weights has the similar equations as (6.49), (6.50) and (6.51) but with different \mathbf{m} and Λ .

In order to sequentially update α , we can maximize the type-II marginal likelihood $p(D|\alpha)$. The fast algorithm to optimize the type-II marginal likelihood is to decompose $p(D|\alpha)$ into two parts, one part denoted by $p(D|\alpha_{\setminus i})$, that does not depend on α_i and another that does, i.e.,

$$p(D|\alpha) = p(D|\alpha_{\setminus i}) + l(\alpha_i), \quad (6.52)$$

where $l(\alpha_i)$ is a function that depends on α_i .

The updating rule for α_i can be obtained with the derivation of marginal likelihood (Faul and Tipping, 2002). The details have been presented in appendix C.

6.3.5 Algorithm Description

Based on the above subsections, the predictive ensemble pruning algorithm by expectation propagation is summarized as follows:

1. Include a number of learning machines in the ensemble and initialize the hyperparameters α .

6.3 Predictive Ensemble Pruning by Expectation Propagation

2. Train EP algorithm with the current hyperparameters α and sequentially update α by maximizing the type-II marginal likelihood $p(D|\alpha)$. Based on the updated values of α , we choose to add one learner to the ensemble, delete one existing ensemble member or re-estimate the hyperparameter α . Repeat this process until the algorithm converges.
3. Choose the ensemble from the sequential updates with the minimum leave-one-out error estimation. As the leave-one-out error is discrete, so in case of a tie, choose the first ensemble in the tie, i.e., the one with the smaller marginal likelihood¹.

6.3.6 Comparison of Expectation Propagation with Markov Chain Monte Carlo

Expectation Propagation is a kind of integral approximation technique. It is better to know the difference between the approximation and the exact distribution. As the truncated Gaussian prior is used in this paper, the exact posterior distribution is unknown. In this section, we employ Markov Chain Monte Carlo (MCMC) method to simulate the exact posterior distribution for the comparison with EP.

MCMC methods (Andrieu et al., 2003) are a class of algorithms for sampling from probability distributions based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. MCMC may be too slow for many practical applications, but has the advantage that it becomes exact in the limit of long runs. Thus, MCMC can provide a standard way to measure the accuracy of integral approximation methods, such as expectation propagation in this paper.

This paper uses one of the most well-known MCMC algorithms, Metropolis-Hastings algorithm (Andrieu et al., 2003) to investigate regression and classification ensembles, respectively. In our experiments, a Bagging ensemble with 100

¹Qi et al. Qi et al. (2004) pointed out that optimization of marginal likelihood can lead to over-fitting and leave-one-out error with smaller marginal likelihood is a better choice for model selection.

6.3 Predictive Ensemble Pruning by Expectation Propagation

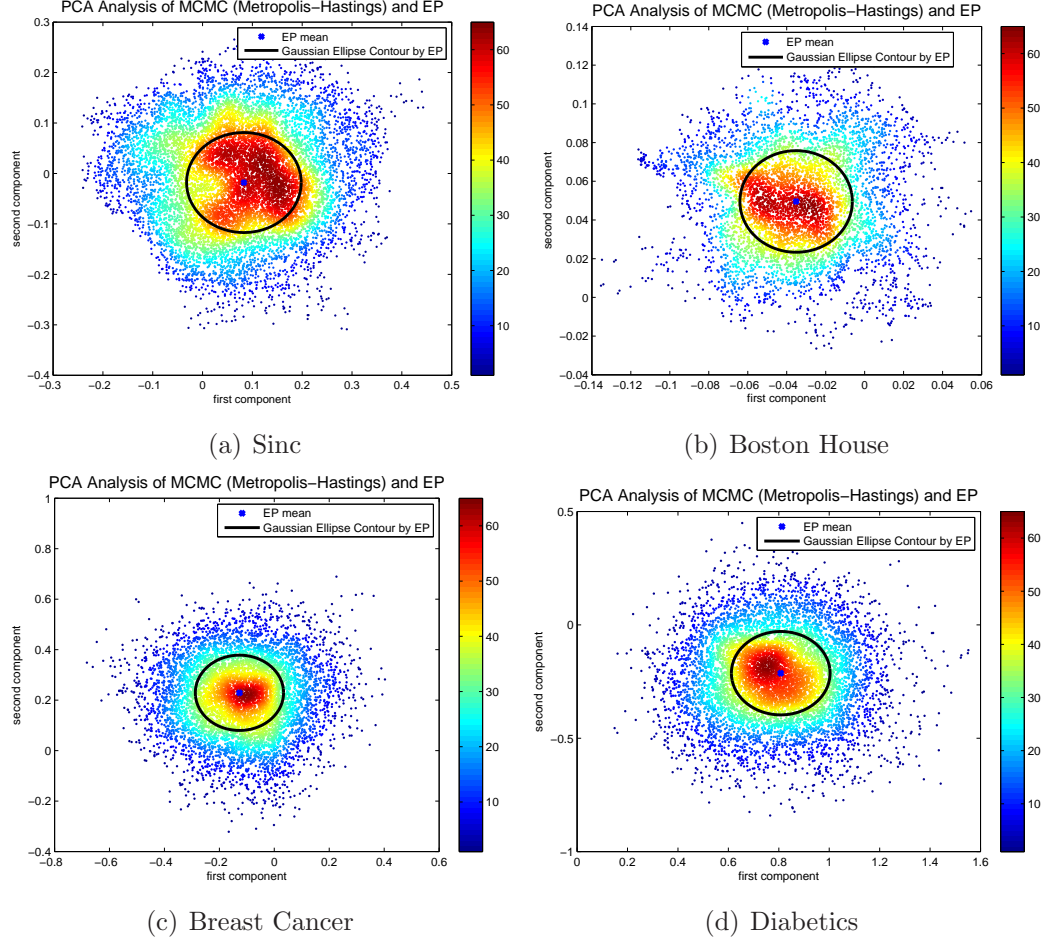


Figure 6.2: The posteriors of combination weights calculated by MCMC (30000 sampling points) and EP. The color bar indicates the density (the number of overlapping points) in each place.

Classification And Regression Trees (CARTs) is generated. MCMC and EP with the hyperparameters optimization algorithm are employed for ensemble pruning.

In most of the cases, the pruned ensemble size is larger than 2 or 3 which makes it inconvenient to directly visualize the resulting distribution. To facilitate the visualization, principal components analysis (PCA) is performed and the first two components are used for visualization. Figure 6.2 illustrates the first two components, calculated by PCA, of the posterior of weights calculated by MCMC and EP for regression and classification ensembles.

Figure 6.2 illustrates the posteriors of combination weights calculated by MCMC (30000 sampling points) and EP. We use sinc (with 0.1 Gaussian noise), Boston house, breast cancer and diabetics data sets in this figure. Note that the hyperparameters and noise terms are estimated in the hyperparameters optimization step by maximizing the marginal likelihood in both EP and MCMC methods. The posteriors of weights calculated by MCMC have irregular boundaries for these problems and EP approximates the posteriors well by picking a Gaussian to cover the densest area, although the distribution are not Gaussians, for both regression and classification problems.

The pruned ensemble sizes and the error for both regression and classification problems are shown in Table 6.1. From the table, EP and MCMC achieve similar performance in terms of both accuracy and ensemble size. EP uses much less time than MCMC.

Both figures and table indicate that EP approximates the posterior well in this ensemble pruning model with truncated Gaussian priors for regression and classification problems.

6.4 Numerical Experiments

This section presents the experimental results of expectation propagation pruning algorithm for regression problems and classification problems, respectively.

In chapter 5, we generate a population of networks using evolutionary multiobjective algorithm and include all of the individual networks in the ensemble. In this chapter, we use the ensemble pruning algorithm (EP pruning) to select the classifiers, generated by MRNCL, to produce small ensembles.

Table 6.1: The pruned ensemble size, error rate and computational time of MCMC, EP and unpruned ensembles.

Regression	Sinc			House		
	Size	MSE	Time	size	MSE	Time
MCMC	7	0.0082	343.1s	11	11.4892	398.5s
EP	8	0.0087	8.7s	11	11.5725	11.6s
Unpruned	100	0.0103	-	100	11.8464	-
Classification	Cancer			Diabetics		
	Size	%error	Time	size	error	Time
MCMC	10	26.34	676.2s	19	24.58	986.3s
EP	11	26.93	19.1s	18	24.73	62.6s
Unpruned	100	27.64	-	100	24.65	-

In chapter 2, we reviewed a number of ensemble pruning algorithms, such as least-square (LS) pruning, Bayesian automatic relevance determination (ARD) pruning and so on. These algorithms are employed to compared with our EP pruning algorithm in this section.

6.4.1 Synthetic Data Sets

As the first experiment, we compare EP-pruned ensembles with un-pruned ensembles on some synthetic data sets, including one regression data set, Sinc, and two classification data sets: synth and banana.

Figure 6.3 shows the output of EP pruning and original Bagging ensembles, which consists of 100 MLPs. We notice that EP pruning is a little better than the original ensemble in the left tail of the sinc function. With respect to ensemble size, EP pruning only picks 9 neural networks vs. 100 neural networks in the original ensemble.

In the following synthetic classification data sets, we select the Adaboost of neural networks as the ensemble algorithm because large Adaboost is prone to overfitting the noise in the training set. Figure 6.4 illustrates the decision boundaries of EP pruning and original Adaboost for both problems. Not surprisingly,

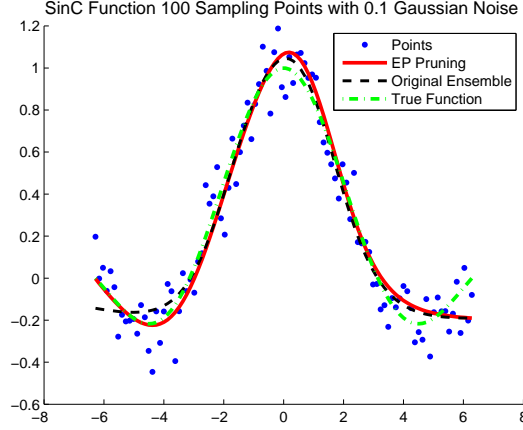


Figure 6.3: Comparison of EP-pruned ensembles and un-pruned Bagging ensembles on sinc data set. The sinc data set is generated by sampling 100 data points with 0.1 Gaussian noise from the sinc function. The Bagging ensemble consists of 100 neural networks with random selected hidden nodes (3-6 nodes).

Adaboost with 100 neural networks overfits the noise and generates the twisty boundaries.

With small ensemble size (16 for synth and 12 for banana), EP pruning removes those overfitting individuals and generates better (smoother) decision boundaries for both classification problems.

Based on these initial experiments with synthetic data, we observe that EP pruning performs better than the original ensembles by utilizing a small amount of individuals.

The following experiments will investigate EP pruning for benchmark problems. In the benchmark problems, we utilize decision trees, i.e. classification and regression trees (CART), as base learners to generate different kinds of ensembles, such as Bagging, Adaboost and Random Forest (both for classification ensembles only). Each ensemble consists of 100 CARTs.

6.4.2 Results of Regression Problems

The information on the data sets used for regression is tabulated in Table 4.1. These data sets have been used in chapter 4 to evaluate the performance of RNCL.

6.4 Numerical Experiments

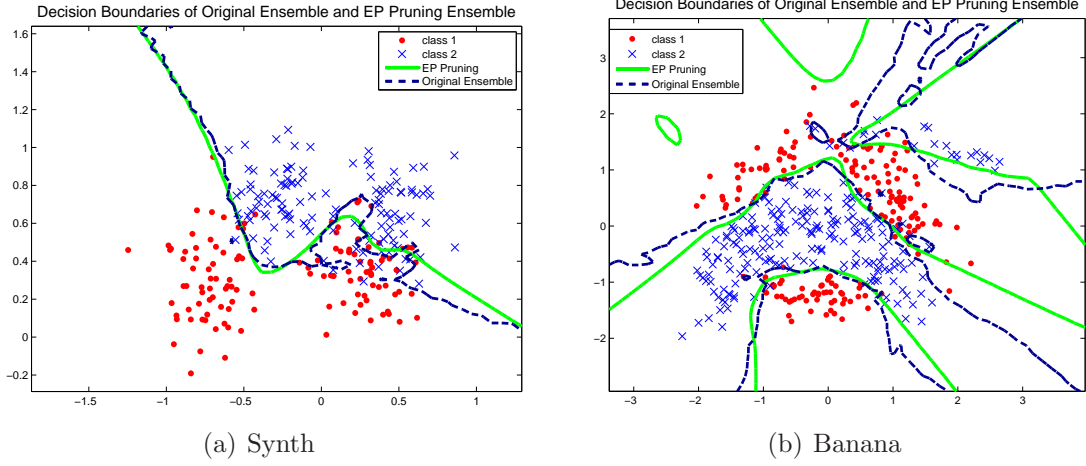


Figure 6.4: Comparison of EP-pruned ensembles and un-pruned Adaboost ensembles on Synth and banana data sets. The Adaboost ensemble consists of 100 neural networks with random selected hidden nodes (3-6 nodes).

Table 6.2: Average Test MSE, Standard Deviation for seven regression Benchmark Data sets based on 100 runs for Bagging. EP, ARD, LS, Random stand for EP pruning, ARD pruning, least square pruning and random pruning, respectively.

Bagging	EP	RVM	LS	Random	Non-pruned
Sinc	0.0091±0.0019	0.0158±0.0026	0.0254±0.0036	0.0158±0.0018	0.0102±0.0017
Friedman	4.4765±0.4287	4.6327±0.4079	4.9594±0.4355	4.7711±0.4107	4.6094±0.4196
Gabor	0.0272±0.0094	0.0289±0.0080	0.0311±0.0087	0.0518±0.0118	0.0497±0.0010
Multi	0.1472±0.0206	0.1606±0.0198	0.1994±0.0236	0.1753±0.0206	0.1537±0.0190
Plane	0.0011±0.0002	0.0015±0.0003	0.0026±0.0004	0.0018±0.0002	0.0010±0.0002
Polynomial	0.3261±0.0522	0.3614±0.0499	0.4813±0.0664	0.3765±0.0515	0.3349±0.0487
House	11.5725±4.0741	11.5803±4.3361	12.3202±4.8322	12.2844±4.8322	11.8464±4.4938
W-L-T	-	0-7-0	0-7-0	0-7-0	1-6-0
Significant	-	0-4-3	0-6-1	0-6-1	0-2-5

Table 6.3: Size of Pruned Ensemble with standard deviation for Different Algorithms for Bagging. The results are based on 100 runs.

Bagging	EP	RVM	LS	Random	Non-pruned
Sinc	7.9±1.7	21.4±5.3	100	25	100
Friedman	12.2±1.9	36.3±5.3	100	25	100
Gabor	9.6±2.0	44.3±4.6	100	25	100
Multi	13.6±1.7	34.9±5.0	100	25	100
Plane	9.3±1.5	24.4±6.3	100	25	100
Polynomial	11.2±2.1	31.3±5.3	100	25	100
House	10.5±1.5	44.0±4.4	100	25	100

Please refer to chapter 4 for detailed information.

In order to evaluate the performance of EP pruning algorithm, we have implemented Bayesian ARD pruning without negative constraints, least square pruning algorithm and random pruning algorithm on 8 regression benchmark problems for comparison purposes.

For every data set, we run independent 100 times and record the average mean squared error (MSE) and the standard deviation on the test set for different algorithms. The performance of four pruning algorithms with seven data sets has been presented in Table 6.2. A win-loss-tie summary based on mean values and t-test (95% significance level) is attached at the bottom of the table.

Table 6.2 shows that EP pruning outperforms all the other methods in 6 out of 7 data sets, comes second in one other case. Although ARD pruning uses Bayesian inference for ensemble pruning as well, it seems that adopting the negative combination weights leads to suboptimal results. The baseline algorithm, random pruning, fails to compete with EP and ARD pruning. In most cases, the least square algorithm is worse than other algorithms, which indicates that the least square algorithm often overfits the noise and does not work well in practice.

Another interesting point is that EP pruning achieves better performance by employing only a few of the ensemble members, as shown by Table 6.3. For the random pruning algorithm, we fix the size of pruned ensemble as 25 since

6.4 Numerical Experiments

Table 6.4: Average Test error, Standard Deviation for 13 classification Benchmark Data sets based on 100 runs for Bagging algorithm. EP, ARD, Kappa, CP, LS, Random stand for EP pruning, ARD pruning, kappa pruning, concurrency pruning, least square pruning and random pruning.

Bagging	EP	ARD	Kappa	CP	LS	Random	Non-pruned
Banana	12.74±0.78	13.14±0.67	13.74±0.73	13.32±0.87	14.54±0.94	14.93±0.85	12.75±0.79
Cancer	26.81±4.74	30.96±4.73	28.81±4.54	30.30±4.54	34.51±5.03	29.35±4.33	27.42±4.54
Diabetics	24.88±1.93	25.76±1.90	26.30±1.79	25.08±1.97	26.15±1.89	27.89±1.95	24.62±1.81
Solar	35.18±1.80	36.33±1.93	36.80±1.98	36.00±1.77	37.15±1.93	37.88±1.82	35.96±1.82
German	22.15±2.21	24.19±2.23	24.12±2.21	24.67±2.30	25.01±2.27	25.80±2.05	23.63±2.17
Heart	19.13±3.64	22.17±3.56	20.01±3.99	20.71±4.11	26.42±3.98	22.08±3.75	19.09±3.89
Image	2.04±0.48	2.20±0.47	2.40±0.54	2.31±0.48	2.35±0.46	2.40±0.53	2.32±0.50
Ringnorm	8.68±1.16	10.44±1.53	8.88±1.16	9.26±1.42	8.00±1.20	9.84±1.36	8.08±1.37
Splice	5.03±0.67	5.18±0.69	5.13±0.75	5.15±0.69	5.04±0.70	5.71±0.79	5.02±0.77
Thyroid	6.27±3.04	7.03±5.14	6.87±3.04	7.35±2.95	9.19±3.25	7.49±3.21	6.83±3.20
Titanic	22.36±1.31	23.72±5.61	22.56±1.67	24.00±1.64	22.49±1.21	24.50±0.96	22.57±1.01
Twonorm	7.24±1.04	12.55±6.48	7.98±0.88	8.67±1.60	7.18±0.98	10.44±1.21	6.55±1.34
Waveform	13.12±0.57	14.35±0.48	14.10±0.65	13.52±0.63	13.84±0.89	14.64±0.68	13.67±0.71
W-L-T	-	0-13-0	0-13-0	0-13-0	2-11-0	0-13-0	5-8-0
Significant	-	0-7-6	0-5-8	0-7-6	0-8-5	0-11-2	1-3-9

previous empirical research suggests that, in most cases, most or all of the generalization gain in a well-constructed ensemble comes from the first 25 learners added (Breiman, 1996a; Opitz and Maclin, 1999).

According to Table 6.3, EP pruning consistently uses significantly fewer ensemble members than other algorithms, including ARD pruning.

In general, the performance of EP pruning on these benchmark problems is better than non-pruned ensembles in terms of generalization ability and sparsity.

6.4.3 Results of Classifier Ensembles

For classifier ensembles, we use Rätsch’s data sets, which have been used in chapter 4 and 5, to make the thesis consistent. Please refer to section 4.4 for the detailed description of these data sets.

In order to compare our algorithm with others, we have implemented ARD

6.4 Numerical Experiments

Table 6.5: Size of Pruned Ensemble with standard deviation with Different Algorithms for Bagging. The results are based on 100 runs.

Bagging	EP	RVM	Kappa	CP	LS	Random	Non-pruned
Banana	10.2±2.7	12.7±1.7	25	25	100	25	100
Cancer	9.8±2.7	17.5±2.0	25	25	100	25	100
Diabetics	17.5±2.3	18.6±1.8	25	25	100	25	100
Solar	5.7±1.8	7.0±1.0	25	25	69.4±4.3	25	100
German	17.9±3.7	25.0±2.5	25	25	100	25	100
Heart	10.1±1.7	10.1±1.5	25	25	100	25	100
Image	9.4±1.4	9.5±1.5	25	25	100	25	100
Ringnorm	10.1±1.6	8.0±2.4	25	25	100	25	100
Splice	11.4±2.4	12.2±1.7	25	25	100	25	100
Thyroid	5.1±1.2	4.9±2.2	25	25	43±6.2	25	100
Titanic	3.3±1.1	25.9±22.5	25	25	74.8±1.4	25	100
Twonorm	10.6±1.5	5.3±3.5	25	25	100	25	100
Waveform	10.3±1.8	10.8±1.2	25	25	100	25	100

6.4 Numerical Experiments

Table 6.6: Average Test error, Standard Deviation for 13 classification Benchmark Data sets based on 100 runs for Adaboost algorithm. EP, ARD, Kappa, CP, LS, Random stand for EP pruning, ARD pruning, kappa pruning, concurrency pruning, least square pruning and random pruning.

Adaboost	EP	ARD	Kappa	CP	LS	Random	Non-pruned
Banana	13.49±0.65	14.19±0.76	15.85±1.48	13.40±0.72	14.23±0.67	16.65±0.69	13.51±0.60
Cancer	31.88±4.15	31.64±6.21	37.40±6.58	38.86±5.87	32.70±4.97	36.94±6.42	31.16±4.47
Diabetics	25.72±2.42	28.78±2.33	28.21±2.52	28.13±2.07	26.25±1.92	29.15±2.12	26.06±1.99
Solar	34.28±1.87	36.37±1.98	39.46±2.38	40.95±2.38	38.59±1.97	41.95±5.43	36.26±1.78
German	24.37±2.55	27.39±2.43	26.73±2.35	27.21±2.57	24.21±2.05	28.05±2.44	24.06±2.19
Heart	18.40±4.25	23.33±3.41	28.33±4.68	22.65±4.00	21.74±3.76	22.79±3.98	20.82±3.97
Image	1.23±0.54	1.78±0.72	1.46±0.55	1.40±0.37	1.15±0.35	1.40±0.42	1.12±0.35
Ringnorm	3.82±0.53	4.65±0.61	5.14±1.08	6.43±0.78	4.37±0.49	6.89±0.75	4.09±0.47
Splice	4.17±0.85	4.40±0.62	6.4±0.68	4.40±0.58	3.94±0.49	6.12±0.77	3.55±0.54
Thyroid	5.09±2.70	7.24±2.90	8.39±8.04	5.58±2.57	7.41±3.24	7.00±3.38	4.69±2.40
Titanic	21.40±0.79	23.76±0.82	28.98±0.84	29.12±0.79	23.22±0.96	32.90±1.07	21.98±0.70
Twonorm	3.52±0.41	5.49±0.52	6.08±0.52	6.01±0.83	4.14±0.44	5.85±0.39	3.79±0.30
Waveform	10.47±0.52	11.88±0.62	14.12±0.64	12.83±0.70	11.58±0.59	14.88±0.47	11.42±0.50
W-L-T	-	1-12-0	0-13-0	1-12-0	3-10-0	0-13-0	5-8-0
Significant	-	0-7-5	0-10-3	0-9-4	0-5-7	0-12-1	2-4-7

6.4 Numerical Experiments

Table 6.7: Size of Pruned Ensemble with standard deviation with Different Algorithms for Adaboost. The results are based on 100 runs.

Adaboost	EP	RVM	Kappa	CP	LS	Random	Non-pruned
Banana	10.9±1.9	10.6±1.8	25	25	100	25	100
Cancer	11.4±2.3	13.2±1.7	25	25	65.0±19.2	25	100
Diabetics	12.5±1.9	12.4±2.1	25	25	100	25	100
Solar	8.3±2.8	11.8±1.7	25	25	49.8±17.1	25	100
German	12.4±1.8	12.3±1.2	25	25	100	25	100
Heart	10.9±1.6	11.0±2.1	25	25	100	25	100
Image	8.8±2.3	6.2±2.7	25	25	100	25	100
Ringnorm	10.4±2.7	8.7±2.5	25	25	100	25	100
Splice	8.9±1.2	8.3±2.2	25	25	87.3±12.6	25	100
Thyroid	5.9±0.9	5.1±2.9	25	25	87.3±18.3	25	100
Titanic	4.9±0.9	5.6±0.7	25	25	10.7±1.8	25	100
Twonorm	11.3±1.6	8.6±1.4	25	25	100	25	100
Waveform	10.5±2.0	9.1±2.9	25	25	100	25	100

pruning, kappa pruning, concurrency pruning, least square pruning and random pruning. As same as the previous subsection, we fix the pruned ensemble size at 25 for kappa pruning, concurrency pruning and random pruning.

Tables 6.4, 6.6, and 6.8 report the performance of these algorithms with 13 benchmark data sets using Bagging, Adaboost and Random forests, respectively. The size of the ensembles is recorded in Tables 6.5, 6.7 and 6.9.

In this chapter, we also apply the EP pruning to select the best subset of individual classifiers, generated by the multiobjective regularized negative correlation learning (MRNCL) algorithm in chapter 5, to produce the ensemble. Tables 6.10 and 6.11 reports the performance on MRNCL, and the size of pruned ensembles, respectively.

According to these tables, EP pruning compares quite favorably against these different ensemble algorithms. For example, for Bagging (Table 6.4) EP pruning outperforms all the other methods, including the non-pruned ensemble on eight

6.4 Numerical Experiments

Table 6.8: Average Test error, Standard Deviation for 13 classification Benchmark Data sets based on 100 runs for Random forests algorithm. EP, ARD, Kappa, CP, LS, Random stand for EP pruning, ARD pruning, kappa pruning, concurrency pruning, least square pruning and random pruning.

Random Forest	EP	RVM	Kappa	CP	LS	Random	Non-pruned
Banana	12.76±0.44	13.83±0.42	13.70±0.49	13.89±0.57	13.89±0.89	16.24±0.79	12.72±0.48
Cancer	24.86±4.66	26.66±4.65	26.86±4.69	28.18±4.47	34.79±4.69	28.43±4.21	24.92±4.10
Diabetics	24.67±1.98	24.35±2.14	30.79±2.38	25.12±1.76	26.80±2.10	26.45±2.18	25.20±1.74
Solar	34.90±1.79	36.31±1.90	36.78±2.76	39.70±4.60	37.37±2.06	38.85±2.31	34.59±1.91
German	23.64±2.37	24.96±2.38	27.23±2.48	24.51±2.29	25.72±2.29	26.18±2.24	24.32±2.33
Heart	18.08±4.16	18.31±4.27	19.34±4.20	19.63±4.06	26.32±4.11	18.71±3.87	17.43±3.72
Image	1.81±0.40	1.83±0.58	2.37±0.81	1.98±0.48	1.67±0.42	2.01±0.49	1.84±0.44
Ringnorm	3.71±0.63	4.07±0.76	5.50±0.65	6.17±0.98	5.19±0.68	6.00±0.85	4.63±0.68
Splice	3.63±0.51	3.70±0.68	3.64±1.42	3.99±0.48	3.30±0.39	3.84±0.48	2.91±0.35
Thyroid	5.25±2.84	6.13±2.83	8.36±2.87	5.78±2.54	8.28±3.92	6.21±3.14	5.71±2.78
Titanic	22.44±1.31	22.76±2.97	24.19±2.20	24.23±2.11	22.41±1.17	23.96±2.73	23.55±2.15
Twonorm	3.89±0.51	4.21±0.81	5.93±0.56	6.74±0.75	5.39±0.53	6.06±0.60	4.31±0.40
Waveform	11.98±0.75	12.06±0.85	13.06±0.85	12.65±0.76	13.88±0.76	12.59±0.72	11.57±0.63
W-L-T	-	1-12-0	0-13-0	0-13-0	3-10-0	0-13-0	5-8-0
Significant	-	0-5-8	0-9-4	0-7-6	0-9-4	0-7-6	1-2-10

6.4 Numerical Experiments

Table 6.9: Size of Pruned Ensemble with standard deviation with different algorithms for random forest. The results are based on 100 runs.

Random Forest	EP	RVM	Kappa	CP	LS	Random	Non-pruned
Banana	14.1±2.4	15.1±4.7	25	25	76.0±4.5	25	100
Cancer	6.7±1.6	19.3±3.1	25	25	98.5±1.4	25	100
Diabetics	16.8±4.2	20.3±2.0	25	25	99.9±0.1	25	100
Solar	7.1±2.6	11.9±2.0	25	25	84.5±3.6	25	100
German	16.0±4.8	27.1±2.2	25	25	100	25	100
Heart	11.1±1.6	11.2±1.6	25	25	100	25	100
Image	10.1±1.5	8.9±2.9	25	25	100	25	100
Ringnorm	10.7±1.4	6.4±2.6	25	25	100	25	100
Splice	12.8±1.9	12.1±2.3	25	25	100	25	100
Thyroid	5.6±1.2	4.9±2.0	25	25	82.5±5.9	25	100
Titanic	3.7±1.5	22.3±18.1	25	25	96.8±1.4	25	100
Twonorm	10.8±1.4	4.8±2.9	25	25	100	25	100
Waveform	11.1±1.4	10.4±2.2	25	25	100	25	100

out of thirteen data sets, comes second in two cases and third in the remaining three. Comparing with the original ensemble, EP pruning employs much fewer ensemble members but performs better than the original ensemble. Take the Adaboost (Table 6.6) as an example, EP pruning performs better than non-pruned ensemble in eight out of thirteen cases, in which four wins are statistically significant; EP pruning loses five times, where two losses are statistically significant.

Least square (LS) pruning, which minimizes the training error, performs well only on data set with little noise, for example Image. In most situations, LS pruning does not reduce the size of an ensemble. Random pruning, which serves as the baseline algorithm, is not comparable to the original ensemble and other pruning algorithms.

According to these tables, the previous finding that Adaboost is prone to overfitting the noise in the training set is also confirmed as Adaboost performs well on data sets with little noise, such as Image, Twonorm, but worse on noise-corrupted data sets.

6.4 Numerical Experiments

Table 6.10: Average Test error, Standard Deviation for 13 classification Benchmark Data sets based on 100 runs for MRNCL algorithm. EP, ARD, Kappa, CP, LS, Random stand for EP pruning, ARD pruning, kappa pruning, concurrency pruning, lease square pruning and random pruning.

MRNCL	EP	RVM	Kappa	CP	LS	Random	Non-pruned
Banana	10.75±0.82	11.62±0.79	11.70±0.89	11.93±0.62	12.31±0.87	13.44±0.92	10.68±0.72
Cancer	24.86±4.26	26.47±4.72	26.92±4.61	27.27±4.38	30.79±4.76	28.39±4.89	26.43±4.61
Diabetics	23.53±1.86	23.38±2.14	27.72±2.28	26.25±1.79	26.58±2.11	27.40±2.14	23.26±1.68
Solar	32.43±1.79	35.36±1.90	35.78±2.36	38.57±2.63	37.42±2.07	38.95±2.31	33.10±1.76
German	23.68±2.37	25.06±2.28	26.88±2.49	26.62±2.49	25.83±2.29	27.36±2.93	24.23±2.12
Heart	16.04±3.62	16.42±3.68	17.19±4.21	17.63±4.06	19.47±3.89	19.75±4.92	15.62±3.03
Image	2.83±0.73	3.06±0.78	2.89±0.82	2.98±0.84	2.56±0.69	3.48±0.99	2.62±0.70
Ringnorm	1.58±0.25	1.89±0.36	1.66±0.32	1.92±0.35	1.72±0.22	2.69±0.63	1.61±0.23
Splice	9.96±0.51	10.13±0.61	10.34±0.83	11.36±0.86	11.87±0.60	12.37±0.94	9.89±0.59
Thyroid	4.58±2.44	4.86±2.63	4.97±2.78	5.38±2.54	4.39±3.10	6.48±3.53	4.53±2.12
Titanic	22.24±1.31	22.78±1.89	23.09±2.10	23.29±2.46	27.11±1.48	27.96±2.39	22.28±1.08
Twonorm	2.49±0.16	2.6±0.18	2.89±0.32	3.42±0.63	2.83±0.21	5.32±0.64	2.29±0.12
Waveform	10.24±0.72	11.26±0.75	11.04±0.82	12.25±0.76	13.47±0.80	14.22±0.85	10.42±0.61
W-L-T	-	0-13-0	0-13-0	0-13-0	2-11-0	0-13-0	7-6-0
Significant	-	0-5-8	0-8-5	0-6-7	0-9-4	0-10-3	2-2-9

6.4 Numerical Experiments

Table 6.11: Size of Pruned Ensemble with standard deviation with different algorithms for MRNCL. The results are based on 100 runs.

MRNCL	EP	RVM	Kappa	CP	LS	Random	Non-pruned
Banana	24.3±6.4	29.1±4.7	25	25	64.0±7.8	25	100
Cancer	18.9±3.2	24.3±5.1	25	25	100	25	100
Diabetics	36.8±4.2	31.3±4.0	25	25	100	25	100
Solar	19.2±2.6	25.9±2.0	25	25	100	25	100
German	36.0±4.1	29.1±5.2	25	25	100	25	100
Heart	18.5±2.4	19.2±1.9	25	25	66.5±7.6	25	100
Image	20.1±1.5	18.9±4.7	25	25	100	25	100
Ringnorm	16.3±2.3	12.4±3.5	25	25	100	25	100
Splice	23.4±2.2	27.1±4.3	25	25	100	25	100
Thyroid	25.9±4.2	16.4±3.8	25	25	92.5±2.9	25	100
Titanic	24.7±7.5	29.3±8.1	25	25	100	25	100
Twonorm	27.4±3.2	19.4±3.5	25	25	98.6±1.1	25	100
Waveform	30.7±2.8	26.7±3.4	25	25	100	25	100

Table 6.12: Running Time of EP pruning, ARD pruning and EM pruning on Regression Data Sets in seconds. Results are averaged over 100 runs.

Time	Sinc	Fried.	Gabor	Multi	Plane	Poly.	House
EP	8.7s	9.3s	8.6s	7.6s	7.2s	9.2	11.6s
EM	0.6s	0.5s	0.7s	0.6s	0.3s	0.6s	1.1s
ARD	0.3s	0.2s	0.3s	0.3s	0.1	0.3s	0.4s

Overall, EP pruning achieves significant sparseness in ensembles and performs better or as well as the original ensemble. It provides a way to reduce the computational complexity at the *test* stage and make the ensemble more compact. There are two possible reasons to explain the success of EP pruning.

1. EP pruning benefits from the truncated Gaussian priors. As negative combination weights are not intuitive and unreliable, especially when individual learners are highly correlated, the truncated Gaussian prior not only satisfies the constraint but also leads to a sparse ensemble. This prior controls the complexity by generating appropriate sparseness, and thus improves the generalization.
2. EP pruning employs the leave-one-out error together with the Bayesian evidence as the criterion for model selection, which is more efficient than the other algorithms.

6.4.4 Computational Complexity and Running Time

The improved performance of our algorithm comes with a price: more computation time during the *training* stage. Tables 6.12 and 6.13 show the average running time of EP pruning and other pruning algorithms over 100 runs for regression and classification problems, respectively. The computational environment is Windows XP with Intel Core 2 Duo 1.66G CPU and 2G RAM. These algorithms are implemented in MATLAB.

According to the algorithm in section 6.3, EP pruning is an iterative algorithm and it consists of two major parts: EP training and sequential update of hyperparameters α .

6.4 Numerical Experiments

Table 6.13: Running Time of EP pruning, ARD pruning, EM pruning, Kappa pruning and concurrency pruning on Classification Data Sets in seconds. Results are averaged over 100 runs.

Time	Banana	Cancer	Diabetics	Solar	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
EP	56.3s	19.1s	62.6s	42.2s	88.4s	21.4s	184.4s	83.5s	136.2s	21.7s	3.1s	84.6s	82.5s
EM	1.6s	1.1s	3.4s	2.6s	4.9s	1.7s	6.3s	4.9s	3.8s	1.3s	0.9s	4.7s	5.3s
ARD	0.7s	0.3s	1.3s	0.7s	2.0s	0.4s	2.5s	1.8s	1.6s	0.4s	0.2s	1.8s	1.8s
Kappa	0.8s	0.7s	0.9s	1.0s	1.0s	0.7s	1.5s	0.9s	1.3s	0.7s	0.6s	0.8s	0.8s
CP	1.2s	0.6s	1.3s	2.1s	2.7s	0.6s	7.2s	1.2s	4.1s	0.5s	0.5s	1.2s	1.2s

In the first part, EP processes each data point in $O(M^2)$ time, where M is the size of *current* ensemble. Assuming the number of iterations is constant, which seems to be true in practice, the computational complexity of EP training in the first part is $O(NM^2)$, where N is the number of training points. In the second step, the major running time is consumed in calculating vector products, which can be done quickly. Most of the computation time is consumed in the first part.

Although we cannot prove the convergence of EP, in our experiments it always converges for ensemble pruning with Gaussian (for regression) or probit (for classification) likelihood. In practice, 200 iterations have been adopted in our ensemble pruning algorithm. Therefore, the total estimated computational complexity of EP pruning is around $O(iter * NM^2)$, where *iter* is the number of iterations.

As indicated in the introduction, ensemble pruning algorithms can improve accuracy and reduce the test time, but will lead to a longer training time. Ensemble pruning algorithms are particularly suited to the applications that are characterized by small training but large test sets. In this subsection, we will provide an example with a relatively small training set and a large test set, namely the poker hand data set from the UCI machine learning repository (Asuncion and Newman, 2007).

The data set consists of 25010 training examples and 1 million test examples. Each example represents a hand holding five playing cards drawn from a standard deck of 52. Each card is described by two attributes (suit and rank), for a total of 10 predictive attributes (corresponding to the 5 cards). There are ten classes in

6.4 Numerical Experiments

the data set and we merge the ten classes into 2 classes. One class means nothing¹ in hand and another class means that there is something (one pair, two pairs, flush, royal flush, etc.) in hand. The percentages of the two classes are nearly balanced in both training and test data sets. Although a manually-specified rule can successfully classify the data set, this task is not easy for a machine learning algorithm operating on the provided vectorial feature representation.

We use 100 CART trees to generate a Bagging ensemble and we use different ensemble pruning algorithms to prune the ensemble. The experimental results are summarized in Table 6.14. According to Table 6.14, the EP pruning algorithm has improved both accuracy and efficiency. For example, the pruned ensemble outperformed the unpruned ensemble in terms of accuracy (error rate 25.68% vs. 27.94%). The total time (2806.1 seconds) including the EP training time and the application time for the pruned ensemble is much smaller than the application time for the unpruned ensemble (6154.9 seconds).

Table 6.14: Summary of EP, EM, ARD, LS, Kappa, CP, random and other unpruned ensembles with poker hand problem (Train points 25010 and Test points 1 mil.). The results are averaged over ten runs.

Summary	Error %	Size	Train Time	Test Time	Total Time
Unpruned	27.94	100	-	6154.9s	6154.9s
EP	25.68	11.6	2129.4s	677.4s	2806.8s
EM	27.84	19.2	2463.7s	1181.7s	3645.4s
ARD	28.42	29.6	2236.2s	1829.6s	4065.8s
LS	29.13	100	6.8s	6189.2s	6196.0s
Kappa	31.15	25	195.7s	1455.6s	1651.3s
CP	28.73	25	458.6s	1479.3s	1937.9s
Random	31.92	25	1.1s	1464.3s	1465.4s

¹In the five cards, there is *no* one pair, two pairs, three of a kind, straight (five cards, sequentially ranked with no gaps), flush (five cards with the same suit), full house (pair plus different rank three of a kind), four of a kind (four equal ranks within five cards), straight flush (straight plus flush) or royal flush (Ace, King, Queen, Jack, Ten plus flush).

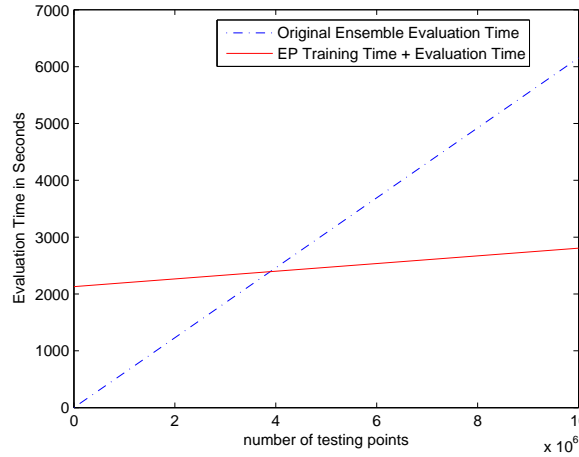


Figure 6.5: Comparison of evaluation time of each pruning method averaged.

Figure 6.5 illustrates the total time as a function of the number of test examples. According to the figure, though EP needed more time in training, the pruned ensemble is much smaller and thus consumed considerably less time in testing than the unpruned ensemble. When the number of test examples increased, EP used less time than the unpruned ensemble.

6.5 Summary

This chapter considers how to prune the ensemble to reduce the computational complexity and improve the generalization performance. This chapter proposes a probabilistic ensemble pruning algorithm in order to get a set of sparse combination weights to prune the ensemble. As the previous research implied that negative combination weights in the ensemble will degrade the performance, we introduce a left-truncated and non-negative Gaussian prior for every combination weight in this probabilistic model. The integral to calculate the normalization term is intractable in Bayesian inference after incorporating the truncated Gaussian prior. Therefore, we propose to use expectation propagation to approximate the posterior calculation in this thesis. An estimate of the leave-one-out (LOO) error can be obtained in the training of EP. The LOO error is used together with Bayesian evidence for ensemble pruning. An empirical study on several regression

and classification benchmark data sets shows that our algorithm utilizes far less component learners but performs as well as, or better than, the non-pruned ensemble. The results are promising compared with ARD pruning and some other heuristic algorithms.

In total, the contributions of the chapter includes a new ensemble pruning algorithms based on Bayesian probabilistic and a thorough analysis and empirical comparison of different combining strategies.

Chapter 7

Conclusions and future research

This chapter summarizes the work presented in the previous chapters. The potential future research on some further questions and possible extensions are also discussed.

7.1 Conclusions

This thesis focuses on analyzing and application of diversity and regularization in ensemble systems. With this aim, we investigated the theoretical and empirical analysis of diversity in classifier ensembles in chapter 3, and investigated a special kind of diversity, error diversity, using negative correlation learning (NCL) in detail, and discovered that regularization should be used to address the overfitting problem of NCL. Then we proposed the regularized negative correlation learning (RNCL) algorithm to improve the noise-robustness ability of NCL. Finally, we investigated ensemble pruning as one way to balance diversity, regularization and accuracy and studied one ensemble pruning algorithm in chapter 6. The details of these contributions and significance are described as follows.

In chapter 2 we comprehensively reviewed the literature on ensemble of learning machines from the following aspects: (i) some popular ensemble learning methods; (ii) three important decompositions for analyzing ensemble models and the current literature on analysis of diversity for classifier ensembles; (iii) some developments and applications of negative correlation learning algorithm; (iv) a number of ensemble combination and pruning methods. For the first point, we

studied the current techniques of ensemble learning and their advantages and disadvantages. The second point introduced three fundamental theoretical results for regression ensembles, the bias-variance decomposition, bias-variance-covariance decomposition and ambiguity decomposition. Thirdly, we made a thorough survey and analysis of negative correlation learning (NCL) algorithm and pointed out the potential problems in NCL, which ignited the explosion of regularized negative correlation learning techniques in the thesis. In the last part, we summarized various selection-based and weight-based algorithms for ensemble pruning, which aims to reduce the size of ensemble and simultaneously improve the generalization performance by balancing diversity, regularization and accuracy in the ensemble. The aim of this chapter is to identify the wider body of literature to which the thesis contributes.

In chapter 3, we proposed an ambiguity decomposition for *classifier* ensembles with *zero-one* loss for the first time. The proposed ambiguity decomposition is fundamental for our understanding of classifier ensembles and the decomposition can be employed to analyze the classifier ensembles. For example, a new diversity measure has been defined based on the decomposition. The superiority of the new diversity measure is confirmed by the numerical experiments against other nine diversity measures. Furthermore, we used ten diversity measures to examine the relationship between diversity and generalization for Bagging. The results showed that diversity highly correlated with generalization error when the diversity was small, and the correlation reduced after diversity exceeded a threshold. These findings explain the conflicting empirical results (García et al., 2005; Kuncheva and Whitaker, 2003) in ensemble research and point out that large diversity does not always help the generalization of ensemble.

Chapter 4 focused on the application of one specific diversity “error diversity” in negative correlation learning and pointed out two problems of NCL: (1) non-regularized NCL is prone to overfitting the noise; (2) there is no formulated approach to select the correlation coefficient. To address these problems, regularization should be used to address the overfitting problem of NCL and we proposed the regularized negative correlation learning (RNCL) by incorporating an additional regularization term for NCL. The Bayesian formulation of the RNCL and

an algorithm to optimize these regularization parameters based on Bayesian inference were proposed in this chapter. The numerical results on synthetic data as well as real-world data sets demonstrated that RNCL achieved better performance than NCL, especially when the noise level was non-trivial in the data set. There are two major contributions in this chapter. The first contribution is that we give the first theoretical and empirical analysis demonstrating that negative correlation learning (NCL) is prone to overfitting the noise. The second contribution includes the regularized negative correlation learning algorithm and the inference of regularization parameters. These work improve the noise-robustness of NCL, enhance our understanding to ensemble models and bridge the gap between ensemble research and Bayesian analysis. Another significance of the chapter is that RNCL can be viewed as a general framework, rather than an algorithm itself, which means many methods can make use of it.

The major contribution of chapter 5 is to formulate the regularized negative correlation learning as a multiobjective optimization problem. The resulting algorithm MRNCL can effectively search the best trade-off among these three terms. To facilitate the evolutionary process, the crossover and mutation operators were defined to vary the structure of RBF networks. The nondominated sorting algorithm with fitness sharing and linear rank-based fitness assignment were employed to promote diversity in MRNCL. The numerical results and visualization on both the synthetic data sets and the benchmark datasets have demonstrated that MRNCL achieves better performance than MNCL. The comparison with other state-of-the-art algorithms also demonstrates the superiority of MRNCL. The research in this chapter explicitly indicates that evolutionary multi-objective algorithms provide another way to implement effective ensemble models for supervised learning problems. In chapter 5, we also compared gradient descent based RNCL with multiobjective based MRNCL and presented their respective advantages.

To reduce the computational complexity and improve the generalization performance, chapter 6 investigated ensemble pruning as one way to balance the trade-off among diversity, regularization and accuracy. We proposed an ensemble pruning algorithm based on predictive expectation propagation, where an estimate of the leave-one-out (LOO) error can be obtained without extra computa-

tion. The LOO error is used together with Bayesian evidence for model selection. An empirical study on several regression and classification benchmark data sets showed that our algorithm utilizes far less component learners but performs as well as, or better than, the non-pruned ensemble. Moreover, we have conducted a thorough analysis and empirical comparison of different combining strategies and the results of our algorithm are promising compared with ARD pruning and some other algorithms. EP pruning offers a way to estimate the combination weights and prune the ensemble with the following compelling advantages: (1) Good generalization ability. Although our algorithm employs only a few of the ensemble members, they performs as well as, or better than, the non-pruned ensemble; (2) The highly sparse model is obtained by the sparseness-inducing prior and behaves optimally compact; (3) No parameters to tune.

The results are also positive when EP pruning algorithm is used to select the classifiers from the population, generated by multi-objective regularized negative correlation learning algorithm, to produce effective and efficient ensembles by balancing the diversity, regularization and accuracy.

7.2 Future work

This section discusses several ideas for further research that may extend and improve the methods described in this work.

7.2.1 Reduce the Computational Complexity of EP Pruning

In chapter 6, we proposed an ensemble pruning algorithm based on expectation propagation. Although this algorithm outperforms other algorithms, it does cost extra computational resources. In the future, we are interested in improving the efficiency of the algorithm.

As the source of the low efficiency in EP pruning algorithm originates from the integral approximating technique expectation propagation, one future opinion is to employ other integral approximating techniques, such as Laplace approximation (MacKay, 1998) and variational methods (Jordan et al., 1999), for the

probabilistic model. Then, we can evaluate different approximating techniques for this probabilistic ensemble pruning model.

7.2.2 Theoretical Analysis of Ensemble

Throughout the theoretical analysis in this thesis, we mainly focused on developing decompositions for binary classification problems and we empirically studied the relationship between diversity and generalization for Bagging. In the future, we would like to extend the definition of diversity to multi-class classification problems and study the relationship between diversity and generalization error for other ensemble algorithms, such as Boosting, negative correlation and random forests. After we examine the relationship between generalization error and diversity, we are interested in employing the relationship to improve the performance of ensemble.

7.2.3 Semi-supervised Regularized Negative Correlation Learning

In this work, we concentrated on developing ensemble methods for supervised learning. However, the practical dataset might consist of labeled data and a large amount of unlabeled data, and unlabeled examples are much easier to obtain than labeled ones in many real-world applications. Semi-supervised learning (Chapelle et al., 2006; Zhu, 2007), where unlabeled data is used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy.

We are interested in the application of ensemble algorithms to semi-supervised learning problems. In fact, the graph-based methods can be employed to extend the RNCL algorithm for semi-supervised learning. Graph-based semi-supervised methods define a graph where the nodes are labeled and unlabeled examples in the dataset, and edges (may be weighted) reflect the similarity of examples. These methods usually assume label smoothness over the graph.

Given a set of N labeled examples $\{\mathbf{x}_n, y_n\}_{n=1}^N$ and a set of u unlabeled examples $\{\mathbf{x}_l\}_{l=N+1}^{N+u}$, an easy extension of RNCL for semi-supervised learning is

presented as follows:

$$E_{ens} = \sum_{n=1}^N (f_{ens}(\mathbf{x}_n) - y_n)^2 + \sum_{i=1}^M \alpha_i \mathbf{w}_i^T \mathbf{w}_i + \gamma \|f_{ens}\|_I^2, \quad (7.1)$$

where γ is the coefficient parameter and $\|f_{ens}\|_I^2$ is the manifold regularization term (Belkin et al., 2006)

$$\|f_{ens}\|_I^2 = \frac{1}{(N+u)^2} \sum_{n,l=1}^{N+u} (f_{ens}(x_n) - f_{ens}(x_l))^2 W_{n,l}, \quad (7.2)$$

where $W_{n,l}$ are edge weights in the data adjacency graph and $\frac{1}{(N+u)^2}$ is the normalizing coefficient.

The formulation of semi-supervised RNCL considers the relationship among these data points, specified by the edge weights $W_{n,l}$, and this formulation can be effectively implemented by gradient descent methods.

7.2.4 Multi-objective Ensemble Learning

This thesis has utilized multi-objective algorithm to implement RNCL algorithm. There are still a lot of room to extend our work in the future. For example, we proposed the ambiguity decomposition for *classifier* ensembles in chapter 3 and empirically validated that the ambiguity term is a better diversity measure than other diversity measures. In the future, we can consider more objectives, such as the accuracy term and ambiguity term, in the multi-objective ensemble algorithms and investigate their impacts to the ensemble performance.

In the last stage of multi-objective ensemble learning, we need to aggregate these individual learners in the population to form an ensemble. Evolutionary algorithm often employs a large population for large search space and some theoretical and empirical evidences have shown that small ensembles may be better than large ensembles.

In the future, we are interested in considering the ensemble combination as an evolving population together with the network population and using cooperative coevolution for the ensemble generation.

The future work for this study also includes a more in-depth study of different evolutionary operators and fitness ranking methods and more comprehensive evaluation of the proposed algorithm on large noise data sets.

Appendix A

Diversity Measures

The existing diversity measures of classifier ensembles could be grouped to two categories: pairwise diversity measures, which are based on the measurement of any pairwise classifiers, e.g. Q statistic, Kappa statistic, correlation coefficient, disagreement measure and non-pairwise diversity measures, e.g. entropy measure, Kohavi-Wolpert variance and generalized diversity (Kuncheva and Whitaker, 2003).

A.1 Pairwise Diversity Measures

Pairwise diversity measures compute the relationship between any pairwise classifiers. All of pairwise diversity measures mentioned in this chapter rely on Table A.1. where f_i and f_j are two classifiers and N^{ab} is the number of data points for which f_i and f_j are correct/wrong when $a = 1/0$ and $b = 1/0$.

Table A.1: A 2×2 table of the relationship between a pair of classifiers f_i and f_j .

	f_j correct(1)	f_j wrong(0)
f_i correct(1)	N^{11}	N^{10}
f_i wrong(0)	N^{01}	N^{00}

- The Q statistic

Yule's Q statistic (Yule, 1900) computes the "coefficient of association" for two classifiers, f_i and f_j , is

$$Q_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}, \quad (\text{A.1})$$

For statistically independent classifiers, the expectation of $Q_{i,j}$ is 0. Q varies between -1 and 1. Classifiers that tend to recognize the same objects correctly will have positive values of Q , and those which commit errors on different objects will render negative Q value. For an ensemble of M classifiers, the averaged Q statistic over all pairs of classifiers is

$$Q_{av} = \frac{2}{M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^M Q_{i,j}. \quad (\text{A.2})$$

- Kappa statistic (Margineantu and Dietterich, 1997)

Kappa statistic compares the agreement against that which might be expected by chance. Kappa can be thought of as the chance-corrected proportional agreement, and possibly ranges from +1 (when the two classifiers agree on every example) via 0 (when the agreement of the two classifiers equals that expected by chance) to -1 (negative values occur when agreement is weaker than expected by change). Kappa statistic is defined by followings:

$$Kappa_{i,j} = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2}, \quad (\text{A.3})$$

where

$$\Theta_1 = \frac{N^{11} + N^{00}}{N}, \quad (\text{A.4})$$

and

$$\Theta_2 = \frac{(N^{11} + N^{10})(N^{11} + N^{01}) + (N^{00} + N^{01})(N^{00} + N^{10})}{N^2}, \quad (\text{A.5})$$

where N is the total number of points, i.e. $N = N^{11} + N^{10} + N^{00} + N^{01}$.

- The correlation coefficient CC (Sneath and Sokal, 1973)

The correlation coefficient indicates the strength and direction of a linear relationship between two classifiers. In general, correlation refers to the departure of two variables from independence. The correlation between two binary classifiers, f_i and f_j , is

$$CC_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{(N^{11} + N^{10})(N^{01} + N^{00})(N^{11} + N^{01})(N^{10} + N^{00})}}. \quad (\text{A.6})$$

- The disagreement measure Dis (Ho, 1998)

The disagreement measure is the ratio between the number of observations on which one classifier is correct and the other is incorrect to the total number of observations. This measure was used by (Skalak, 1996) to characterize the diversity between a base classifier and a complementary classifier, and then by (Ho, 1998) for measuring diversity in decision forests. In our notation,

$$Dis_{i,j} = \frac{N^{01} + N^{10}}{N^{11} + N^{10} + N^{01} + N^{00}}. \quad (\text{A.7})$$

For all pairwise measures we used the averaged values of the diversity matrix, calculated similarly to equation (A.2). We note that all these pairwise measures have been proposed as measures of (dis)similarity in the numerical taxonomy literature (Sneath and Sokal, 1973).

A.2 Non-pairwise Diversity Measures

- The entropy measure E (Cunningham and Carney, 2000)

The entropy among an ensemble is defined as follows:

$$E = \frac{1}{N} \sum_{n=1}^N \frac{1}{(M - \lceil M/2 \rceil)} \min\{l(\mathbf{x}_n), M - l(\mathbf{x}_n)\}. \quad (\text{A.8})$$

where $l(\mathbf{x}_n)$ denotes the number of classifiers that correctly recognize \mathbf{x}_n , and $\lceil M/2 \rceil$ denotes round toward the nearest integers greater than or equal to $M/2$. If these classifiers all were 0's or all were 1's, there is no disagreement, and the classifiers cannot be deemed diverse. E varies between 0

A.2 Non-pairwise Diversity Measures

and 1, where 0 indicates no difference and 1 indicates the highest possible diversity.

- Kohavi-Wolpert variance KW (Kohavi and Wolpert, 1996)

Kohavi and Wolpert presented a bias-variance decomposition of expected misclassification rate. They gave an expression of the variability of the predicted class label y for \mathbf{x} , across training sets, for a specific classifier model

$$\text{variance}_x = \frac{1}{2}(1 - \hat{p}(y = 1|\mathbf{x})^2 - \hat{p}(y = -1|\mathbf{x})^2), \quad (\text{A.9})$$

and averaging over the whole data set. Setting the KW measure of diversity to be

$$KW = \frac{1}{NM^2} \sum_{n=1}^N l(\mathbf{x}_n)(M - l(\mathbf{x}_n)). \quad (\text{A.10})$$

From (Kuncheva and Whitaker, 2003), KW differs from the averaged disagreement measure Dis_{av} by a coefficient,

$$KW = \frac{M-1}{2M} Dis_{av}. \quad (\text{A.11})$$

- The measure of “difficulty” θ (Hansen and Salamon, 1990)

Hansen and Salamon defined a discrete random variable θ taking values in $\{\frac{0}{M}, \frac{1}{M}, \dots, 1\}$, which denotes the proportion of classifiers in M that correctly classify an input \mathbf{x} drawn randomly from the distribution of the problem. For each input x , the estimate of $\theta(x)$ is the fraction of the M classifiers which classify x incorrectly. This measure is to measure the distribution of difficulty and in order to capture the distribution shape, variance of θ is employed in this measure. Based on this, the measure of “difficulty” θ is defined to be $Var(X)$.

- Generalized diversity GD (Partridge and Krzanowski, 1997)

In the definition of generalized diversity and coincident failure diversity, Partridge and Krzanowski assumed that maximum diversity occurs when failure of one of M classifiers in the ensemble is accompanied by correct labeling by the other classifier. The probability of both classifiers failing

is the same as the probability of one randomly picked classifier failing. Minimum diversity occurs when failure of one is always accompanied by failure of the other, then the probability of both classifiers failing is the same as the probability of one randomly picked classifier failing.

In order to quantify the maximum and minimum diversity, $p(1)$, the probability that one randomly chosen classifier will fail on a randomly chosen point and $p(2)$, two randomly chosen classifiers simultaneously fail on an input point, are defined as follows:

$$p(1) = \sum_{i=1}^M \frac{i}{M} p_i, \text{ and } p(2) = \sum_{i=1}^M \frac{i(i-1)}{M(M-1)} p_i, \quad (\text{A.12})$$

where the relative frequency p_i represents the probability that i , ($i = 0, 1, \dots, M$) classifiers will fail simultaneously on a randomly chosen input from these populations of inputs and classifiers.

Based on these assumptions, the generalization diversity measure GD is defined as

$$GD = \frac{p(1) - p(2)}{p(1)} = 1 - \frac{p(2)}{p(1)}. \quad (\text{A.13})$$

GD varies between 0 (minimum diversity when $p(2) = p(1)$) and 1 (maximum diversity when $p(2) = 0$).

- Coincident failure diversity CFD (Partridge and Krzanowski, 1997)

In order to refine the generalized diversity, Partridge and Krzanowski proposed coincident failure diversity, a modification of GD . The formula to compute the coincident failure diversity is illustrated in the following

$$CFD = \begin{cases} 0 & \text{if } p_0 = 1 \\ \frac{1}{1-p_0} \sum_{i=1}^M \frac{M-i}{M-1} p_i & \text{if } p_0 < 1 \end{cases}. \quad (\text{A.14})$$

Coincident failure diversity varies from a minimum value of 0, when all classifiers are identical, to a maximum value 1, when all points failure occur on exactly one classifier.

Appendix B

Further Details of RNCL using Bayesian Inference

B.1 Further Details of Gaussian Posterior

Considering the normalization term, the posterior of weigh vector \mathbf{w} is described as

$$P(\mathbf{w} \mid D) = \frac{\exp(-J_1(\mathbf{w}))}{\int \exp(-J_1(\mathbf{w}))d\mathbf{w}}. \quad (\text{B.1})$$

In order to obtain the result, the Taylor expansion of $J_1(\mathbf{w})$ is employed at point \mathbf{w}_{MP} .

$$J_1(\mathbf{w}) = J_1(\mathbf{w}_{MP}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP}), \quad (\text{B.2})$$

where \mathbf{w}_{MP} is the most probable weight vector, and A is the Hessian matrix of $J_1(\mathbf{w})$.

$$A = \nabla \nabla J_1 = \nabla \nabla \left(\sum_{i=1}^M \frac{\mu_i}{2} \mathbf{w}_i^T \mathbf{w}_i + \frac{\beta}{2} \sum_{n=1}^N e_n^2 \right) = \text{diag}(\Lambda) + \beta \nabla \nabla \left(\frac{1}{2} \sum_{n=1}^N e_n^2 \right), \quad (\text{B.3})$$

where $\Lambda = (\mu_1^{(1)}, \dots, \mu_1^{(n_1)}, \mu_2^{(1)}, \dots, \mu_2^{(n_2)}, \dots, \mu_M^{(1)}, \dots, \mu_M^{(n_M)})^T$ and the superscript indicates the number of repetitions of μ_i .

The integral can be computed as below:

$$\begin{aligned} \int \exp(-J_1(\mathbf{w}))d\mathbf{w} &= \int \exp(-J_1(\mathbf{w}_{MP}) - \frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP}))d\mathbf{w} \\ &= \exp(-J_1(\mathbf{w}_{MP})) \cdot (2\pi)^{W/2} \det A^{-\frac{1}{2}}. \end{aligned} \quad (\text{B.4})$$

Based on these equations, the approximated posterior of \mathbf{w} is obtained as follows

$$P(\mathbf{w} \mid D) = \frac{\exp(-J_1(\mathbf{w}))}{\int \exp(-J_1(\mathbf{w})) d\mathbf{w}} = \frac{\exp(-\frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A (\mathbf{w} - \mathbf{w}_{MP}))}{(2\pi)^{W/2} \det A^{-\frac{1}{2}}}. \quad (\text{B.5})$$

B.2 Details of Parameter Updates

The update rule for $\alpha_i = \mu_i/\beta$ is can be obtained from the derivation of J_2 .

$$J_2 = \frac{1}{2} \sum_{i=1}^M \mu_i \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} + \frac{1}{2} \beta \sum_{n=1}^N e_{n,MP}^2 - \frac{1}{2} \sum_{i=1}^M n_i \log \mu_i - \frac{1}{2} N \log \beta + \frac{1}{2} \log \det A. \quad (\text{B.6})$$

In order to apply the partial derivation to J_2 , we need to apply partial derivation to $\log \det A$.

Since $\det A = \prod_{j=1}^W (\beta \lambda_j + \Lambda_j)$, where λ_j is the eigenvalue of the Hessian matrix $\nabla \nabla \left(\frac{1}{2} \sum_{n=1}^N e_n^2 \right)$ and W is the number of weighs in ensemble.

$$\begin{aligned} \frac{\partial}{\partial \mu_i} \log \det A &= \frac{\partial}{\partial \mu_i} \log \left(\prod_{j=1}^W (\beta \lambda_j + \Lambda_j) \right) = \sum_{j \in n_i} \frac{1}{\beta \lambda_j + \mu_i}, \\ \frac{\partial}{\partial \beta} \log \det A &= \frac{\partial}{\partial \beta} \log \left(\prod_{j=1}^W (\beta \lambda_j + \Lambda_j) \right) = \sum_j \frac{\lambda_j}{\beta \lambda_j + \Lambda_j}, \end{aligned} \quad (\text{B.7})$$

where $j \in n_i$ indicates the range $\left[\sum_{t=1}^{i-1} n_t + 1, \dots, \sum_{t=1}^i n_t \right]$.

The gradient of $\log P(D \mid \mu, \beta)$ toward μ_i and β are:

$$\frac{\partial J_2}{\partial \mu_i} = \frac{1}{2} \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} - \frac{1}{2} \frac{n_i}{\mu_i} + \frac{1}{2} \sum_{j \in n_i} \frac{1}{\beta \lambda_j + \mu_i}, \quad (\text{B.8})$$

$$\frac{\partial J_2}{\partial \beta} = \frac{1}{2} \sum_{n=1}^N e_{n,MP}^2 - \frac{N}{2\beta} + \frac{1}{2} \sum_j \frac{\lambda_j}{\beta \lambda_j + \Lambda_j}. \quad (\text{B.9})$$

Setting the gradient to zero and the optimal μ_i and β can be obtained:

$$\mu_i^{new} = \frac{1}{\mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP}} \left(n_i - \sum_{j \in n_i} \frac{\mu_i}{\beta \lambda_j + \mu_i} \right), \quad (\text{B.10})$$

$$\beta^{new} = \frac{1}{\sum_{n=1}^N e_{n,MP}^2} \left(N - \sum_{j=1}^W \frac{\beta \lambda_j}{\beta \lambda_j + \Lambda_j} \right). \quad (\text{B.11})$$

B.2 Details of Parameter Updates

Combining both equations (B.10) and (B.11) and the relation $\alpha_i = \mu_i/\beta$, we obtain the following equation:

$$\beta \left[\sum_{i=1}^M \alpha_i \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} + \sum_{n=1}^N e_{n,MP}^2 \right] = N. \quad (\text{B.12})$$

In the following, we reformulate the optimization problem, equation (B.6), in μ_i and β into a scalar optimization problem in $\alpha_i = \mu_i/\beta$. Therefore, we firstly replace that optimization problem by an optimization problem in β and α_i by the relation $\mu_i = \beta\alpha_i$. As the equation (B.12) also holds in the scalar optimization, we search for the optimum only along this curve in the $(\alpha_i$ and $\beta)$ space.

By elimination of β from equation (B.12), the minimization problem from J_2 is obtained in a straightforward way:

$$J_3 = \sum_{j=1}^W \log\left(1 + \frac{\lambda_j}{\hat{\alpha}_j}\right) + N \log \left(\sum_{i=1}^M \alpha_i \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} + \sum_{n=1}^N e_{n,MP}^2 \right), \quad (\text{B.13})$$

where $\hat{\alpha}_j = \Lambda_j/\beta$ and $\Lambda = (\mu_1^{(1)}, \dots, \mu_1^{(n_1)}, \mu_2^{(1)}, \dots, \mu_2^{(n_2)}, \dots, \mu_M^{(1)}, \dots, \mu_M^{(n_M)})^T$.

Setting $\frac{\partial J_3}{\partial \alpha_i} = 0$, the update rule $\alpha_i^{new} = \mu_i/\beta$ can be obtained as follows:

$$\alpha_i^{new} = \frac{\sum_{n=1}^N e_{n,MP}^2}{\mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP}} \frac{\left(n_i - \sum_{j \in n_i} \frac{\alpha_i}{\lambda_j + \alpha_i} \right)}{\left(N - \sum_{j=1}^W \frac{\lambda_j}{\lambda_j + \hat{\alpha}_j} \right)}, \quad (\text{B.14})$$

where $j \in n_i$ indicates the range $\left[\sum_{t=1}^{i-1} n_t + 1, \dots, \sum_{t=1}^i n_t \right]$.

Appendix C

Further Details of Hyperparameters Optimization in EP

The following analysis is based on the sequential analysis of sparse Bayesian learning (Faul and Tipping, 2002). Please refer to (Faul and Tipping, 2002) for more details.

To have a sequential update on α_i , we explicitly decompose $p(D|\alpha)$ into two parts, one part denoted by $p(D|\alpha_{\setminus i})$, that does not depend on α_i and another that does, i.e.,

$$p(D|\alpha) = p(D|\alpha_{\setminus i}) + \frac{1}{2}(\log \alpha_i - \log(\alpha_i + r_i) + \frac{u_i^2}{\alpha_i + r_i}), \quad (\text{C.1})$$

where $r_i = F_i C_{\setminus i}^{-1} F_i^T$, $u_i = F_i C_{\setminus i}^{-1} \mathbf{m}$, and $C_{\setminus i} = \Lambda^{-1} + \sum_{m \neq i} F_m^T F_m$. Here F_i and F_m are the i^{th} and the m^{th} rows of the ensemble matrix \mathbf{F} respectively.

Using the above equation, $p(D|\alpha)$ has a maximum with respect to α_i :

$$\alpha_i = \frac{r_i^2}{u_i^2 - r_i}, \quad \text{if } \eta_i > 0, \quad (\text{C.2})$$

$$\alpha_i = \infty, \quad \text{if } \eta_i \leq 0, \quad (\text{C.3})$$

where $\eta_i = u_i^2 - r_i$. Thus, in order to maximize the evidence, we introduce the i^{th} learner when $\alpha_i = \infty$ and $\eta_i > 0$, exclude the i^{th} learner when $\alpha_i < \infty$ and $\eta_i \leq 0$, and re-estimate α_i according to (C.2) when $\alpha_i < \infty$ and $\eta_i > 0$.

References

- H. A. Abbass. A memetic pareto evolutionary approach to artificial neural networks. In *Proceedings of the fourteenth Australian Joint Conference on Artificial Intelligence*, volume 2256, pages 1–12, 2000. 19
- C. Andrieu, N. d. Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1-2):5–43, 2003. 113
- A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL <http://mllearn.ics.uci.edu/MLRepository.html>. 38, 69, 71, 129
- R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Ensemble diversity measures and their application to thinning. *Information Fusion*, 6(1):49–62, 2005. 100
- J. M. Bates and C. W. J. Granger. The combination of forecasts. *Operations Research*, 20:451–468, 1969. 29
- M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006. 138
- J. A. Benediktsson, J. R. Sveinsson, O. K. Ersoy, and P. H. Swain. Parallel consensual neural networks. *IEEE Transaction on Neural Networks*, 8(1):54–64, 1997. 30
- C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. 3

REFERENCES

- L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996a. [4](#), [5](#), [8](#), [15](#), [23](#), [32](#), [50](#), [69](#), [100](#), [120](#)
- L. Breiman. Arcing classifier. *Annals of Statistics*, 26(3):801–849, 1998. [8](#), [100](#)
- L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999. [8](#), [16](#), [17](#), [37](#), [100](#)
- L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. [4](#), [5](#), [18](#), [23](#)
- Leo Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996b. [101](#), [102](#)
- G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods: A survey and categorisation. *Journal of Information Fusion*, 6(1):5–20, 2005a. [6](#), [33](#)
- G. Brown, J. Wyatt, and P. Tiño. Managing diversity in regression ensembles. *Journal of Machine Learning Research*, 6:1621–1650, 2005b. [21](#), [22](#), [27](#)
- P. Buhlmann and B. Yu. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002. [8](#), [50](#), [51](#)
- P. K. Chan, W. Fan, A. L. Prodromidis, and S. J. Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems*, 14(6):67–74, 1999. [29](#)
- A. Chandra and X. Yao. Divace: Diverse and accurate ensemble learning algorithm. In *Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning*, volume 3177, pages 619–625, 2004. [19](#)
- A. Chandra and X. Yao. Ensemble learning using multi-objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms*, 5(4):417–445, 2006a. [19](#), [23](#)
- A. Chandra and X. Yao. Evolving hybrid ensembles of learning machines for better generalisation. *Neurocomputing*, 69(7-9):686–700, 2006b. [5](#), [19](#)
- A. Chandra, H. Chen, and X. Yao. Trade-off between diversity and accuracy in ensemble generation. In Y. Jin, editor, *Multi-objective Machine Learning*, pages 429–464. Springer, 2006. [12](#)

REFERENCES

- O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006. [137](#)
- N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *Journal of Machine Learning Research*, 5:421–451, 2004. [28](#)
- H. Chen and X. Yao. Evolutionary multiobjective ensemble learning based on bayesian feature selection. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 1141, pages 267–274, 2006. [13](#)
- H. Chen and X. Yao. Evolutionary random neural ensemble based on negative correlation learning. In *Proceedings of The 2007 IEEE Congress on Evolutionary Computation (CEC'07)*, pages 1468–1474, 2007a. [13](#), [27](#), [53](#)
- H. Chen and X. Yao. Evolutionary ensemble for in silico prediction of ames test mutagenicity. In *Proceedings of International Conference on Intelligent Computing 2007*, pages 1162–1171, 2007b. [13](#)
- H. Chen and X. Yao. When does diversity in classifier ensembles help generalization? *Machine Learning*, 2008. In Revise. [12](#)
- H. Chen and X. Yao. Multiobjective regularized negative correlation learning for neural network ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 2009a. In Press. [12](#), [27](#)
- H. Chen and X. Yao. Regularized negative correlation learning for neural network ensembles. *IEEE Transactions on Neural Networks*, 2009b. In Press. [12](#), [27](#)
- H. Chen, P. Tiño, and X. Yao. A probabilistic ensemble pruning algorithm. In *Workshops on Optimization-based Data Mining Techniques with Applications in Sixth IEEE International Conference on Data Mining*, pages 878–882, 2006. [13](#), [31](#), [99](#), [101](#)
- H. Chen, P. Tiño, and X. Yao. Predictive ensemble pruning by expectation propagation. *IEEE Transactions on Knowledge and Data Engineering*, 21(7): 999–1013, 2009a. [12](#)

- H. Chen, P. Tiño, and X. Yao. Probabilistic classification vector machine. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):901–914, 2009b. [11](#)
- X. Chen and M. Liu. Prediction of protein–protein interactions using random decision forest framework. *Bioinformatics*, 21(24):4394–4400, 2005. [19](#)
- K. J. Cherkauer. Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks. In *Proceedings of AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, pages 15–21, 1996. [5](#)
- R. T. Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5(4):559–583, 1989. [29](#)
- T. Cox and M. Cox. *Multidimensional Scaling*. Chapman Hall, London, 1994. [39](#)
- P. Cunningham and J. Carney. Diversity versus quality in classification ensembles based on feature selection. In *ECML'00: Proceedings of the 11th European Conference on Machine Learning*, pages 109–116, 2000. [24](#), [142](#)
- H. H. Dam, H. A. Abbass, C. Lokan, and X. Yao. Neural-based learning classifier systems. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):26–39, 2008. [27](#)
- P. Darwen and X. Yao. Every niching method has its niche: fitness sharing and implicit sharing compared. In *Proceedings of Parallel Problem Solving from Nature (PPSN) IV*, volume 1141, pages 398–407, Berlin, Germany, 1996. [82](#)
- P. J. Darwen and X. Yao. Speciation as automatic categorical modularization. *IEEE Transactions Evolutionary Computation*, 1(2):101–108, 1997. [6](#), [33](#)
- A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002. [30](#)
- R. Diaz-Uriarte and S. Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006. [5](#)

REFERENCES

- T. G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000. [9](#), [24](#), [100](#)
- T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2003. [17](#)
- P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997. [20](#)
- Pedro Domingos. A unified bias-variance decomposition and its applications. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 231–238, Morgan Kaufmann, 2000. [21](#)
- R. Ebrahimpour, E. Kabir, and M. R. Yousefi. Face detection using mixture of mlp experts. *Neural Processing Letters*, 26(1):69–82, 2007. [15](#)
- B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, London, U.K., 1993. [38](#)
- W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. Adacost: Misclassification cost-sensitive boosting. In *ICML’99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 97–105, 1999. [16](#)
- A. Faul and M. Tipping. Analysis of sparse bayesian learning. In *Advances in Neural Information Processing Systems 14*, pages 383–389, 2002. [111](#), [112](#), [148](#)
- J. H. Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997. [16](#)
- N. García, C. Hervás, and D. Ortiz. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Transactions on Evolutionary Computation*, 9(3):271–302, 2005. [6](#), [19](#), [23](#), [24](#), [27](#), [33](#), [50](#), [51](#), [52](#), [134](#)
- Y. Ge and W. Jiang. A note on mixtures of experts for multiclass responses: approximation rate and consistent bayesian inference. In *ICML’06: Proceedings of the 23rd international conference on Machine learning*, pages 329–335, 2006. [15](#)

REFERENCES

- S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992. [20](#), [57](#), [79](#)
- T. Van Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle. Bayesian framework for least-squares support vector machine classifiers, gaussian processes, and kernel fisher discriminant analysis. *Neural Computation*, 14(5):1115–1147, 2002. [3](#), [62](#)
- G. Giacinto and F. Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9-10):699–707, 2001. [6](#), [33](#)
- P. O. Gislason, J. A. Benediktsson, and J. R. Sveinsson. Random forests for land cover classification. *Pattern Recognition Letters*, 27(4):294–300, 2006. [19](#)
- R. B. Gramacy and H. K. H. Lee. Gaussian processes and limiting linear models. Technical report, Department of Applied Mathematics and Statistics, University of California, Santa Cruz, 2005. [88](#)
- Yves Grandvalet. Bagging equalizes influence. *Machine Learning*, 55(3):251–270, 2004. [16](#)
- J.V. Hansen. *Combining predictors: Meta machine learning methods and bias/variance and ambiguity decompositions*. PhD thesis, Department of Computer Science, University of Aarhus, Denmark, 2000. [69](#)
- L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990. [4](#), [5](#), [14](#), [24](#), [32](#), [143](#)
- L. K. Hansen, L. Liisberg, and P. Salamon. Ensemble methods for handwritten digit recognition. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 333–342, 1992. [5](#)
- S. Hashem. *Optimal Linear Combinations of Neural Networks*. PhD thesis, Purdue University, 1993. [29](#), [101](#), [102](#)

REFERENCES

- T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2001. [101](#), [102](#)
- S. He, H. Chen, X. Li, and X. Yao. Profiling of mass spectrometry data for ovarian cancer detection using negative correlation learning. In *Proceedings of the 19th International Conference on Artificial Neural Networks (ICANN'09)*, 2009. [12](#)
- T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998. [4](#), [18](#), [24](#), [27](#), [142](#)
- T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994. [4](#)
- A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000. ISSN 0040-1706. [57](#)
- R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90, 1993. [20](#)
- F. J. Huang, T. Chen, Z. Zhou, and H. Zhang. Pose invariant face recognition. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, pages 245–250, Washington, DC, USA, 2000. [5](#)
- D. Husmeier and S. J. Roberts. Regularisation of rbf-networks with the bayesian evidence scheme. In *Proceedings of the 8th International Conference on Artificial Neural Networks(ICANN99)*, pages 533–538, 1999. [3](#)
- M. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transaction on Neural Networks*, 14(4):820–834, 2003. [21](#), [27](#), [53](#)
- R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. [14](#), [15](#)

REFERENCES

- Gareth James. Variance and bias for general loss functions. *Machine Learning*, 51(2):115–135, 2003. [21](#)
- M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994. [4](#), [15](#)
- M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999. [136](#)
- Y. Kim, W. N. Street, and F. Menczer. Meta-evolutionary ensembles. In *Proceedings of the 2002 International Joint Conference on Neural Networks*, volume 3, pages 2791–2796, 2002. [30](#), [101](#)
- Ron Kohavi and David H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 275–283, Morgan Kaufmann, 1996. [6](#), [21](#), [24](#), [33](#), [143](#)
- A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, volume 4, pages 950–957, 1992. [7](#), [56](#), [79](#)
- A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems 7*, pages 231–238, 1995. [5](#), [6](#), [22](#), [24](#), [29](#), [32](#), [33](#), [34](#), [79](#)
- S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951. [103](#), [104](#)
- L. I. Kuncheva. A Theoretical Study on Six Classifier Fusion Strategies. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, pages 281–286, 2002. [4](#)
- L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003. [6](#), [23](#), [24](#), [33](#), [43](#), [46](#), [50](#), [51](#), [52](#), [134](#), [140](#), [143](#)

REFERENCES

- M. Leblanc and R. Tibshirani. Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91(436):1641–1650, 1996. [101](#), [102](#)
- X. Liao, H. Li, and L. Carin. Quadratically gated mixture of experts for incomplete data classification. In *ICML'07: Proceedings of the 24th international conference on Machine learning*, pages 553–560, 2007. [15](#)
- Y. Liao and J. Moody. Constructing heterogeneous committees using input feature grouping: Application to economic forecasting. In *Advances in Neural Information Processing Systems*, pages 921–927, 1999. [18](#)
- Y. Liu and X. Yao. Negatively correlated neural networks can produce best ensembles. In *Australian Journal of Intelligent Information Processing Systems* 4(3/4), pages 176–185, 1997. [4](#), [26](#)
- Y. Liu and X. Yao. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(6):716–725, 1999a. [5](#), [7](#), [21](#), [23](#), [26](#), [33](#), [53](#), [75](#)
- Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999b. [4](#), [7](#), [21](#), [23](#), [26](#), [32](#), [33](#), [53](#), [75](#)
- Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transaction on Evolutionary Computation*, 4(4):380–387, 2000. [7](#), [23](#), [26](#), [33](#), [53](#), [75](#)
- D. J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(3):720–736, 1992a. [3](#)
- D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992b. [3](#)
- D. J. C. MacKay. Choice of basis for laplace approximation. *Machine Learning*, 33(1):77–86, 1998. [136](#)

REFERENCES

- H. Mamitsuka. Empirical evaluation of ensemble feature subset selection methods for learning from a high-dimensional database in drug design. In *Proceedings of Third IEEE Symposium on BioInformatics and BioEngineering*, pages 253–257, 2003. [18](#)
- D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 211–218, 1997. [28](#), [100](#), [141](#)
- T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001. [101](#), [103](#), [106](#), [107](#), [108](#)
- M. Møller. *Efficient Training of Feed-Forward Neural Networks*. PhD thesis, University of Aarhus, Denmark, 1993a. [73](#)
- M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Network*, 6(4):525–533, 1993b. ISSN 0893-6080. [57](#)
- L. S. Oliveira, M. Morita, R. Sabourin, and F. Bortolozzi. Multi-objective genetic algorithms to create ensemble of classifiers. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, volume 87, pages 592–606, 2005. [18](#), [19](#), [23](#)
- D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999. [9](#), [100](#), [120](#)
- D. W. Optiz. Feature selection for ensembles. In *Proceedings of the 16th International Conference on Artificial Intelligence*, pages 379–384, 1999. [18](#)
- J. O’Sullivan, J. Langford, R. Caruana, and A. Blum. Featureboost: A meta-learning algorithm that improves model robustness. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 703–710, San Francisco, CA, USA, 2000. [18](#)
- D. Partridge and W. J. Krzanowski. Software diversity: Practical statistics for its measurement and exploitation. *Information and Software Technology*, 39: 707C717, 1997. [6](#), [24](#), [33](#), [143](#), [144](#)

- B. A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, 1994. [73](#)
- M. P. Perrone. *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization*. PhD thesis, Brown University, USA, 1993. [29](#)
- A. Prodromidis and P. Chan. Meta-learning in a distributed data mining system: Issues and approaches. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 211–218, 1998. [29](#)
- Y. A. Qi, T. P. Minka, R. W. Picard, and Z. Ghahramani. Predictive automatic relevance determination by expectation propagation. In *ICML’04: Proceedings of the twenty-first international conference on Machine learning*, pages 85–92, New York, NY, USA, 2004. [113](#)
- G. Rätsch, T. Onoda, and K. R. Müller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001. [x](#), [17](#), [37](#)
- G. Rätsch, A. Demiriz, and K. P. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1/3):189–218, 2002. [18](#)
- L. Reyzin and R. E. Schapire. How boosting the margin can also boost classifier complexity. In *ICML’06: Proceedings of the 23rd international conference on Machine learning*, pages 753–760, 2006. [8](#)
- G. Ridgeway, D. Madigan, and T. Richardson. Boosting methodology for regression problems. In *Proceedings of Artificial Intelligence and Statistics*, pages 152–161, 1999. [69](#)
- B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. [65](#)
- R. Rosipal and M. Girolami. An expectation-maximization approach to nonlinear component analysis. *Neural Computation*, 13(3):505–510, 2001. [63](#), [73](#)
- R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990. [16](#)

REFERENCES

- R. E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406, 1999. [4](#), [8](#), [16](#), [33](#), [100](#)
- R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998. [37](#)
- D. Skalak. The sources of increased accuracy for two proposed boosting algorithms. In *AAAI’96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, 1996. [142](#)
- P. Sneath and R. Sokal. *Numerical Taxonomy*. W H Freeman & Co, 1973. [24](#), [142](#)
- Y. Song, D. Zhou, J. Huang, I. G. Councill, H. Zha, and C. L. Giles. Boosting the feature space: Text classification for unstructured data on the web. In *ICDM ’06: Proceedings of the Sixth International Conference on Data Mining*, pages 1064–1069, 2006. [18](#)
- N. Srinivas and K. Deb. Multiobjective function optimization using nondominated sorting genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995. [82](#)
- E. K. Tang, P. N. Suganthan, and X. Yao. An analysis of diversity measures. *Machine Learning*, 65(1):247–271, 2006. [5](#), [23](#), [33](#)
- J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000. [41](#)
- M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1(3):211–244, 2001. [3](#), [30](#)
- A. Tsymbal, P. Cunningham, M. Pechenizkiy, and S. Puuronen. Search strategies for ensemble feature selection in medical diagnostics. In *Proceedings of 16th IEEE Symposium on Computer-Based Medical Systems*, pages 124–129. IEEE Computer Society, 2003. [18](#)

REFERENCES

- N. Ueda. Optimal linear combination of neural networks for improving classification performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(2):207–215, 2000. [30](#)
- P. Ueda and R. Nakano. Generalization error of ensemble estimators. In *Proceedings of International Conference on Neural Networks*, pages 90–95, 1996. [21](#)
- V. N. Vapnik. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995. [79](#)
- V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998. [2](#), [3](#), [7](#)
- P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR'01: Proceedings of 2001 International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001. [5](#)
- G. Wahba, X. Lin, F. Gao, D. Xiang, R. Klein, and B. Klein. The bias-variance tradeoff and the randomized gacv. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 620–626, 1999. [21](#)
- J. A. E. Weston, M. O. Stitson, A. Gammerman, V. Vovk, and V. Vapnik. Experiments with support vector machines. Technical Report CSD-TR-96-19, Royal Holloway University of London, London, 1996. [69](#)
- M. Woodbury. Inverting modified matrices. *Memorandum Report*, 42, 1950. [106](#)
- X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 28(3):417–425, 1998. [8](#), [30](#), [100](#)
- X. Yao, M. Fischer, and G. Brown. Neural network ensembles and their application to traffic flow prediction in telecommunications networks. In *Proceedings of International Joint Conference on Neural Networks*, pages 693–698, 2001. [7](#), [53](#)
- X. Yin, C. Liu, and Z. Han. Feature combination using boosting. *Pattern Recognition Letters*, 26(14):2195–2205, 2005. [18](#)

REFERENCES

- L. Yu, H. Chen, S. Wang, and K. K. Lai. Evolving least squares support vector machines for stock market trend mining. *IEEE Transactions on Evolutionary Computation*, 2008. In Press. [12](#)
- U. Yule. On the association of attributes in statistics. *Philosophical Transactions of the Royal Society of London. Series A*, 194:257–319, 1900. [24](#), [141](#)
- Z. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137(1-2):239–263, 2002. [8](#), [29](#), [30](#), [100](#)
- X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin, Madison, 2007. URL http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf. [137](#)