DATA TRANSFORMATION FOR DECISION TREE ENSEMBLES

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2009

By
Amir Ahmad
School of Computer Science

Contents

Abstract **18**

Declaration **19**

Copyright **20**

Acknowledgements **21**

1 Introduction **23**
1.1 A Committee Decision 23
1.2 Data Transformation and Ensembles in Machine Learning 24
1.3 Statement of Problems Tackled 25
1.3.1 Decision Tree Ensembles - The Representational Problem . . 25
1.3.2 Decision Tree Ensembles - Data fragmentation problem . . . 26
1.3.3 Our Approach 27
1.4 Thesis Structure 29
1.5 Publications Resulting from the Thesis 30
1.6 Notations 31

2 Literature Survey **33**
2.1 Supervised Learning 33
2.2 Decision Trees 33
2.2.1 Splitting Criteria 37
2.2.2 Node splits for Continuous Attributes 38
2.2.3 Binary Split or Multi-Way Split for Categorical Attributes? . . 39
2.3 Types of Decision Nodes 42
2.4 Motivation for Classifier Ensembles 44
2.5 Theoretical Models for Classifier Ensembles 46

2.6 Methods of Constructing Classifier Ensembles 47
2.6.1 Changing the Distribution of Training Data Points 47
2.6.2 Changing the Attributes Used in the Training 48
2.6.3 Output Manipulation 48
2.6.4 Injecting Randomness into the Learning Algorithm 48
2.6.5 Combination of Different Ensemble Methods 49
2.7 Some Popular Ensemble Methods 49
2.7.1 Bagging 49
2.7.2 Boosting 50
2.7.3 MultiBoosting 50
2.7.4 Random Subspaces 51
2.7.5 Dietterich's Random Trees 51
2.7.6 Random Forests 51
2.7.7 Extremely Randomized Trees 52
2.7.8 The Random Oracle Framework 52
2.8 Conclusion 54

3 Data Transformation Techniques 55
3.1 Different Data Transformation Techniques 55
3.2 Principal component analysis (PCA) 56
3.3 Random Projection (RP) 57
3.4 Discretization 60
3.4.1 Discretization Methods 61
3.4.2 Effect of the Discretization Process on Different Classifiers . . 64
3.5 Data Transformation in Classifier Ensembles 65
3.6 Conclusion 66

4 A Study of Random Linear Oracle Framework and Its Extensions 67
4.1 Diverse Linear Multivariate Decision Trees 67
4.2 Random Linear Oracle Ensembles 70
4.3 Learned-Random Linear Oracle 72
4.4 Multi-Random Linear Ensembles 74
4.5 Experiments 78
4.6 Results 82
4.6.1 Ensembles of Linear Multivariate Decision Trees 82
4.6.2 Comparative Study of RLO, LRLO and Multi-RLE 85

4.7 Conclusion 86

5 A Novel Ensemble Method for the Representational Problem 88
5.1 Random Discretized Ensembles (RDEns) 88
5.1.1 Data Generation 89
5.1.2 Learning 91
5.2 Motivation For Random Discretization Ensembles 92
5.3 Related Work 94
5.4 Experiments 95
5.5 Analysis 96
5.5.1 Noisy Data 96
5.5.2 The Study of the Ensemble Size 99
5.5.3 The Effect of the Number of Discretized Bins 101
5.5.4 The Study of Time/Space Complexities 104
5.6 Combining Random discretized Ensembles with Multi-RLE 107
5.7 Motivation for Random Projection Random Discretization Ensembles (RPRDE) 108
5.8 Experiments 109
5.8.1 Parameters for RPRDE 110
5.8.2 Controlled Experiment 110
5.8.3 Comparative Study 111
5.8.4 The Study of Ensemble Diversity 112
5.8.5 RPRDE against the Other Classifiers 115
5.8.6 Noisy Data 115
5.8.7 Combining RPRD with Other Ensemble Methods 117
5.9 Weaknesses 122
5.10 Conclusion 122

6 A Novel Ensemble Method to Reduce the Data Fragmentation Problem 123
6.1 Data fragmentation problem 123
6.2 Random Ordinality Ensembles 124
6.2.1 Data Generation 124
6.2.2 Learning 125
6.3 Empirical Evaluation of RO: Trees and Ensembles 127
6.3.1 Experiments with a Single RO Tree 127
6.3.2 Experiments with RO Ensembles 127

6.4 Study of RO attributes in the information theoretic framework 131
6.5 Controlled Experiments 134
6.5.1 Discussion 138
6.6 Analysis 139
6.7 Analysis of RO Ensembles 141
6.7.1 The Effect of the Data Fragmentation 142
6.7.2 RO Tree Sizes 142
6.7.3 The Diversity - Accuracy Trade Off 144
6.7.4 The Effect of the Ensemble Size 147
6.7.5 Combinations of RO with the Other Ensemble Methods . . . 155
6.8 Conclusion 157

7 Conclusion and Future work **158**
7.1 Contributions of the Thesis 158
7.1.1 Conclusion 159
7.2 Future Work 160

A **162**
A.1 Datasets 162
A.2 The Kappa measure 162
A.3 Results for RPRDE 164

Bibliography **169**

List of Tables

2.1 Continuous data. 38
2.2 Tennis Data. 40

3.1 A continuous dataset. We present discretization of this dataset by different methods. 62
3.2 Different ensemble methods that use data transformation. 66

4.1 Comparative chart of RLO, RLO′, LRLO and Multi-RLE on the basis of number of possibilities to be considered at the root node and other nodes. '-' means split points are created randomly and '+' means split points are created by using the selected split criteria. m is the number of the original attributes, d is the number of new attributes created by RP and n is the number of data points in the training data. 79
4.2 Classification errors in % for the linear multivariate ensemble method. 1,5 and 10 new attributes, created by using random projections, are added. We also presented the results with other ensemble methods. Results suggest that the Adaboost.M1 and Random Forests generally perform better than the proposed method. 83
4.3 Classification errors in % of Bagging and its combination with RLO, LRLO and Multi-RLE. Bold numbers show the best performance. Results suggest that creating a large number of new attributes and concatenating with the original features is the best strategy in the RLO framework. 84

5.1 A two dimensional numeric dataset. 91
5.2 Classification errors (in %) for different ensembles methods on different datasets, bold numbers show the best performance. RD ensembles and ERD ensembles generally perform similar to or better than bagging and quite competitive with Adaboost.M1 and Random Forests. . 97

5.3 Comparison Table- '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm. RD ensembles perform similar to or better than bagging and quite competitive with Adaboost.M1 and Random Forests. 98

5.4 Classification errors (in %) for different ensembles methods for the Pendigit dataset with different levels of noise, bold numbers show the best performance. 99

5.5 Comparison table for the Pendigit dataset with different levels of noise - '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm. 99

5.6 Classification errors (in %) for different ensembles methods for the Segment dataset with different levels of noise, bold numbers show the best performance. 99

5.7 Comparison table for the Segment dataset with different levels of noise - '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm. 100

5.8 Classification errors (in %) for different ensembles methods for the Vowel dataset with different levels of noise, bold numbers show the best performance. 100

5.9 Comparison table for Vowel data with different levels of noise - '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm. 100

5.10 Classification errors (in %) for different ensembles methods for the Waveform dataset with different levels of noise, bold numbers show the best performance. 100

5.11 Comparison table for the Waveform data with different levels of noise - '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm. 101

5.12 Classification errors (in %) for different ensembles methods for the Pendigit dataset with different number of discretized bins. Last four columns show the classification error of a single decision tree, bold numbers show the best performance. 104

5.13 Classification errors (in %) for different ensembles methods for the Segment dataset with different number of discretized bins. Last four columns show the classification error of a single decision tree, bold numbers show the best performance. 104

5.14 Classification errors (in %) for different ensembles methods for the Vowel dataset with different number of discretized bins. Last four columns show the classification error of a single decision tree, bold numbers the show best performance. 105

5.15 Classification errors (in %) for different ensembles methods for the Waveform dataset with different number of discretized bins. Last four columns show the classification error of single decision tree, bold numbers show the best performance. 105

5.16 Time in sec. taken in the tree growing phase for different trees. 106

5.17 Complexities of different trees. 106

5.18 Classification errors with the simulated data, bold numbers show the best results. Results suggest that RPRDE ensembles can learn a diagonal problem very well. This shows that these ensembles have good representational power. 111

5.19 Classification error(in %) for different ensembles methods on different dataset, bold numbers show best performance. Ensemble size 10. . . . 113

5.20 Classification errors (in %) for different ensemble methods on different datasets, bold numbers show best performance, the ensemble size 100. RPRD ensembles generally perform similar to or better than other ensemble methods, however, their competitive advantage is more for smaller ensembles. 114

5.21 Average classification errors (in %) of different methods on different datasets, bold numbers show the best performance. 115

5.22 Classification errors (in %) for different ensemble methods on different datasets, bold numbers show best performance, the ensemble size 10, the class noise is 10%. RPRD ensembles generally perform similar to or better than other ensemble methods and their competitive advantage is more for the noisy data. 118

5.23 Classification errors (in %) for different ensemble methods on different datasets, bold numbers show best performance, the ensemble size 100, the class noise is 10%. RPRD ensembles generally perform similar to or better than other ensemble methods, however, their competitive advantage is more for smaller ensembles. 119

5.24 Comparative study of Bagging against RPRD + Bagging. '+/-' shows that performance of RPRD + Bagging is statistically better/worse than Bagging for that dataset. or most of the data studied, the combination of RPRD with Bagging has positive effect. 120

5.25 Comparative study of AdaBoost.M1 against RPRD + AdaBoost.M1. '+/-' shows that performance of RPRD + AdaBoost.M1 is statistically better/worse than AdaBoost.M1 for that dataset. The combination of RPRD with AdaBoost.M1 is less successful than the combination of RPRD with Bagging. 121

6.1 Original Dataset - All attributes are categorical. 125

6.2 New continuous data created from the dataset presented in Table 6.1 with ordering of attribute 1 values as Dog<Cow<Rat<Cat and attribute 2 values as Deer<Bird<Sheep<Bat. 126

6.3 New continuous data created from the dataset presented in Table 6.1 with ordering of attribute 1 values as Dog<Rat<Cow<Cat and attribute 2 values as Sheep<Bat<Deer<Bird. 126

6.4 Average classification error of single decision tree (J48) with original data and single decision tree (J48) with RO attributes. On 9/13 datasets, the average errors of the RO trees are lower than standard multi-way decision trees trained on the original data (multi-way split). 128

6.5 Classification error in % for different ensembles (rank on the basis of average classification accuracy is given in brackets), bold numbers show best performance. ROE ensembles generally perform similar to or better than other ensemble methods. 129

6.6 Comparative Study of ROE with J48 and ROE with RT. Results are presented *ROE with J48*/*ROE with RT*. if performance of these ensembles are different. '+/-' shows that performance of ROE is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between ROE and that algorithm. ROE ensembles generally perform similar to or better than other ensemble methods. 130

6.7 Information gain ratio of attributes with different numbers of attribute values. RO attributes have better information gain ratio than multi-way splits. 132

6.8 Testing error in % (bold numbers indicate the best performance) for $Odd\_Even\_Data\_4\_6$ dataset, '+' suggests that RO ensembles are statistically better than that ensemble method. 139

6.9 Testing error in % (bold numbers indicate the best performance) for $Odd\_Even\_Data\_4\_10$, '+' suggests that RO ensembles are statistically better than that ensemble method. 139

6.10 Testing error in % (bold numbers indicate the best performance) for $Odd\_Even\_Data\_8\_6$, '+' suggests that RO ensembles are statistically better than that ensemble method. 140

6.11 Testing error in % (bold numbers indicate the best performance) for $Odd\_Even\_Data\_8\_10$, '+' suggests that RO ensembles are statistically better than that ensemble method. 140

6.12 Testing error in % (bold numbers indicate the best performance) for $Categorical\_11 - Multiplexer$, the attribute cardinality is 6, '+' suggests that RO ensembles are statistically better than that ensemble method. 140

6.13 Testing error in % (bold numbers indicate the best performance) for $Categorical\_11 - Multiplexer$, the attribute cardinality is 10, '+' suggests that RO ensembles are statistically better than that ensemble method. 140

6.14 Testing error in % (bold numbers indicate the best performance) for $Categorical\_20 - Multiplexer$, the attribute cardinality is 6, '+' suggests that RO ensembles are statistically better than that ensemble method. 141

6.15 Testing error in % (bold numbers indicate the best performance) for $Categorical\_20 - Multiplexer$, the attribute cardinality is 10, '+' suggests that RO ensembles are statistically better than that ensemble method. 141

6.16 The average sizes of RO trees and multi-split J48 trees for different datasets. RO trees are smaller than multi-way trees. 144

6.17 Comparative Study of Bagging against RO + Bagging. '+/-' shows that performance of RO + Bagging is statistically better/worse than Bagging for that dataset. Results suggest that RO can be combined with Bagging to improve the performance of Bagging. 156

6.18 Comparative Study of AdaBoost.M1 against RO + AdaBoost.M1. '+/-' shows that performance of RO + AdaBoost.M1 is statistically better/worse than AdaBoost.M1 for that dataset. Results suggest that RO can be combined with AdaBoost.M1 to improve the performance of AdaBoost.M1. 156

A.1 Datasets used in experiments. All datasets are categorical. 162

A.2 Datasets used in experiments. These datasets are pure continuous datasets. 163

A.3 Comparison Table - The ensembles size 10, '+' shows that performance of RPRDE is statistically better than that algorithm for that dataset, '-' shows that RPRDE is statistically worse for that dataset than this algorithm, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RPRDE and that algorithm. 165

A.4 Comparison Table - The ensemble size 100, '+' shows that performance of *RPRD* is statistically better than that algorithm for that dataset, '-' shows that RPRDE is statistically worse for that dataset than this algorithm, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RPRDE and that algorithm. . 166

A.5 Comparison Table - The ensembles size 10, '+' shows that performance of *RPRD* is statistically better than that algorithm for that dataset, '-' shows that *RPRD* is statistically worse for that dataset than this algorithm, 'Δ' shows that there is no statistically significant difference in performance for this dataset between *RPRD* and that algorithm. The class noise is 10%. 167
A.6 Comparison Table - The ensembles Size 100, '+' shows that performance of *RPRD* is statistically better than that algorithm for that dataset, '-' shows that RPRDE is statistically worse for that dataset than this algorithm, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RPRDE and that algorithm. The class noise is 10%. 168

List of Figures

1.1 The left figure shows the true diagonal decision boundary and three staircase approximations to it (of the kind that are created by decision tree algorithms). The right figure shows the voted decision boundary, which is a much better approximation to the diagonal boundary. The figure is taken from [29]. 26

1.2 The framework of the thesis. Gray colour boxes show research works carried out in this thesis. 28

2.1 An example of a decision tree. 34

2.2 ID3 decision tree algorithm. 35

2.3 An example of a multi-way split and a binary split for the Tennis data for the Outlook attribute. 41

2.4 Graph for number of possible splits against the attribute cardinality. . 42

2.5 Examples of univariate(solid line), linear multivariate(dotted line) and non-linear multivariate(dashed line) splits. 43

2.6 Three reasons why an ensemble works better than a single classifier. the figure is taken from [29]. 45

2.7 XOR classification problem and its solution using a linear oracle and two linear subclassifiers [68]. 54

3.1 A two dimensional dataset, the variation for this data is in different directions (principal components) and not in the natural directions. . . 57

3.2 A method to create Random matrix for RP. 58

3.3 Summary of Discretization Methods [32]. 61

4.1 Algorithm for ensembles of linear multivariate decision trees. 69

4.2 The RLO (hyperplane) is generated by taking two random points A and B from the training set and calculating the heperplane perpendicular to the line segment between the points and running through the middle point. 71
4.3 Project all data points on a random direction and spit the data by selecting the random point C. 72
4.4 A RLO′ omnivariate decision tree with a random hyperplane at the root node. 73
4.5 The original Random Linear Oracle (RLO) algorithm [68]. The highlighted portion of the algorithm is modified in the proposed RLO′ and LRLO. 74
4.6 Random Linear Oracle (RLO) algorithm by using RP. The highlighted portion of the algorithm is different from the original RLO. We define this algorithm as RLO′. 75
4.7 A LRLO ominivariate decision tree with a decision stump at the root node. 76
4.8 Learned-Random Linear Oracle (LRLO) algorithm. The highlighted portion of the algorithm is different from the original RLO. 77
4.9 Multi-Random Linear Ensembles (Multi-RLE) algorithm. d new attributes are created and concatenated with the original features. 79
4.10 RLO′, LRLO and Multi-RLE trees. A dotted line represents random hyperplane, a solid line represents a decision stump trained on new features created using random projections. 80

5.1 Random Discretization (RD) method. 90
5.2 Random Distretization Ensembles (RDEns) algorithm. 92
5.3 Division of axis by trees is uniform and fine grained. There is a diagonal concept. The combination of trees approximates the diagonal concept. Right side of the figure shows a small portion of the concept and its approximation by the ensemble of ERD trees. 93
5.4 Classification errors of various ensemble methods for the Pendigit dataset against the size of the ensemble. 101
5.5 Classification errors of various ensemble methods for the Segment dataset against the size of the ensemble. 102
5.6 Classification error of various ensemble methods for the Vowel dataset against the size of the ensemble. 102

5.7 Classification error of various ensemble methods for the Waveform dataset against the size of the ensemble. 103

5.8 RPRDE algorithm. In this method attributes created by using RD and by using RP are concatenated. 108

5.9 Kappa-error plots for four ensemble methods, First column- RPRDE, second column - Bagging, third column - AdaBoost.M1, fourth column - MultiBoosting and last column RF. x-axis - Kappa, y-axis - the average error of the pair of classifiers. Axes scales are constant for various ensemble methods for a particular dataset (each row). Lower κ represents a higher diversity. The plots suggest that RPRDE classifiers are accurate with reasonable diversity. 116

6.1 The example of multi-valued categorical attributes having four values A, B, C, D and are converted to ordinal data by imposing random ordinality, A = 4, B = 3, C = 1, D = 2. 125

6.2 Algorithm for Random Ordinality Ensembles(ROE). 126

6.3 Information gain ratio for RO attributes, Random Split and multi-way splits. RO attributes have better information gain ratio than multi-way splits and random splits. 133

6.4 Information gain ratio for attributes created by using the RO method, for attributes with different cardinalities. Left column - probability vs gain ratio, right column - cumulative probability vs information gain ratio. Small cumulative probability at low information gain ratio suggests that splits for RO attributes are good for classification. 135

6.5 Information gain ratio for attributes created using random splits, for attributes with different cardinalities. Left column - probability vs information gain ratio, right column - cumulative probability vs information gain ratio. Large cumulative probability at low information gain ratio suggests that these random splits are not as good as splits created for RO attributes for classification. 136

6.6 Cumulative probability for information gain ratio for an attribute of cardinality 12 (Left- RO attribute, Right - Random split). Smaller cumulative probability at low information gain ratio suggests that splits created for RO attributes are better for classification. 137

6.7 The effect of equal width discretization on various ensemble methods for the Vehicle dataset. RO ensembles are quite robust to data fragmentation. 143

6.8 The effect of equal width discretization on various ensemble methods for the Segment dataset. RO ensembles are quite robust to data fragmentation. 143

6.9 Kappa-error diagrams for three ensemble methods, Left column- ROE with J48, middle column - ROE with RT, right column - Bagging. x-axis - Kappa, y-axis - the average error of the pair of classifiers. Axes scales are constant for various ensemble methods for a particular dataset (each row). Lower κ represents higher diversity. RO ensembles have accurate classifiers with reasonable diversity. 145

6.10 Part of the dataset available at each node for different depth, for decision trees with different number of splits at each node. 147

6.11 Tree depth ratio ($\frac{\vartheta_2}{\vartheta_{|A|}}$) for different number of splits ($|A|$) such that N($\vartheta_{|A|}$) = N(ϑ_2) where N($\vartheta_{|A|}$) is the number of points at each node at depth ϑ_k, for trees having $|A|$ splits at each node. 148

6.12 Classification error (with 95% confidence interval) of various ensemble methods vs size of the ensemble for different datasets. 149

6.13 Classification error (with 95% confidence interval) of various ensemble methods vs size of the ensemble for different datasets. 150

6.14 Classification error (with 95% confidence interval) of RO ensembles (top fig. ROE with J48 and bottom fig. ROE with RT, solid line) for the Car dataset with expected classification error (dotted line) using Fumera et al. [42] framework. The Y-axis of the graph represents testing error in % of the ensemble, and the X- axis represents the number of classifiers in the ensemble. 151

6.15 Classification error (with 95% confidence interval) of RO ensembles (top fig. ROE with J48 and bottom fig. ROE with RT, solid line) for the DNA dataset with expected classification error (dotted line) using Fumera et al. [42] framework. The Y-axis of the graph represents testing error in % of the ensemble, and the X- axis represents the number of classifiers in the ensemble. 152

6.16 Classification error (with 95% confidence interval) of RO ensembles (top fig. ROE with J48 and bottom fig. ROE with RT, solid line) for the Promoter dataset with expected classification error (dotted line) using Fumera et al. [42] framework. The Y-axis of the graph represents testing error in % of the ensemble, and the X- axis represents the number of classifiers in the ensemble. 153
6.17 Classification error (with 95% confidence interval) of RO ensembles (top fig. ROE with J48 and bottom fig. ROE with RT, solid line) for the Tic-tac-toe dataset with expected classification error (dotted line) using Fumera et al. [42] framework. The Y-axis of the graph represents testing error in % of the ensemble, and the X- axis represents the number of classifiers in the ensemble. 154

Abstract

In pattern recognition fields classifiers, computer programs that take decisions, are extensively used. Finding the right problem representation can make a huge difference to a classifier the study of data transformations is therefore important. Taking different opinions before reaching to a final decision is an important part of the decision making process. Ensembles are combination of multiple classifiers. Ensembles have been shown to produce better results than individual models, if the models, in the ensembles, are *accurate* and *diverse*. Ensemble approaches allow us to increase *robustness* by using *multiple* different and complementary representations. In this thesis, we study data transformation techniques from the perspective of decision tree ensembles.

Random Linear Oracle is an ensemble technique introduced by Kuncheva and Rodriguez [68]. We demonstrate that RLO can be viewed as a data transformation technique using random projections. This observation allows us to develop various extensions and a generalized RLO framework.

Decision Trees suffer from two significant problems - "representation" and "data fragmentation". The first refers to the fact that Decision trees have limitation in learning non-orthogonal problems (complex problems). The second refers to small number of learning examples at the lower level of decision trees, hence statistical decisions at lower levels of decision trees are not reliable. We present two novel transformation methods to address each of these problems. The first projects from a continuous space to a categorical space (Random Discretization, Chapter 5) and is helpful in creating diverse decision trees and ensembles of these trees can learn non-orthogonal problems. The second projects from a categorical space to a continuous space (Random Ordinality, Chapter 6) and is useful to reduce the data fragmentation problem for multi-valued categorical datasets.

One of the advantages of using these data transformation techniques to create ensembles of decision trees is that these data transformations can be combined with the existing ensemble methods to improve them.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

Acknowledgements

I express my deep gratitude and sincere thanks to my research supervisor Dr. Gavin Brown for his invaluable guidance, inspiring discussions, critical review, care and encouragement throughout this PhD work. His ideas, stimulating comments, interpretations and suggestions increased my cognitive awareness and have helped considerably in the fruition of my objectives. I remain obliged to him for his help and able guidance through all stages of this. His constant inspiration and encouragement towards my efforts shall always be acknowledged.

My sincere thanks are due to my PhD advisor Dr. Jon Shapiro for his help and sopport in defining my research problems.

I am also grateful to all Machine Learning Optimization (MLO) group staff members, Dr. Neil Lawrence, Dr. Magnus Rattray, Dr. Joshua Knowles, Dr. Ke Chen, Dr. Pedro Mendes, Dr. Richard Neville, Dr. Sridhar Rajagopalan and Xiaojun Zeng for providing me all kind of supports without which this work would not have been possible.

I am thankful to all MLO group members Kevin Sharp, Richard Allmendinger, Mauricio Alvarez, Adam Pocock, Richard Stapenhurs, Ruofei He, Ahmad Salman, Jennifer Withers, John Butterworth, Stefan Haflidason, Seemab Latif, Chong Liu, Zarrar A Malik, Arslan Shaukat, Shihai Wang, Yun Yang for their affection and support during my research work. I am thankful to past MLO group members Richard Pearson, Hao Wu, Gwenn Englebienne, Hussein Sharif for their help and support.

I am also equally grateful to Dr. L.I. Kuncheva, School of Computer Science,Bangor University for inviting me to a workshop on Classifier Ensembles, Feature Selection and fMRI Data Analysis in Bangor and for providing me feedback on my work. I am thankful to J. J. Rodriguez,School of Informatics and Systems, University of Burgos, Spain, for the useful discussion on ensemble methods. Special thanks to Mr. Chris Whitaker, Mr Thomas Christy and Dr Ik Soo Lim (all from Bangor, University) for organizing the conference.

My esteemed thanks are also due to my parents, whose sacrifices made me what I am today and without their support it would have been impossible to complete this research work. I would like to remember and thank to my brother and sister. Thanks are also due to my wife for her moral support and helps. Special thanks and love for my daughter Ayesha.

Debts being various, are not easy to remember, hence I convey my heartiest thanks to all those who helped me or blessed me in making this milestone. I deeply regret here for not mentioning these individuals.

Chapter 1

Introduction

In this chapter, we present the different problems studied in this thesis. We discuss our proposed solutions to these problems. We also present the structure of the thesis in the chapter.

1.1 A Committee Decision

Decision making is an important part of everyone's life. Different processes are applied to improve the accuracy of these decisions. One of the methods is to take the decision of the person who is an expert in that problem domain. For example, if a person is ill he takes the decision of a doctor. However, when the problem is complex, instead of relying on one expert, people generally take decisions of many experts and the final decision is taken on the basis of these decisions. By taking the decisions of various experts, the accuracy and the confidence in the final decision are improved as we get different views of the problem. For example, if a person is critically ill, generally a team of doctors who are experts in different fields of medical science takes the decision.

Taking different opinions before reaching to the final decision is an important part of the effective decision making process. A committee of experts, a decision by majority voting, taking a decision on the basis of different reviews (e.g. movie reviews, product reviews etc.) etc. are the examples of such decision making processes.

Presenting the different views of the problem to the same person may improve the decision of the person rather than presenting only one view. For example, supposing a person has to decide about the quality of a car, if he decides only on the basis of the one aspect of the car like the size of the car he may not make the right decision. However, if he decides on the basis of different aspects of the car like design, size, colour, fuel

efficiency, maintenance cost etc., he may make better judgement about the quality of the car.

The decision maker may have some weaknesses for example colour blindness; in which he may not able to recognise the colour of the car properly. One possible solution to this problem is that an expert who understands these weaknesses, suggests the other properties of the car that the person can use in his judgement. This process may improve the decision making capability of the person. My PhD thesis addresses similar problems.

1.2 Data Transformation and Ensembles in Machine Learning

Different classifiers have different properties. The performance of classifiers is dependent on the problem representation. The data transformations is a process by which the problem representation is changed. For example, the discretization process [32] is used to convert a continuous dataset to a categorical dataset. Naive Bayes classifiers have shown better performance with discretized datasets [32]. This suggests that the data transformation is an important research field in machine learning.

Ensembles are a combination of multiple base models [29, 63, 67, 48, 90, 79] for which the final classification depends on the combined outputs of individual models. Classifier ensembles have been shown to produce better results than single models, if the classifiers in the ensemble are *accurate* and *diverse* [48, 90].

There are different methods to create these ensembles. One popular method is to present different problem representations to a classifier algorithm. We can train many classifiers using this process. These classifiers are different because they are trained on various problem representations. Their decisions are combined to get the final decision. This final decision is generally better than a single classifier [67]. Several methods have been proposed to create different problem representations.

In the present thesis, we study ensembles of decision tree classifiers [14, 80]. The decision tree is one of the most popular classification algorithms in the pattern recognition field [14, 80] because univariate decision trees algorithms are computationally efficient and decision trees are able to generate understandable rules [14, 80]. Decision trees ensembles are very popular ensembles because decision trees are *unstable* classifiers– classifiers whose output undergoes significant changes, in response to small

changes in training data. If decision trees are trained with different training data representations, decision trees will be diverse and the final result will be better than or similar to a single classifier. Mansilla and Ho study the domain of dominant competence of various popular classifiers in a space of data complexity measurements. They observe that the sophisticated ensemble classifiers tend to be robust for wider types of problems and are largely equivalent in performance. Despite the fact that an ensemble of decision trees requires many decision trees, the low computational cost of growing a single decision tree makes ensembles of decision trees attractive classification algorithms.

1.3 Statement of Problems Tackled

Decision trees are very popular, however, decision trees have some weaknesses; a representational problem [14, 16](an univariate decision tree does not learn non-orthogonal decision surfaces properly) and a large data fragmentation [14, 35, 10] problem (decisions have little or no statistical support) due to multi-way splits for multi-valued categorical datasets.

1.3.1 Decision Tree Ensembles - The Representational Problem

For pure continuous datasets, decision trees may have representation problem. Generally decision trees like CART [14], C4.5 [80] etc. are univariate decision trees. At each node, a univariate decision tree can take the decision only on the basis of a single attribute. That restricts the representational power of decision trees. Any decision surface that is not perpendicular to a attribute axis is approximated by these decision trees. Very large decision trees can approximate these boundaries well. However, to grow a very large decision trees we need a sufficiently large dataset. The lack of a large dataset often restricts the representational power of a decision trees. Ensembles of decision trees generally perform better than a single decision trees as they have better representational powers (an ensemble of small decision trees act as a large decision tree [29]) (Fig. 1.1). The development of ensembles that have very good representational power is the key to good performance of ensembles.

Linear multivariate decision trees (at each node a linear multivariate decision tree can take the decision on the basis of a linear combination of the attributes) algorithms [14, 16] is the other strategy to improve the representational power of decision trees.

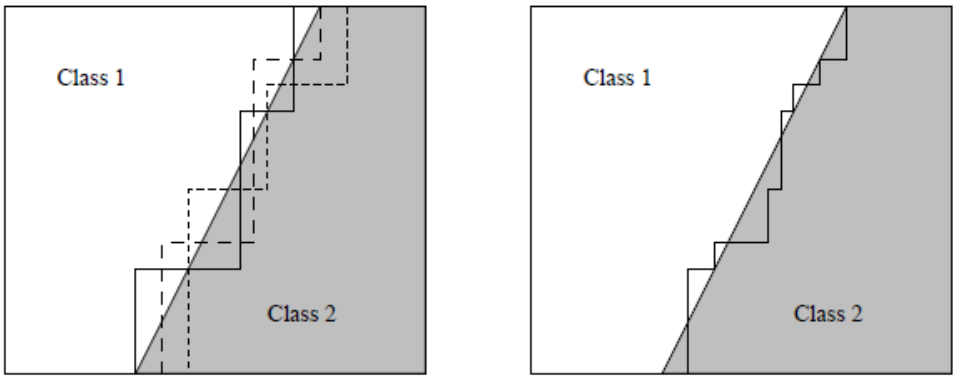


Figure 1.1: The left figure shows the true diagonal decision boundary and three staircase approximations to it (of the kind that are created by decision tree algorithms). The right figure shows the voted decision boundary, which is a much better approximation to the diagonal boundary. The figure is taken from [29].

Linear multivariate decision trees have orthogonal (orthogonal to a attribute axis) and non-orthogonal decision surfaces. However, it is computational expensive (NP-hard) to create multivariate decision trees [49].

Omnivariate decision trees [98, 99] can take decisions on the non-linear combination of attributes. They have better representational power than univariate decision trees and linear univariate decision trees. However, they are the most computationally expensive.

1.3.2 Decision Tree Ensembles - Data fragmentation problem

Variables having categories without a natural ordering are called categorical [3]. The analysis of categorical datasets is quite popular in bioinformatics, social sciences, marketing research etc. [3, 92]. Decision trees can handle categorical data well. However, datasets with multi-valued categorical attributes can cause major problems for decision trees. While multi-way splits produce a more comprehensible tree, they may increase the *data fragmentation* problem [94]; the continuous partitioning of the training set at every tree node reduces the number of examples at lower-level nodes. As decisions in the lower levels nodes are based on increasingly smaller fragments of the data, some of them may not have much statistical significance. Creating binary splits by splitting the attribute values into two groups is a method to avoid multi-splits. Breiman [14] suggests exhaustive search to find the best binary split. If the number of attribute values is $|A|$ then the number of nontrivial binary splits is given by $2^{(|A|-1)} - 1$. Selecting

the best split by this method is computationally expensive. Geurts et al. [47] suggest a randomized method to create binary attributes from the multi-valued attributes; they divide the attribute values randomly into the two categories. As in this method the node split decision is taken without considering the output, the classification accuracy of the tree may be poor.

1.3.3 Our Approach

In the thesis, we present various data transformation methods that are designed for decision tree ensembles keeping in view their (decision trees) weaknesses. In other words, our methods act as the experts who understand the weaknesses of the classifiers (decision trees) and present the different problem representations so that the classifiers can use these problem representations in a proper way. Fig. 1.2 shows the framework of the thesis.

In this thesis, first we study and propose ensemble methods for better representational power. We investigate how we can create ensemble of *linear multivariate* decision trees by using *univariate* decision tree algorithms. We then show that this method is a generalization of an existing ensemble method; the random linear oracle framework [68]. We then present random discretization methods to create diverse discretized datasets. *In these methods, bin boundaries are created randomly.* We introduce a novel ensemble method, in which each decision tree is trained on one dataset from a pool of different discretized datasets created by random discretization methods. Different decision trees trained on datasets having different discretization boundaries are diverse. These ensembles are simple but quite accurate. Theoretical analysis shows that these ensembles have good representational power and can approximate any decision surface. We then combine two proposed approaches (ensemble of linear multivariate decision trees and random discretized ensembles) to create a solution that is better than both of these solutions.

Next, we propose a method to create diverse and accurate binary decision trees that reduces the data fragmentation problem without a large computational complexity. *This technique is based on the data manipulation by imposing random ordinality on categorical attribute values*. This implies a random projection of a categorical attribute into a continuous space. A decision tree that learns on this new continuous space is able to use binary splits, hence reduces the data fragmentation problem. Decision trees trained on the diverse datasets are themselves diverse and accurate. A majority-vote ensemble is then constructed with several trees. RO ensembles resist the data

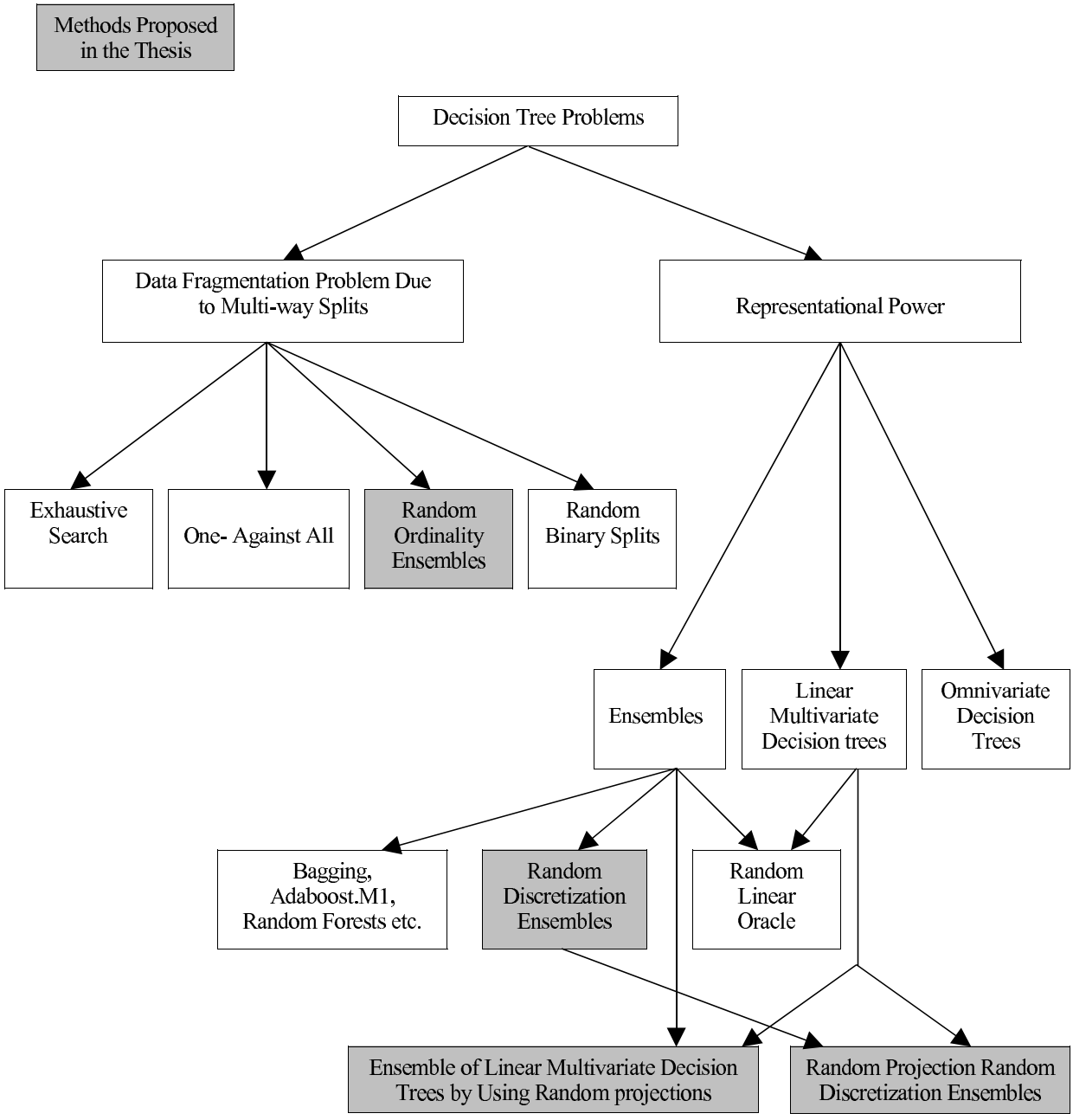


Figure 1.2: The framework of the thesis. Gray colour boxes show research works carried out in this thesis.

fragmentation problem, and provide significantly improved accuracies over current ensemble methods.

1.4 Thesis Structure

The present thesis is divided into 7 chapters. In Chapter 2, we present the relevant literature survey for the present thesis. First, we discuss the decision tree algorithms. We also present various popular split criteria. Linear multivariate decision trees are discussed in detail. Next, we discuss the philosophy of ensemble methods. Different strategies, used to create ensembles, are discussed. Furthermore, we review various popular ensemble methods.

In Chapter 3, various popular data transformation techniques are discussed. We also present the use of different data transformation techniques in the creation of ensembles.

In Chapter 4, we use the random projection technique to study the random linear oracle (RLO) framework that is proposed to improve the performance of various ensemble methods [68, 84]. We propose two new variants of the random linear oracle approach (Learned-Random Linear Oracle and Multi-random linear oracle) that extend the philosophy of the RLO approach. The comparative study of these three methods is presented, which suggests that the Multi-random linear oracle method generally gives superior performance.

In Chapter 5, we suggest a method Random Discretization (RD) to create diverse discretized datasets. We introduce a novel ensemble method Random Discretized Ensembles (RDEns), in which each decision tree is trained on one dataset from the pool of different datasetss created by RD. Ensembles created by using the RD process are simple but quite accurate. The theoretical analysis shows that RD ensembles have good representational power and can approximate any decision surface. We discuss the results of experiments to study the performance of RDEns against other popular ensemble techniques. Results suggest that RDEns matches or outperforms Bagging and Random Forests and is competitive with AdaBoost.M1. We also discuss the experiments on the noisy data and also present the analysis of RD trees and RD ensembles.

We then present Random Projection Random Discretized Ensembles (RPRDE) to create ensembles of multivariate decision trees using a univariate decision tree algorithm. This method combines the RD technique and the Multi-random linear oracle method. We discuss the results of our experiments comparing RPRDE with other

popular ensemble methods. Detailed results suggest that RPRDE performs similar or better than other ensemble methods. However, it has more competitive advantages at smaller ensembles. We also present results of experiments on the noisy data which suggests that RPRDE is quite robust to the noisy data. The accuracy-diversity experiments are presented to have better understanding of RPRD ensembles.

In Chapter 6, we describe a data transformation technique, Random Ordinality (RO), for multi-valued categorical attributes. We study the attributes, created by using the Random Ordinality (RO) technique, by using the information-theoretic framework. The study suggests that these RO attributes are good for classification. Decision trees created by using RO attributes are binary, thereby reduce the data fragmentation problem. We create RO ensembles by using RO trees as these are diverse. The proposed methods outperforms other popular ensemble methods. We also present results of controlled experiments to study how RO trees are affected by the data fragmentation (also called of curse of dimensionality) problem, the error-diversity trade-off, and the applicability of a recently proposed theoretical framework [43] in predicting the performance of Random Ordinality ensembles. We also present the results of combining RO with Bagging and AdaBoost.M1.

Chapter 7 concludes the thesis by summarizing the contributions of the present thesis and present future work.

1.5 Publications Resulting from the Thesis

The work in the present thesis resulted in the following publications:
(1)- Ahmad A. and Brown G., *Random Ordinality Ensembles : A Novel Ensemble Method for Multi-Valued Categorical Data*, Intl. Workshop on Multiple Classifier Systems. Iceland, June 2009.
(2)- Ahmad A. and Brown G., *A study of Random Linear Oracle*, Intl. Workshop on Multiple Classifier Systems. Iceland, June 2009.

1.6 Notations

Symbols

T	Training data
x	An input vector
k	The number of classes in the training dataset
C	Class attribute
$C_i(i = 1..k)$	Classes
n	The number of data points in the training dataset
m	The number of attributes in the training dataset
A	An attribute with different attribute values $a_1, a_2, .., a_{\|A\|}$
$\sharp T_1, \sharp T_2, .., \sharp T_{\|A\|}$	The partition of the training data by attributes
D_i	i^{th} Decision tree of an ensemble
M	Ensemble size
d	The number of new features created by using random projection
R	Random matrix for random projection
ρ	One dimensional data created using random projection
r_{ij}	Elements of the random matrix R
Λ	Random Discretization
ϑ	The height of a tree
θ_1	The observed agreement between the classifiers
θ_2	Agreement-by-chance between the classifiers
κ	The measure of the diversity of the two classifiers
$q_i^{(j)}$	The difference between the error rates of the two classifiers on fold j of replication i
σ	The average correlation of the errors of the base models
$p(C_i)$	A priori probability of the class C_i
$H(C)$	Entropy of the class attribute
$H(C\|A)$	The average specific conditional entropy of C
$I(C; A)$	The reduction in uncertainty (entropy) about the value of C when we know the value of A
$E(X)$	The expected value of X

Abbreviations

RP	Random Projection
RLO	Random Linear Oracle
LRLO	Learned-Random Linear Oracle
Multi-RLE	Multi-Random Linear Oracle
RD	Random Discretization
ERD	Extreme Random Discretization
RDEns	Random Discretized Ensembles
RPRD	Random Prpjection Random Discretization
RPRDE	Random Prpjection Random Discretization Ensembles
RO	Random Ordinality
ROE	Random Ordinality Ensembles
RS	Random Subspaces
RF	Random Forests

Chapter 2

Literature Survey

This chapter gives an introduction to decision tree classifiers. It discusses the motivation of ensembles and summarizes various ensemble techniques. In the end, several data transformation techniques are described along with their applications for creating ensembles of classifiers.

2.1 Supervised Learning

In the pattern recognition field, a pattern is defined by the feature x_i which represents the pattern and its related value y_i. For a classification problem, y_i represents a class or more than one class to which the pattern belongs. For a regression problem, y_i is a real value. For a classification problem, the task of a classifier is to learn from the given training dataset in which patterns with their classes are provided. The output of the classifier is a model or hypothesis h that provides the relationship between the attributes x_i and the class y_i. This hypothesis h is used to predict the class of a pattern depending upon the attributes of the pattern.

Neural networks [8, 74] , naive Bayes [8, 74], decision trees [14, 80] and support vector machines [93, 17] etc. are popular classifiers. In this thesis, we concentrate on decision trees.

2.2 Decision Trees

Decision trees are very popular tools for classification [14, 80]. As discussed in Chapter 1, the attractiveness of decision trees is due to the fact that, decision trees represent rules. Rules can readily be expressed so that humans can understand them. Decision

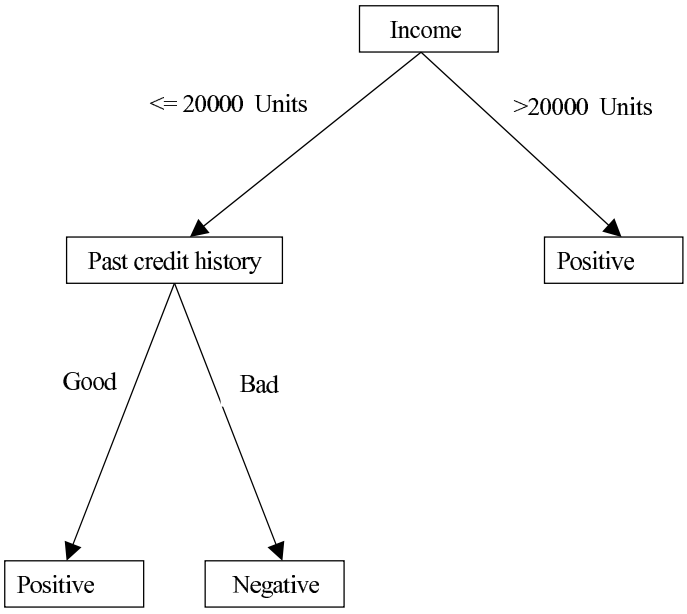


Figure 2.1: An example of a decision tree.

trees provide the information about which attributes are most important for prediction or classification. A decision tree is in the form of a tree structure, where each node is either:

- *A leaf node* - indicates the value of the target class of examples, or
- *A decision node* - specifies some test to be carried out on a single attribute-value, with two or more than two branches and each branch has a sub-tree.

A decision tree can be used to classify an example by starting at the root of the tree and moving through it until a leaf node, which provides the rules for classification of the example. For example, in Fig. 2.1 we have three rules for a positive credit rating or a negative credit rating.

1. If income $>$ 20000 units, the credit rating positive.
2. If income $<=$ 20000 units and the past credit history = good, the credit rating positive.
3. If income $<=$ 20000 units and the past credit history = bad, the credit rating negative.

Decision trees are constructed in a top-down manner. The ID3 decision tree learning algorithm is shown in Fig. 2.2. It starts with a node that grows recursively. At each

ID3 (Examples, Target_Attribute, Attributes)
Create a root node for the tree
if All examples are positive **then**
  Return the single-node tree Root, with label = positive.
else if All examples are negative **then**
  Return the single-node tree Root, with label = negative.
else if The number of predicting attributes is empty **then**
  Return the single node tree Root, with label = most common value of the target attribute in the examples.
else
  Begin
  Find the best attribute A by using the selected splitting criterion (the information gain ratio).
  Decision Tree attribute for Root = A.
  for all Possible values, a_i, of A **do**
    Add a new tree branch below Root, corresponding to the test A = a_i.
    Let Examples(a_i), be the subset of examples that have the value a_i for A
    if Examples(a_i) is empty **then**
      Below this new branch add a leaf node with label = most common target value in the examples.
    else
      Below this new branch add the subtree ID3 (Examples(a_i), Target_Attribute, Attributes - $\{$A$\}$)
    end if
  end for
  End
end if
Return Root.

Figure 2.2: ID3 decision tree algorithm.

node it checks two conditions.
(1) If all the examples are of the same class, then the algorithm just returns a leaf node of that class.
(2) If there are no attributes left with which to construct a nonterminal node, then the algorithm has to return a leaf node.
It returns a leaf node with the class with most likelihood. If none of these conditions is true, then the algorithm finds the best attributes using the selected split criteria to split the training points available at the node into different groups. For a binary split there are two groups and for a multi-way split there are more than two groups. Each group of data points form a new node. These new nodes will be added as children to the node. The decision tree algorithm is called recursively on each of these new nodes. C4.5 and CART are popular decision tree algorithms;

1. **C4.5 Decision trees -** C4.5 [80] made a number of improvements to ID3 (presented in Fig. 2.2). It can handle both continuous (see section 2.2.2 for details) and categorical attributes, whereas ID3 can handle only categorical attributes. In order to avoid overfitting, it uses pruning trees after creation - C4.5 goes back through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes, whereas ID3 decision trees have no pruning. In this thesis, we used J48 (Weka [96] implementation of C4.5) decision trees.

   Decision Stumps - A decision stump is a one level decision tree [96]. If m attributes are present in a dataset, a decision stump selects the attribute that gives the best information gain ratio.

2. **Classification and regression trees (CART) -** The basic methodology of divide and conquer, described in C4.5, is also used in CART [14]. The differences are in the tree structure, the splitting criteria and the pruning method. CART constructs trees that have only binary splits, whereas C4.5 decision trees may have multi-way node splits for multi-valued categorical attributes. CART decision trees use the Gini index as the splitting criterion whereas C4.5 decision trees uses the information gain ratio as the splitting criterion. C4.5 decsion trees and CART have different pruning methods. The time complexity of the pruning step when 10-fold cross- validation is used is a factor of 10 more expensive for CART than C4.5's pruning, but it does tend to produce smaller trees [77].

2.2.1 Splitting Criteria

The splitting criterion in the decision tree algorithm is used to test all available attributes at each decision node in the tree. The goal is to select the attribute that is most useful for classifying examples. Different split criteria have been suggested. The *information gain* and the *information gain ratio* are two popular split criteria [80]. The CART [14] procedure proposed by Breiman uses the *Gini index* as its splitting criterion.

The information gain is a popular split criterion. It is derived using information theory.

If a dataset T, with the class attribute C, contains k classes (C_i for $i = (1, 2, ..., k)$) and T is partitioned into $\sharp T_1, \sharp T_2, ..., \sharp T_{|A|}$ subsets by an attribute A with $|A|$ different values, $(a_1, a_2, ...a_{|A|})$.

The entropy of the message that identifies the class of a data point in the sample T is

$$H(C) = -\sum_{i=1}^{k} p(C_i) \log_2 p(C_i), \tag{2.1}$$

where $p(C_i) = \dfrac{|C_i|}{|T|}$.

If T is partitioned by the attribute A, the entropy is

$$H(C|A) = \sum_{v=1}^{|A|} \frac{|\sharp T_v|}{|T|} H(\sharp T_v). \tag{2.2}$$

The information gain $I(A; C)$ of a given attribute A with respect to the class attribute C is the reduction in uncertainty (entropy) about the value of C when we know the value of A,
The information gain $I(A; C)$ is defined as

$$I(A; C) = H(C) - H(C|A). \tag{2.3}$$

The information gain is biased for attributes with large number of attribute values. The information gain ratio overcomes this weakness of gain ratio by also taking into account the potential information from the partition itself. The information gain ratio penalizes the large number of branches. The gain ratio is defined as

$$\text{Gain ratio} = I(C; A)/H(A), \tag{2.4}$$

Data Point	Attribute A	Class
A	1	+
B	2	+
C	3	+
D	4	-
E	5	-
F	6	-
G	7	-
H	8	-
I	9	+

Table 2.1: Continuous data.

where H(A) is the partition entropy.

The Gini index is another popular split criterion. It is defined as

$$Gini(C) = \sum_{C_i \neq C_j} p(C_i)p(C_j). \tag{2.5}$$

If T is partitioned by the attribute A, the Gini index is

$$Gini(A) = \sum_{v=1}^{|A|} \frac{|\sharp T_v|}{|T|} \sum_{C_i \neq C_j} p(C_i|\sharp T_v)p(C_j|\sharp T_v). \tag{2.6}$$

Gini gain is defined as

$$\text{Gini gain= } Gini(C) \text{ - } Gini(A). \tag{2.7}$$

The attribute with best Gini gain is used to split the data.

2.2.2 Node splits for Continuous Attributes

For a continuous attribute, the split is selected in terms of a threshold point, creating a binary split. If an attribute A_i has numeric values, the form of the test is $A_i \leq \theta$ with outcomes true and false, where θ is a constant threshold. The θ is selected that gives the maximum information gain ratio. Possible values of θ are found by sorting the distinct values of A_i that appear in the training set, then identifying one threshold between each pair of adjacent values. If the cases in the training set have N distinct values for A_i, N-1 thresholds are considered. Fayyad and Irani [34] show that the

value θ for attribute A_i that minimizes the average class entropy for a training set must always be a value between two examples of different classes in the sequence of sorted examples. This result decreases the number of possible splits. For example in table 2.1 for the best split point we need to check only two split points; one between C and D, and the second between H and I. The point between C and D will be selected as the spit point in a C4.5 decision tree as it gives more information gain ratio.

2.2.3 Binary Split or Multi-Way Split for Categorical Attributes?

Whether there should be a *binary split* or *multi-way split* at each decision node has been a question of extensive research [56, 14, 80, 35, 10, 81]. As discussed in Chapter 2, While multi-way splits produce a more comprehensible tree, the high branching factor can lead to the data fragmentation problem, where decisions have little or no statistical support [14, 35, 10, 94].

For example in the Tennis data (Table 2.2) , if we have multi-split on the basis of the Outlook attribute, we have three branches. The number of data points in the Sunny branch is 5, the number of data points in the Rain branch is 5 and the number of data points in the Overcast branch is 4 (Fig. 2.3). However, if we create a binary split as given in Fig. 2.3, one branch for Sunny and another branch for Rain/Overcast then the number of data points in the second branch is 9. As the number of data points is large we have better estimates in the binary split case. However, creating binary split is not straightforward as there is no *intrinsic order* for categorical values.

As a multi-way split (for multi-valued categorical attributes) with the Gini index favours those with more values, CART enforces binary splits to overcome this problem. CART procedure builds binary trees; the values of the categorical attribute at the node have to be divided into two groups. If the number of attribute values is $|A|$ then the number of nontrivial binary splits is given by,

$$\frac{\sum_{k=1}^{|A|-1} \binom{|A|}{k} - 2}{2} = 2^{(|A|-1)} - 1. \tag{2.8}$$

For example, when the attribute cardinality is 3, the number of splits is 3, however, when the attribute cardinality is 10, the number of splits is 511 (Fig. 2.4). This shows that the selecting the best split by this method, with a large attribute cardinality, is quite computationally expensive. Breiman [14] shows that for two class problems the best split can be found by examining only $|A| - 1$ possibilities.

Data Point	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Table 2.2: Tennis Data.

C4.5 as proposed by Quinlan [80] uses the *information gain ratio* as the splitting criterion. C4.5 builds a binary tree for continuous data. There are two methods in C4.5 to handle multi-valued categorical attributes. In the first, it allows the multi-way split of nodes (one branch for each attribute value). In the second method, it uses a greedy approach to iteratively merge the attribute values into two groups. Ho and Scott [53] study various modifications of the grouping method, suggested for C4.5. They use different grouping criteria; the information gain ratio and a combination of χ^2 and Cramer's V [3]. They also study the effect of global (for all the dataset) and local (at the node) grouping of attribute values. Results suggest that the local methods are not consistently more accurate than the multi-way splits, and generally global methods are less accurate.

Another way to obtain a binary split for a multi-valued categorical attribute is to partition the data points using an attribute value [14, 56, 35]. In this method, all the data points with that attribute value form one group, whereas the other group is formed with the other examples. Fayyad and Irani [35] propose a binary tree hypothesis; *A top down, non-backtracking decision tree generation algorithm (i.e. using greedy search) that uses any appropriate selection measure (such as entropy or any impurity measure) to branch a single attribute pair rather than on all values of the selected attribute, is likely to lead to a tree with fewer leaves for any given dataset.* Fayyad and Irani [35] were not able to prove or disprove the stronger version of the hypothesis; "will always

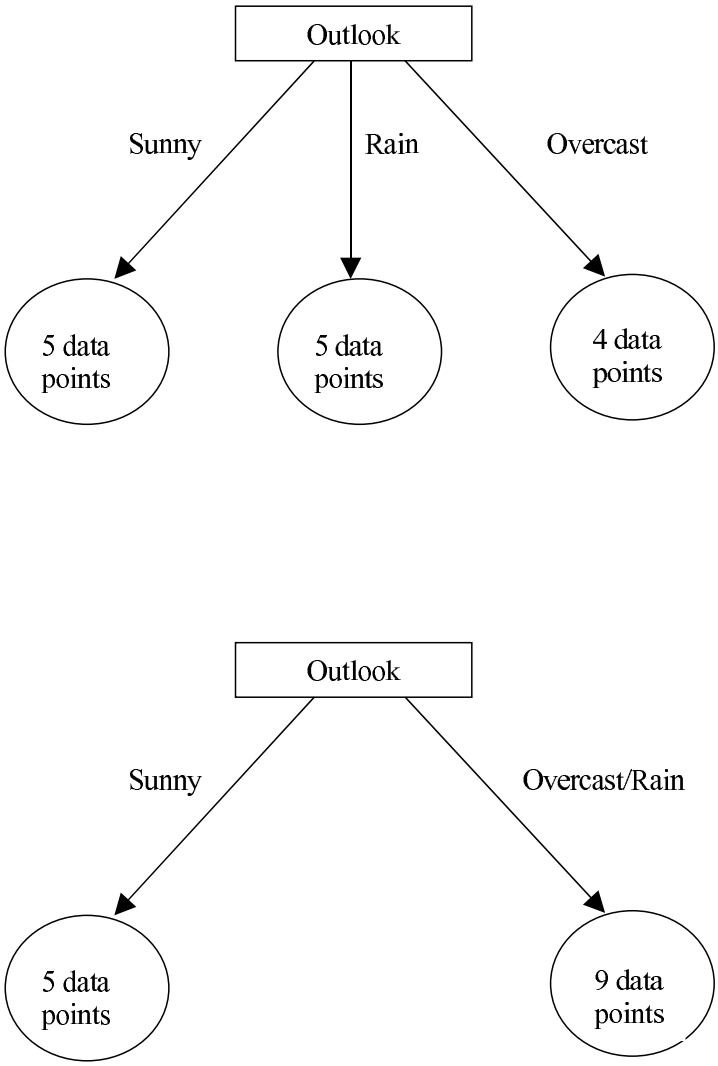


Figure 2.3: An example of a multi-way split and a binary split for the Tennis data for the Outlook attribute.

lead" instead of "is likely to lead". Kononenko [66] presents a counter example to the stronger version of this binary tree hypothesis.

Geurts et al. [47] suggest a randomized method to create binary splits from the multi-valued attributes; it draws a random subset of its possible values. As in this method, the division of attribute values is done without considering the output, the classification accuracy of the tree may be poor.

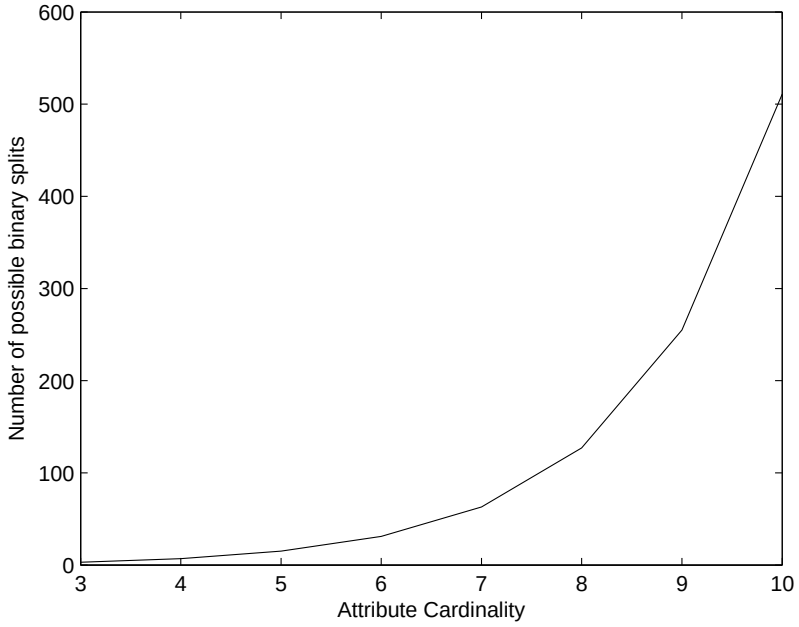


Figure 2.4: Graph for number of possible splits against the attribute cardinality.

2.3 Types of Decision Nodes

Depending upon the decision node of a tree, decision trees for continuous attributes can be divided into three categories:

1. **Univariate Decision Trees -** At each node, they take the decision on one attribute. They create orthogonal splits (orthogonal to the attributes) (Fig. 2.5). They are computationally efficient, however, if the decision boundaries are not orthogonal to a attribute's axis, they are generally not very accurate. C4.5 [80] and CART [14] are examples of these kinds of trees.

2. **Linear Multi-variate Decision Trees -** Multivariate decision trees [14, 16] overcome the representational limitation of univariate decision trees (Fig. 2.5). Linear multivariate decision trees allow the node to test a linear combination of the numeric attributes (2.5). This test can be presented as,

$$\sum_{i=1}^{m} c_i x_i \leq \theta \tag{2.9}$$

where x_i are the numeric attributes, c_i are the corresponding real valued coefficients, and θ is a numeric constant. These trees are also called oblique decision

trees as they uses oblique (non-axis-parallel) hyperplanes to partition the data. In some problem domains, multivariate decision trees performs better than univariate decision trees [59]. However, oblique trees are not very popular as it is computationally expensive to create these trees [49].

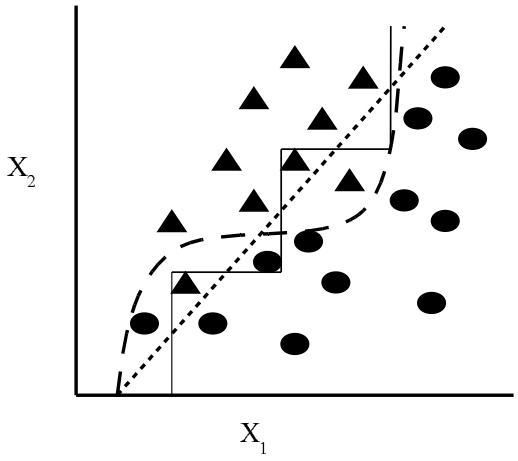


Figure 2.5: Examples of univariate(solid line), linear multivariate(dotted line) and non-linear multivariate(dashed line) splits.

Different approaches have been proposed to create oblique decision trees. Breiman et al. [14] suggest a method to create multivariate decision trees that uses a perturbation algorithm. SADT [49] uses simulated annealing to compute hyperplanes. Simulated annealing introduces an element of randomness so SADT generates a different decision tree in each run. Heath et al. [50] use these trees to create committees of decision trees. OC1 [76] improves SADT by combining deterministic hill climbing with randomization. Cantu-Paz and Kamath [19] use evolutionary algorithms to induce oblique decision trees. Ltree [44] and HOT [59] are able to define decision surfaces both orthogonal and oblique to the axes defined by the attributes of the input space. In these approaches, the axis-parallel tree inducer remains unchanged, but new oblique attributes are added. Gama [45] introduces a simple unifying framework for multivariate tree learning. This framework combines a univariate decision tree with a linear function by means of constructive induction. Decision trees derived from the framework are able to use decision nodes with multivariate tests, and leaf nodes that make predictions using linear functions.

3. **Omnivariate Decision Trees -** At each node, they can take decisions on the non-linear combination of attributes (Fig. 2.5). They have better representational

power than univariate decision trees and linear univariate decision trees. However, they are the most computationally expensive. Non-linear decision nodes can generate an arbitrarily complex decision boundary and provide the strongest discriminant power. However, these trees can be easily influenced by the noise in the data. [98, 99] are the examples of these kinds of trees.

2.4 Motivation for Classifier Ensembles

Ensembles are a combination of multiple base models [29, 63, 67, 48, 90, 79]. Final classification results depend on the combined outputs of individual models. Classifier ensembles have shown to produce better results than single models, if the classifiers, in the ensembles, are accurate and diverse[48]. A classifier is accurate, if it performs better than random guessing of the test data point class. Two classifiers are diverse if they make different errors on data points. Ensembles perform better when base models are unstable–classifiers whose output undergoes significant changes in response to small changes in the training data. Decision-trees, neural networks and rule learning algorithms are all unstable. Support vector machine and naive Bayes algorithms are generally very stable. There are many reasons for using an ensemble system:

1. **Statistical Reason** - Due to the limited amount of the data, the learning algorithm can find many hypotheses with similar training accuracy. Few of these classifiers may be poor performing (poor generalization accuracy). By constructing an ensemble out of all these classifiers, we reduce the risk of selecting a poor performing classifier because in an ensemble, we combine the outputs of all the classifiers in the ensemble [29]. For example in Fig. 2.6 (top left) the outer surface denotes hypothesis space H. Inner surface denotes all the hypotheses that give good accuracy for the given training dataset. f is the true hypothesis, one can get better approximation of the true function if the average of accurate hypotheses is taken.

2. **Computational Reason** - Learning algorithms that work by performing some form of local search, may get stuck in the local minima, so chances are that, we do not get the optimal classifier. For example, optimal training of neural networks and decision tree is NP-hard [9, 57]. If we combine the outputs of many classifiers for final decision, there is a high probability that it gives a better approximation of the true unknown function than any of the individual classifiers

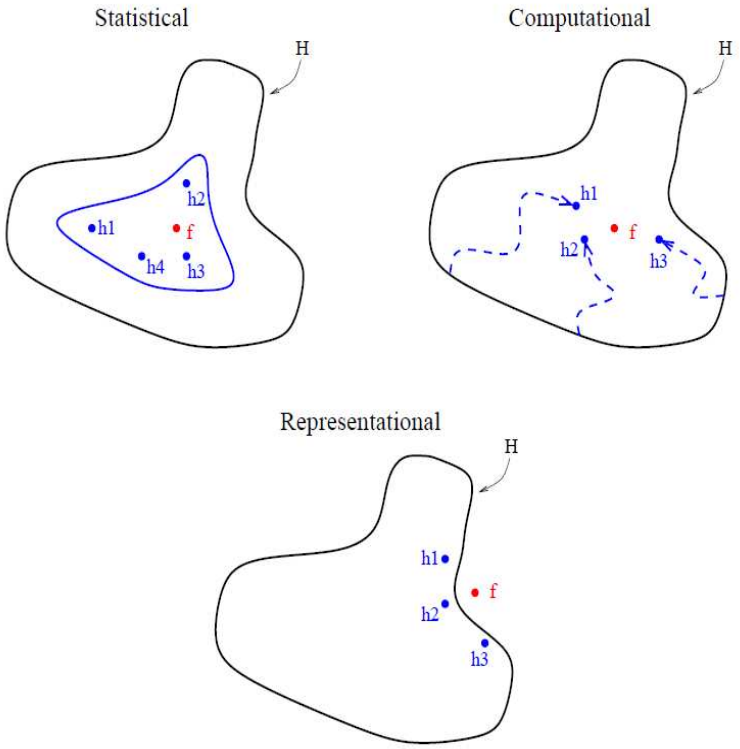


Figure 2.6: Three reasons why an ensemble works better than a single classifier. the figure is taken from [29].

(Fig 2.6 (top right)).

3. **Representational Reason** - Sometimes, decision boundaries lie outside the space of functions that can not be learned by the chosen classifier model. Weighted sum of different hypotheses can have better representational power (Fig. 2.6 (bottom)). Dietterich [29] uses C4.5 trees to explain this for the learning problem in which the true decision boundary is not orthogonal to coordinate axis (Fig. 1.1). At each node a univariate decision tree can take the decision only on the basis of a single attribute. That restricts the representational power of a decision trees. Any decision surface that is not perpendicular to the attribute axis is approximated by these decision trees. Very large decision trees can approximate these boundaries well. However, to grow a very large decision trees we need a sufficiently large dataset. The lack of a large dataset often restricts the representational power of a decision trees. Large decision trees suffer from the overfitting problem. Ensembles of decision trees generally perform better than a

single decision trees as they have better representational power (an ensemble of small decision trees act as a large decision tree [29]).

4. **Data Fusion**-If the data is coming from various sources, where the attributes are different (heterogeneous attributes), it is difficult for a single classifier to learn on this data correctly. In such cases, each dataset from one source can be used to train a different classifier.

5. **Missing Data** - The missing data is an important problem in real datasets. Ensemble learning provides an elegant solution to this problem [27]. Each of the classifiers in the ensemble is trained on a different subset of the available attribute space. Only those classifiers whose training dataset did not include the currently missing attributes are used to classify data points with missing attributes.

2.5 Theoretical Models for Classifier Ensembles

Many studies have shown the advantages of combining classifiers over the single classifiers [29, 63, 67, 48, 90]. The superior performance of an ensemble can be explained by using the bias-variance model proposed by Tumer and Ghosh [90]. They show for an ensemble that computes the average of the base models' output, if the correlations of the errors made by the base models decrease the variance of the error of the ensemble decreases. For uncorrelated base models, the variance of the error of the ensemble is less than the variance of the error of any single base model. They develop following relationship-

$$E_{add}(average) = \frac{(1 + \sigma(M-1))E_{add}}{M}, \tag{2.10}$$

where,

- E_{add} is the average additional error of the base modes, that is due to the learning with finite training dataset (beyond the Bayes error, which is the minimum possible error that can be obtained),
- $E_{add}(average)$ is the additional error of the ensemble based on the base models,
- σ is the average correlation of the errors of the base models,
- M is the number of classifiers in the ensemble.

This relationship suggests that performance of an ensemble depends on σ. If all the models are same (σ = 1), we are not getting any improvement. Whereas, when σ = 0, the added error is reduced by a factor of M, relative to base model added error. This shows that creating the diverse base models is necessary for the good performance of the ensembles.

Fumera et al. [42] suggest an analytic relationship between the expected misclassification probability of the ensemble and the expected misclassification probability of an individual classifier, as a function of the ensemble size. Their theoretical results show that the expected misclassification probabilities of Bagging [11] has the bias component as the bias component of the base model, whereas the variance component is reduced by a factor M. They also suggest that this relationship is true for all ensemble methods based on randomization. They develop the following relationship for the ensemble classification error, the following relationship suggests that as we increase the size of the ensemble, the variance part of the error (hence the total error) gets reduced.

$$E = E(B) + \frac{E(V)}{M}, \tag{2.11}$$

where,

- E is the classification error of ensemble
- $E(B)$ corresponds to the sum of the Bayes error and of the bias component of the error,
- $E(V)$ is the variance part of the error.

2.6 Methods of Constructing Classifier Ensembles

There are five popular approaches for creating classifier ensembles.

2.6.1 Changing the Distribution of Training Data Points

In this approach, each classifier, in the ensemble, is generated using a different sample of the training sets. Bagging [11] and Boosting [40, 46] are the examples of this kind of approach. This is a general approach and works with any classifier.

2.6.2 Changing the Attributes Used in the Training

In this approach, the attribute space of dataset is manipulated. Each classifier is trained on different attribute sets. These attributes may be from the training data or new attributes that are created. Random Subspaces [54] and Rotation Forests [83] are the examples of this approach.

2.6.3 Output Manipulation

In this approach, the output of the training data is manipulated to create diverse datasets. Error-correcting codes [31] and introducing noise in the output [12] are the examples of this approach.

Error-correcting codes is useful for multi-class problems. Suppose the given learning problem has K classes, then new learning problems can be created by randomly partitioning the K classes into two subsets; K_1 and K_2, where K_2 = K - K_1 and K_1 and K_2 are subsets of K classes. For example, if a dataset has 5 classes, $\{1, 2, 3, 4, 5\}$, we may have random partitions like K_1= $\{1, 3, 4\}$ and K_2 = $\{2, 5\}$. The input data is relabeled so that all the original classes in the subset K_1 are given the new label 1, whereas the original classes in the subset K_2 are given the new label 2. In other words, a multi-class problem is converted to a two-class problem. The new relabeled data is then given to a classifier algorithm that generates a classifier h_i. By repeating this process L times, we create L different classifiers (by creating L different subsets K_1 and K_2), these classifiers are combined to create an ensemble. During testing, each classifier gives a vote to a class depending upon whether the class is present in the predicted new class, for example, if the predicted class is 1 then all the classes present in K_1 will get one vote. The class with the maximum number of votes is the final class.

Breiman [12] introduces a ensemble method in which a random noise is introduced. To add classification noise at a rate r, a fraction r of the instances is randomly chosen and their class labels are changed to be incorrect, choosing uniformly from the set of incorrect labels. Diverse datasets are created with different random noise. Diverse datasets create diverse classifiers. These classifiers are combined to create an ensemble.

2.6.4 Injecting Randomness into the Learning Algorithm

This technique is popular in creating decision tree ensembles. In a decision tree, split attributes and split points are selected that optimize the splitting criterion. Different

methods have been proposed to introduce randomness in node splitting criterion. Dietterich [30] proposes an approach that randomly selects a test among the K best splits. In Random Forests [13], the split attributes are selected among the K randomly selected attributes. Extremely Randomized Trees, proposed by Geurts [47], consist of randomizing strongly both attributes and cut-points while splitting a tree node. In this approach, at each level of the tree, K attributes are randomly selected, each attribute is split randomly. Out of these K splits, the best split, based on the split criterion, is selected for that level. In this approach, the split is independent of the output (random split). PERT and Random Tress [33] are the other examples of this kind of ensembles.

2.6.5 Combination of Different Ensemble Methods

Some of the ensemble methods are based on different mechanisms like Bagging [11], AdaBoost.M1[39] and Random Subspaces [54] etc.. Whereas, some of the ensemble methods combine techniques that have different mechanisms, for example Random Forests [13] combine Bagging with Random Subspaces, MultiBoosting [95] combines Bagging with AdaBoost and Rotation Forest [83] combines randomization in the attribute space division with Bagging. The basic idea behind these ”hybrid” ensemble techniques is that as the mechanisms differ for different ensemble methods, their combination may out-perform either in isolation.

2.7 Some Popular Ensemble Methods

In this section we discuss some of the popular classifier ensemble methods in detail.

2.7.1 Bagging

Bagging (Bootstrap Aggregation) [11] generates different bootstrap training datasets from the original training dataset and uses each of them to train one of the classifiers in the ensemble. For example, to create a training set of N data points, it selects one point from the training dataset, N times without replacement. Each point has equal probability of selection. In one training dataset, some of the points get selected more than once, whereas some of them are not selected at all. Different training datasets are created by this process. When different classifiers of the ensemble are trained on different training datasets, diverse classifiers are created. Bagging does more to reduce the variance part of the error of the base classifier than the bias part of the error.

2.7.2 Boosting

Boosting [40, 46] generates a sequence of classifiers with different weight distribution over the training set. In each iteration, the learning algorithm is invoked to minimize the weighted error, and it returns a hypothesis. The weighted error of this hypothesis is computed and applied to update the weight on the training examples. The final classifier is constructed by a weighted vote of the individual classifiers. Each classifier is weighted according to its accuracy on the weighted training set that it has trained on.

The key idea, behind Boosting is to concentrate on data points that are hard to classify by increasing their weights so that the probability of their selection in the next round is increased. In subsequent iteration, therefore, Boosting tries to solve more difficult learning problems.

Boosting reduces both bias and variance parts of the error. As it concentrates on hard to classify data points, this leads to the decrease in the bias. At the same time classifiers are trained on different training data sets so it helps in reducing the variance. Boosting has difficulty in learning when the dataset is noisy. In each iteration, the weights assigned to the noisy data points increases so in subsequent iteration it concentrates more on the noisy data points; it leads to overfitting of the data. AdaBoost.M1 [40, 39], AdaBoost.M2 [39], Arcx4 [15] are some of the examples of boosting algorithms. Breiman [15] shows that the process of selecting data samples (concentrating on hard to classify data objects) is the reason of the better performance of classifier ensembles, not the method by which the data set is produced. To show this point, a different procedure is used to make data samples such that the hard to classify data points get more weight. Similar results as produced by AdaBoost.M1 are achieved with this procedure.

2.7.3 MultiBoosting

MultiBoosting [95] combines Adaboost.M1 with wagging (a kind of Bagging). It combines Adaboost's high bias and variance reduction with wagging's high variance reduction. It is based on the observation that most of the performance advantage of an ensemble is due to the first few members. It forms subcommittees by using an Adaboost algorithm. Different starting datasets for different subcommittees are created by using the wagging method. To create an ensemble of the size L, u subcommittees of the size v are created; where $uv = L$. As the wagging method (a kind of Bagging) creates different datasets, u different datasets are created by using the wagging

method. Then an Adaboost algorithm, for the ensemble size v, runs on each of these datasets independently. The difference between Adaboost and Multiboost is that in Adaboost, for the ensemble size L, data weights are initialized (all initial weights are taken $1/n$) once, whereas, in Multiboost, after each v iterations, a new dataset (created by the wagging method) is used and data weights are initialized (all initial weights are taken $1/n$). *In other words, in Multiboost, Adaboost run multiple times, but each time a different dataset (created by the wagging method) is used*. Decisions of all subcommittees are combined to get the final decision. Using C4.5 as the base learning algorithm, Multiboosting is demonstrated [95] to produce decision committees with lower error than either AdaBoost or wagging significantly more often than the reverse over a large representative cross-section of UCI data sets. Experimental results suggest that MultiBoost achieves most of the bias reduction of AdaBoost together with most of the variance reduction of wagging. MultiBoost offers a potential computational advantage over AdaBoost in that it is amenable to parallel execution. Each subcommittee may be learned independently of the others.

2.7.4 Random Subspaces

In this method [54], the classifiers of an ensemble are trained on the different sets of attributes of training data. As different classifiers are trained on the different sets of attributes, diverse classifiers are created. Ho [54] suggests that this method is more effective when datasets have large numbers of irrelevant attributes.

2.7.5 Dietterich's Random Trees

Dietterich [30] proposes a method to grow randomized trees for an ensemble that consists of random trees. This method introduces randomization while splitting the nodes of decision trees. Instead of selecting the best split, it selects a test among the best *K* tests.

2.7.6 Random Forests

Breiman [13] combines Bagging with Random Subspace method to create Random Forests. To build the tree, it uses a bootstrap replica of the training sample. During the tree growing phase, at each node the optimal split is derived by searching a random subset of size *K* of the candidate attributes. As this method combines two random

processes, it is able to produce diverse trees. It compares favourably to Adaboost but is more robust with respect to the class noise.

2.7.7 Extremely Randomized Trees

Geurts et al. [47] proposed *Extremely Randomized Trees* (ERT). It consists of randomizing both attribute and cut-point choice while splitting a tree node. It takes a decision of cut-point without considering the outputs. At every level, they select *K* attributes. It takes decision of cut-points without considering the outputs. These *K* cut-points are then evaluated to select the best split. In the extreme case (*K*=1), it builds totally randomized trees whose structures are independent of the output values of the learning sample. Every tree is trained on a full dataset. Experiments show, that it performs similar to or better than other randomization based ensemble methods. Besides accuracy, the other strength of the resulting algorithm is good computational efficiency. The geometrical analysis has shown that Extra-Trees asymptotically produce continuous, piecewise multi-linear functions. Bias/variance study suggests that randomization increases bias and variance of individual trees, the part of the variance due to randomization can be cancelled out by averaging over a sufficiently large ensemble of trees.

Extreme randomized trees have been used to create coding for image classification [75]. Experiment results suggest that they provide more accurate results and much faster training and testing as compared to traditional methods.

2.7.8 The Random Oracle Framework

Kuncheva and Rodriguez [68, 84] propose classifier ensembles with random linear oracle. In this method, every classifier in the ensemble is replaced by a pair of classifiers. These two classifiers learn on different subspaces decided by a random hyperplane. In the RLO framework, the hyperplane (random linear oracle) is generated by taking two random points A and B from the training set and calculating the hyperplane perpendicular to the line segment between the points A and B, and running through the middle point (Fig. 2.7). The location of the hyperplane is between these two points that ensures that there will be at least one point on the either side of the hyperplane. While testing, first the position of the testing data point is decided and the decision of the classifier is used that is trained in that subspace.

They also suggest random spherical oracle [84]. In this method, a space is divided

into two regions: inside and outside a hypersphere in a random subspace. The procedure for selecting the sphere is:

- Draw a random feature subset containing at least 50% of the features.
- Select a random training instance as the centre of the sphere.
- Find the radius of the sphere as the median of the distances from the centre to S randomly selected training instances. (For no specific reason, $S = 7$ is used)

The selection of a feature subset is used to increase the diversity of the oracles (and therefore, of the random oracle classifiers). If the distances are always the same for a pair of objects, two close objects would be in the same subspace for the majority of random oracles. Authors argue that the effect of using such feature subset in is that the distance between two objects can be different for different oracles.

Two reasons are suggested for the success of the random oracle approach.

1. As linear oracle splits the space into two subspaces, the classification task is easier, that may lead to the better classification accuracy (for a pair of classifiers) than the classifier trained on complete space.
2. The second reason is that with random linear oracle, diverse classifiers are created.

Kuncheva and Rodriguez [68, 84] present experiments with decision trees ensembles and naive Bayes ensembles. For decision tree ensembles, the random linear oracle ensemble method itself is not a very strong method. However, when different ensemble methods are examined with and without the oracle, the results suggest that all ensemble methods benefited from the new approach, most markedly so random subspaces and bagging.

They also study with naive Bayes classifiers, their experiments consider two random oracles types (linear and spherical). Experiments show that ensembles based solely upon the spherical oracle (and no other ensemble heuristic) outperform Bagging, Wagging, Random Subspaces, AdaBoost.M1, MultiBoost and Decorate. Moreover, all these ensemble methods are better with any of the two random oracles than their standard versions without the oracles.

Peterson and Coleman [78] suggest Principal Direction Linear Oracle (PDLO) in which the hyperplane is learned such that it (the hyperplane) maximizes the separation of samples between the pair of miniclassifiers. The hyperplane is based on rotations

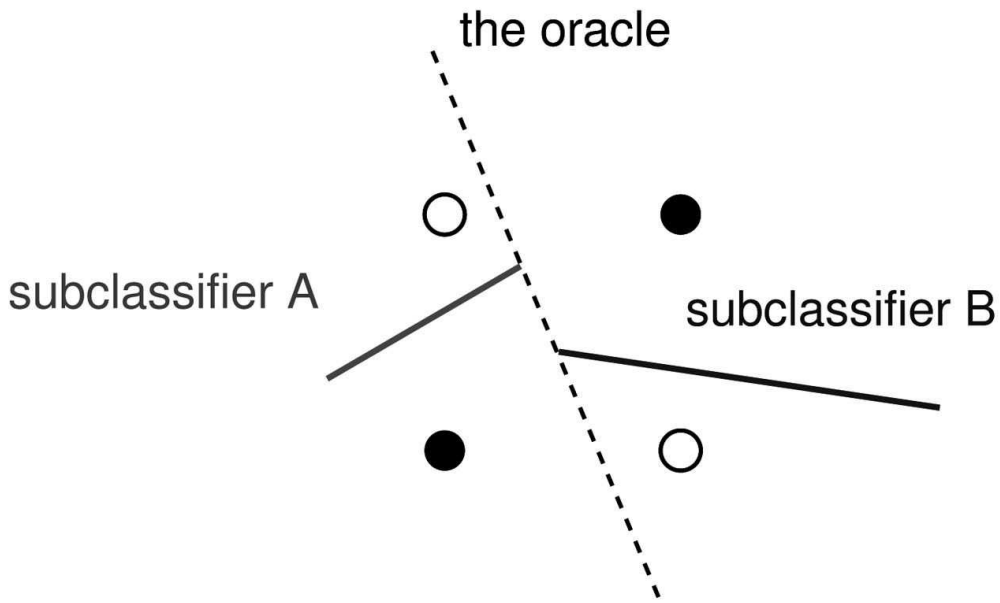


Figure 2.7: XOR classification problem and its solution using a linear oracle and two linear subclassifiers [68].

of principal components extracted from sets of filtered attributes. The motivation for PDLO is that if a standardized data set primarily consists of two largely separated clusters, then by theory the first eigenvector associated with the largest eigenvalue will form a straight line connecting the centres of the two clusters, since the two clusters will define the greatest amount of variation in the data.

2.8 Conclusion

Decision trees are very popular classifiers, however, they are not very accurate. Decision tree ensembles generally produce better results than a single decision tree. Several ensemble methods have been proposed. The core idea of these methods is to produce accurate and diverse decision trees. The balance of accuracy and diversity is the key to success of an ensemble method. In the next chapter, we study various transformation techniques and their applications in creating classifier ensembles.

Chapter 3

Data Transformation Techniques

In the previous chapter, we studied different classifiers and several ensemble methods. This chapter discusses various data transformation techniques. Principal component analysis and random projection techniques and their applications in machine learning is presented in detail. The discretization process is also introduced is this chapter. Several popular discretization methods are also presented in this chapter. In the end, we present the use of different data transformation techniques in creating classifier ensembles.

3.1 Different Data Transformation Techniques

A pattern is represented by attributes. A data transformation is a process by which representation of the pattern is changed by creating new attributes. These new attributes may be a subset of the original attributes or derived by using the original attributes. There are different data transformation techniques that are used for different reasons for example PCA [88, 87] is used to reduce the number of attributes required to represent the pattern. Random projection [23], discretization [32] etc. are other data transformation techniques that are quite popular in the pattern recognition field. We will discuss these techniques in detail.

Kernel machines [93, 17] solve the pattern recognition problems in high dimensional attribute spaces that are derived by using the original attributes. Support vector machines [93, 17] are successful because they use "kernel-trick" which allows kernelized algorithms to operate in high dimensions without incurring a corresponding cost. The idea behind this is that if the data is not linearly separable in the original attribute space, kernel methods may be able to find a linear separator in a high dimensional

space. In kernel machines, the data transformation is not done explicitly, however, the similarity between two data points are computed in high dimensional attribute space that are derived by using the original attributes.

3.2 Principal component analysis (PCA)

PCA is widely used in signal processing, statistics, and image compression [88, 61, 87]. PCA is used to transform a high dimensional data into a low dimensional data with a small loss of information. It transforms a number of possibly correlated attributes into a smaller number of uncorrelated attributes called principal components (for an example see Fig. 3.1). These new attributes are linear combinations of the original attributes. These new attributes are determined by the eigenvectors of the covariance matrix. Each eigenvalue indicates the portion of the variance that is correlated with each eigenvector. Thus, the sum of all the eigenvalues is equal to the sum squared distance of the points with their mean divided by the number of dimensions. Generally the first few principal components account for the majority of the observed variation.

Following assumptions are behind PCA when these assumptions are not correct PCA may perform poorly [88, 87],

1. **Linearity -** Linearity frames the problem as a change of basis. In other words, it asssumes that new important attributes are linear combinations of the original attributes.

2. **Large variances have important structure -** This assumption suggest that principal components with larger associated variances represent interesting structure, while those with lower variances represent noise. This is a strong, and sometimes, incorrect assumption.

3. **The principal components are orthogonal -** This assumption provides an intuitive simplification that makes PCA soluble with linear algebra decomposition techniques.

Kernel principal component analysis [86] is a method for performing a nonlinear form of principal component analysis. It combines the philosphy of PCA with kernel-trick. In experiments comparing the utility of kernel PCA features for pattern recognition using a linear classiffier, Schölkopf et al. [86] shows two advantages of nonlinear kernel PCA: first, nonlinear principal components afforded better recognition rates than corresponding numbers of linear principal components; and second, the

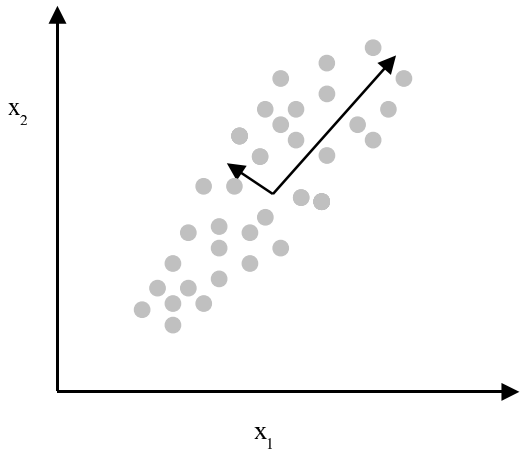


Figure 3.1: A two dimensional dataset, the variation for this data is in different directions (principal components) and not in the natural directions.

performance for nonlinear components can be further improved by using more components than possible in the linear case. In linear PCA, one can find at most d (number of attributes) nonzero eigenvalues, whereas, kernel PCA can find up to n (number of data points) nonzero eigenvalues. Thus, this is not necessarily a dimensionality reduction.

3.3 Random Projection (RP)

In this section, we discuss random projection and its applications in machine learning. RP is a technique of mapping a number of points in a high-dimensional space into a low dimensional space with the property that the Euclidean distance of any two points is approximately preserved through the projection.

In RP, the original data is projected onto a lower dimensionality subspace using a random matrix whose columns have unit lengths. Using matrix notation where $D_{m\times n}$ is the original set of n, m dimensional observations, the projection of the data onto a lower d-dimensional subspace is defined as

$$D^{RP}_{d\times n} = R_{d\times m} D_{m\times n} \tag{3.1}$$

where $R_{d\times m}$ is a Random Matrix and $D^{RP}_{d\times n}$ is the new $d \times n$ projected matrix.

The entries of the matrix R can be calculated using the algorithm presented in Fig. 3.2. The main reason for orthogonalizing the random vectors is to preserve the similarities between the original vectors in the low-dimensional space. However, in

Random Matrix R for RP
1. Set each entry of the matrix to an i.i.d. N(0; 1) value.
2. Orthogonalize the d rows of the matrix using the Gram-Schmidt algorithm.
3. Normalize the rows of the matrix to unit length.

Figure 3.2: A method to create Random matrix for RP.

high-dimensional spaces, there exist a much larger number of almost orthogonal vectors than orthogonal vectors. Thus, high-dimensional vectors having random directions are very likely to be close to orthogonal [51]. Hence, it is possible to save computation time by avoiding the orthogonalization step without affecting much the quality of the projection matrix.

Achlioptas [1] show that the Gaussian distribution can be replaced by a much simpler distribution such as

$$r_{ij} = \pm\sqrt{3} \text{ with probability 1/6 each or 0 with 2/3 probability.} \tag{3.2}$$

$$r_{ij} = (\pm 1) \text{ with probability 1/2.} \tag{3.3}$$

The key idea behind RP is Johnson and Linden Strauss theorem [60, 24] which states that if points in a vector space are projected onto a randomly selected subspace of suitably high dimension, then the distances and relative angles between the points are approximately preserved. To decide the dimension d of the projected data for the practical applications is an open problem as Fern and Brodley [36] suggest "*to our knowledge it is still an open question how to choose the dimensionality for a random projection in order to preserve separation among clusters in general clustering algorithms.*" However, Dasgupta [22] shows that the data from a mixture of K Gaussians can be projected into just $O(\log K)$ dimensions while retaining the approximate level of separation between clusters. This projected dimension is independent of the number of data points and of their original dimension. Dasgupta [22] also concludes that RPs result in more spherical clusters than those in the original dimension. This is important because raw high-dimensional data can be expected to form very eccentric clusters. and he combines RP with the Expectation Maximization (EM) algorithm and applies it to a hand-written digit dataset, achieving good results. Indyk and Motwani

[58] apply RP to the nearest neighbor problem. This leads to an approximate algorithm with polynomial preprocessing and query time polynomial in d and $\log n$ for a d-dimensional Euclidean space. Achlioptas et al. [2] suggest RP as a way of speeding up kernel computations in methods such as kernel PCA.

Franklin and Madigan [38] report a number of experiments to evaluate RP in the context of inductive supervised learning. They also compare RP and PCA on a number of different datasets and using different machine learning methods. In these experiments, datasets are projected into lower dimensional datasets and experiments are carried out to see how different classifiers behave on these low-dimensional datasets. Dimensions of these new spaces are varied and the performance of different classifiers are studied on these new spaces (created by RP or PCA). It is expected that as the dimension of the new space is increased, the performance of the classifier on the new space will approach the performance of classifier on the original space because as the dimension of the new space is increased, the information loss due to the data transformation (RP or PCA) is reduced.

They select nearest neighbour (NN) method [8], C4.5 decision trees [80] and linear SVM [93] for their experiments. They find that nearest neighbor methods are least affected by reduction in dimensionality through PCA or RP - their performance deteriorates less than that of C4.5 or of SVM. Interestingly, in some cases, PCA projection into a low dimensional space actually improves the performance of nearest neighbour methods. As compared to SVM and decision trees, NN methods performance, with RP, approach those in the original space quite rapidly as the dimension of the space, created by RP, increases. Such behaviour of NN methods is to be expected since they (NN methods) rely on distance computations for their performance and are not concerned with separation of classes or informativeness of individual attributes. Thus one might expect that NN methods would stand to benefit most from RP. Both RP and PCA adversely affect the performance of SVM. While PCA does outperform RP at lower dimensions, the difference diminishes as the dimensionality of projections increases. For some datasets, C4.5 does very well with low-dimensional PCA projections, but its performance deteriorates after that and doesn't improve. Its performance with RP is also poor. Authors argue that since decision trees rely on informativeness of individual attributes and construct axis-parallel boundaries for their decision, they don't always deal well with transformations of the attributes. They suggest that "Random Projections and decision trees might not be a good combination."

Hegde et al. [52] claim that for a wide variety of machine-learning algorithms, the

performance of these algorithms when given access to only a randomly projected version of a dataset is essentially the same as its performance on the original dataset. This suggests that with only a low-dimensional, easily obtainable representation, these machine learning algorithms can achieve similar performance as with the high-dimensional dataset. In other words, random projections can be used as a universal, inexpensive preprocessing step to many machine learning tasks. However, it is not precisely clear how the presence of data noise affects the learning performance in the compressed domain.

Random projections have been useful for creating cluster ensembles [36, 91, 7]. In this method a dataset is projected into new data space by using random projections and a cluster algorithm is used on this new data. Various random projections give diverse datasets, hence different clustering results are obtained by using these datasets. These results are combined to get the final result. Empirical results [36, 7] suggest that these cluster ensembles achieve better and robust results as compared to single runs of clustering algorithms.

3.4 Discretization

Discretization [32] is a process that divides continuous numeric values into a set of intervals that can be considered as categorical values. Discretization is used for two main reasons:

1. **Accuracy -** Many classification algorithms work well for nominal data, whereas the data at hand might be purely continuous. The discretization process may improve the accuracy of classification algorithms. For example, the naive Bayes classifier requires the computation of the conditional probabilities of the classes given the example. For the categorical attributes this can be computed with frequencies obtained from the training data. For the continuous attributes, an assumption about the data distribution is needed. Generally, the normal distribution is assumed. When this assumption is not true, the naive Bayes classifier performs poorly. Dougherty et al. [32] show that the performance of naive Bayes algorithm significantly improves when the continuous attributes are discretized using the entropy-based methods. Yans and Webb [97] also present those similar findings; that the discretization process is helpful for naive Bayes classifiers. Dougherty et al. [32] also show that in some cases, the performance of the C4.5 decision tree induction algorithm significantly improves with discretized attributes.

	Global	Local
Supervised	1RD Fayyad and Irani Supervised MCC D-2 Adaptive Quantizers	Vector Quantization Fayyad and Irani C4.5
Unsupervised	Equal Width Discretization Equal Frequency Discretization Unsupervised MCC	k-mean clustering

Figure 3.3: Summary of Discretization Methods [32].

2. **Computational Complexity -** The discretization process improves the speed of the tree induction process. At each node, to split the node, all available split points are considered to get the best split point. If possible split points are few, the computation time to decide the best split point will be small. The discretization process is used to reduce the number of possible spilt points. The use of histograms, to approximate the split at a node, has been proposed to reduce the time to create a decision tree using a very large dataset [20]. For each attribute, instead of sorting the instances at a node, a histogram is created, and bin boundaries are used as potential splits. Since no sorting is done and fewer potential split points are evaluated, it takes less time to create a tree using histograms. The proposed tree may have less accuracy as split points have been approximated.

3.4.1 Discretization Methods

Dougherty et al. [32] define three axes upon which discretization methods can be classified; global vs. local, supervised vs. unsupervised and static vs. dynamic. Supervised methods use the information of class labels whereas, unsupervised methods do not. Local methods as the one used in C4.5, produce partitions that are applied to localized regions of the instance space. Global methods are applied to the entire dataset. In static methods attributes are discretized independently of each other, whereas, dynamic methods take into account the interdependencies between them. Equal width intervals, equal frequency intervals and unsupervised monothetic contrast criteria (MCC) [26] are unsupervised methods. Whereas, discretization methods based on entropy (supervised MCC [26], entropy minimization discretization [34], D-2 [20]), 1DR [55], adaptive quantizers [21] and Vector Quantization [65] etc. are supervised methods.

Data Point	Attribute A	Class
A	1	+
B	2	+
C	3	-
D	4	-
E	5	-
F	6	-
G	7	-
H	8	-
I	9	+
J	10	+
K	11	+
L	12	+
M	13	+
N	14	+
O	15	-
P	16	-

Table 3.1: A continuous dataset. We present discretization of this dataset by different methods.

Equal width intervals and equal frequency intervals are global methods. Whereas, the discretization used in C4.5 decision tree growing phase and Vector Quantization are local methods. All these methods are static methods. Fig. 3.3 shows the detailed classification of different discretization methods.

Dynamic methods are a promising area of research. As these methods are able to capture interdependencies between attributes, it may improve the accuracy of decision rules [69]. Kwedlo and Kretowski [69] show that static methods (that do not capture interdependencies) run the risk of missing information necessary for correct classification. Following are some popular discretization methods,

1. **Equal Width Discretization (EWD) -** Equal Width Discretization (EWD) is a simple and popular discretization algorithm. EWD [32] divides the feature values into equal sized bins. Hence, for K bins, the bin boundaries are x_{min} + w, x_{min} + 2w,..., x_{min} + (K-1)w, where w = (x_{max} - x_{min})/K. This is a an unsupervised method, which does not take into account the information of class labels. In the given example (table 3.1), there are 16 points. In order to create 4 bins, different boundaries are created by the equal width method. The width of the bin is (16 - 1)/4 = 3.75. Hence, bin boundaries are 1, 1 + 3.75, 1 + 2$\times$3.75 and 1 + 3$\times$3.75. By using these boundaries, four bins will be (A, B, C, D), (E,

F, G, H), (I, J, K, L) and (M, N, O, P).

2. **Equal Frequency Discretization (EFD) -** EFD [32] divides the sorted values into K bins so that each bin contains approximately the same number of training instances. In other words, each bin contains n/K (possibly duplicated) adjacent values. As there are 16 data points in the given example (table 3.1), for 4 bins, each bin will have 16/4 = 4 points, hence, four bins will be (A, B, C, D), (E, F, G, H), (I, J, K, L) and (M, N, O, P).

3. **Entropy Minimization Discretization -** Fayyad and Irani [34] use recursive entropy minimization for discretization. For evaluating each candidate bin boundary, the attribute is discretized into two bins and the resulting class information entropy (Eq. 2.3) is calculated. A binary discretization is determined by selecting the bin boundary for which the entropy is minimal amongst all candidates. This method then is applied to both the bins recursively until some stopping criterion is achieved. A minimum description length criterion (MDL) is applied to decide when to stop the recursive process. It is a supervised method as it uses class labels to compute the entropy of different partitions. First the data, given in the example (table 3.1) , is divided into two bins such that these two bins have minimum entropy, hence, the first split will be between H and I points. The new two bins will be further divided, the first bin will be divided between B and C points and the second bin will be divided between N and O points to have minimum entropy. Hence, four bins are (A, B), (C, D, E, F, G, H), (I, J, K, L, M, N) and (O, P).

4. **Error-based Discretization -** Maass [71] proposes an algorithm to discretize a continuous feature with respect to error on the training set. This algorithm discretizes a continuous feature by producing an optimal set of K or fewer intervals that results in the minimum training error if the instances were to be classified using only that single feature after discretization. We will have four bins for the example (table 3.1). Four bins are (A, B), (C, D, E, F, G, H), (I, J, K, L, M, N) and (O, P), as with these bins we have zero training error.

5. **1R discretizer -** Holte [55] proposes a one-level decision tree (also called a decision stump). This is used to create disctetization. In this method, the observed values of a continuous feature is sorted and divided into different bins such that each bin contains only the instance of a particular class. As, this procedure can

lead to the same number of bins as the number of data points; only one data point in each bin, this algorithm is constrained to have minimum number of data points in each bin (except the rightmost bin). For the data given in the example (table 3.1) all adjacent points with same class will be combined. Hence, four bins (A, B), (C, D, E, F, G, H), (I, J, K, L, M, N) and (O, P) are created.

6. **D-2 Discretizer -** Catlett [20] explores the use of discretization to improve the speed of decision tree algorithm for datasets with large number of continuous attributes. They use an entropy-based method for discretization. This method uses several criteria to stop the recursive partitioning of each attribute: the maximum number of bins, the minimum number of data points in each bin, the minimum information gain.

7. **Monothetic Contrast Criteria (MCC) -** Van de Merckt [26] proposes two methods under the general heading of Monothetic Contrast Criteria (MCC). The first criterion, makes use of an unsupervised clustering algorithm that seeks to find the partition boundaries that "produce the greatest contrast" according to a given contrast function. An unsupervised monothetic contrast criterion is defined in following way,

$$Contrast(N_1; N_2; A) = \frac{N_1 N_2}{N_1 + N_2}(m_{A1} - m_{A2})^2 \tag{3.4}$$

where N_1 and N_2 are number of instances of the resulting binary split, m_{A1} is the mean value for attribute A of N_1 instances and m_{A2} is the mean value for attribute A of N_1 instances.

In the given example (table 3.1) the best contrast will be between H and I points. The second method, that combine supervised and unsupervised methods redefines the objective function to be maximized by dividing the previous contrast function by the entropy of a proposed partition.

3.4.2 Effect of the Discretization Process on Different Classifiers

Dougherty et al. [32] study the effect of different discretization methods (as a prepossessing process) on naive Bayes classifiers and C4.5 decision trees. All the discretization methods improve the naive Bayes classifier. However, this improvement is

more for entropy-based discretization methods. As discussed earlier, for continuous attributes, a naive Bayes classifier assumes Gaussian distribution which is not valid for many domains. A discretization process helps overcome the normality assumption, hence, improves the performance of naive Bayes classifiers.

The study with C4.5 decision trees suggests that the discretization process has not much effect on C4.5 classifiers. Authors suggest that no degradation in the performance with the global entropy-based discretization process indicates that the C4.5 algorithm is not taking full advantage of local discretization methods for the datasets tested.

Kohavi and Sahami [64] compare an error-based discretization method [71] with an entropy-based discretization method (with minimum description length heuristic) [34] for naive Bayes classifiers and C4.5 decision trees. Results suggest that the entropy-based discretization method generally outperforms the error-based discretization method. Authors carried out analysis with simulated data to understand this behaviour. They showed that error-based discretization methods have inherent limitation which allows a less number of partitions whereas there is no such limitation for entropy-based methods and can partition the space as long as the class distribution between the different partitions is different. Their analysis suggest that the entropy-based discretization method might perform better because of the feature interaction; as long as the distribution is different, a threshold will be formed, allowing other features to make final discrimination. In other words, the error-based discretization method is inappropriate in cases where features interact.

3.5 Data Transformation in Classifier Ensembles

Random subspaces [54] (RS) is a popular ensemble method in which each classifier is trained on a random subset of the attributes. In other words, for every classifier a pattern is represented by a subset of original attributes. For the dataset with a small number of attributes, Ho [54] suggests that new attributes may be created by taking the random linear combinations of the attributes and the RS technique can be applied on the dataset that is the combination of original attributes and the new attributes. In Rotational Forests [83], new attributes are created by using PCA technique. In one of the variants of Random Forests [13], Breiman defines new attributes by taking the random linear combinations of the attributes. Kamath et al. [62] introduces randomization by discretizing continuous attributes. This method uses histograms to determine the split

Ensemble Method	Data transformation method	Classifier
Random Subspace [54]	A subset of attributes	Decision trees
Random Forests [13]	A subset of attributes at nodes	Decision trees
Rotational Forests [83]	PCA	Decision Trees
Kamath et al. [62]	Discretization at nodes	Decision trees
Schclar and Rokach [85]	Random Projection	KNN

Table 3.2: Different ensemble methods that use data transformation.

at each node of a decision tree. It evaluates the score only at the bin boundaries, after selecting the best bin boundary, it selects the split point randomly in some interval around the best interval boundary. This way diverse decision trees are created. Cai and Wu [18] propose the use of the discretization process to create ensembles of support vector machines. This algorithm uses the rough sets and boolean reasoning approach to construct diverse classifiers. Schclar and Rokach [85] introduce an ensemble method by using random projections. In this method, new datasets are created by using random projections. Classifiers learn on these new diverse datasets are diverse, hence diverse classifiers are created. Authors use nearest-neighbour classifier as the base classifier to show that this method outperform the Bagging algorithm. However, no comparison with more complex ensemble methods like AdaBoost.M1 is presented. The summary of these methods are provided in table 3.2.

Generally in most of the ensemble methods a subset of features is used, hence, it would be interesting to study those ensemble methods that use extended feature spaces. Discretization has been used for ensemble methods, however, there has been no research for using global discretization methods for ensemble methods.

3.6 Conclusion

Several data transformation techniques were presented in this chapter. Principal component analysis, random projection and discretization techniques and their applications in machine learning were discussed in detail. One can conclude from the present research trend that RP is very active research area as it is quite useful in transforming high-dimensional space into low dimensional space. RP can also be useful for creating ensembles as it creates different representations of same problem that can be used to create diverse classifiers.

Chapter 4

A Study of Random Linear Oracle Framework and Its Extensions

In the previous chapter, we studied about random projections. In this chapter, first we present a method that uses random projections and a univariate decision tree algorithm to create ensembles of linear multivariate decision trees. In the second part of this chapter, we study the random linear oracle (RLO) framework [68, 84]. We propose two new variants of the random linear oracle approach (Learned-RLO and Multi-RLE) that extend the philosophy of the RLO approach. We then show that the method proposed to create ensembles of linear multivariate decision trees is the generalization of the RLO framework.

4.1 Diverse Linear Multivariate Decision Trees

As discussed in Chapter 2, linear multivariate decision trees have better representational powers as compared to univariate decision trees, however, they are more computationally expensive. Linear multivariate decision trees can have decisions at each node on the basis of linear combinations of attributes. Generally, the possible linear combination of attributes is learnt at each node [59, 16, 49, 44] this makes the learning process computationally expensive. The other possible method could be that we generate new attributes that are linear combinations of the original attributes, then we can use these attributes with univariate decision tree algorithms. The final decision trees will be multivariate decision trees in the original attribute space. As discussed in Chapter 2, there are two popular methods: (PCA and random projections) that create good linear combinations of attributes (both create small number of linear attributes

with minimal loss of information). We propose a method to create ensembles of linear multivariate decision trees that uses random projections. In the present method, we use RP for the following reasons;

1. **Datasets created by using random projections create linear multivariate decision trees -** RP project the original data to a new attribute space. These attributes are linear combinations of the original attributes. *We concatenate these new attributes with the original attributes*. As discussed in Chapter, 2 RP has been used for classifier ensembles [85] and cluster ensembles [36, 91, 7]. *No one has used the extended space as proposed by us for ensembles*. If we induce a univariate decision tree on this concatenated data, we get a linear multivariate tree in the original attribute space. The main difference between this approach of creating oblique decision trees and other approaches discussed in Chapter 2, is that in this approach, the orientations of non-orthogonal hyperplanes are fixed (that depend on new attributes), only the locations of these hyperplanes are learned during the tree growing phase whereas in the other approaches both the locations and the orientations of the non-orthogonal hyperplanes are computed during the tree growing phase. Experiments suggest that C4.5 trees do not perform well with random projections [38]. One of the possible reasons for this is that the orientations of the hyperplanes are fixed that so limiting the representational power of decision trees. Fradkin and Madigan [38] suggest "Random Projections and decision trees are perhaps not a good combination". However, in the present approach, we concatenate new attributes with the original attributes, finding the technique extremely profitable.

2. **Different random projections create different datasets, hence we get diverse decision trees after learning on these datasets -** To build a classifier ensemble, we need diverse decision trees. RP helps in creating these diverse decision trees. Different random projections of a dataset create different new datasets (a dataset is created by concatenating new attributes with the original attributes). If we train univariate decision trees on these new datasets, we get diverse decision trees that are linear multivariate decision trees in the original attribute space. We can combine these trees to get an *ensemble of linear multivariate decision trees*.

In summary, each tree of the ensemble has decision surfaces defined by the original attributes and the linear combinations of original attributes (created using RP) and each

Input- Original dataset *T* with m continuous attributes and k classes.
M the size of the ensemble.
Training Phase
for i=1...M **do**
Data Generation
a - Use random projection matrix (R_i) to create *d* dimensional dataset R_i.
b - Combine *T* and R_i datasets to get *m* + *d* dimensional dataset T_i.
Learning Phase
Treating dataset T_i as continuous, learn decision tree D_i on it.
end for
Classification Phase
For a given data point **x**
for i=1...M **do**
a - Convert **x** into *m* + *d* dimensional data point $\mathbf{x_i}$ using the original attribute and random projection matrix (R_i).
b - Find the decision of D_i.
end for
Combine the decisions of all the selected classifiers by the chosen combination rule (we use majority voting scheme).

Figure 4.1: Algorithm for ensembles of linear multivariate decision trees.

tree has different decision surfaces (RP creates different attributes), hence these trees may produce good ensembles. The method is presented in Fig. 4.1.

Creating new attributes and augmenting with original attributes has been suggested by Balcan and Blum [5]. However, their proposal is only for a single classifier whereas our proposed method creates classifier ensembles. Balcan and Blum [5] use similarity functions to create new attributes, whereas we use random projections to create new attributes.

Maudes et al. [73] propose an ensemble method that builds some new features to be added to the training dataset of the base classifier. Those new features are computed by using a Nearest Neighbor (NN) classifier built from a few randomly selected instances. An experimental study by Maudes et al. [73] with decision tree as the base classifier suggests that for traditional ensemble methods, the ensemble accuracy and the base classifiers diversity are usually improved. In our proposed method, we are using random projections to create new attributes that are linear combinations of the original attributes whereas there is no such property of the new attributes created by the method proposed by Maudes et al. [73], hence we expect that our proposed method has better representational power.

In the next part of this chapter, we will show that the method proposed in this section is the generalization of the RLO framework [68] for decision trees ensembles.

4.2 Random Linear Oracle Ensembles

Kuncheva and Rodriguez [68, 84] propose classifier ensembles with Random Linear Oracle (RLO). As discussed in Chapter 2, in this method, every classifier in the ensemble is replaced by a pair of classifiers. These two classifiers learn on different subspaces decided by a random hyperplane. While testing, first the position of the testing data point with respect to the random hyperplane is decided and the decision of classifier, trained in that subspace, is evaluated.

As discussed in Chapter 2, having the decision tree as the base classifier, RLO ensemble members can be defined as "omnivariate decision trees" (linear multivariate decision trees) [98], where there is a random hyperplane at the root node followed by two standard univariate decision trees. One of the reasons for the success of RLO ensembles is that RLO trees are quite diverse as different RLOs at the root node of different trees create diversity.

As discussed in chapter 2, in the RLO framework, the hyperplane is generated by taking two random points A and B from the training set and calculating the hyperplane perpendicular to the line segment between the points A and B and running through the middle point (Fig. 4.2). The location of the hyperplane is between these two points that ensures that there will be at least one point on the either side of the hyperplane.

Another interpretation of this approach is, that by this method every training data point is projected on the line segment between the points A and B, a split point that is the middle point of the position of the data point A and the position of the data point B on this dimension is selected to split the training data into two new datasets. *The philosophy of the RLO framework can be seen as projecting the data onto a randomly selected attribute that is a linear combination of original attributes and selecting a split point on this attribute such that there is at least one point on either side of this split point* (Fig. 4.3). To decide the new random attribute, we may use the rich literature of random projections [37, 22] that are generally used to create a low dimensional data from a high dimensional data such that the loss of the information is minimal.

Following the philosophy of the RLO framework, the dataset can be divided into two subsets using the random projection technique (we will call this the RLO$'$ framework). The data is randomly projected onto a one dimensional space (ρ_1) and the data

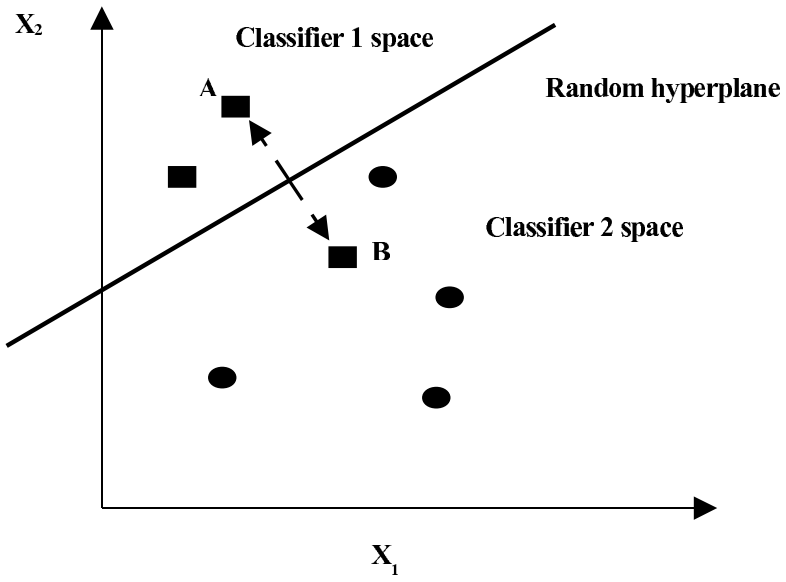


Figure 4.2: The RLO (hyperplane) is generated by taking two random points A and B from the training set and calculating the heperplane perpendicular to the line segment between the points and running through the middle point.

is split into two subsets using a random split point s such that

$$\rho_1 = \sum_{i=1}^{m} w_i x_i \leq s. \tag{4.1}$$

where x_i are the numeric attributes, w_i are the corresponding real valued coefficients and $\rho_{1(min)} < s < \rho_{1(max)}$.

RLO′ ensemble members created using the proposed method can also be defined as linear multivariate decision trees [98], where there is a random hyperplane defined as Eq. 4.1 at the root node followed by two standard univariate decision trees (Fig. 4.7). The RLO algorithm for the Bagging ensemble method in the original form is presented in Fig. 4.5. The RLO′ algorithm for the Bagging ensemble method by using RP (as proposed in this section) is presented in Fig. 4.6. An empirical evaluation will follow in section 4.5.

In RLO′ ensembles, random hyperplanes are created such that the orientations and the locations of these hyper planes are random, that may adversely affect the accuracy of individual classifiers of the ensembles. In the next section, we propose a new variant of RLO′ ensembles, Learned-RLO (LRLO) ensembles, in which the locations of the hyperplanes are decided using the selected split criterion.

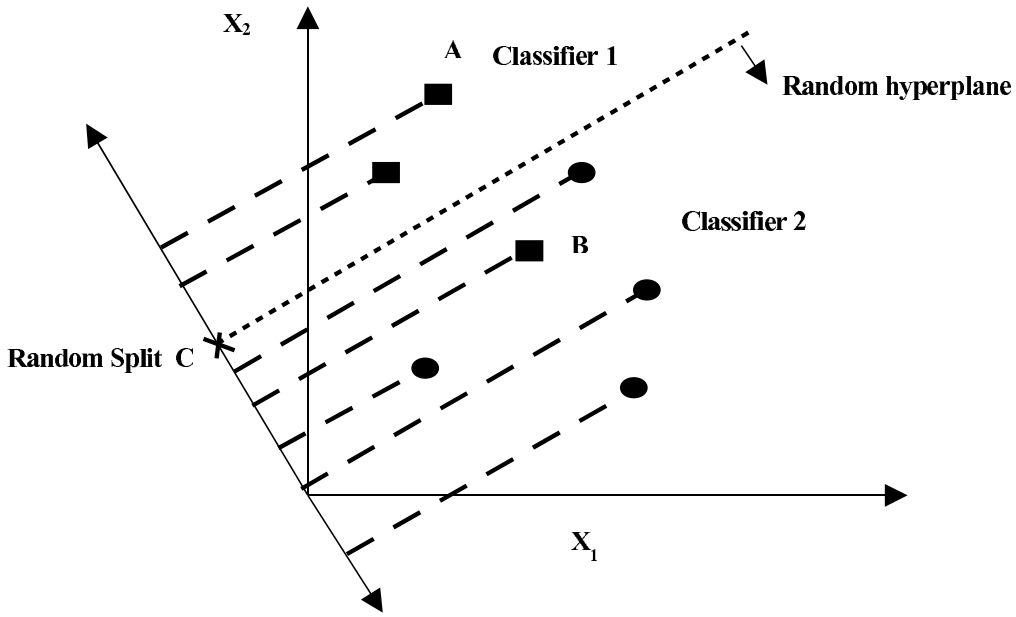


Figure 4.3: Project all data points on a random direction and spit the data by selecting the random point C.

4.3 Learned-Random Linear Oracle

In RLO′ ensembles, the random hyperplanes are created such that the orientations and the locations of these hyperplanes are random. We may extend this approach by developing Learned-RLO (LRLO) in which the orientations of these hyperplanes are random, however, the locations of these hyperplanes are decided using the selected split criterion. The advantage of this approach is that it may improve the accuracy of decision trees as we have eliminated one of the random elements of a RLO′ tree (their locations; in RLO′ random hyperplanes are placed randomly). One may argue that by eliminating one of the random elements of RLO trees, the diversity of the RLO ensemble is reduced that is created by this random element (the locations of hyperplanes) and that may adversely effect the final accuracy of the ensembles. In Learned-RLO, the orientation of the hyperplane is random for each LRLO tree. In other words, every LRLO tree has different hyperplane that is the source of the diversity in LRLO ensembles. Hence, the new step may improve the overall performance of ensembles.

In the LRLO approach, the data is randomly projected in one dimension ρ_1 and any selected splitting criterion is used to split the data (in the RLO′ approach, the data is split randomly),

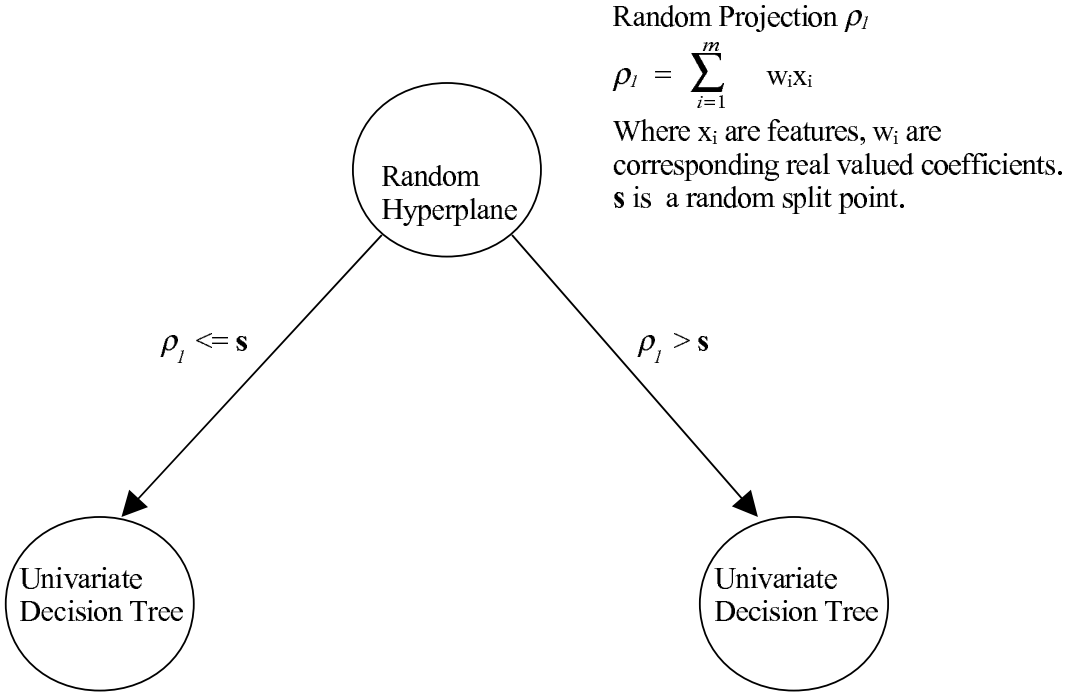


Figure 4.4: A RLO′ omnivariate decision tree with a random hyperplane at the root node.

$$\rho_1 = \sum_{i=1}^{m} w_i x_i \leq l \tag{4.2}$$

where l is decided by the selected splitting criterion.

Then two univariate decision trees are trained on two data subsets created using this split. LRLO ensemble members can be defined as linear multivariate decision trees, there is a decision stump (a decision stump is a single level decision tree created by using a single attribute) at the root node that takes the decision on rp_1 (defined as Eq. 4.2) attribute followed by two univariate decision trees (Fig. 4.4).

We propose a variant of the LRLO approach that instead of creating random projection in one dimension, the data is randomly projected onto more than one dimensional space and the best split point is selected from the new attributes. In other words, if the data is randomly projected on d dimensions, a decision stump is learnt on this d dimensional data. As the decision stump take the decision on the basis of a single attribute, the best attribute of these new d attributes is selected by the decision stump. As there are more options to select from, it may further improve the accuracy of decision trees. The LRLO algorithm for the bagging ensemble method is presented in Fig. 4.8.

This variant of LRLO is similar to the Principal Direction Linear Oracle (PDLO)

Random Linear Oracle Algorithm
Initialisation- Choose the ensemble size *M*, the base classifier model *D* and classification problem *P* defined as a labelled training set *T* with *m* attributes.

Ensemble Construction-
for i=1...*M* **do**
a- Create a dataset T_i from the training dataset *T* using Bagging process.
b- Draw a random hyperplane h_i in the feature space of T_i.
d- Split the training set T_i into T_i^+ and T_i^- depending on the which side of h_i the points lie.
e- Train a classifier for each side, D_i^+ = *D*(T_i^+) and D_i^- = *D*(T_i^-). Add the mini-ensemble of the two classifiers and the split point, (s_t,D_i^+,D_i^-), to the current ensemble.
end for

Classification
For a new object **x**,
for i=1...*M* **do**
For a new object **x**, find the decision of each ensemble member by choosing D_i^+ and D_i^- depending on which side of h_i, **x** is. Combine the decisions of all the selected classifiers by the chosen combination rule (we use majority voting scheme).
end for

Figure 4.5: The original Random Linear Oracle (RLO) algorithm [68]. The highlighted portion of the algorithm is modified in the proposed RLO′ and LRLO.

[78] approach (discussed in Chapter 2) in a sense that both learn the best hyperplane from the different choices. However, principal direction linear oracle (PDLO) [78] uses PCA and whereas in LRLO, we use random projections.

4.4 Multi-Random Linear Ensembles

In the RLO′ framework, a random hyperplane is placed at the root node of a RLO′ tree, whereas in a LRLO tree a decision stump trained on the new attribute (created by using the random projection method) is placed at the root node of a LRLO tree. In both cases *the original attributes are not examined to select the split point at the root node*.

In a normal decision tree growing phase, at each level of the decision tree, a split point is decided by the selected split criterion. In other words, at each level all the available attributes are examined and the best attribute is selected (decided by the splitting

Random Linear Oracle Algorithm by using RP
Initialisation- Choose the ensemble size *M*, the base classifier model *D* and classification problem *P* defined as a labelled training set *T* with *m* attributes.

Ensemble Construction-
for i=1...*M* **do**
a- Create a dataset T_i from the training dataset *T* using Bagging process.
b- Create random projection ρ_i of the data T_i on 1 dimensions by using matrix R_i.
c-Select a split point S_t on ρ_i attribute randomly.
d- Split the training set T_i into T_i^+ and T_i^- depending on the value of attribute ρ_i $<= s_i$ or $> s_i$.
e- Train a classifier for each side, D_i^+ = *D*(T_i^+) and D_i^- = *D*(T_i^-). Add the mini-ensemble of the two classifiers and the split point, (s_t,D_i^+,D_i^-), to the current ensemble.
end for

Classification
For a new object **x**,
for i=1...*M* **do**
a- Use R_i matrix to create new d attributes.
b- Find the decision of each ensemble member by choosing D_i^+ or D_i^- depending on the value of ρ_i attribute, $<= s_i$ or $> s_i$.
end for
Combine the decisions of all the selected classifiers by the chosen combination rule (we use majority voting scheme).

Figure 4.6: Random Linear Oracle (RLO) algorithm by using RP. The highlighted portion of the algorithm is different from the original RLO. We define this algorithm as RLO′.

criterion). Hence, at the root node the best split point is selected from all the attributes. Whereas, RLO′ trees and LRLO trees do not consider the original attributes at the root node. The new attribute (the random projection of the data to one dimensional space) may not be very significant; even then the split point at the root node is decided by this attribute. This may adversely affect the classification accuracy of the decision tree. To overcome this problem, we propose the Multi-RLE framework that has a similar tree growing phase as a univariate decision tree.

In the Multi-RLE framework, the new attribute (created using random projection of the data) is combined with the original attributes. Hence, if the number of attributes are m, each data point is represented by m + 1 attributes. A decision tree is trained

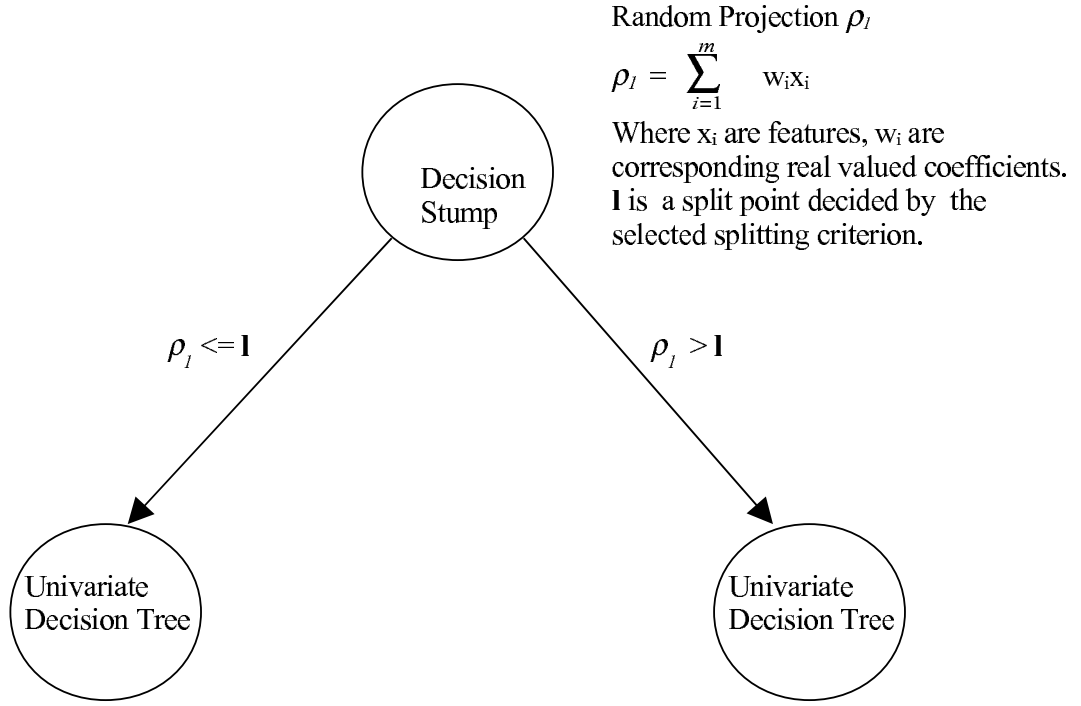


Figure 4.7: A LRLO ominivariate decision tree with a decision stump at the root node.

on this new data, now the new attribute can be selected anywhere in the tree and there is no guarantee that it is selected at the root node of the tree as in a RLO′ tree or a LRLO tree. It may help in improving the classification accuracy of Multi-RLE trees as the new attribute created using RP may not be good for classification. Its selection at the root node may badly affect the accuracy of a RLO′ or a LRLO tree. However, by making the new attributes a part of the data, we do not get very diverse decision trees.

This can be explained by the fact that the diversity is created by the new attribute. If the new attribute is not very significant, it will be selected at the lower level of the decision tree, and two decision trees which are not similar only at lower levels will not be as diverse as decision trees which are not similar at higher levels (as in RLO′ and LRLO trees, the root nodes are different). Decision trees are very unstable, a small change, in one node, can bring a large change in the structure of decision trees. Therefore, different RLO′ and LRLO trees may have different tree structures.

In the Multi-RLE framework the data is not divided into two subsets, in other words a single tree is trained whereas, in the RLO′ framework and in the LRLO framework a pair of trees is trained.

In one of the variants of Multi-RLE ensembles, instead of creating a random projection on one dimension, the random projection of the data is created on more than one

Learned-Random Linear Oracle Algorithm
Initialisation- Choose the ensemble size *M*, the base classifier model *D* and classification problem *P* defined as a labelled training set *T* with *m* attributes.

Ensemble Construction-
for i=1...*M* **do**
a- Create a dataset T_i from the training dataset *T* using Bagging process.
b- Create random projection $T_{i(d)}$ of the data T_i on d dimensions by using random projection matrix R_i
c- Learn a decision stump on this d dimensional data $T_{i(d)}$, and save split point S_t of the selected S attribute (as the decision stump decides on one attribute out of d attributes).
d- Split the training set T_i into T_i^+ and T_i^- depending on the value of S attribute $<= l_t$ or $> l_t$.
e- Train a classifier for each side, D_i^+ = *D*(T_i^+) and D_i^- = *D*(T_i^-). Add the mini-ensemble of the two classifiers and the split point, (l_t,D_i^+,D_i^-), to the current ensemble.
end for

Classification
For a new object **x**,
for i=1...*M* **do**
a- Use R_i matrix to create new d attributes.
b- Find the decision of each ensemble member by choosing D_i^+ or D_i^- depending on the value of S attribute, $<= l_t$ or $> l_t$.
end for
Combine the decisions of all the selected classifiers by the chosen combination rule (we use majority voting scheme).

Figure 4.8: Learned-Random Linear Oracle (LRLO) algorithm. The highlighted portion of the algorithm is different from the original RLO.

dimension (d attributes) and these d attributes are added to the original m attributes. Better accuracy of individual classifiers may be one of the advantages of this approach, the other important advantage of this approach is better diversity. As we are creating d attributes, there is more probability (as compared to when a random projection creates one attribute) that some of these d attributes may be good for the classification and may be selected at higher levels, hence decision trees will be diverse. We will like to emphasise that in this process the diversity is created by creating new attributes and the tree growing phase is not changed (no randomization is introduced during tree growing phase) that ensures the good classification accuracy of decision trees. The Multi-RLE algorithm for the bagging method is presented in Fig. 4.9.

The second version of the Multi-RLE is identical to the method to create ensembles of linear multivariate decision trees suggested in the first part of this chapter. This shows the relationship between the method presented in first part of the chapter and the RLO framework. These two algorithms have different motivations; the first algorithm (Fig. 4.1) was developed to increase the representational power of ensembles whereas the second algorithm (Fig. 4.9) has its root in the RLO philosophy which improves the diversity of trees. This shows that one may study RLO ensembles by using the representational power framework.

If that data is projected onto more than one dimensional space, a Multi-RLE tree can use all the new attributes, whereas a RLO$'$ tree and a Multi-RLE tree can use only one attribute. This suggests that a Multi-RLE tree has more expressive powers as compared to a RLO$'$ tree and a LRLO tree.

We summarize these three approaches in Table 4.1 and Fig. 4.10. In a decision tree rules are combinations of attributes. Creating a best decision tree is a NP-hard problem [57]. As discussed in Chapter 2 a decision tree is created by using a greedy heuristic in which the best attribute is selected at each node. There is no guarantee that this will give the best tree, however, this method gives good results. Following the philosophy of this heuristic, we can conclude on the basis of random elements in the tree growing phase that RLO$'$ trees are least accurate and Multi-RLE trees are most accurate. However, good ensembles consists of diverse and accurate decision trees, in the next section, we present experimental results.

4.5 Experiments

We carried out two kinds of experiments;

Multi-RLE Algorithm
Initialisation- Choose the ensemble size M, the base classifier model D and classification problem P defined as a labelled training set T with m attributes.

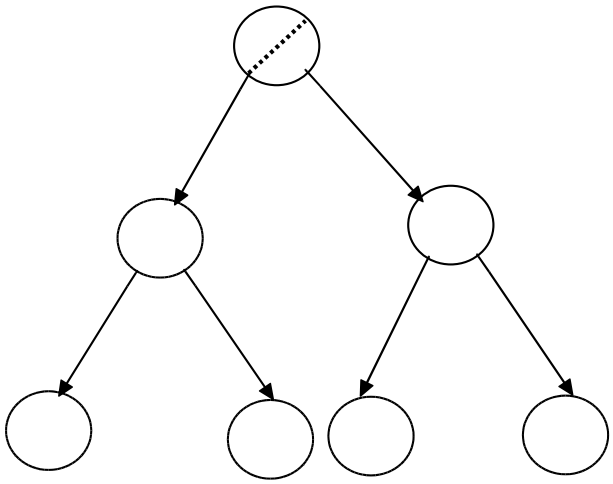
Training Phase
for i=1...M **do**
Data Generation
a- Create a dataset T_i from the training dataset T using Bagging process.
a-Create random projection $T_{i(d)}$ of the data on d dimensions by using matrix R_i.
b-Combine T and $T_{i(d)}$ datasets to get $m + d$ dimensional dataset.
Learning Phase
Learn D_i decision tree on it.
end for

Classification Phase
For a given data point **x**
for i=1...M **do**
a- Convert **x** into $m + d$ dimensional data point $\mathbf{X_i}$ using original data and matrix R_i.
b- Get the prediction for $\mathbf{X_i}$ by the decision tree D_i.
end for
Combine the results of *M* decision trees to get the final classification result.

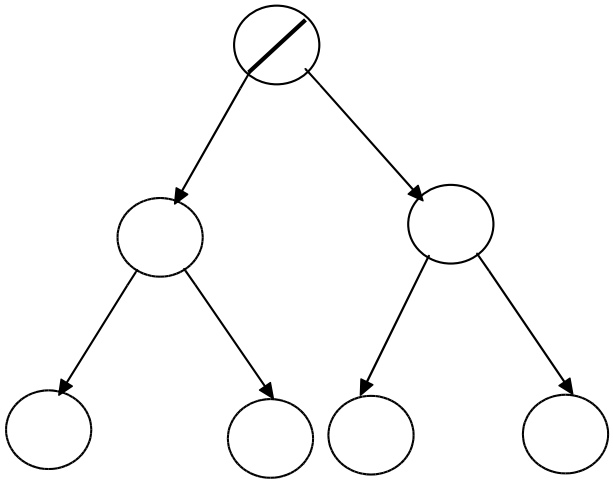
Figure 4.9: Multi-Random Linear Ensembles (Multi-RLE) algorithm. d new attributes are created and concatenated with the original features.

Method	Number of possibilities to be considered at the root node	Number of attributes to be considered at other nodes
RLO	$\binom{n}{2}$	m(+)
RLO′	d (-)	m (+)
LRLO	d (+)	m (+)
Multi-RLE	$(m + d)$ (+)	$(m + d)$ (+)

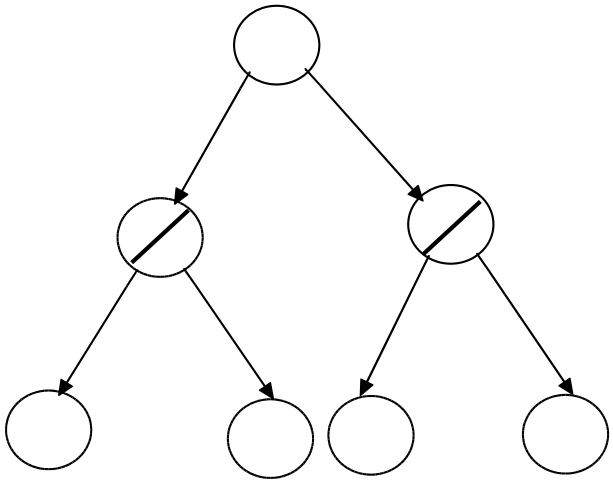
Table 4.1: Comparative chart of RLO, RLO′, LRLO and Multi-RLE on the basis of number of possibilities to be considered at the root node and other nodes. '-' means split points are created randomly and '+' means split points are created by using the selected split criteria. m is the number of the original attributes, d is the number of new attributes created by RP and n is the number of data points in the training data.



(a) RLO



(b) LRLO



(c) Multi-RLE

Figure 4.10: RLO′, LRLO and Multi-RLE trees. A dotted line represents random hyperplane, a solid line represents a decision stump trained on new features created using random projections.

- To study how the ensembles of linear multivariate decision trees perform, we carried out comparative study of the proposed ensemble methods against Bagging, Adaboost.M1 and RF.

- Different popular ensemble methods are benefited from the RLO framework, most markedly so Random Subspace and Bagging[68]. Hence, In the second part of experiments, we study the how Bagging is affected by RLO, LRLO and Multi-RLE methods.

We carried out experiments with unpruned J48 (Weka implementation of C4.5) decision trees. The size of the ensembles was 50 for these experiments. For RLO′ ensembles and LRLO ensembles, 50 pairs of classifiers are trained. We used Bagging, AdaBoost.M1 with J48 decision trees (with unpruned option) implementation of Weka. We also carried out the experiments with Random Forests implementation) of Weka. Apart from the size of the ensemble (which was 50) default settings were used for these modules.

As discussed in Chapter 3, the elements r_{ij} of Random Matrix R are often Gaussian distributed. We used the following matrix as suggested by Achlipotas [1] for RP in our experiments as it has benefit of being easy to implement and compute.

$$\mathrm{r}_{ij} = \pm\sqrt{3} \text{ with probability 1/6 each or 0 with 2/3 probability.} \tag{4.3}$$

The experiments were conducted following 5×2 cross-validation [28]. The original *t* test proposed by Dietterich [28] to compare the performance of classifiers suffers from low power and low replicability. Alpaydin [4] propose a modification called 5×2 cross-validation *F* test. We used this test for our experiments. The test has following steps,

$q_i^{(j)}$ is the difference between the error rates of the two classifiers on fold $j = 1, 2$ of replication $i = 1, 2, 3, 4, 5$. The average of the replica i is $\bar{q}_i = \frac{(q_i^{(1)} + q_i^{(2)})}{2}$, the estimated variance is $\nu_i^2 = (q_i^{(1)} - \bar{q}_i)^2 + (q_i^{(2)} - \bar{q}_i)^2$ and f is defined as

$$f = \frac{\sum_{i=1}^{5}\sum_{j=1}^{2}(q_i^{(j)})^2}{\sum_{i=1}^{5}\nu_i^2}, \tag{4.4}$$

the value of f is approximately F distributed with 10 and 5 degrees of freedom. We considered a confidence level of 95% for this test.

We projected the data onto one-dimensional space, five-dimensional and ten-dimensional spaces. For linear multivariate decision trees, these new attributes were added to the original attributes. When the dataset was projected onto one-dimensional space, for a LRLO tree, a decision stump was trained on that new attribute, whereas a random split point was used for RLO′ for that new attribute. When the dataset was projected on more than one-dimensional space, for a MLO tree, a decision stump was learnt on new attributes and the best split point was selected. For the proper comparative study, the attribute that was selected for the split (by the decision stump) in a LRLO tree was used with the random split point for a RLO′ tree.

4.6 Results

Table 4.2 shows the classification accuracy for Bagging RLO, LRLO, and Multi-RLE methods for different datasets. We summarize our results in following points;

4.6.1 Ensembles of Linear Multivariate Decision Trees

1. Except for the Two-Norm dataset, the average accuracy of linear multivariate decision trees are similar to J48 decision trees (univariate decision trees). As discussed earlier that the attributes created by random projection are not good attributes for a decision trees [38] and decision trees are using predominantly the original attributes. Therefore, results are similar to J48 decision trees.

2. For ensembles of linear multivariate decision trees, there is a steady increase in the classification accuracy with the increase in the number of new attributes. A Multi-RLE tree can use all the new attributes and these attributes can be the part of the tree at any level. This ensures a more accurate decision tree in some cases. At the same time when we have a large number of new attributes, there is a greater probability that some of these attributes are selected at higher levels of the tree. This helps in getting diverse decision trees. Results suggest that the contribution of this second point is more in the performance of ensembles.

3. Generally, the performance of ensembles of linear multivariate decision trees (with 10 added attributes) is statistically similar to (Vowel and Waveform datasets)

Dataset	No. of attributes added-1 (ensemble)	No. of attributes added-5 (ensemble)	No. of attributes added-10 (ensemble)	Bagging	AdaBoost.M1	Random Forests	No. of attributes added-1 (single)	No. of attributes added-5 (single)	No. of attributes added-10 (single)	Single Tree
Pendigit	4.62	1.93	1.44	2.23	0.94	1.15	4.56	4.64	4.51	4.77
TwoNorm	14.94	3.45	2.73	4.34	3.32	4.11	14.41	12.25	10.68	15.83
Vowel	29.6	21.56	16.90	18.62	11.61	10.74	30.03	30.16	28.72	29.70
Waveform	23.53	20.92	17.53	17.61	16.25	16.54	23.97	24.26	24.32	24.60

Table 4.2: Classification errors in % for the linear multivariate ensemble method. 1,5 and 10 new attributes, created by using random projections, are added. We also presented the results with other ensemble methods. Results suggest that the Adaboost.M1 and Random Forests generally perform better than the proposed method.

Dataset	RLO with 1 new attribute	RLO with 5 new attributes	RLO with 10 new attributes	LRLO with 1 new attribute	LRLO with 5 new attributes	LRLO with 10 new attributes	Multi-RLE with 1 new attribute	Multi-RLE with 5 new attributes	Multi-RLE with 10 new attributes	Bagging	Single Tree
Pendigit	1.81	1.68	1.67	1.72	1.64	1.46	2.13	1.52	**1.42**	2.23	4.77
TwoNorm	3.90	3.91	3.82	3.67	3.24	3.17	3.80	3.38	**2.81**	4.34	15.83
Vowel	16.43	16.25	16.38	15.74	15.21	14.61	17.52	14.73	**13.21**	18.62	29.70
Waveform	17.26	17.13	17.18	16.48	16.26	16.61	17.13	16.62	**16.21**	17.61	24.60

Table 4.3: Classification errors in % of Bagging and its combination with RLO, LRLO and Multi-RLE. Bold numbers show the best performance. Results suggest that creating a large number of new attributes and concatenating with the original features is the best strategy in the RLO framework.

or statistically better (Pendigit and TwoNorm) than Bagging, whereas generally it is statistically worse than AdaBoost.M1 and Random Forests.

4.6.2 Comparative Study of RLO, LRLO and Multi-RLE

Results are shown in Table 4.3. We summarize our results in following points.

1. All methods generally improve the performance of Bagging. This indicates that similar to the RLO′ approach [68], the LRLO approach and the Multi-RLE approach can also be used to improve the performance of Bagging ensembles.

2. The performance of RLO′ ensembles are almost similar when 1, 5 and 10 new attributes are used. The split point in a RLO′ tree root node is decided randomly (the attribute is the same as is used in the LRLO tree), hence creating large number of attributes are not very useful in this case.

3. For two datasets (Vowel and Waveform), the performances of LRLO ensembles with different new attributes (1, 5 and 10) are similar. For two datasets (Pen and Two-Norm), the performance of LRLO ensembles with 10 new attributes is statistically better than LRLO ensembles with 1 new attributes. A LRLO tree has a decision stump at the root node. The decision stump is trained on new attributes. When we have a large number of new attributes, the decision stump has more choices. Hence, a more accurate decision stump is highly probable with a large number of new attributes. A more accurate decision stump at the root node may lead to a more accurate LRLO tree. This may improve the performance of LRLO ensembles.

4. For Multi-RLE ensembles, there is a steady increase in the classification accuracy with the increase in the number of new attributes.

5. For two datasets (Pendigit and TwoNorm), the performance of LRLO ensembles with 10 new attributes is statistically better than all RLO′ ensembles. For the other two datasets (Vowel and Waveform) LRLO ensembles and RLO ensembles are statistically similar.

6. For all the datasets, the performance of Multi-RLE ensembles with 10 new attributes is statistically better than all RLO′ ensembles.

7. The performance of Multi-RLE ensembles with 10 new attributes is statistically similar to all LRLO ensembles for two datasets (Pendigit and Waveform). For two datasets (TwoNorm and Vowel) the performance of Multi-RLE ensembles is statistically better than all LRLO ensembles.

Results suggest that in the RLO framework the best strategy is to create a sufficiently large number of attributes using the random projection transformation and concatenate these attributes with the original attributes, then learn a univariate tree.

4.7 Conclusion

In this chapter, first we presented a method to create linear multivariate decision trees using an univariate decision tree algorithm. Random projections are used to create new features that are linear combinations of the original features, these new features are concatenated with the original features and decision trees are trained on these new datasets. These trees are diverse linear multivariate decision trees as they use new features along with original features. Different random projections create diversity. The difference between this approach and the other popular techniques (to create linear multivariate decision trees) discussed in Chapter 2 is that in this approach, we do not learn linear combinations of attributes during the tree growing phase as they are decided by random projections. Hence, the complexity of the tree growing phase does not increase much (it is affected by the increased size of new datasets due to new attributes).

We further show that the RLO′ framework can be studied by using random projections. We then showed that the method presented in first part of the chapter is the generalization of the RLO′ framework. The experimental study suggest that the generalization of the RLO′ framework perform better than the original RLO framework. The proposed method can use all the new attributes created by random projections, whereas the original RLO′ framework uses only one new attribute. Hence, the proposed decision trees have more decision surfaces. It helps them learning complex decision boundaries. The position of a new random hyperplane in a RLO′ tree is decided randomly and a random hyperplane is always placed at the top of decision tree this may adversely affect the accuracy of the decision tree. However, in the proposed method the positions of random hyperplanes are learnt and these hyperplanes can be anywhere in the decision trees (this is decided by the decision tree algorithm), hence, these tress are likely to be more accurate than RLO′ trees. These are the reasons for

the success of the proposed method.

In this chapter, we presented an ensemble method that uses random projections. In the next chapter, we will study how the discretization process (a data transformation process) can be used to address the representational problem.

Chapter 5

A Novel Ensemble Method for the Representational Problem

In the previous chapter we used random projections to address the representational problem of decision trees. In this chapter, two discretization methods, Random Discretization (RD) and Extreme Random Discretization (ERD) are presented that create diverse discretized datasets. We then show that these two methods can be used to create ensembles of decision trees. We discuss the relationship between the proposed ensemble methods and existing ensemble methods. We present a theoretical study that suggests that these ensembles have excellent representational power. We compare the proposed ensemble method with other popular ensemble methods; Bagging, random Forests and Adaboost.M1. We also study the effect of combining Multi-RLE (presented in the last Chapter) with random discretized ensembles.

5.1 Random Discretized Ensembles (RDEns)

This work explores a use of the discretization technique for creating decision tree ensembles. Discretization is a process that divides continuous values into a set of intervals that can be considered as categorical values. As discussed in Chapter 2, all popular discretization methods [32] create a unique discretized dataset. However, to create diverse classifiers, we need diverse datasets. In this work, we develop a novel discretization method Random Discretization (RD) that create diverse discretized datasets. This method in principle can be applied to any classifier (the learning process of which is perturbed with different discretization) though here we evaluate only decision trees. RD ensembles are created by using the RD method. RD ensembles are simple but

surprisingly accurate. The simple structure of a RD tree is helpful in the theoretical analysis of RD ensembles.

In this section, we present Random Discretization Ensembles (RDEns) method. First, we describe two novel methods Randomized Discretization (RD) and Extremely Randomized Discretization (ERD) that create diverse discretized datasets.

5.1.1 Data Generation

Discretization divides an attributes values into different categories depending upon the intervals they fall into. For example, to discretize a attribute into three categories, we choose two points x_1 and x_2 between the minimum (x_{min}) and the maximum (x_{max}) values of the attribute, if $x_1 < x_2$, the attribute values are discretized using the following rules;

- if $(x \leq x_1)$, the category of x = 1,
- if $(x > x_1)$ and $(x \leq x_2)$, the category of x = 2,
- if $(x > x_2)$, the category of x = 3,
  where x is the attribute value.

This example suggests that to create *K* categories, we need $K-1$ points. There are different methods to create these points [32]. However, all these methods produce a single and unique discretized dataset. For creating ensembles, we need diverse datasets so that the learning on these datasets creates diverse classifiers. We propose a novel method Random Discretization (RD) to select these $K-1$ points. In this method, we take into account the interdependencies of attributes to create discretization of various attributes. To create K categories $K-1$ *data points are selected randomly from the training data*. For each attribute, every selected data point has one value, this way we can have $K-1$ data points for every attribute. The attribute can be discretized into K categories using these $K-1$ data points. It is possible that for some attributes, we have less than $K-1$ boundaries, as two or more selected data points may have same values for these attributes. That will produce a fewer number of categories for those attributes. In the extreme case, for some attributes, all the selected points may have attributes values equal to minimum or maximum values of the attributes. In other words, we have no point for the attributes between the minimum and the maximum values of the attributes. $K-1$ points are selected randomly between the minimum and

---

Random_Discretization
Input- Numeric training dataset T with *n* data points and *m* continuous attributes.
Output- Discretized training data set.
Begin
1- For K categories in each dimension, select K-1 data points randomly from the training data.
for j=1...*m* **do**
  2.1- Get the j^{th} attributes values of K - 1 points and sort them.
  2.2- If all points are equal to the minimum or the maximum values of the attribute. Select K-1 points randomly having values between minimum or maximum values of the attribute.
end for
for i=1...*n* **do**
  3- Discretize the i^{th} data point using the values got in step 2.
end for
End.

Figure 5.1: Random Discretization (RD) method.

the maximum values of the attribute for these kinds of attributes. The proposed RD algorithm is presented in Fig. 5.1. Table 5.1 shows a two dimensional dataset. Two points are needed to divide every dimension into three categories. Two data points are selected randomly, for example, we select data points 2 and 4. For the attribute A_1 , 3.7 and 6.8 are the two points for discretization. For the attribute A_2, 5.8 and 4.5 are two points for discretization so the dataset is discretized using the following rules,

1. **For A_1 attribute**
   - if an attribute value$\leq$ 3.7, the category of the attribute value = 1.
   - if 6.8 $\geq$ an attribute value $>$ 3.7, the category of the attribute value = 2.
   - if an attribute value $>$ 6.8, the category of the attribute value = 3.
2. **For Y attribute**
   - if an attribute value$\leq$ 4.5, the category of the attribute value = 1.
   - if 5.8 $\geq$ an attribute value $>$ 4.5, the category of the attribute value = 2.
   - if an attribute value $>$ 5.8, the category of the attribute value = 3.

Data point	Attribute A_1	Attribute A_2	Class
1	2.3	6.8	C_1
2	3.7	5.8	C_2
3	3.8	9.8	C_1
4	6.8	4.5	C_1
5	3.2	8.1	C_2
6	7.7	4.2	C_2

Table 5.1: A two dimensional numeric dataset.

We also carried out experiments with the Extremely Randomazied Discretization (ERD) method. In this method, to discretize a attribute, we randomly generate $K - 1$ points such that these points lie between the minimum and the maximum values of the attribute.

5.1.2 Learning

In this section, we discuss the learning process of decision trees on discretized datasets created by using RD and ERD.

Each decision tree in the ensemble learns on one discretized dataset from a pool of different datasets created by RD. If the order of values of attributes is maintained, a discretized dataset is an ordinal dataset. It can be treated as a continuous/integer dataset or a categorical dataset. The learning of some classifiers depends on whether the training dataset is continuous or categorical. In this section, we focus our discussion on decision trees (C4.5 as we have used J48 decision tree as base classifier in our experiments).

For multi-valued categorical attributes, generally, we get multi-way splits at each node. In categorical attributes, categories are already provided so there is no concept of finding the best boundary for split. However, when this attribute is treated as a continuous attribute, a binary split is obtained and each node is split at the best boundary. This can be explained by the following example. Suppose that, at a node, the attribute has four categories (1, 2, 3, 4). In this case, the node has four branches; one for each value, if the attribute is treated as a categorical attribute. Whereas, if it is treated as continuous there will be a binary split. There are three possible splits ({1,},{2,3,4}), ({1,2},{3,4}), ({1,2,3},{4}) and the best split will be selected using the selected split criterion. We expect more accurate decision trees in this case as binary decision trees avoid the data fragmentation problem [94] associated with trees with multi-way splits

Input- Dataset *T* with *m* continuous attributes and M size of the ensemble.

Training Phase
for i=1...M **do**
  Data Generation
  Use Random Discretization Λ_i to generate integer valued dataset T_i.
  Learning Phase
  Treat dataset T_i as continuous and learn D_i decision tree on it.
end for

Classification Phase
For a given data point **x**
for i=1...M **do**
  Convert **x** into a discretized data point $\mathbf{x_i}$ by using Random Discretization (Λ_i.
  Get the prediction for $\mathbf{x_i}$ by the decision tree D_i.
end for
Combine the results of M decision trees by the chosen combination rule to get the final classification result (We use majority voting to combine the results of classifiers).

Figure 5.2: Random Distretization Ensembles (RDEns) algorithm.

nodes. However, the tree growing phase takes more time when the dataset is considered as continuous as compared to when the dataset is treated as categorical, because in a continuous dataset the best split boundary has to be searched at each node. In RDEns, each classifier is trained by using the complete training dataset that helps improve the accuracy of a RD tree. RDEns algorithm is presented in Fig. 5.2.

5.2 Motivation For Random Discretization Ensembles

In this section, we focus our discussion on C4.5 [80] type decision trees (univariate decision trees). In an ensemble, accurate and diverse classifiers are required. *Extreme Random Discretization* (ERD) builds an ensemble of classifiers by changing category boundaries. As discussed in Chapter 2, decision trees have the representational problem because of their orthogonal properties. They have difficulty in learning non-orthogonal decision boundaries. We will discuss the representational power of infinite ERD decision trees for a diagonal decision boundary. We will discuss in this section that when we have infinite ERD decision trees in a ensemble, *a piece-wise continuous function produced by the ERD ensemble approximates to the diagonal concept for a*

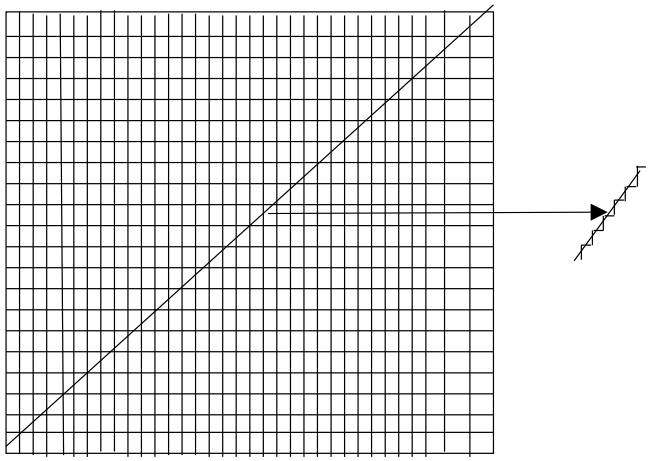


Figure 5.3: Division of axis by trees is uniform and fine grained. There is a diagonal concept. The combination of trees approximates the diagonal concept. Right side of the figure shows a small portion of the concept and its approximation by the ensemble of ERD trees.

two dimensional data. Hence, ERD ensembles have better representational power as compared to a single decision tree.

Decision trees have difficulty in learning a diagonal decision boundary. Ensembles of decision trees solve this problem as combined results of decision trees produce a good approximation of a diagonal decision boundary. As discussed in Chapter 2, Dietterich [29] shows that for majority voting an ensemble of small size decision trees are similar to a large size decision tree and can create a good approximation of a diagonal decision boundary (Fig. 1.1). To get diverse decision trees, we need trees with different split points. In Extreme Random Discretization, the dataset is discretized randomly. When decision trees learn on discretized datasets, nodes can split only on the bin boundaries. As the number of boundaries is small (for K categories, there are only K-1 category boundaries) and these boundaries are random, there is a small probability that decision trees trained on different discretized datasets have same node splits for an attribute. *In other words, there is a high probability for that each decision tree divides the data space at different points*. This situation is similar to the Fig. 5.3, which suggests that with ERD ensembles, diagonal decision boundary can be learned properly. In case of infinite ERD decision trees in an ensemble, a piece-wise continuous function produced by ERD ensembles approximates the diagonal concept. We may extend this argument to the other decision boundaries to show that ERD ensembles have good representational power. On the basis of the above argument, it can be hypothesized that a ERD ensemble has better representational power as compared to a

single decision tree.

The success of ensemble methods depends on the creation of uncorrelated classifiers [90]. An RD tree has limited tree growing options as it has to follow bin boundaries. In other words, diverse decision trees are produced as different options are provided (different bin boundaries) at the tree growing phase. Very accurate trees may not be obtained by RD, however, this technique ensures very diverse decision trees with good representation powers for RD ensembles. We have discussed characteristics of ERD ensembles, however as RD ensembles and ERD ensembles are quite similar, we expect that RD ensembles have the same properties. In the next section, we present an experimental study of RD ensembles.

5.3 Related Work

RD trees are similar to Extremely Randomized Tree (ERT) [47] and ensembles of trees based on histograms [62]. Possible split boundaries in RDEns and these two methods are created without considering the output (class labels); the best split cut-points are selected using the selected split criteria from these possible split boundaries. In RD ensembles, the features are discretized, this is same as decision trees based on histograms [20]. However, in ensembles of trees based on histograms, the discretization is done at node. In other words, the discretization is local and in RD trees the discretization is global. In ensembles of trees based on histograms, bin boundaries are not selected as cut-points (after selecting the best bin boundary, they select the split point randomly in some interval around the best interval boundary) whereas in a RD tree, one of the bin boundaries is selected as the cut-point. In ERT, nodes are split at randomly selected points. This can be treated as discretizing feature randomly into two bins. It is also a case of local discretization. At each level, K features are randomly selected, each feature is split randomly. Out of these K splits, the best split based on the selected split criterion is selected for that level. Hence, if a feature is not selected for the split, chances are that it will be considered at lower levels with different cut-points (as the cut-point is selected randomly). In a RD tree, the data is discretized globally so during the tree building the cut-points do not change (bin boundaries remain the same). In the RD approach, bin boundaries of different features are related whereas in these approaches, the discretization of one feature is independent of other features.

RD is a general approach in that, it can be used with any classifier in which different discretization affects the learning of classifiers (for example *Naive Bayes* classifiers

[97]). In contrast, ensembles of trees based on histograms and ERT are constrained to decision trees as they are based on the discretization at the nodes during the tree growing phase.

5.4 Experiments

We carried out experiments on 16 pure continuous datasets taken from the UCI repository to study RDEns. The information about these datasets is given in Table A.2. We used the J48 tree (the C4.5 decision tree [80] implementation) of Weka software [96], with the unpruned option, for our experiments. The size of classifier ensembles was set to 50. Bagging [11], AdaBoost.M1 [41] modules of Weka with the J48 decision tree were used for comparison. We also did experiments with Random Forests [13]. Default parameters (other than the size of the ensembles) were used in all cases. Attributes were discretized into 5 categories by using the RD method for RD ensembles and by using the ERD method for ERD ensembles.

As discussed in section 5.1.2, during the learning phase a discretized dataset can be treated as a categorical dataset or a continuous dataset. We carried out experiments with both kinds of learning. Symbols of four different kinds of ensemble methods are given below;

1. RD(cat.) - Discretized datasets are created by using RD, and during the tree growing phase, discretized datasets are treated as categorical datasets.

2. RD(cont.) - Discretized datasets are created by using RD, and during the tree growing phase, discretized datasets are treated as continuous datasets.

3. ERD(cat.) - Discretized datasets are created by using ERD, and during the tree growing phase, discretized datasets are treated as categorical datasets.

4. ERD(cont.) - Discretized datasets are created by using ERD, and during the tree growing phase, discretized datasets are treated as continuous datasets.

We followed the 5×2 cross validation methodology [28, 4] discussed in Chapter 4 for these experiments.

Table 5.2 presents the results of various ensemble methods. The numbers in bold

show the best performances for the dataset. Results suggest that the learning discretized datasets as continuous datasets give better results as compare to when discretized datasets are treated as categorical datasets. For two datasets (Waveform, Ring-Norm and Two-Norm) ERD ensembles perform very well, whereas for the Spambase dataset, all ensemble methods perform better than a single classifier, the performance of ERD ensembles is similar to a single classifier. In these experiments, we select RD(cont.) method for comparative study.

Table 5.3 shows the comparative performance of RD(cont.) on different datasets. For 14 out of 16 datasets, RD(cont.) performed statistically better than single decision trees. Its performance is statistically similar with Bagging for 8 datasets, better for 8 datasets. RD(cont.) performs statistically better than AdaBoost.M1 for 3 datasets and statistically worse than AdaBoost.M1 for 3 datasets. Its performance is similar to Random Forest for 12 datasets, it is better than Random Forest for 4 datasets. The comparative study indicates that RD(cont.) performs similar to or better than Bagging and Random Forests, whereas it is quite competitive to AdaBoost.M1. In the next section, we analyse RD ensembles and ERD ensembles to understand their behaviour.

5.5 Analysis

In the previous section, we showed that the RD ensemble method is very competitive when compared with the other ensemble methods. In this section, we carried out following analysis to get insights of RD ensembles and ERD ensembles. We used four datasets Pendigit, Segment, Vowel and Waveform for our analysis.

5.5.1 Noisy Data

As most real datasets have class noise, it is important to understand the robustness of RD ensembles and ERD ensembles for noisy data. In this section, we present the study of RD ensembles and ERD ensembles for the noisy data.

To add noise to the class labels, we followed the method proposed by Dietterich [30] as discussed in Chapter 5. We carried out this exercise for the noise levels 10%, and 20% for four datasets. We used again 5x2 cross validation for this study. The ensemble size was 50.

The results are presented in Table 5.4 - Table 5.11. Results suggest that RD ensembles and ERD ensembles are quite robust to the class noise. With the exception of

Data	RD (cat.)	RD (cont.)	ERD (cat.)	ERD (cont.)	Bagging	AdaBoost.M1	Random Forests	Single
Balance	14.5	**12.5**	16.4	14.4	17.2	20.9	18.6	22.4
Breast Cancer	3.8	3.6	3.5	3.8	4.0	3.6	**3.3**	6.2
Ecoli	17.4	14.6	16.5	**14.2**	17.8	18.4	18.1	19.5
Glass	**25.9**	27.9	33.6	30.8	30.6	28.2	27.5	37.8
Ionosphere	7.1	**6.3**	7.0	6.8	7.2	7.7	7.5	10.1
Letter	6.2	5.4	6.9	6.5	8.5	**4.4**	5.3	15.7
Pendigit	1.6	1.0	1.2	1.0	2.3	**0.9**	1.2	4.7
Pima-diabetes	26.2	26.1	25.4	**24.7**	26.1	28.1	25.6	28.4
RingNorm	3.5	2.9	2.5	**2.3**	5.5	2.6	4.5	9.9
Segment	2.7	2.6	4.0	3.8	3.7	**2.3**	2.7	4.6
Sonar	24.6	21.1	21.0	**19.8**	26.8	23.0	21.8	30.9
Spam	6.0	5.5	8.3	8.3	6.1	5.4	**5.1**	8.5
TwoNorm	4.2	3.8	3.1	**3.1**	4.0	3.2	4.0	16.0
Vehicle	27.5	26.3	25.8	26.6	27.0	**24.1**	26.2	30.6
Vowel	11.3	11.2	11.7	11.1	17.8	11.9	**10.8**	30.0
Waveform	16.6	15.8	**15.3**	15.4	17.3	16.0	15.7	25.2

Table 5.2: Classification errors (in %) for different ensembles methods on different datasets, bold numbers show the best performance. RD ensembles and ERD ensembles generally perform similar to or better than bagging and quite competitive with Adaboost.M1 and Random Forests.

Dataset	Bagging	AdaBoost.M1	Random Forests	Single Tree
Balance	+	+	+	+
Breast Cancer	Δ	Δ	Δ	+
Ecoli	+	+	+	+
Glass	Δ	Δ	Δ	+
Iono	Δ	+	Δ	+
Letter	+	-	Δ	+
Pendigit	+	Δ	+	+
Pima-Dia	Δ	Δ	Δ	Δ
RingNorm	+	Δ	+	+
Segment	+	-	Δ	+
Sonar	Δ	Δ	Δ	+
Spam	Δ	Δ	Δ	+
TwoNorm	Δ	-	Δ	+
Vehicle	Δ	Δ	Δ	Δ
Vowel	+	Δ	Δ	+
Waveform	+	Δ	Δ	+
win/draw/lose	8/8/0	3/10/3	4/12/0	14/2/0

Table 5.3: Comparison Table- '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm. RD ensembles perform similar to or better than bagging and quite competitive with Adaboost.M1 and Random Forests.

Noise in %	RD (cat.)	RD (cont.)	ERD (cat.)	ERD (cont.)	Bagging	AdaBoost.M1	Random Forests	Single Tree
0	1.6	1.0	1.2	1.0	2.3	**0.9**	1.2	4.7
10	1.6	1.2	1.3	**1.1**	2.1	1.7	1.4	10.9
20	1.7	1.2	1.4	**1.1**	2.4	3.3	1.9	20.8

Table 5.4: Classification errors (in %) for different ensembles methods for the Pendigit dataset with different levels of noise, bold numbers show the best performance.

Noise in %	Bagging	AdaBoost.M1	Random Forests	Single Tree
0	+	Δ	+	+
10	+	+	+	+
20	+	+	+	+

Table 5.5: Comparison table for the Pendigit dataset with different levels of noise - '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm.

the Waveform data, the performance of RD(cont.) is statistically better than the other ensemble methods. For the Waveform data, RD(cont.) is similar to AdaBoost.M1 and Random Forests, whereas it is better than Bagging.

5.5.2 The Study of the Ensemble Size

We studied the effect of ensemble sizes on the performance of ensembles. Fig. 5.4 - 5.7 show the classification errors against ensemble sizes for different ensemble methods for different datasets. We used 5x2 cross validation for this study. The behaviour of RD(cont.) is very similar to that of Random Forests. A single decision tree trained using RD has more classification error as compared to a single J48 tree but as the size of the ensemble is increased, RD ensembles perform better than a single decision tree. This verifies the fact that in RD(cont.), we have diverse classifiers that account

Noise in %	RD (cat.)	RD (cont.)	ERD (cat.)	ERD (cont.)	Bagging	AdaBoost.M1	Random Forests	Single Tree
0	2.7	2.6	4.0	3.8	3.7	**2.3**	2.7	4.6
10	3.5	**3.1**	5.0	4.6	4.2	6.3	4.3	10.8
20	4.2	**4.1**	5.6	5.5	6.4	10.1	6.8	19.7

Table 5.6: Classification errors (in %) for different ensembles methods for the Segment dataset with different levels of noise, bold numbers show the best performance.

Noise in %	Bagging	AdaBoost.M1	Random Forests	Single Tree
0	+	-	Δ	+
10	+	+	+	+
20	+	+	+	+

Table 5.7: Comparison table for the Segment dataset with different levels of noise - '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm.

Noise in %	RD (cat.)	RD (cont.)	ERD (cat.)	ERD (cont.)	Bagging	AdaBoost.M1	Random Forests	Single Tree
0	11.3	11.2	11.7	11.1	17.8	11.9	**10.8**	30.0
10	12.0	11.5	12.6	**11.0**	16.8	16.0	13.8	33.1
20	17.3	**15.6**	16.9	16.1	25.0	24.1	21.6	39.2

Table 5.8: Classification errors (in %) for different ensembles methods for the Vowel dataset with different levels of noise, bold numbers show the best performance.

Noise in %	Bagging	AdaBoost.M1	Random Forests	Single Tree
0	+	Δ	Δ	+
10	+	+	+	+
20	+	+	+	+

Table 5.9: Comparison table for Vowel data with different levels of noise - '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm.

Noise in %	RD (cat.)	RD (cont.)	ERD (cat.)	ERD (cont.)	Bagging	AdaBoost.M1	Random Forests	Single Tree
0	16.6	15.8	15.3	**15.4**	17.3	16.0	15.7	25.2
10	16.6	16.5	15.9	**15.8**	17.8	16.8	16.2	31.3
20	17.5	17.1	16.9	**16.9**	18.3	18.3	16.9	37.4

Table 5.10: Classification errors (in %) for different ensembles methods for the Waveform dataset with different levels of noise, bold numbers show the best performance.

Noise in %	Bagging	AdaBoost.M1	Random Forests	Single Tree
0	+	Δ	Δ	+
10	+	Δ	Δ	+
20	+	Δ	Δ	+

Table 5.11: Comparison table for the Waveform data with different levels of noise - '+/-' shows that performance of RD(cont.) is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RD(cont.) and that algorithm.

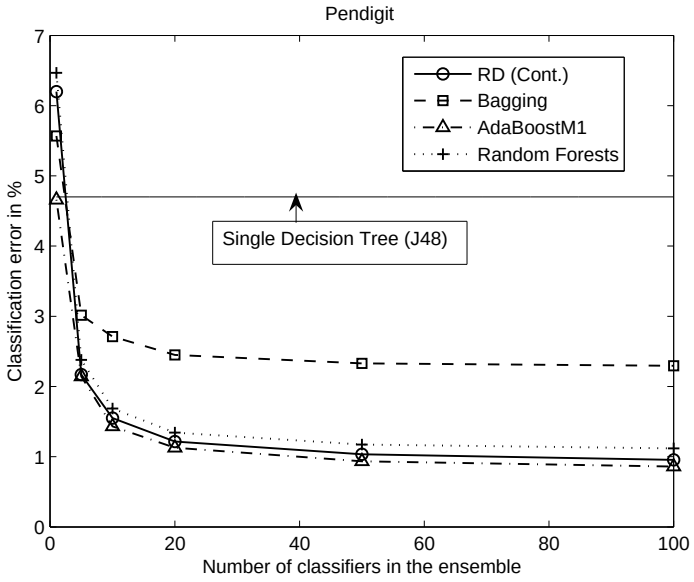


Figure 5.4: Classification errors of various ensemble methods for the Pendigit dataset against the size of the ensemble.

for its better performance. We expect that RD increases the bias and the variance of individual decision trees however the part of the variance is cancelled out by averaging over a sufficiently large number of trees in the RD ensemble. As in RD trees there is a less dependence of nodes splits on the output, there is a increase in the bias and the variance but highly randomized method of discretization is helpful in creating diverse decision trees that reduces the variance in the ensemble.

5.5.3 The Effect of the Number of Discretized Bins

In RD and ERD algorithms, the number of discretized bins is a user defined variable. We carried out experiments with different number of bins to understand its effect on

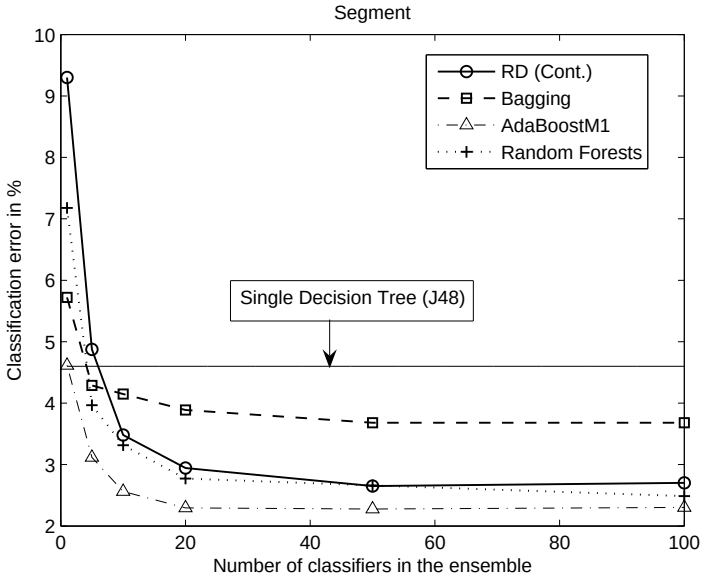


Figure 5.5: Classification errors of various ensemble methods for the Segment dataset against the size of the ensemble.

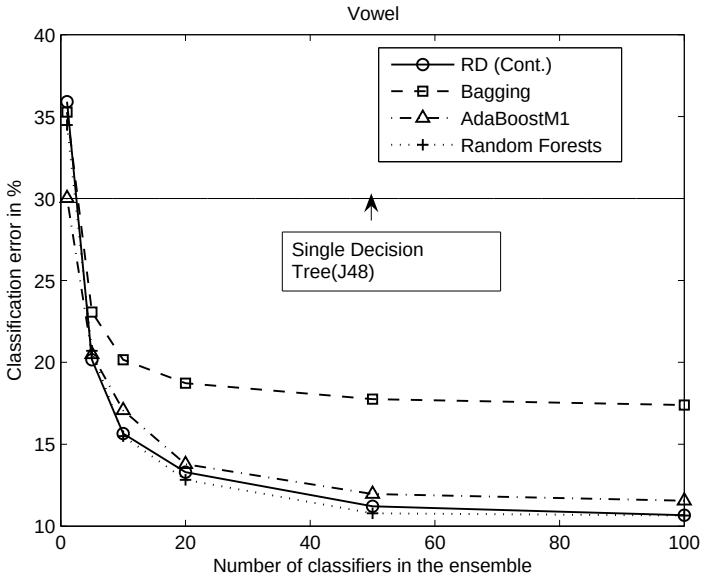


Figure 5.6: Classification error of various ensemble methods for the Vowel dataset against the size of the ensemble.

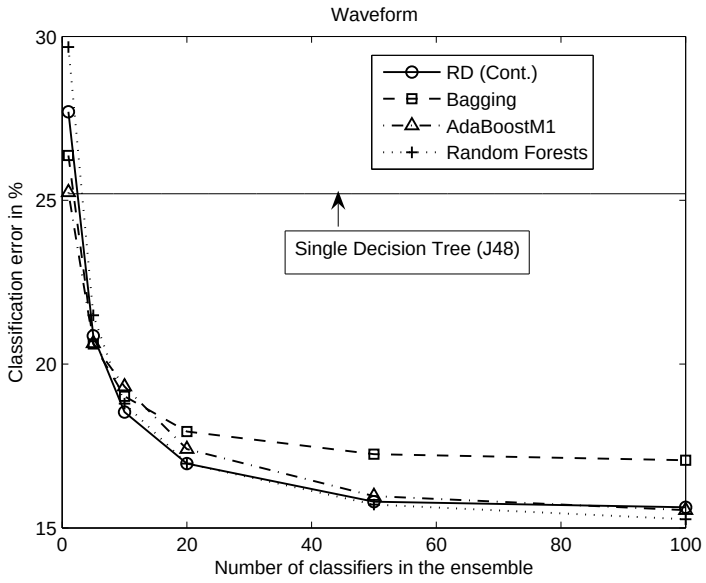


Figure 5.7: Classification error of various ensemble methods for the Waveform dataset against the size of the ensemble.

an ensemble classification accuracy. We used the 5x2 cross validation for this study. The ensemble size was 50.

We tested RD ensembles and ERD ensembles on four datasets (Pendigit, Segment, Vowel and Waveform) with 3, 5 and 10. Results are presented in Table 5.12- 5.15. Results suggest that generally the performance of ensembles is best when we have 5 or 10 bins. These results can be explained easily for RD(cont.) and ERD(cont.). For small number of bins, there will be a large loss of information that leads to poor decision trees accuracy. When we increase the number of bins the discretized data becomes more similar to the original data. For example, if the original data is $\{1,2,3,4,5,6\}$, and we have two equal width bins, the discretized data is $\{1,1,1,2,2,2\}$, whereas, if we have 6 equal width bins, the discretized data will be $\{1,2,3,4,5,6\}$ that is similar to the original data. As the discretized data becomes more similar to the original data, it improves the accuracy of the classifier trained on the discretized data. Results for the single decision trees show this trend. However, a large number of bins reduces the diversity of decision trees as discretized datasets are similar. With 5 and 10 numbers of bins, it seems that we are getting good enough combination of accuracy and diversity.

Results show the similar behaviour in RD(cat.) and ERD(cat.). If the data is categorical, it is difficult to explain this relationship using the above argument as there are two factors working together ; the loss of the information because of the discretization

Number of bins	RD (cat.)	RD (cont.)	ERD (cat.)	ERD (cont.)	Single RD (cat.) Tree	Single RD (cont.) Tree	Single ERD (cat.) Tree	Single ERD (cont.) Tree
3	1.6	1.5	1.5	1.5	9.5	8.9	8.9	8.3
5	1.6	1.0	1.2	1.0	8.7	6.2	8.1	5.7
10	2.2	**0.9**	2.3	1.1	10.2	5.3	11.1	4.8

Table 5.12: Classification errors (in %) for different ensembles methods for the Pendigit dataset with different number of discretized bins. Last four columns show the classification error of a single decision tree, bold numbers show the best performance.

Number of bins	RD (cat.)	RD (cont.)	ERD (cat.)	ERD (cont.)	Single RD (cat.) Tree	Single RD (cont.) Tree	Single ERD (cat.) Tree	Single ERD (cont.) Tree
3	4.2	4.1	6.8	6.9	16.5	16.2	17.7	17.5
5	2.7	2.6	4.0	3.8	10.4	9.3	11.3	10.7
10	3.0	**2.5**	3.2	2.9	8.9	6.8	8.7	7.1

Table 5.13: Classification errors (in %) for different ensembles methods for the Segment dataset with different number of discretized bins. Last four columns show the classification error of a single decision tree, bold numbers show the best performance.

and the data fragmentation problem. If the number of discretized bins is low, there will be more loss of the information, but when the number of discretized bins is increased, classifiers suffer due to the data fragmentation problem.

Different datasets may have a different number of discretized bins for their optimal performance. The validation data may be used to search for it.

5.5.4 The Study of Time/Space Complexities

As discussed in Chapter 3, Catlett [20] suggests the use of a histogram to approximate the split at the node to reduce the time to create a decision tree. We studied the time required by the tree growing phase in RD trees and ERD trees. We also studied the size complexity of RD decision trees to see their relationship with normal decision trees. We used the 5x2 cross validation for this study. In each run, 50 unpruned trees were created and the average results are presented. The number of discretized bins was 5. Results are shown in Table 5.16 and Table 5.17. We observed following behaviour;

1. As expected the time taken in the tree growing phase of RD(cat.) trees and

Number of bins	RD (cat.)	RD (cont.)	ERD (cat.)	ERD (cont.)	Single RD (cat.) Tree	Single RD (cont.) Tree	Single ERD (cat.) Tree	Single ERD (cont.) Tree
3	19.1	12.9	19.6	19.7	44.9	43.9	54.3	54.2
5	11.3	11.2	11.7	**11.1**	40.4	35.9	42.7	40.5
10	12.8	13.1	13.0	10.4	41.6	30.9	40.0	34.2

Table 5.14: Classification errors (in %) for different ensembles methods for the Vowel dataset with different number of discretized bins. Last four columns show the classification error of a single decision tree, bold numbers the show best performance.

Number of bins	RD (cat.)	RD (cont.)	ERD (cat.)	ERD (cont.)	Single RD (cat.) Tree	Single RD (cont.) Tree	Single ERD (cat.) Tree	Single ERD (cont.) Tree
3	15.4	15.4	15.4	**15.3**	30.4	29.5	33.4	33.1
5	16.6	15.8	**15.3**	15.4	30.1	27.7	30.5	29.6
10	19.2	16.8	16.6	15.7	32.1	26.5	30.7	27.3

Table 5.15: Classification errors (in %) for different ensembles methods for the Waveform dataset with different number of discretized bins. Last four columns show the classification error of single decision tree, bold numbers show the best performance.

ERD(cat.) trees when the data is treated as categorical is less than the time taken in a normal decision tree. For example, for the Pendigit data, the average tree growing time for the normal decision tree is 1.75 sec., whereas, the average time for RD(cat.) is 0.39 sec. and for ERD(cat.) is 0.40 sec..

2. RD(cat.) trees and ERD(cat.) trees are shallow; there is not much difference between the size of the trees and the number of leaves. For example, for the Pendigit data the average number of leaves/size of tree for RD trees are 2089/2311, whereas for ERD trees, the average number of leaves/size of tree are 2152/2392. As we have five branches (as the number of attribute values is five) at each node, the branching occurs at higher rate (as compared to binary split) that leads to a large number of leaves.

3. Generally, the tree growing phase for RD(cont.) trees and ERD(cont.) trees is faster than a J48 decision tree. For example, for the Segment data, on average a RD tree took 0.22 sec.(the ERD tree took 0.22 sec.), and a J48 decision tree took 0.29 sec.. In the discretized data, we have few points to evaluate for the best split criterion that leads to a faster tree growing phase.

Name of data	Single RD (cat.)	Single RD (cont.)	Single ERD (cat.)	Single ERD (cont.)	J48 with the original data
Pendigit	0.39	1.71	0.40	1.73	1.75
Segment	0.11	0.22	0.11	0.22	0.29
Vowel	0.09	0.13	0.10	0.13	0.20
Waveform	0.51	1.53	0.55	1.72	2.21

Table 5.16: Time in sec. taken in the tree growing phase for different trees.

Name of data	Single RD (cat.) num. of leaves/ size	Single RD (cont.) num. of leaves/ size	Single ERD (cat.) num. of leaves/ size	Single ERD (cont.) num. of leaves/ size	J48 with the org. data num. of leaves/ size
Pendigit	2089/2311	227/453	2152/2392	209/417	151/300
Segment	532/591	76/151	388/431	52/103	32/63
Vowel	622/691	107/213	658/731	102/203	68/135
Waveform	856/951	242/483	1045/1161	391/781	185/369

Table 5.17: Complexities of different trees.

4. The tree size complexity of RD (cont.) and ERD(cont.) trees is greater than the normal J48 decision trees, for example, for the Pendigit data, the average number of leaves/size of RD tree are 227/453 (the average number of leaves/size of tree of ERD trees are 209/417), whereas for the normal decision trees, the average number of leaves/size of tree are 151/300. As we get better node splits in normal decision trees, it leads to shorter decision trees, whereas, split points in RD trees and ERD trees are not optimal as we are using discretized data (the discretiztion leads to the loss of the information). This leads to complex decision trees. This also explain why the tree growing phase for the Pendigit data for RD(cont.) trees and ERD(cont.) trees took almost the same time as a normal decision tree. The time, taken in the tree growing phase for a tree, is decided by the two factors, one the time taken in a node split and the second the number of nodes in the tree. In RD(cont.) and ERD(cont.) trees a node split takes less time, but the higher size complexity of the tree increases the time for the tree growing phase. In the Pendigit data, it appears that both terms neutralize each other and we get almost the same tree growing phase time for RD trees as the normal decision tree.

5.6 Combining Random discretized Ensembles with Multi-RLE

In Chapter 4, we presented the Multi-RLE framework that is useful for improving different ensemble methods. In this section, we present Random Projection Random Discretized Ensembles (RPRDE) that combines RD with the Multi-RLE technique. In RD, we create a m dimensional discretized dataset (for m dimensional dataset), whereas in Multi-RLE a d dimensional space is created by using RP and combined with the original attributes. *In RPRDE, this d dimensional space is concatenated with the m dimensional discretized dataset*. A univariate decision tree is trained on this $m+d$ dimensional dataset. Though, we train a univariate decision tree, we get decision surfaces both orthogonal (due to the m discretized attributes) and oblique to the axes defined by the attributes of the input space (due to the new d attributes). The algorithm to create RPRD ensembles is presented in Fig. 5.8.

Experiments suggest that C4.5 trees do not do well with random projections [38]. As discussed in Chapter 3, Fradkin and Madigan[38] suggests "Random projections and decision trees are perhaps not a good combination". This means that new attributes created by using random projections are not as informative as the original attributes. Hence, when we combine the original attributes with the attributes created by using random projections and train a univariate decision tree on it, then there is a strong probability, that the original attributes are selected at higher levels as they are more informative, whereas the attributes created using random projections will be selected at lower levels as they are less informative. This suggests that these trees are not very diverse (as they are similar at the higher levels).

In RPRD trees, the discretized original attributes are used. As there is a loss of information due to the discretization, it makes the original attributes less informative. Hence, when the discretized attributes are used, then there is more probability that attributes created by using random projections will be selected at higher levels of decision trees. That ensures more diverse trees as different trees use different new attributes (different trees use different attributes created by using different random projections).

Input- Original dataset *T* with m continuous features and k classes $(C_1, C_2, .., C_k)$. M the size of the ensemble.
Training Phase
for i=1...M **do**
Data Generation
1- Use Random Discretization Λ_i to create a *m* dimensional discretized dataset.
2- Use Random Projection R_i to create a *d* dimensional dataset.
3- Combine S_i and R_i datasets to get *m* + *d* dimensional dataset T_i.
Learning Phase
Treating dataset T_i as continuous, learn D_i decision tree on it.
end for
Classification Phase
For a given data point **x**
for i=1...M **do**
1- Convert **x** into *m* + *d* dimensional data point $\mathbf{x_i}$ by using Random Discretization Λ_i and Random Projection R_i.
2- Let $p_{i,j}(x)$ be the probability for $\mathbf{x_i}$ by the decision tree D_i to the hypothesis that **x** comes from class C_j. Calculate $p_{i,j}(x)$ for all classes (j = 1..k).
end for
Calculate the confidence $P(j)$ for each class C_j (j = 1..k) by the average contribution method, $P(j) = \frac{1}{L}\sum_{i=1}^{M} p_{i,j}(x)$.
The class with the largest confidence will be the class of **x**.

Figure 5.8: RPRDE algorithm. In this method attributes created by using RD and by using RP are concatenated.

5.7 Motivation for Random Projection Random Discretization Ensembles (RPRDE)

In this section, we discuss why RPRDE should perform well:

1. We discussed in the last chapter (by using the random linear oracle framework) that combining new random attributes (these attributes are the linear combinations of the original attributes) with the original attributes can improve the performance of different ensemble techniques. Therefore, we expect that by combining the attributes created by using random projection to the discretized attributes (created by using RD) may improve the performance of RD ensembles.

2. As discussed in Chapter 2, some of the ensemble methods combine ensemble

methods that have different mechanisms, for example Random Forests [13] combine Bagging with Random Subspaces, MultiBoosting [95] combines Bagging with AdaBoost and Rotation Forest [83] combines randomization in the attribute space division with Bagging. RD and RP have different mechanisms; RD is based on the random discretization of the attributes whereas RP creates random attributes that are the linear combinations of the original attributes. Hence, combining these two approaches may be beneficial.

3. As some of the real datasets have class noise, it is important that ensemble methods should be robust to the class noise to handle these kinds of datasets. AdaBoost.M1 is a quite successful ensemble algorithm, however, its performance degrades in the presence of the class noise whereas ensemble algorithms based on pure randomization are quite robust to the class noise [30, 13]. RPRDE uses two randomization processes; random projections and random discretization (proposed in the thesis). Similar to the other ensemble methods that are created by using only randomized process, RPRDE is expected to be quite robust to the class noise.

5.8 Experiments

We carried out a comparative study of RPRD ensembles against the other popular ensemble methods to test the effectiveness of the RPRDE approach. In this section, we present our experimental results.

We did experiments with unpruned J48 (the Weka implementation of C4.5) decision trees. We carried out experiments with Bagging, Adaboost.M1, MultiBoosting using J48 (unpruned) as the base model, and Random Forests. The sizes of the ensembles were 10 and 100 for these experiments. For MultiBoosting, when the ensemble size was 10, the number of subcommittees was chosen as 3 whereas for ensemble size 100, the number of subcommittees was chosen as 10. Default settings were used for the rest of the parameters. Datasets were normalized to bring all attributes on the same scale. We also present results with RP ensembles in which each classifier is trained on the d dimensional space created by RP. The experiments were conducted following the 5×2 cross-validation [28, 4] as discussed in Chapter 4.

5.8.1 Parameters for RPRDE

There are three parameters for RPRDE;

1. **Number of bins** - As in Chapter 5, 5 bins are created by using 4 data points from the training data for the discretization process. For each tree in the ensemble, different 4 points were selected randomly.

2. **Dimension d of the datasets created using RP** - As discussed in Chapter 3, Dasgupta [22] shows that the data from a mixture of J Gaussians can be projected into just $O(\log J)$ dimensions while retaining the approximate level of separation between clusters. We selected d as $2(\log m)$ where m is the number of attributes. As in almost all datasets we tested, the number of classes (k) $<$ the number of attributes(m) and it is assumed that each class probability distribution is represented by a Gaussian distribution. Factor 2 was taken to have a large dimension of the projected data so that the most of the information is preserved. There was no guarantee that with this assumption, the correct value of d is obtained. However, these new d attributes were added to the original attributes (discretized attributes) so even if these new d attributes did not have all the information of the dataset, it was expected that their combination with original attributes improved the representational power of the decision trees.

3. **Matrix for the Random Projection-** We used following random matrix R as discussed in Chapter 3,

$$\mathrm{r}_{ij} = \pm\sqrt{3} \text{ with probability 1/6 each or 0 with 2/3 probability.} \quad (5.1)$$

5.8.2 Controlled Experiment

As discussed in Chapter 2, univariate decision trees like C4.5 have difficulty in learning non-orthogonal decision boundary. This experiment was carried out to study the performance of RPRD ensembles for learning a non-orthogonal concept. We did experiment with a diagonal decision boundary.

We tested different ensemble methods on a simulated dataset. It was a 10 dimensional data having a 5 dimensional diagonal concept. The i^{th} attribute of the pattern x was defined by x_i, (i = 1 to 10) where x_i is a random number between 0 to 1. Two classes were defined as

Size of the ensemble	RPRDE	RD	RP	Bagging	AdaBoostM1	MultiBoost	Random Forests	Single Tree
10	**8.01**	8.93	10.65	10.44	10.16	9.68	11.07	15.76
100	**4.92**	6.41	5.09	9.12	6.95	6.31	8.30	15.76

Table 5.18: Classification errors with the simulated data, bold numbers show the best results. Results suggest that RPRDE ensembles can learn a diagonal problem very well. This shows that these ensembles have good representational power.

$$\sum_{i=1}^{5} x_i \leq 5/2, \tag{5.2}$$

and

$$\sum_{i=1}^{5} x_i > 5/2. \tag{5.3}$$

We created 2000 data points. Experiments were done using 5×2 cross-validation. Results are presented in Table 5.18. Test results indicate that RPRD ensembles perform statistically better than other popular ensemble methods. This shows that RPRD ensembles can learn non-orthogonal concepts very well. This vindicates our hypothesis that RPRD ensembles have good representational power.

5.8.3 Comparative Study

In the second part of the experiments, we selected 21 pure continuous data sets from the UCI Machine Learning Repository. The information about the datasets is presented in Table A.2.

For the ensemble size 10, results are presented in Table 5.19 and Table A.3. RPRDE is statistically similar to or better than other popular ensemble methods (Bagging (11 Wins/10 Draws), AdaBoost.M1 (9 Wins/12 Draws), MultiBoosting (9 Wins/12 Draws) and Random Forests (9 Wins/12 Draws)). For the ensemble size 100, results are presented in Table 5.20 and Table A.4. For the ensemble size 100, generally RPRDE performs similar to or better than other popular ensembles methods, however, its comparative advantage decreases (Bagging (13 Wins/8 Draws), AdaBoost.M1 (7 Wins/13 Draws/ 1 Loss), MultiBoosting (8 Wins/13 Draws/1 Loss) and Random Forests (5 Wins/16 Draws)).

RPRDE is the combination of two ensemble methods; RP ensembles and RD ensembles. Our motivation, to combine these two approaches, is that they are based on

different mechanisms so their combination may produce good results. Results indicate that for almost all the dataset (except the RingNorm dataset, RP performed statistically better than RPRD when the ensemble size was 100) RPRDE has better results than both of these methods or similar to the better one. For example, when ensemble size was 10, for Phoneme dataset, Pendigit dataset, Segment dataset and Letter dataset, RPRD performs better than both the methods, whereas for Optical dataset and Spambase data, it is similar to RD ensembles and for RingNorm dataset, it is similar to the RP ensembles. This behaviour suggests that RPRD ensembles have got best of both methods.

Experimental results suggest that RPRD ensembles are comparatively better when ensemble sizes are small. This is true with ensembles consisting of accurate classifiers with reasonable diversity. To understand the diversity-error behaviour of RPRD ensembles, we study kappa-error plots [72] of various ensemble methods.

5.8.4 The Study of Ensemble Diversity

Kappa-error plots [72] is a method to understand the diversity-error behaviour of an ensemble (for detail see Appendix 1.3). These plots represent a point for each pair of classifiers in the ensemble. The *x* coordinate is a measure of diversity of the two classifiers D_i and D_j known as the *kappa* measure (low values suggest high diversity). The *y* coordinate is the average error $E_{i,j}$ of the two classifiers D_i and D_j. When the agreement of the two classifiers equals that expected by chance, $\kappa = 0$; when they agree on every instance, $\kappa = 1$. Negative values of κ mean a systematic disagreement between the two classifiers.

We draw kappa-error plots of for five datasets (Pen, Phoneme, Segment, Vowel, Waveform40) for different ensemble methods. The scales of κ and $E_{i,j}$ are same for each given dataset so we can easily compare different ensemble methods. The size of the ensembles was 10, so the total number of points was 45 in each plot. Plots are presented in Fig. 5.9. RPRDE is not as diverse as AdaBoost.M1, MutiBoosting and Random Forests, however, generally RPRDE is more diverse than Bagging. Classifiers created by using RPRDE and Bagging generally have similar accuracy performance, whereas they are generally more accurate than all other ensemble methods. One may conclude that RPRDE behaviour is midway between these two types of methods Bagging (classifiers; more accurate, less diverse) and Adaboost.M1 (classifiers; less accurate, more diverse). RPRDE is able to improve diversity, but to a lesser degree than Adaboost.M1 and Random Forests, without affecting accuracy of individual classifiers

Data	RPRDE	RD	RP	Bagging	Ada BoostM1	MultiBoost	Random Forests	Single Tree
Balance	**13.41**	15.04	14.88	18.40	20.51	19.30	19.75	21.95
Breast Cancer	**3.41**	3.67	**3.41**	4.64	4.04	4.23	4.12	5.87
Ecoli	16.13	16.72	**14.99**	18.45	18.24	18.09	18.45	20.65
Glass	29.34	29.35	32.34	31.12	31.04	29.90	**26.54**	34.02
Ionosphere	7.06	7.26	**6.04**	7.47	7.92	9.57	7.52	10.48
Letter	**6.56**	7.32	13.32	9.94	6.63	7.78	7.94	15.47
Optical	3.48	3.77	5.20	5.71	**3.22**	3.90	4.17	11.03
Pendigit	**1.06**	1.41	1.50	2.85	1.37	1.74	1.53	4.69
Phoneme	**11.40**	14.54	14.86	13.04	12.37	12.45	11.99	15.63
Pima-diabetes	24.89	25.39	24.97	**24.82**	27.08	26.01	25.72	26.89
RingNorm	3.24	4.41	**3.23**	6.04	4.14	4.91	5.84	10.10
Satimage	**10.38**	10.70	11.02	11.38	10.67	10.88	10.62	15.39
Segment	**2.68**	3.36	3.42	4.11	2.71	3.23	3.28	4.51
Sonar	**21.54**	21.63	23.37	24.61	24.90	24.13	23.87	27.88
Spam	5.92	5.94	10.03	6.13	5.53	**5.31**	5.99	8.16
TwoNorm	**3.71**	5.73	4.28	6.23	5.90	6.16	6.23	16.16
Vehicle	26.70	26.55	30.71	27.07	**26.33**	27.03	27.30	30.36
Vowel	**11.39**	12.87	13.23	20.49	15.45	18.56	15.31	30.24
Waveform21	**16.99**	17.52	16.82	18.67	18.86	18.32	18.50	24.45
Waveform40	**17.70**	18.11	20.26	18.92	19.24	18.66	19.17	25.40
Yeast	**42.06**	43.60	42.78	42.21	44.92	42.65	43.45	47.25

Table 5.19: Classification error(in %) for different ensembles methods on different dataset, bold numbers show best performance. Ensemble size 10.

Data	RPRDE	RD	RP	Bagging	Ada BoostM1	MultiBoost	Random Forests	Single Tree
Balance	11.30	13.56	**11.26**	18.18	21.86	20.35	18.91	21.95
Breast Cancer	3.23	3.38	**2.98**	4.69	3.61	3.49	3.89	5.87
Ecoli	15.06	**14.40**	14.48	16.79	16.49	16.40	16.79	20.65
Glass	27.85	27.29	30.00	28.88	27.57	28.60	**24.67**	34.02
Ionosphere	5.81	6.56	**5.53**	6.89	7.86	7.79	6.50	10.48
Letter	4.22	4.93	6.39	8.52	**4.08**	4.10	4.92	15.47
Optical	2.45	2.76	2.11	4.60	**1.91**	2.06	2.11	11.03
Pendigit	**0.73**	0.93	0.95	2.29	0.83	0.91	1.08	4.70
Phoneme	10.54	13.96	13.86	12.21	**10.48**	10.50	10.61	15.63
Pima-diabetes	23.57	23.62	25.47	**23.36**	26.20	25.21	23.88	26.85
RingNorm	1.90	2.61	**1.55**	5.30	2.27	2.44	4.45	9.82
Satimage	9.11	9.47	9.90	10.40	**8.91**	9.02	9.21	15.39
Segment	**2.07**	2.77	2.82	3.70	2.13	2.40	2.55	4.51
Sonar	18.85	19.90	**18.17**	22.50	20.58	22.01	19.04	27.88
Spam	5.45	5.59	8.54	5.92	5.29	4.42	5.06	8.16
TwoNorm	**2.47**	3.64	**2.47**	3.69	2.83	2.87	3.65	16.16
Vehicle	24.42	24.97	29.43	26.87	**23.86**	24.04	25.90	30.36
Vowel	7.23	8.67	**7.17**	17.23	10.77	11.27	9.88	30.24
Waveform21	14.64	15.41	**14.36**	17.14	15.62	15.61	15.62	24.45
Waveform40	15.28	15.58	**15.18**	17.29	15.26	15.26	15.49	25.40
Yeast	39.50	40.04	**39.49**	40.07	42.51	41.33	40.61	47.25

Table 5.20: Classification errors (in %) for different ensemble methods on different datasets, bold numbers show best performance, the ensemble size 100. RPRD ensembles generally perform similar to or better than other ensemble methods, however, their competitive advantage is more for smaller ensembles.

Dataset	RPRDE	SVM	AdaBoost with RBF-Network
Banana	12.11	**11.53**	12.26
Breast-Cancer	28.44	**26.04**	30.36
Daibetes	24.34	**23.53**	26.47
Flare-Solar	35.41	**32.43**	35.70
German Credit	**23.32**	23.61	27.45
Heart	17.73	**15.95**	20.29
Image	**1.71**	2.96	2.73
Ring-Norm	3.03	**1.66**	1.93
Splice	**4.46**	10.88	10.14
Thyroid	**4.35**	4.80	4.40
Titanic	**22.29**	22.42	22.58
Two-Norm	**2.81**	2.96	3.03
Waveform	10.39	**9.88**	10.84

Table 5.21: Average classification errors (in %) of different methods on different datasets, bold numbers show the best performance.

as much as Adaboost.M1, Random Forests and MutiBoosting.

Kappa-error plots indicate that *accurate classifiers with reasonable diversity* is the reason for the success of RPRDE.

5.8.5 RPRDE against the Other Classifiers

We carried out the comparative study of RPRDE against other classifiers. We selected support vector machines (SVM) for this comparison. We also present results for AdaBoost with RBF-Network. The information about different datasets used in the experiments is given in [82]. We carried out experiments on all the realizations of different datasets given in http://ida.first.fraunhofer.de/projects/bench/. Classification results for SVM and AdaBoost with RBF-Network were taken from [82]. As the size of the AdaBoost with RBF-Network was 200 [82], the size of RPRD ensembles was chosen to be 200. Results are presented in Table 5.21. Results suggest RPRDE is quite competitive against SVM.

5.8.6 Noisy Data

As most real datasets have class noise, it is important to understand the robustness of RPRD ensembles for the noisy data. As suggested in Chapter 2, the performance of boosting methods degrade in the presence of the class noise. In this section, we present

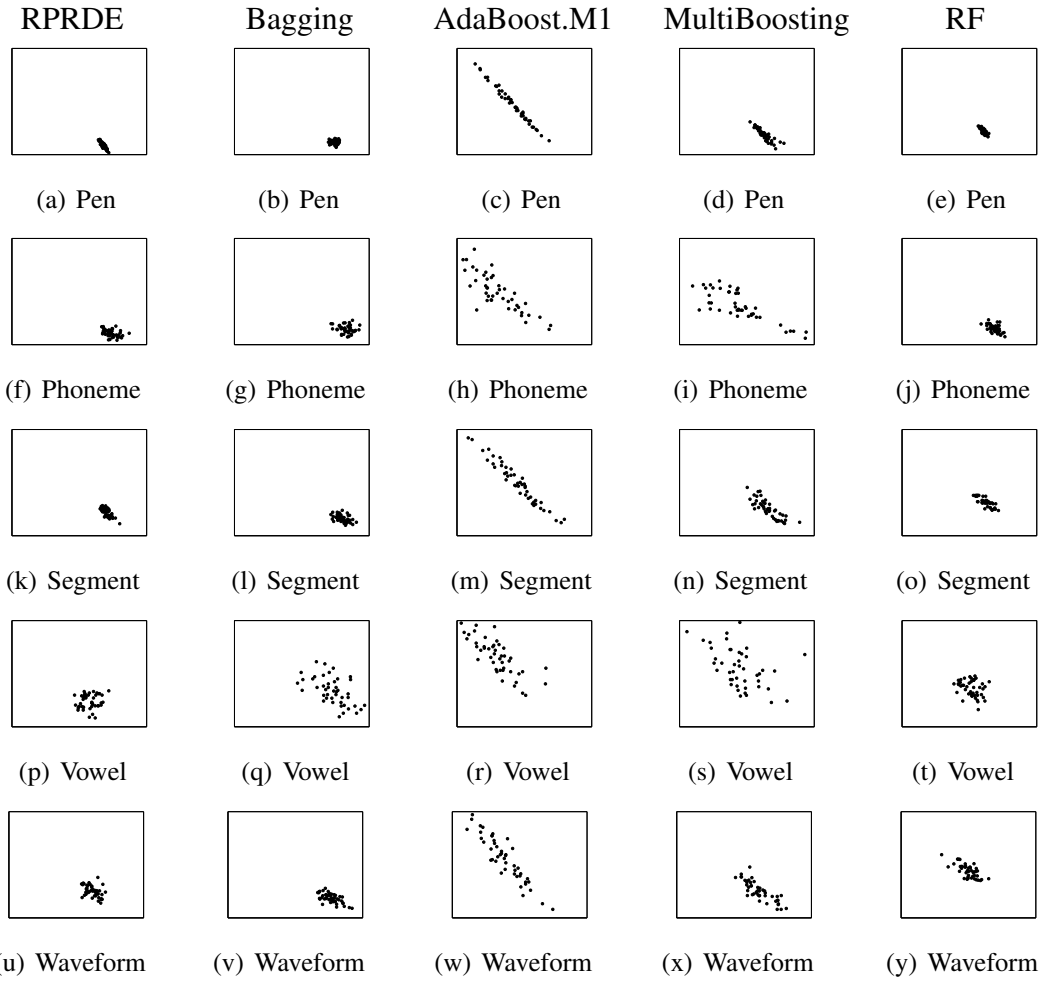


Figure 5.9: Kappa-error plots for four ensemble methods, First column- RPRDE, second column - Bagging, third column - AdaBoost.M1, fourth column - MultiBoosting and last column RF. x-axis - Kappa, y-axis - the average error of the pair of classifiers. Axes scales are constant for various ensemble methods for a particular dataset (each row). Lower κ represents a higher diversity. The plots suggest that RPRDE classifiers are accurate with reasonable diversity.

our experimental results to study the sensitivity of RPRDE to the class noise. To add noise to the class labels, we followed the method proposed by Dietterich [30]. To add classification noise at a rate r, we chose a fraction r of the instances and changed their class labels to be incorrect, choosing uniformly from the set of incorrect labels. We carried out experiments for the noise levels 10% for all the datasets.

Results, when the ensemble size was 10, are presented in Table 5.22 and A.5. Results indicate that RPRDE is quite robust to the class noise. Its comparative advantage increases (as compared to without noise data) for the noisy data. Except Bagging (the performance of RPRDE is statistically worse than Bagging for the Spambase dataset) RPRD performed statistically similar to or better than other ensemble methods (Bagging (12 Wins/8 Draws/ 1 Loss), AdaBoost.M1 (16 Wins/5 Draws), MultiBoosting (15 Wins/6 Draws/1 Loss) and Random Forests (18 Wins/8 Draws)).

We also carried out same experiments when the size of ensemble was 100. Results are presented in Table 5.23 and Table A.6. Except Bagging (the performance of RPRDE is statistically worse than Bagging for the Spambase data) RPRDE performed statistically similar to or better than other ensembles methods, however, its comparative advantage decreases (as compared to ensembles of size 10) (Bagging (13 Wins/8 Draws), AdaBoost.M1 (14 Wins/7 Draws), MultiBoosting (11 Wins/10 Draws) and Random Forests (10 Wins/11 Draws)).

Results demonstrate that RPRDE is quite robust to the class noise. This is probably due to the fact that RPRDE does not put so much emphasis on incorrectly classified instances as AdaBoost does. Overfitting is one of the weaknesses of oblique decision trees [59]. We create ensembles of RPRD trees this helps in avoiding overfitting problem that is associated with single oblique decision tree as in an ensemble the final result is the combination of all trees. For attributes created by RD, decision boundaries are created without considering the output. During the learning phase, the output is used only to decide on the split among these boundaries. The less participation of the output in generating a RPRDE tree is one of the probable reason for its robustness to the class noise.

5.8.7 Combining RPRD with Other Ensemble Methods

In this section, we study how two popular ensemble methods, Bagging and AdaBoost.M1 get affected by RPRD. As discussed in Chapter 2, different ensemble methods are combinations of two ensemble methods. We follow the philosophy of Multiboosting (discussed in Chapter 2) to combine RPRD with Bagging and AdaBoost.M1. We created

Data	RPRDE	RD	Bagging	Ada Boost.M1	MultiBoosting	Random Forests	Single Tree
Balance	**15.49**	15.34	19.74	26.07	21.82	24.31	25.14
Breast Cancer	5.67	**5.47**	6.72	8.75	6.35	7.63	8.49
Ecoli	17.69	**17.40**	19.70	25.27	21.42	21.18	24.79
Glass	31.94	**31.39**	34.63	37.13	37.13	**31.94**	40.93
Ionosphere	10.85	10.68	11.59	14.09	13.58	11.19	16.42
Letter	**7.81**	8.63	11.15	12.92	11.41	11.81	18.41
Optical	**3.94**	4.40	6.70	6.01	6.15	5.35	18.43
Pendigit	**1.29**	1.42	3.42	3.46	3.06	2.61	11.42
Phoneme	**13.49**	15.22	14.71	18.44	15.24	15.09	18.80
Pima-diabetes	28.44	27.69	**27.32**	29.50	28.04	28.29	29.66
Ring-Norm	**5.06**	6.48	6.43	8.57	6.72	7.08	12.29
Satimage	**10.80**	11.24	12.35	12.77	12.17	11.47	22.98
Segment	**3.36**	3.54	5.20	7.02	5.61	5.66	10.93
Sonar	**27.59**	27.02	29.81	30.96	30.96	24.42	34.61
Spam	9.62	8.63	8.36	11.38	**8.96**	8.80	11.63
Two-Norm	**5.36**	7.21	7.36	10.32	7.89	8.12	17.88
Vehicle	28.04	27.71	28.42	**29.69**	28.02	29.33	34.34
Vowel	**15.40**	18.31	23.30	22.52	23.06	22.92	33.49
Waveform21	**18.11**	18.70	19.78	21.08	19.78	19.84	29.45
Waveform40	**19.05**	19.13	20.08	21.22	20.30	20.49	31.68
Yeast	42.90	**42.56**	43.67	47.83	45.35	45.16	52.17

Table 5.22: Classification errors (in %) for different ensemble methods on different datasets, bold numbers show best performance, the ensemble size 10, the class noise is 10%. RPRD ensembles generally perform similar to or better than other ensemble methods and their competitive advantage is more for the noisy data.

Data	RPRDE	RD	Bagging	Ada Boost.M1	MultiBoosting	Random Forests	Single Tree
Balance	**13.61**	13.86	18.85	26.35	23.41	21.72	25.14
Breast Cancer	**4.98**	5.32	5.90	8.69	8.57	6.18	8.49
Ecoli	15.44	**15.62**	18.22	22.89	21.48	18.58	24.79
Glass	29.63	**28.61**	31.94	34.72	33.89	30.09	40.93
Ionosphere	**8.92**	9.82	10.68	11.76	11.59	9.54	16.42
Letter	**4.93**	5.67	8.63	11.52	8.51	7.74	18.41
Optical	2.58	2.82	4.66	2.27	2.45	**2.22**	18.43
Pendigit	**0.71**	1.03	2.48	1.24	1.20	1.21	11.41
Phoneme	12.38	14.69	13.85	**18.44**	18.44	13.28	18.80
Pima-diabetes	25.63	26.08	**25.77**	29.56	29.40	26.36	29.66
Ring-Norm	**1.98**	2.77	4.70	3.00	2.74	2.99	12.29
Satimage	9.36	9.69	10.84	**9.40**	9.54	9.33	22.99
Segment	**2.73**	2.84	4.29	6.20	4.52	4.20	10.94
Sonar	25.96	26.06	27.5	26.92	27.12	**22.12**	34.61
Spam	7.98	7.76	7.73	10.48	9.15	**7.04**	11.63
Two-Norm	**2.74**	3.53	3.82	3.90	3.59	3.62	17.88
Vehicle	25.97	26.56	27.24	**26.70**	26.60	27.33	34.34
Vowel	**11.02**	12.33	20.06	18.29	17.86	15.46	33.49
Waveform21	**15.07**	15.89	17.15	16.23	15.70	16.04	29.45
Waveform40	15.51	15.75	17.46	15.83	15.46	**15.42**	31.68
Yeast	**40.64**	40.66	42.25	45.84	43.97	42.39	52.17

Table 5.23: Classification errors (in %) for different ensemble methods on different datasets, bold numbers show best performance, the ensemble size 100, the class noise is 10%. RPRD ensembles generally perform similar to or better than other ensemble methods, however, their competitive advantage is more for smaller ensembles.

Dataset	Bagging	Bagging + RPRD
Balance	18.18 (+)	12.93
Breast Cancer	4.69	4.32
Ecoli	16.79	14.64
Glass	28.88	28.69
Ionosphere	7.88	6.63
Letter	8.24 (+)	5.61
Optical	4.59 (+)	2.74
Pendigit	2.29 (+)	0.96
Pima-diabetes	23.36	23.98
Phoneme	12.21 (+)	11.18
Ring-Norm	5.3 (+)0	2.31
Satimage	10.41 (+)	9.72
Segment	3.70 (+)	2.48
Sonar	22.5	20.19
Spam	5.92	5.70
Two-Norm	3.69 (+)	2.75
Vehicle	26.87	23.12
Vowel	17.23 (+)	9.56
Waveform21	17.14 (+)	14.82
Waveform40	17.29 (+)	15.64
Yeast	40.07	39.64
win/draw/loss		12/9/0

Table 5.24: Comparative study of Bagging against RPRD + Bagging. '+/-' shows that performance of RPRD + Bagging is statistically better/worse than Bagging for that dataset. or most of the data studied, the combination of RPRD with Bagging has positive effect.

100 trees with the original data using Bagging process. In the second process, we created 10 different datasets using RPRD and 10 trees using Bagging are created for each dataset. Hence in both cases 100 trees are trained. The same procedure was followed with AdaBoost.M1. Experiments were carried out with the same 5×2 cross validations methodology as suggested Chapter 4. Results (Table 5.24 and Table 5.25) suggest that for some datasets RPRD has positive effect on Bagging (12Wins/9 Draws for Bagging + RPRD) and AdaBoost.M1 (5 Wins/15 Draws for AdaBoost.M1 + RPRD). This indicates that RPRD can be combined with other ensemble methods to improve their performance.

Dataset	AdaBoost.M1	AdaBoost.M1 + RPRD
Balance	21.86 (+)	14.50
Breast Cancer	3.61	3.48
Ecoli	16.49	15.77
Glass	27.57	26.36
Ionosphere	7.84 (+)	5.93
Letter	4.12	4.49
Optical	1.91	1.87
Pendigit	0.83	0.73
Pima-diabetes	26.2	25.13
Phoneme	10.48	10.38
Ring-Norm	2.27	2.08
Satimage	9.03	8.91
Segment	2.13	2.06
Sonar	20.58 (+)	16.54
Spam	5.29 (+)	4.52
Two-Norm	2.83	2.71
Vehicle	23.86	24.83
Vowel	10.77 (+)	7.47
Waveform21	15.62	15.01
Waveform40	15.26	15.41
Yeast	42.51	41.09
win/draw/loss		5/16/0

Table 5.25: Comparative study of AdaBoost.M1 against RPRD + AdaBoost.M1. '+/-' shows that performance of RPRD + AdaBoost.M1 is statistically better/worse than AdaBoost.M1 for that dataset. The combination of RPRD with AdaBoost.M1 is less successful than the combination of RPRD with Bagging.

5.9 Weaknesses

Both RD and RP can be applied only for the pure continuous datasets. That restricts the application of RPRDE only for the pure continuous datasets. In this approach, we use random projections to create new attributes that add extra computational cost. These new attributes are added to the original attributes that increase the size of the training dataset. Hence, the tree learning phase may need more computational resources as compared to the data with the original dataset. However, the performance of RPRDE justifies the additional computational cost.

5.10 Conclusion

Discretization is a popular data transformation technique in which real data is converted into categorical data by creating category boundaries. In this chapter, we presented two discretization methods that create diverse discretized data. In these methods, category boundaries are created randomly. Different decision trees trained on these diverse datasets are diverse as they have different split points. We showed that decision tree ensembles created by using these two methods can approximate complex decision boundaries, hence, address the representational problem of decision trees. These ensembles are quite competitive to other popular ensemble methods. These ensembles are quite robust to the class noise. The less participation of the output in deciding on the split point is the probable reason for this behaviour.

We also presented RPRD ensembles. That is based on RD and Multi-RLE. We discussed the reasons for the success of RPRD ensembles. Similar to the ensemble methods that combine techniques that have different mechanisms, the combination of two schemes; random discretization and random projections, are the reason for the success of this technique.

In this chapter, we showed that the discretization process can be used to create decision tree ensembles that addresses the representational problem and the discretization process can be combined with random projections to create a successful ensemble method. In the next chapter, we discuss a data transformation scheme, random ordinality that is used to create decision tree ensembles. These ensembles reduce the data fragmentation problem associated with decision trees.

Chapter 6

A Novel Ensemble Method to Reduce the Data Fragmentation Problem

In the last two chapters, we use random projections and the discretization process to address the representational problem of decision trees. In this chapter, we introduce a data transformation technique, random ordinality (RO). to reduce data fragmentation problem associated with decision trees. This technique works for multi-valued categorical datasets. It is based on random projection of the *categorical* data into a *continuous* space. We show how RO can be used to create ensembles of binary decision trees that resist the data fragmentation problem. We also present the study of RO attributes using the information theoretic framework. We compare the RO ensemble method against the other popular ensemble methods. We then present the analysis of RO ensembles.

6.1 Data fragmentation problem

Variables having categories without a natural ordering are called categorical [3]. As discussed in Chapter 2, datasets with multi-valued categorical attributes can cause major problems for decision trees. While multi-way splits produce a more comprehensible tree, they may increase the *data fragmentation* problem [94]; the continuous partitioning of the training set at every tree node reduces the number of examples at lower-level nodes. As decisions in the lower levels nodes are based on increasingly smaller fragments of the data, some of them may not have much statistical significance. Creating binary splits by splitting the attribute values into two groups is a method to avoid multi-splits. Breiman [14] suggests exhaustive search to find the best binary split. If the number of attribute values is $|A|$ then the number of nontrivial binary splits is given

by $2^{(|A|-1)} - 1$. Selecting the best split by this method is computationally expensive. Another way to obtain a binary split for a multi-valued categorical attribute is to partition the data points using an attribute value [14, 56, 35]. In this method, all the data points with that attribute value form one group, whereas the other group is formed with the other examples.

Motivated by the advantages of binary decision trees for *multi-valued categorical data*, in the proposed work we build classifier ensembles of binary decision trees for these kinds of datasets. We solve the node splitting problem under some random constraints. These random constraints are helpful in building classifier ensembles as the randomization helps in creating diversity. Using this method, it is not necessarily true that we get the best split as suggested by Breiman [14] (the best split from all possible binary splits). However, since we want to create an ensemble, different node splits are necessary to create diverse decision trees. Furthermore there is no change in the tree building process so there is no extra computational cost for the tree building phase. We discuss this algorithm in the next section.

6.2 Random Ordinality Ensembles

The handling of categorical attributes is difficult as the categories have no *intrinsic order*. We can exploit this property to build an ensemble of binary decision trees. Our method is based on data manipulation so it is not specific to any split criterion. Random Ordinality (RO) creates diverse *training datasets*. The learning process is exactly a decision tree on standard *continuous data* - each binary split gives maximum information gain based on the selected split criterion, but based on an imposed ordinality.

6.2.1 Data Generation

As there is no natural order given for the categorical attribute values, we can *enforce random ordinality* on these attribute values (Fig. 6.1). This implies a random projection of the categorical attributes into a continuous space. We explain our method using the example data given in Table 6.1. This data has four attribute values (Cow, Dog, Cat, Rat) for one of its attributes (attribute 1). We assign some integer number (1 to *number of attribute values*) to them randomly such that no two attribute values are assigned the same integer value. For example, we assign Dog = 1, Cow = 2, Rat = 3, Cat = 4 to the attribute values of the first attribute. The enforced ordinality is therefore

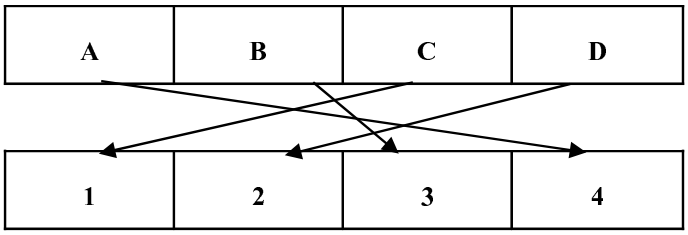


Figure 6.1: The example of multi-valued categorical attributes having four values A, B, C, D and are converted to ordinal data by imposing random ordinality, A = 4, B = 3, C = 1, D = 2.

.

Attribute 1	Attribute 2	Class
Cow	Sheep	1
Dog	Sheep	1
Cow	Bat	1
Dog	Bat	1
Rat	Deer	2
Rat	Bird	2
Cat	Deer	2
Cat	Bird	2

Table 6.1: Original Dataset - All attributes are categorical.

Dog<Cow<Rat<Cat. We follow the same process for all the multi-valued categorical attributes independently. Our final dataset will be integer-valued, therefore having a natural ordering. Following this method we can generate diverse continuous datasets from the given training dataset. Two new datasets are given in Table 6.2 and Table 6.3.

6.2.2 Learning

Each decision tree in the ensemble learns on one dataset from the pool of different datasets created by RO. During learning, these integer-valued attributes are treated as *continuous attributes*. We have binary splits in the tree as for continuous data attributes the node is split at a threshold value. For our example, we have three possible splits, {(1), (2,3,4)}, {(1,2), (3,4)} and {(1,2,3), (4)}. The best split is decided by the desired split criterion. We avoid the data fragmentation problem as there is a binary split. We have presented our proposed algorithm in Fig. 6.2.

Attribute 1	Attribute 2	Class
2	3	1
1	3	1
2	4	1
1	4	1
3	1	2
3	2	2
4	1	2
4	2	2

Table 6.2: New continuous data created from the dataset presented in Table 6.1 with ordering of attribute 1 values as Dog<Cow<Rat<Cat and attribute 2 values as Deer<Bird<Sheep<Bat.

Attribute 1	Attribute 2	Class
3	1	1
1	1	1
3	2	1
1	2	1
2	3	2
2	4	2
4	3	2
4	4	2

Table 6.3: New continuous data created from the dataset presented in Table 6.1 with ordering of attribute 1 values as Dog<Rat<Cow<Cat and attribute 2 values as Sheep<Bat<Deer<Bird.

Input- Dataset *T* and *M* size of the ensemble.

Training Phase
for i=1...M **do**
  Data Generation
  Apply Random Ordinality (O_i) to generate integer valued dataset T_i.
  Learning Phase
  Treat dataset T_i as continuous, and learn decision tree D_i.
end for

Testing Phase
For a given data point x
for i=1...M **do**
  Convert x(categorical) to x'(integer valued) using the ordinality (O_i) of tree D_i.
  Get the prediction for x' from tree D_i.
end for
Combine the predictions of *M* decision trees by the chosen combination rule to get the final classification result.

Figure 6.2: Algorithm for Random Ordinality Ensembles(ROE).

6.3 Empirical Evaluation of RO: Trees and Ensembles

We carried out experiments to study Random Ordinality trees and Random Ordinality ensembles. We note that the principle of RO can be applied to just a single tree, avoiding multi-way split, or it can be applied as an ensemble technique.

In the first part of the experiments, we studied the performance of a single decision tree based on RO. Next, a study was carried out to compare the performance of ROE with Bagging [11], AdaBoost.M1 [41] and Random Forests [13].

6.3.1 Experiments with a Single RO Tree

In the first part of the experiments, we compared the classification error of a decision tree trained on the original data (with multi-way split) and a decision tree created using generated by RO method.

Experimental datasets are taken from UCI repository. The information about these datasets is given in Table A.1. *As RO works only for categorical datasets, we selected pure categorical datasets only*. We used J48 (the Weka [96] implementation of C4.5, with the unpruned option), which has multi-way splits for multi-valued categorical attributes as the default. Following the methodology proposed by Dietterich [28], we performed five replications of a two-fold cross-validation. In each replication, the dataset was divided into two random equal-sized sets. Each learning algorithm was trained on one set at a time and its error was estimated on the other set. The RO process is random, hence with different RO, we get different trees. To take into account this randomness of RO into analysis, For every run, 100 trees were created, each created by using with a different RO transformation. A total of 1000 trees were created (10 runs $\times$ 100 trees in each run). The average testing errors on a 5×2 cross validation of these decision trees are given in Table 6.4.

On 9/13 datasets, the average errors of the RO trees are lower than standard multi-way decision trees trained on the original data (multi-way split).

6.3.2 Experiments with RO Ensembles

In this section, we present a comparative study of the performance of RO ensembles against other popular ensemble methods.

We created two types of classifier ensembles by using ROE. In the first, we used the unpruned J48 decision trees. In the second, we used *Random Trees* (RT) of WEKA,

Dataset	Decision tree (J48) with original data, error in % in multi-way split	Decision tree (J48) with RO attributes, error in %
Promoter	28.5	**25.3**
Hayes-Roth	25.3	**21.7**
Breast Cancer	35.9	**33.4**
Monks-1	**15.9**	26.1
Monks-2	49.6	**32.3**
Monks-3	**0**	0.1
Balance	31.4	**26.6**
Soyalarge	**9.7**	10.5
Tic-tac-toe	18.4	**12.4**
Car	9.2	**6.5**
DNA	8.9	**8.5**
Mushroom	**0**	0.2
Nursery	3.6	**2.2**

Table 6.4: Average classification error of single decision tree (J48) with original data and single decision tree (J48) with RO attributes. On 9/13 datasets, the average errors of the RO trees are lower than standard multi-way decision trees trained on the original data (multi-way split).

Random Trees constructs a tree that considers *K* random attributes at each node. In other words, we combine the attribute randomization of *Random Subspaces* with *Random Ordinality*. We carried out experiments with Bagging and AdaBoost.M1 using J48 (unpruned) as the base model, and Random Forests. For Random Forests, the number of attributes selected from the available attributes at each node is set at $\lfloor \log_2 m + 1 \rfloor$ (default value), where m is the number of attributes in the dataset. The size of the ensembles was set at 50 for these experiments. Following [54], *K* (the number of attributes to randomly investigate) is taken as the half of the attributes for Random Subspaces. Default settings were used for the rest of the parameters. In the experiments, decision tree algorithms were untouched. As open source Weka software was used for these experiments, these experiments can be easily duplicated. The experiments were conducted following the 5×2 cross-validation [28] as discussed in Chapter 3.

Table 6.5 presents classification errors of different ensemble methods on different datasets. It also presents the performance rank of various ensemble methods on different datasets. The average rank of ROE with RT is 1.8, whereas the average rank of ROE with J48 is 2.8. These ranks are better than ranks of Bagging (3.8), AdaBoost.M1 (3.6) and Random Forests (3.3).

Table 6.6 presents the comparative study of the various popular ensemble methods

Dataset	ROE with J48	ROE with RT	Bagging	AdaBoost.M1	Random Forests	Single Tree (J48)
Promoter	13.1(2)	**12.8(1)**	15.5(4)	19.6(5)	13.4(3)	28.5(6)
Hayes-Roth	16.9(2)	**15.9(1)**	22.8(4)	23.1(5)	22.2(3)	25.3(6)
Breast Cancer	30.3(3)	30.1(2)	**29.9(1)**	35.6(5)	32.4(4)	35.9(6)
Monks1	18.3(5)	**1.5(1)**	5.8(3)	5.9(4)	3.3(2)	15.9(6)
Monks2	33.9(2)	**30.9(1)**	46.9(3)	47.5(4)	50.4(6)	49.6(5)
Monks3	0(3.5)	0(3.5)	0(3.5)	0(3.5)	0(3.5)	0(3.5)
Balance	**19.6(1)**	20.0(2)	29.6(4)	30.3(5)	26.9(3)	31.4(6)
Soyalarge	8.8(5)	**7.3(1.5)**	8.2(4)	**7.3(1.5)**	7.9(3)	9.7(6)
Tic-tac-toe	6.6(3)	**3.4(1)**	10.0(5)	3.5(2)	8.6(4)	18.4(6)
Car	**4.1(1)**	4.2(2)	8.3(4.5)	5.9(3)	8.3(4.5)	9.2(6)
DNA	4.5(2)	**4.4(1)**	6.2(5)	5.1(3)	5.8(4)	8.9(6)
Mushroom	0.1(5.5)	0.1(5.5)	**0(2.5)**	**0(2.5)**	**0(2.5)**	0**0(2.5)**
Nursery	1.0(2)	**0.9(1)**	2.8(5)	1.3(3)	2.6(4)	3.6(6)
Average Rank	2.8	1.8	3.8	3.6	3.3	5.5

Table 6.5: Classification error in % for different ensembles (rank on the basis of average classification accuracy is given in brackets), bold numbers show best performance. ROE ensembles generally perform similar to or better than other ensemble methods.

against ROE with J48 and ROE with RT. Results are presented as *ROE with J48*/*ROE with RT* when results are different for these two ensembles. For example, in the Random Forests column for Tic-tac-toe dataset, we have Δ/+, it means for Tic-tac-toe dataset, ROE with J48 is similar to Random Forests, whereas ROE with RT is better than Random Forests. When only one result is presented it means the comparative performance of ROE with J48 and ROE with RT is similar. For example in the Bagging column for Car dataset, we have +, it means for Car dataset both ROE with J48 and ROE with RT are better than Bagging.

Results suggest that, with the exception of Monks1 dataset, the performance of ROE with J48 is either statistically similar or better than other ensemble methods. The performance of ROE with RT is either statistically similar or better than other ensemble methods for all datasets.

For Monks1 dataset, ROE with J48 did not give good results, whereas for Tic-tac-toe and Soyalarge datasets, when we combine RS with RO, we observed great improvement in classification accuracy. We will now discuss these datasets in detail to understand the limitations of ROE.

Monks1 dataset has six attributes and two classes. The classification is Y = 1, if $(x_1 = x_2) \vee (x_5 = 1)$. All the other data points belong to class 2. When we treat

Dataset	Bagging	AdaBoost.M1	Random Forests	Single Tree
Promoter	Δ	Δ	Δ	+
Hayes-Roth	+	+	+	+
Breast Cancer	Δ	Δ	Δ	Δ
Monks1	-/Δ	-/Δ	-/Δ	-/+
Monks2	+	+	+	+
Monks3	Δ	Δ	Δ	Δ
Balance	+	+	+	+
Soyalarge	Δ	Δ	Δ	Δ
Tic-tac-toe	+	Δ	Δ/+	+
Car	+	+	+	+
DNA	+	Δ	Δ	+
Mushroom	Δ	Δ	Δ	Δ
Nursery	+	+	+	+
RO with J48 win/draw/lose	7/5/1	5/7/1	5/7/1	9/3/1
RO with RS win/draw/lose	7/6/0	5/8/0	6/7/0	9/4/0

Table 6.6: Comparative Study of ROE with J48 and ROE with RT. Results are presented *ROE with J48*/*ROE with RT*. if performance of these ensembles are different. '+/-' shows that performance of ROE is statistically better/worse than that algorithm for that dataset, 'Δ' shows that there is no statistically significant difference in performance for this dataset between ROE and that algorithm. ROE ensembles generally perform similar to or better than other ensemble methods.

data as continuous, the first concept ($x_1 = x_2$) is a diagonal concept. J48 trees are restricted to *orthogonal* decision boundaries. In other words, decision trees divide the input attribute space into rectangular regions whose sides are perpendicular to the attribute axis. Decision trees have a representational problem because of the orthogonal property- they have difficulty in learning a diagonal decision boundary. Ensembles of decision trees solve this problem, as combined results of decision trees produce a good approximation of a diagonal concept [29]. The quality of the approximation depends on the diversity of decision trees in the ensemble. ROE with RT trees are more diverse as compared to ROE with J48 trees. Hence, ROE with RT can learn this diagonal decision boundary in Monks1 data better than ROE with J48.

Building a good ensemble depends on the creation of diverse decision trees. We create diverse decision trees by imposing random ordinality to categorical attributes values that in turn create different node splits. Diversity in node splits is the key for

diverse decision trees. For an attribute, the possible number of different splits for the attribute is given by Eq.(1). If $|A|$ is small, there is a large possibility that different trees have same node splits, and we may not get the very diverse trees. When an attribute has only two values, imposing random ordinality is not useful as there is only one way a node can split. Tic-Tac-Toe data has only 3 attribute values for each attribute hence RO alone does not produce very diverse decision trees. Similarly Soyalarge dataset has a large number of binary attributes which is unaffected by RO. Hence, RO trees are not creating diverse trees for Soyalarge dataset. The small cardinality of attributes is the reason for relatively large improvement when RO is combined with RS.

In summary, RO trees are generally better than multi-way split trees and RO ensembles are either statistically similar or better than other ensemble methods for all datasets. To analyse RO attributes, in the next section, we carry out a theoretical study of RO attributes on artificial datasets.

6.4 Study of RO attributes in the information theoretic framework

In RO, new attributes are created by randomly assigning order to different attribute values and treating these new attributes as continuous. The selected splitting criterion is used to decide the best binary split. In this section, we will discuss whether these attributes are good for classification using the information theoretic framework.

Let T be a 2 class (Y = +1 and Y = -1) dataset such that it has the same number of positive and negative examples. Let A be a multi-valued attribute with cardinality $|A|$ again with uniform prior probability. Half of these values correctly identify the positive class, whereas the rest of the values correctly identify the negative class. For example, if attribute values are (a, b, c, d, e, f),

$$p(Y = +1|A = a) = 1. \tag{6.1}$$

$$p(Y = +1|A = b) = 1. \tag{6.2}$$

$$p(Y = +1|A = c) = 1. \tag{6.3}$$

$$p(Y = -1|A = d) = 1. \tag{6.4}$$

$$p(Y = -1|A = e) = 1. \tag{6.5}$$

$$p(Y = -1|A = f) = 1. \tag{6.6}$$

Cardinality $\|A\|$ of the attribute A	Number of random attributes created	Average gain ratio for RO attributes (s.d.)	Average gain ratio for attributes with random split (s.d.)	Gain ratio for multi-way split
4	10^4	0.59(0.29)	0.37(0.26)	0.50
6	10^4	0.47(0.20)	0.20(0.24)	0.39
8	10^6	0.40(0.16)	0.13(0.18)	0.33
10	10^7	0.35(0.12)	0.10(0.14)	0.30
12	10^7	0.32(0.10)	0.08(0.11)	0.28
14	10^7	0.29(0.09)	0.06(0.09)	0.26

Table 6.7: Information gain ratio of attributes with different numbers of attribute values. RO attributes have better information gain ratio than multi-way splits.

We calculate the information gain ratio (discussed in Chapter 2) of different attributes created by RO. We randomly assign order to (a, b, c, d, e, f) and calculate a binary split at each point, the maximum information gain ratio is taken as the information gain ratio associated with this random order. For example, if we assign

$$a < c < f < e < b < d. \tag{6.7}$$

The maximum information gain ratio is based on the split ((a,c) (f,e,b,d)) and this is taken as the information gain ratio associated with the random order presented in Eq. 6.7.

We calculate the average information gain ratio of different possible random orders of attribute values. We carry out this exercise for attributes with different cardinality, whereas datasets and attribute values have same properties as discussed above.

We also calculate the information gain ratio of binary attributes created by random splitting of attribute values into two groups. Results are presented in Table 6.7 and Fig. 6.3. Results indicate that the average gain ratio of attribute created using RO and the gain ratio of multi-valued attributes are quite similar, whereas random splits do not create good attributes. As the number of attribute values increases, the average information gain ratio of RO attributes decreases. The same is true for the multi-way split as the value of the normalizing factor ($\log_2 |A|$) increases. This suggests that on average we are creating continuous attributes from multi-valued categorical attributes that have similar information gain ratio.

For the example data, using RO we get the best gain ratio, when all attributes

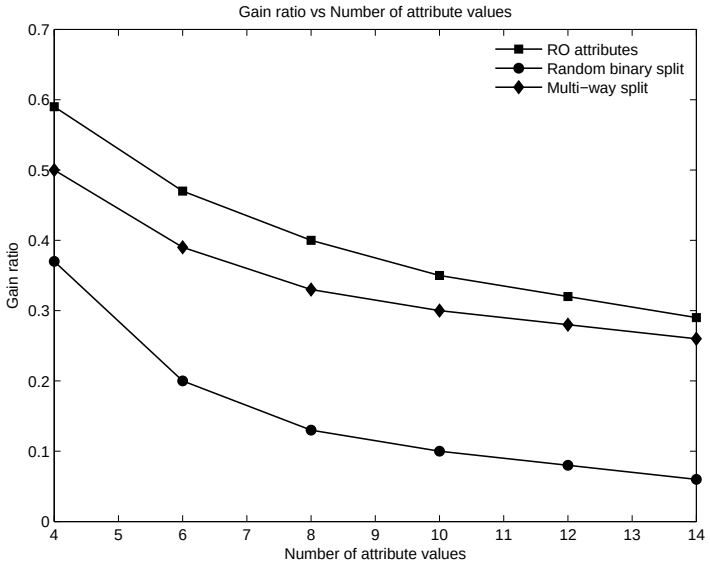


Figure 6.3: Information gain ratio for RO attributes, Random Split and multi-way splits. RO attributes have better information gain ratio than multi-way splits and random splits.

values related with positive class are together, the same should be true with attributes values related with negative class. We have $|A|$ number of attribute values, half of them are related with positive class, whereas the other half are related with negative class. All attributes values $\frac{|A|}{2}$ related with positive class will be either left or right of the attributes values $\frac{|A|}{2}$ related with negative class. The number of possible ways for this kind of combination is $2(\frac{|A|}{2})!(\frac{|A|}{2})!$. The total number of possible combinations with $|A|$ number of attribute values is $|A|!$. Hence, the probability to get attributes with best gain ratio is $2\frac{(\frac{|A|}{2})!(\frac{|A|}{2})!}{|A|!}$, which decreases as $|A|$ increases, this can be observed in Fig. 6.3.

Fig. 6.4 shows the histogram of information gain ratio probability for RO attributes generated from attributes with a different number of attribute values. We also plot cumulative probability for the gain ratio. Fig. 6.5 shows the same for random splits. The best possible cumulative probability is a delta function at gain ratio value 1. In other words, a better cumulative probability curve takes higher values at higher values of the gain ratio. The comparative study of RO attributes and random splits suggests that RO attributes have better cumulative probability curve for gain ratio, for example for an attribute with 10 values there is around 0.68 probability that the gain ratio is

more that 0.32, whereas for random splits that probability is around 0.10 (Fig. 6.6). We have shown in this section that for multi-valued categorical attributes with certain properties, the information gain ratio of a binary split with some random constraints may be similar to a multi-way split.

As in real datasets, we have a large number of attributes, this helps us in selecting better attributes. For example, if all attributes have similar properties, and the number of attributes is m, we are taking the best of m selections of the population which has average gain ratio similar to a multi-way split information gain ratio as at each level a decision tree algorithm selects the best available attribute. In summary, there is a reasonable probability that RO attributes are good for classification.

In the next section, we present the various studies to analyse RO trees and RO ensembles.

6.5 Controlled Experiments

In this section, we study the performance of RO ensembles by varying the number of attributes in the concepts, the attribute cardinality and the number of training data points. We created four datasets for this purpose.

1. $Categorical\_Multiplexer$ - In Multiplexer, an instance is a series of bit of length; a + 2^a, where a is a positive integer. The first a bits represent an index into the remaining bits and the level of the instance is the value of the indexed bit. We created a variant of Multiplexer. This is referred to as $Categorical\_Multiplexer$. In this dataset, attributes are categorical (can take more than 2 values and these values are integers). To decide the concept, we converted the data into binary data by using the following transformations,

$$\text{even number = 0, odd number = 1.} \tag{6.8}$$

For example, for $Categorical\_11 - Multiplexer(a = 3)$, if we have the instance, (2,1,6,5,3,5,7,8,5,6,3), by using the above transformation, the instance is converted to the following binary instance to compute the concept,

$$(0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1). \tag{6.9}$$

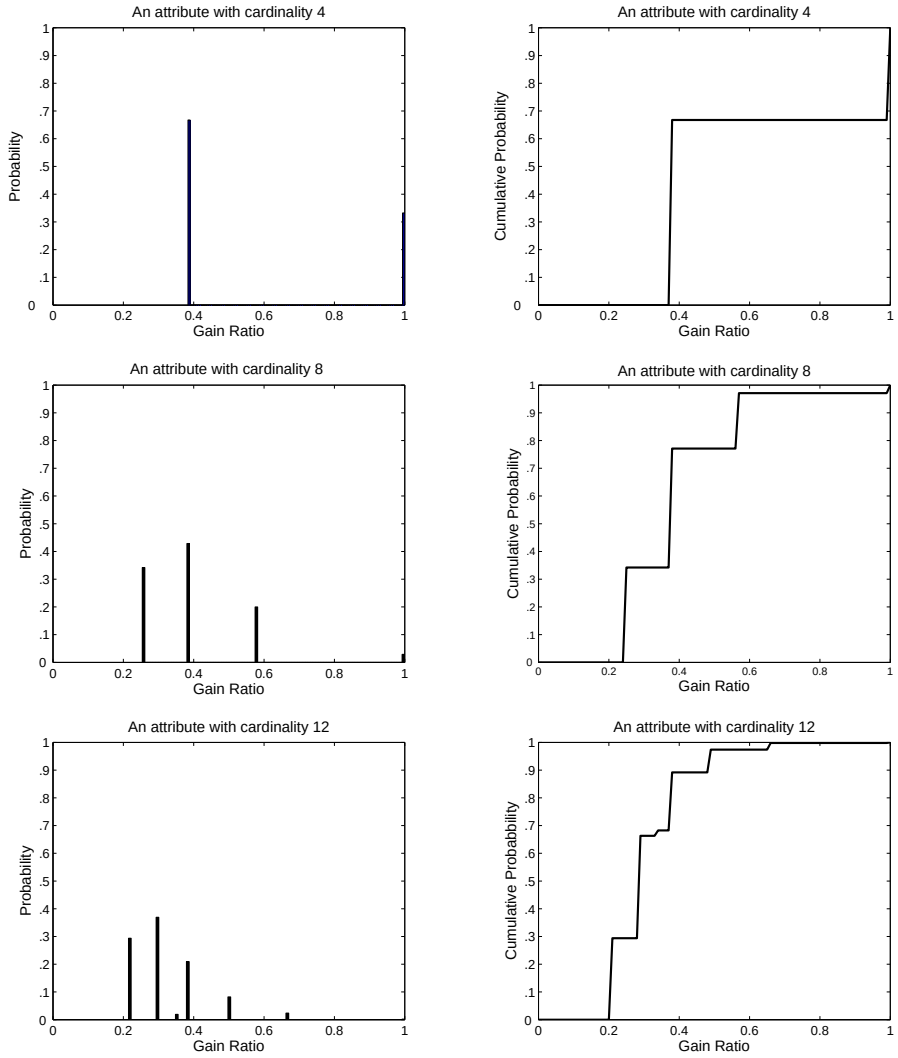


Figure 6.4: Information gain ratio for attributes created by using the RO method, for attributes with different cardinalities. Left column - probability vs gain ratio, right column - cumulative probability vs information gain ratio. Small cumulative probability at low information gain ratio suggests that splits for RO attributes are good for classification.

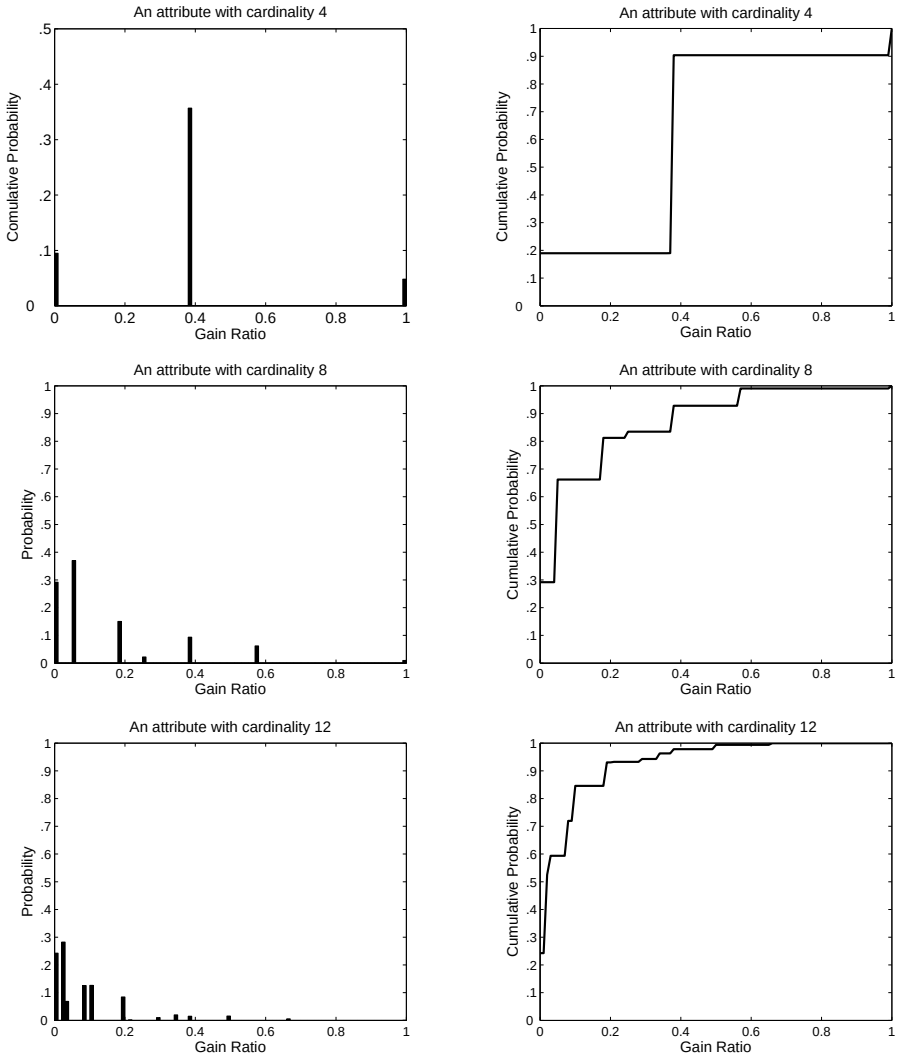


Figure 6.5: Information gain ratio for attributes created using random splits, for attributes with different cardinalities. Left column - probability vs information gain ratio, right column - cumulative probability vs information gain ratio. Large cumulative probability at low information gain ratio suggests that these random splits are not as good as splits created for RO attributes for classification.

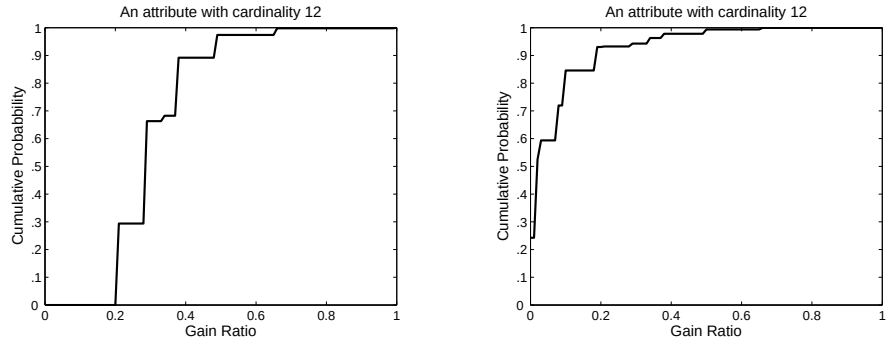


Figure 6.6: Cumulative probability for information gain ratio for an attribute of cardinality 12 (Left- RO attribute, Right - Random split). Smaller cumulative probability at low information gain ratio suggests that splits created for RO attributes are better for classification.

First three attributes, from the right, decide the index into the remaining bits, which is 5 (1×4 + 1×1), so the concept is 0; the value of 6^{th} remaining bits (as the starting point is 0, 9^{th} bit from the right).

The integer valued data is created randomly and the above procedure is used to compute the concepts. As the data is integer valued, it is treated as multi-valued categorical data. The number of integers is varied in every attribute, in other words, the cardinality of attributes is varied. All attributes have the same cardinality. Two types of datasets with a different numbers of attribute cardinalities (6 and 10) are created. We carry out experiments with

- $Categorical\_11 - Multiplexer(a = 3)$ and
- $Categorical\_20 - Multiplexer(a = 4)$.

This problem is different with normal multiplexer task as with a higher cardinality, the complexity of the problem increases due to more number of concepts.

2. Odd_Even_Data This is an integer valued data that is treated as multi-valued categorical data. It has the following 3 concepts,

   - A = (All attribute values are even) or (All attribute values are odd)
   - B = ((First half of the attributes are even) and (Next half of the attributes are odd)) or ((First half of the attributes are odd) and (Next half of the attributes are even))
   - C = ∼(A or B)

Integer valued data points are created randomly and treated as multi-valued categorical data. Two types of datasets are created, one with 4 attributes and the other with 8 attributes. Data points are created such that there is almost equal representation of each class. We have 2 variants of each dataset with different attribute cardinalities (6 and 10). The dataset with 4 attributes and each attribute cardinality 6 is referred as $Odd\_Even\_4\_6$, the dataset with 4 attributes and each attribute cardinality 10 is referred as $Odd\_Even\_4\_10$, the dataset with 8 attributes and each attribute cardinality 6 is referred as $Odd\_Even\_8\_6$ whereas the dataset with 8 attributes and each attribute cardinality 6 is referred as $Odd\_Even\_8\_10$.

6.5.1 Discussion

The experiments are conducted following the 5×2 cross-validation [28] (discussed in experiment section). Experiments are carried out with different sizes of training data/testing; 1000/1000 data points and 4000/4000 data points.

For all kinds of datasets, RO ensembles performed statistically similar to or better than other ensemble methods (Table 6.8 to Table 6.15). On the basis of a single decision tree classification error (Table 6.8 to Table 6.15), one can conclude that Odd_Even_Data with 4 attributes has simplest concepts followed by Odd_Even_Data with 8 attributes. $Categorical\_11-Multiplexer$ and $Categorical\_20-Multiplexer$ datasets have difficult concepts. The number of attributes needed to define the concepts is the possible reason for these characteristics; if a concept is defined by a large number of attributes, it is difficult to learn this concept.

The variants of datasets with higher number attribute cardinality are more difficult to learn as they have more concepts (as compared to the variant of dataset with a lower attribute cardinality) and because of the larger attribute cardinality they are affected more by the data fragmentation problem.

Two kinds of behaviour are observed, for the simpler datasets (Odd_Even_Data), with 1000 data points in the training datasets, there was a large difference between classification errors of RO ensembles (Table 6.8 to Table 6.11) and the other popular ensembles (results are in favour of RO ensembles), however as the size of the training dataset is increased from 1000 to 4000, the relative advantage of RO ensembles decreases (Table 6.8 to Table 6.11). One of the possible reasons is that a large number of data points reduces the data fragmentation problem, hence improves the performance of ensembles of decision trees with multi-splits.

For datasets ($Categorical\_11-Multiplexer$ and $Categorical\_20-Multiplexer$

No. of training/ testing points	ROE with J48	ROE with RT	Bagging	AdaBoost.M1	Random Forests	Single Tree
1000/1000	0.43	**0.34**	8.87(+)	4.27(+)	7.33(+)	10.93(+)
4000/4000	0.01	**0**	2.78(+)	3.25(+)	1.34(+)	3.75(+)

Table 6.8: Testing error in % (bold numbers indicate the best performance) for $Odd\_Even\_Data\_4\_6$ dataset, '+' suggests that RO ensembles are statistically better than that ensemble method.

No. of training/ testing points	ROE with J48	ROE with RT	Bagging	AdaBoost.M1	Random Forests	Single Tree
1000/1000	6.57	**3.87**	20.12(+)	13.40(+)	18.96(+)	28.30(+)
4000/4000	0.20	**0.02**	8.50(+)	4.23(+)	7.17(+)	12.88(+)

Table 6.9: Testing error in % (bold numbers indicate the best performance) for $Odd\_Even\_Data\_4\_10$, '+' suggests that RO ensembles are statistically better than that ensemble method.

datasets) with relatively difficult concepts, the opposite behaviour is observed. RO ensembles are statistically similar to other ensembles of decision trees with multi-splits for 1000 training data points (Table 6.12 to Table 6.15). However, as the number of training data points is increased from 1000 to 4000, the improvement in RO ensembles (except $Categorical\_20 - Multiplexer$ dataset with the attribute cardinality 10) is greater than the other ensemble methods. As these datasets have difficult concepts, 1000 data points are not enough for the learning, however with 4000 training data points RO ensembles learn these concepts better than other ensemble methods.

We showed in this section by varying the number of attribute in the concepts, the cardinality of attributes and the number of training data points that the RO ensembles are statistically similar to or better than other ensemble methods that show the effectiveness of the ROE method. To analyse the reasons for the success of ROE, we focus on two factors; robustness of ROE for data fragmentation problem and error-diversity patterns of ROE.

6.6 Analysis

In the previous sections, we compared the performance of ROE approach with other popular ensemble methods. In this section, we present the following studies to understand the behaviour of RO ensembles;

No. of training/ testing points	ROE with J48	ROE with RT	Bagging	AdaBoost.M1	Random Forests	Single Tree
1000/1000	1.64	**1.56**	10.06(+)	4.61(+)	8.13(+)	18.26(+)
4000/4000	0.19	**0.05**	4.68(+)	2.33(+)	3.44(+)	10.70(+)

Table 6.10: Testing error in % (bold numbers indicate the best performance) for $Odd\_Even\_Data\_8\_6$, '+' suggests that RO ensembles are statistically better than that ensemble method.

No. of training/ testing points	ROE with J48	ROE with RT	Bagging	AdaBoost.M1	Random Forests	Single Tree
1000/1000	**4.68**	5.42	21.50(+)	12.30(+)	19.45(+)	33.05(+)
4000/4000	**1.30**	1.68	10.24(+)	4.24(+)	7.97(+)	20.98(+)

Table 6.11: Testing error in % (bold numbers indicate the best performance) for $Odd\_Even\_Data\_8\_10$, '+' suggests that RO ensembles are statistically better than that ensemble method.

No. of training/ testing points	ROE with J48	ROE with RT	Bagging	AdaBoost.M1	Random Forests	Single Tree
1000/1000	30.66	**30.58**	37.97(+)	33.81	34.90	43.83(+)
4000/4000	**8.21**	13.39	31.15(+)	26.49(+)	29.59(+)	40.48

Table 6.12: Testing error in % (bold numbers indicate the best performance) for $Categorical\_11 - Multiplexer$, the attribute cardinality is 6, '+' suggests that RO ensembles are statistically better than that ensemble method.

No. of training/ testing points	ROE with J48	ROE with RT	Bagging	AdaBoost.M1	Random Forests	Single Tree
1000/1000	40.25	**39.86**	42.76	41.85	41.15	45.95(+)
4000/4000	**28.88**	30.17	38.42(+)	35.98(+)	36.09(+)	44.45(+)

Table 6.13: Testing error in % (bold numbers indicate the best performance) for $Categorical\_11 - Multiplexer$, the attribute cardinality is 10, '+' suggests that RO ensembles are statistically better than that ensemble method.

No. of training/ testing points	ROE with J48	ROE with RT	Bagging	AdaBoost.M1	Random Forest	Single Tree
1000/1000	**43.08**	44.55	45.69	44.88	44.53	48.39(+)
4000/4000	**39.05**	39.28	44.16(+)	44.16(+)	43.38(+)	48.03(+)

Table 6.14: Testing error in % (bold numbers indicate the best performance) for $Categorical\_20 - Multiplexer$, the attribute cardinality is 6, '+' suggests that RO ensembles are statistically better than that ensemble method.

No. of training/ testing points	ROE with J48	ROE with RT	Bagging	AdaBoost.M1	Random Forests	Single Tree
1000/1000	45.42	**44.88**	45.86	46.40	45.38	49.67(+)
4000/4000	44.15	**44.07**	46.25	45.68	45.27	48.7(+)

Table 6.15: Testing error in % (bold numbers indicate the best performance) for $Categorical\_20 - Multiplexer$, the attribute cardinality is 10, '+' suggests that RO ensembles are statistically better than that ensemble method.

6.7 Analysis of RO Ensembles

In the previous sections, we compared the performance of ROE approach with other popular ensemble methods. In this section, we present the following studies to understand the behaviour of RO ensembles;

The Effect of the Data Fragmentation One of the motivations for ROE is to build binary decision trees so that they may not suffer from data fragmentation problem. If the dataset has attributes with large number of values, the performance of decision trees may be affected because of the data fragmentation problem. We carried out a control experiment to study RO ensembles by varying the number of attribute values.

RO Tree Sizes We carried out a study to analyse the RO trees sizes and their advantages for creating more reliable rules.

The Diversity - Accuracy Trade Off A good combination of diversity and accuracy is required for better performance of an ensemble. To understand the behaviour of RO ensembles, we studied diversity-accuracy patterns using kappa-error plots.

The Effect of the Ensemble Size We discussed the effect of the size of RO ensembles on RO ensembles accuracy. We also analysed RO ensembles using the theoretical framework proposed by Fumera et al. [42].

Combinations of RO with the Other Ensemble Methods We studied how Bagging and AdaBoost.M1 are affected by RO.

6.7.1 The Effect of the Data Fragmentation

Data fragmentation may affect the performance of decision trees. We have carried out a controlled experiment to see how different ensemble methods perform with respect to the cardinality of the attribute. For this purpose, we selected two pure continuous datasets; Vehicle and Segment. Vehicle data has 846 data points described by 18 continuous attributes. These data points are distributed into 4 classes. Segment dataset has 2310 data points. These data points are described by 19 continuous attributes and divided into 7 classes. We converted these datasets into categorical datasets using equal width discretization. We studied various ensemble methods on these discretized datasets; varying the numbers of bins to see its effect on different ensemble methods. We performed five replications of a two-fold cross-validation. The size of the ensembles was 50. The results (Fig. 6.7 - Fig. 6.8) suggest that classification errors of RO ensembles are relatively unaffected. When we increase the number of bins we have a small number of points in every bin; that leads to badly estimated probabilities and poor generalization. As ROE consists of binary decision trees so it is more robust to the data fragmentation problem.

6.7.2 RO Tree Sizes

In the previous sections, we have argued that RO trees avoid multi-split. As RO trees having binary splits RO trees are more likely to have smaller sizes than that of multi-split decision trees [35]. Smaller trees have greater statistical evidence at the leaves. We studied RO tree sizes for various datasets. The experiments were conducted following the 5×2 cross-validation and 50 RO trees are created in each run.

In the Table 6.16, we have presented the average sizes of RO trees (J48 decision trees created using datasets generated by the RO method) and normal multi-split J48 decision trees for different datasets. For all the datasets, RO trees are smaller that normal multi-split J48 decision trees. For example, for DNA dataset, the average size

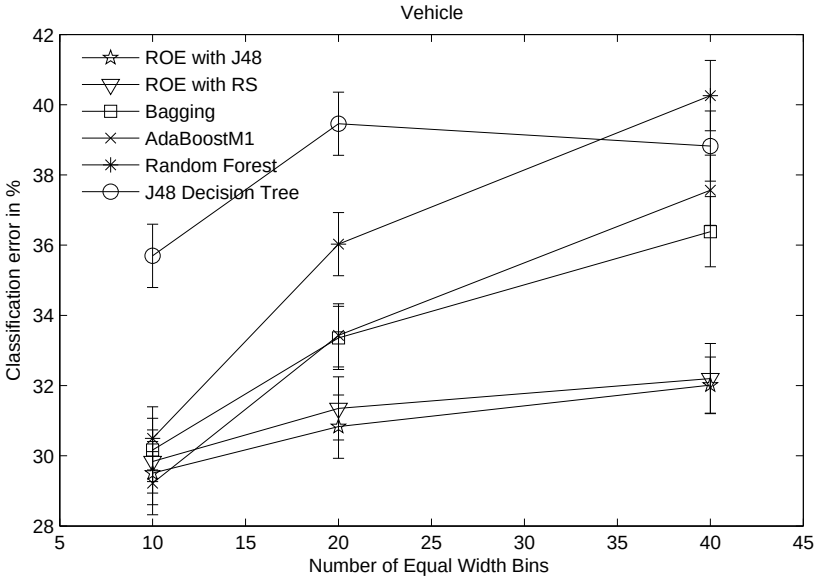


Figure 6.7: The effect of equal width discretization on various ensemble methods for the Vehicle dataset. RO ensembles are quite robust to data fragmentation.

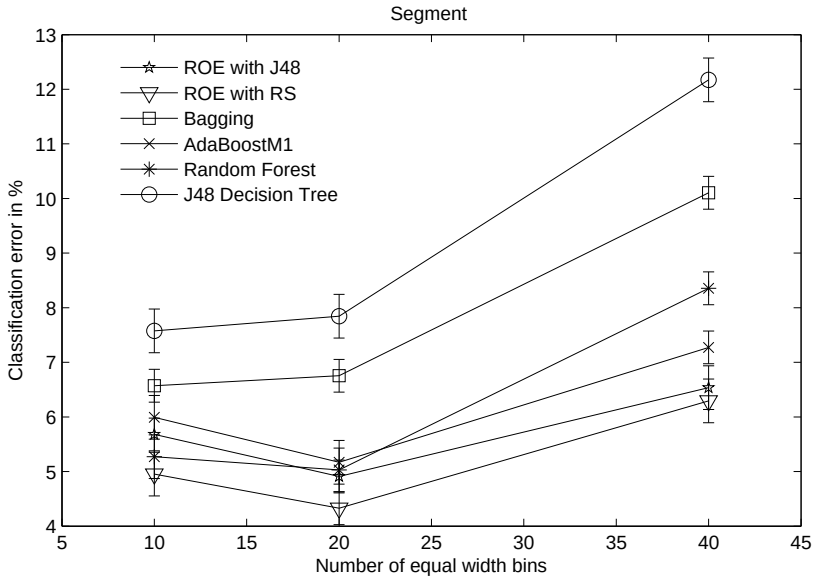


Figure 6.8: The effect of equal width discretization on various ensemble methods for the Segment dataset. RO ensembles are quite robust to data fragmentation.

Name of dataset	Size of the training data	The average number of leaves/size of RO trees (J48)	The average number of leaves/size of multi-split J48 trees
Car	864	54/107	127/174
DNA	1587	151/76	281/211
Tic-Tac-Toe	479	97/49	142/92
Promoter	53	6/11	13/17

Table 6.16: The average sizes of RO trees and multi-split J48 trees for different datasets. RO trees are smaller than multi-way trees.

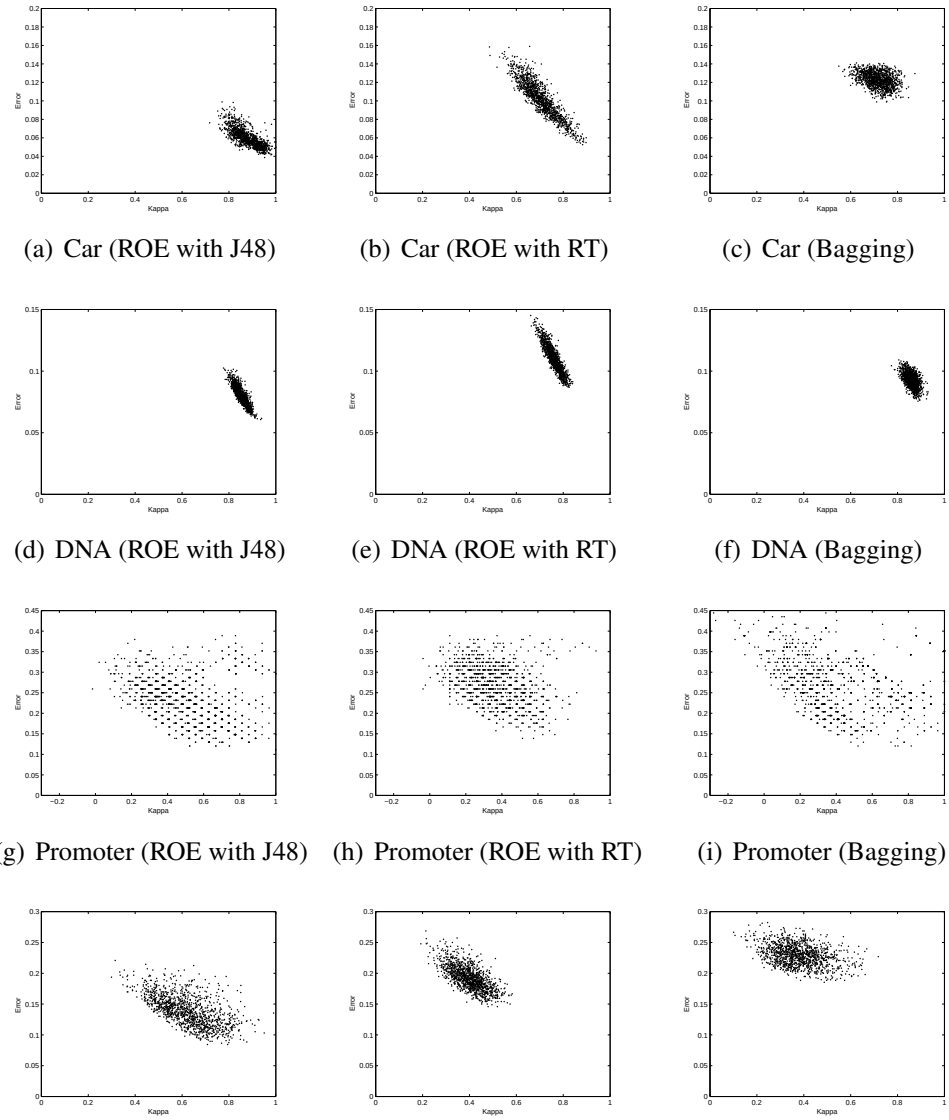
of RO trees is 151 whereas the average size of normal multi-split J48 decision trees is 281. These results indicate that RO helps in creating smaller decision trees.

6.7.3 The Diversity - Accuracy Trade Off

As discussed in chapter 4, Kappa-error plots [72] are a method to understand the diversity-error behaviour of an ensemble (for detail see Appendix). We draw kappa-error plots of our proposed ensemble methods (Fig. 6.9) for four datasets (Car, DNA, Promoter, Tic-Tac-Toe). For comparison, we also draw kappa error plots for Bagging method for these datasets. The scales of κ and $E_{i,j}$ are same for each given dataset so we can easily compare different ensemble methods. As expected, ROE with J48 is less diverse as compared to ROE with RT (selection of attribute out of *K* randomly selected attributes increases the diversity). The diversity of ROE with J48 is less than the diversity of Bagging (except for DNA dataset). However, RO with J48 trees are generally more accurate than ROE with RT trees and trees created using Bagging. This is the reason for better performance of ROE with J48. ROE with RT and Bagging have similar diversity behaviour whereas ROE with RT trees are generally more accurate.

Kappa-error plots suggest that with ROE method, we are able to produce accurate classifiers. This is the reason for the better performance of ROE method. In other words, ROE is able to create accurate classifiers with reasonable diversity.

Results in section 4 and in this section suggest that RO trees are quite accurate. There may be two reasons for this behaviour of RO trees. As we have discussed, these classifiers are quite robust to the data fragmentation problem. This may be one of the reasons for the RO trees good accuracy. In the tree growing phase, we calculate the information gain ratio of all available attributes at each level. The reliability of these calculations depends on the number of data points presents at the nodes (when



(a) Car (ROE with J48) (b) Car (ROE with RT) (c) Car (Bagging)

(d) DNA (ROE with J48) (e) DNA (ROE with RT) (f) DNA (Bagging)

(g) Promoter (ROE with J48) (h) Promoter (ROE with RT) (i) Promoter (Bagging)

(j) Tic-Tac-Toe (ROE with J48) (k) Tic-Tac-Toe (ROE with RT) (l) Tic-Tac-Toe (Bagging)

Figure 6.9: Kappa-error diagrams for three ensemble methods, Left column- ROE with J48, middle column - ROE with RT, right column - Bagging. x-axis - Kappa, y-axis - the average error of the pair of classifiers. Axes scales are constant for various ensemble methods for a particular dataset (each row). Lower κ represents higher diversity. RO ensembles have accurate classifiers with reasonable diversity.

smaller nodes are split, the measure of information gain is more unreliable). We do the comparative study of number of data points available in nodes at different depths of binary split trees and multi-split trees.

If we assume that data points are equally divided in the nodes, *for a decision tree having $|A|$ splits at each node*, at depth ϑ_x, the number of points at each node is N(ϑ_x),

$$N(\vartheta_{|A|}) = \frac{n}{|A|^{\vartheta_{|A|}}}, \tag{6.10}$$

where n is the total number of points. and $|A|^{\vartheta_{|A|}}$ is the number of nodes at depth ϑ_x.

For decision tree having binary splits at each node, at the depth ϑ_2, the number of points at each node is N(ϑ_2),

$$N(\vartheta_2) = \frac{n}{2^{\vartheta_2}}, \tag{6.11}$$

where 2^{ϑ_2} is the number of nodes at depth ϑ_2.

We show in Fig. 6.10 how the data points are split in nodes at different depths of the trees. If each node presents at depth $\vartheta_{|A|}$ of decision tree having $|A|$ splits at each node has the same number of points as the each node present at the depth ϑ_2 of decision tree having binary split

(N(ϑ_2)= N($\vartheta_{|A|}$)), using Eq. 9 and 10.

$$|A|^{\vartheta_{|A|}} = 2^{\vartheta_2}. \tag{6.12}$$

$$\vartheta_2 = \vartheta_{|A|} \log_2 |A|. \tag{6.13}$$

$$\frac{\vartheta_2}{\vartheta_{|A|}} = \log_2 |A|. \tag{6.14}$$

We have presented depth ratio $\frac{\vartheta_2}{\vartheta_{|A|}}$ (for the same number of data points in the nodes, N(ϑ_2)= N($\vartheta_{|A|}$)) for different number of splits in Fig. 6.11.

Hence, if we assume that data points are equally divided in the nodes, in a binary

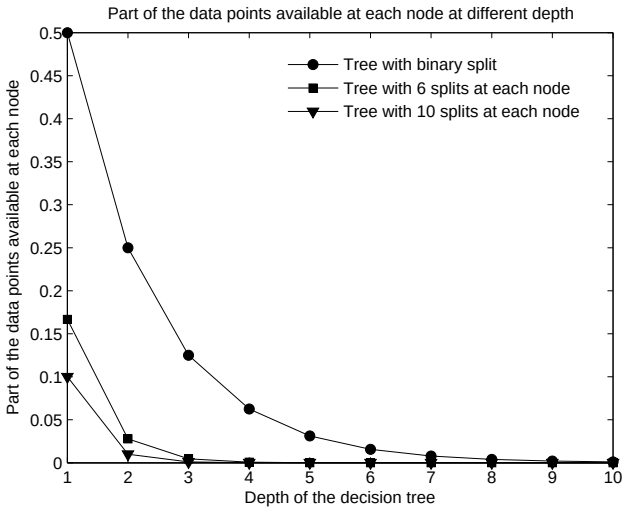


Figure 6.10: Part of the dataset available at each node for different depth, for decision trees with different number of splits at each node.

tree, the number of points at each node present at depth ϑ_2 is the same as the number of points at each node present at depth $\frac{\vartheta_2}{\log_2 |A|}$ for the decision tree having $|A|$ splits at each node. This suggests that in a binary tree, a more reliable decision can be made at lower levels. As the classification rules that we get from the decision tree are the paths from root to leaves of the decision tree, we may get more reliable rules with binary split decision trees.

The second reason, we believe is related with generation of new attributes. RO creates new attributes by imposing random ordinality. In other words, we are creating a numerical representation of attribute values. In real life datasets, we have many attributes, for every attribute RO creates a numerical representation. As we have large numbers of attributes, there is a high probability that some of the randomly generated numerical representations are good for classification (as discussed in section 5). This way we expect good decision trees.

6.7.4 The Effect of the Ensemble Size

We studied the effect of ensemble size on ensemble errors for these four datasets (Car, DNA, Promoter, Tic-Tac-Toe). The results are given in Fig. 6.12 and Fig. 6.12. For comparison the classification errors with Bagging and AdaBoost.M1 are also presented (we did not present results of Random Forests for better visualization as the error of Random Forests consisting of a single classifier is quite large). ROE with J48 achieves

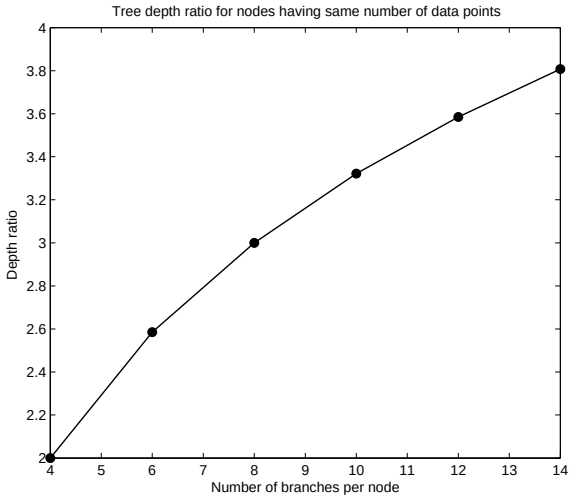


Figure 6.11: Tree depth ratio ($\frac{\vartheta_2}{\vartheta_{|A|}}$) for different number of splits ($|A|$) such that N($\vartheta_{|A|}$) = N(ϑ_2) where N($\vartheta_{|A|}$) is the number of points at each node at depth ϑ_k, for trees having $|A|$ splits at each node.

its maximum performance with a small number of classifiers (around 10), which is a characteristic of an ensemble having accurate but not very diverse classifiers.

As discussed in Chapter 2, Fumera et al. [42] suggest an analytic relationship between the expected misclassification probability of the ensemble and the expected misclassification probability of an individual classifier, as a function of the ensemble size. Their theoretical results show that the expected misclassification probabilities of Bagging [11] has the bias component as the bias component of the base model, whereas the variance component is reduced by a factor M.

$$E = E(B) + \frac{E(V)}{M}, \tag{6.15}$$

where,

- E is the classification error of ensemble
- $E(B)$ corresponds to the sum of the Bayes error and of the bias component of the error,
- $E(V)$ is the variance part of the error.

We study RO ensembles by using the analytical relationship suggested by Fumera et al. [42] to study its applicability for RO ensembles. To compute the values of E_B

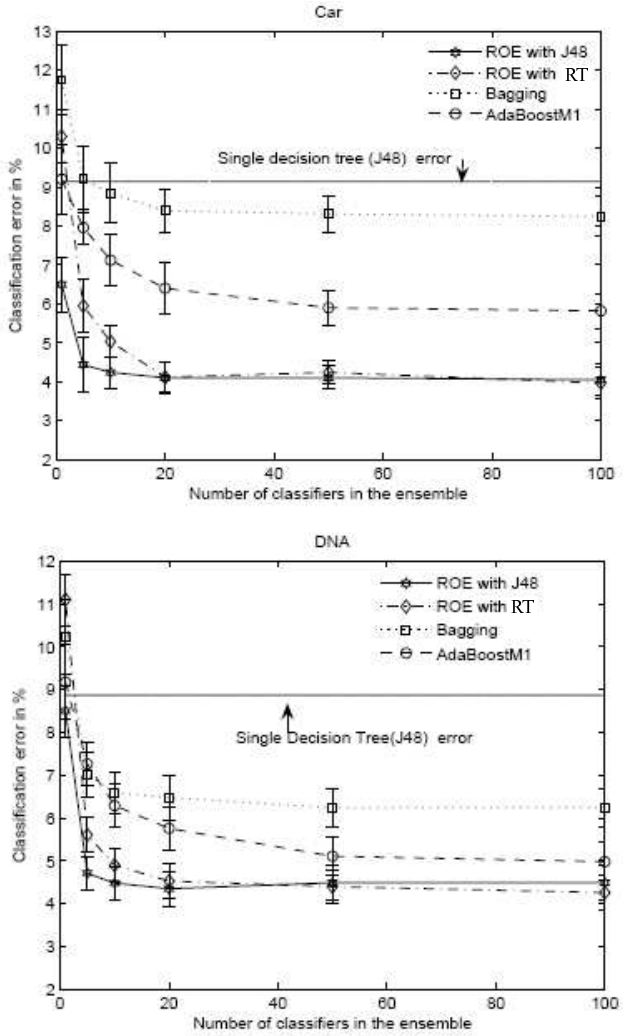


Figure 6.12: Classification error (with 95% confidence interval) of various ensemble methods vs size of the ensemble for different datasets.

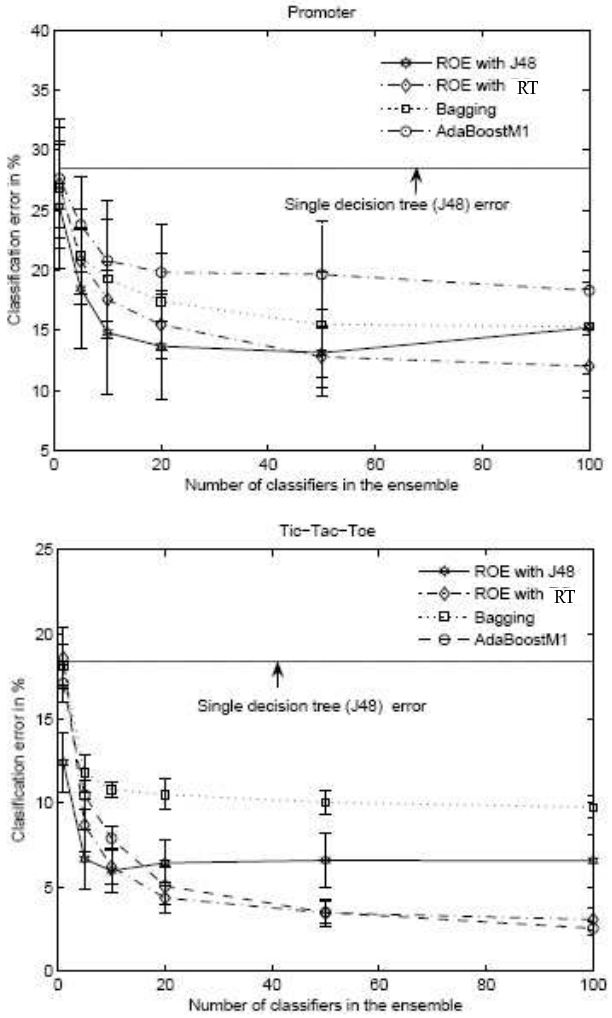


Figure 6.13: Classification error (with 95% confidence interval) of various ensemble methods vs size of the ensemble for different datasets.

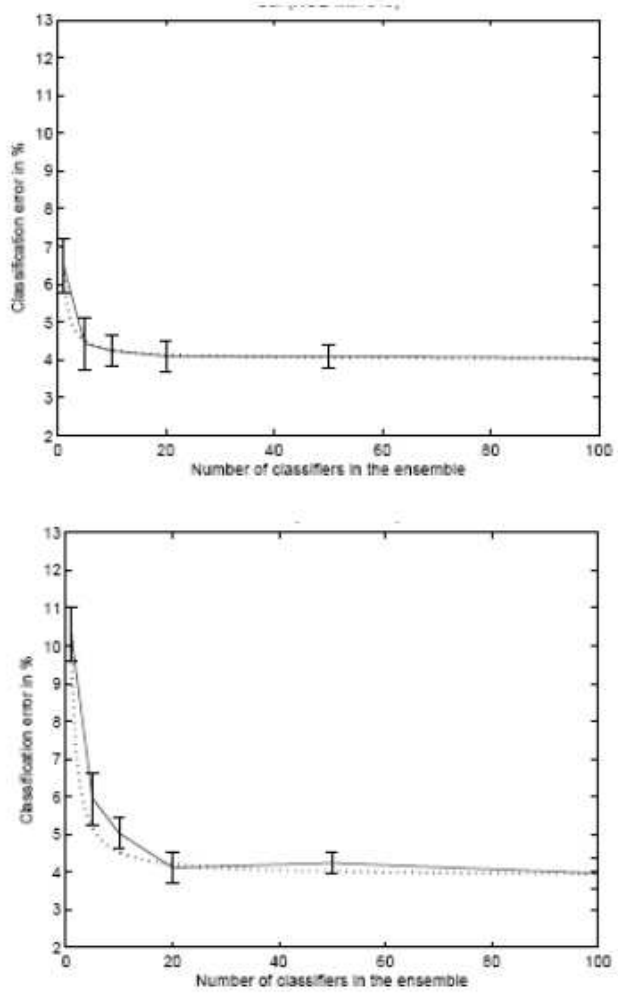


Figure 6.14: Classification error (with 95% confidence interval) of RO ensembles (top fig. ROE with J48 and bottom fig. ROE with RT, solid line) for the Car dataset with expected classification error (dotted line) using Fumera et al. [42] framework. The Y-axis of the graph represents testing error in % of the ensemble, and the X- axis represents the number of classifiers in the ensemble.

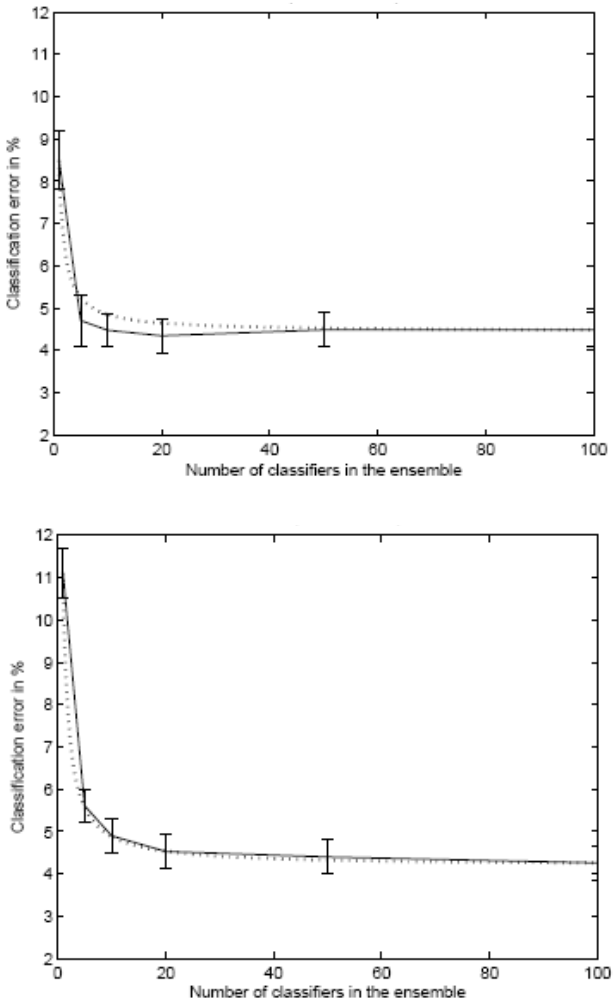


Figure 6.15: Classification error (with 95% confidence interval) of RO ensembles (top fig. ROE with J48 and bottom fig. ROE with RT, solid line) for the DNA dataset with expected classification error (dotted line) using Fumera et al. [42] framework. The Y-axis of the graph represents testing error in % of the ensemble, and the X- axis represents the number of classifiers in the ensemble.

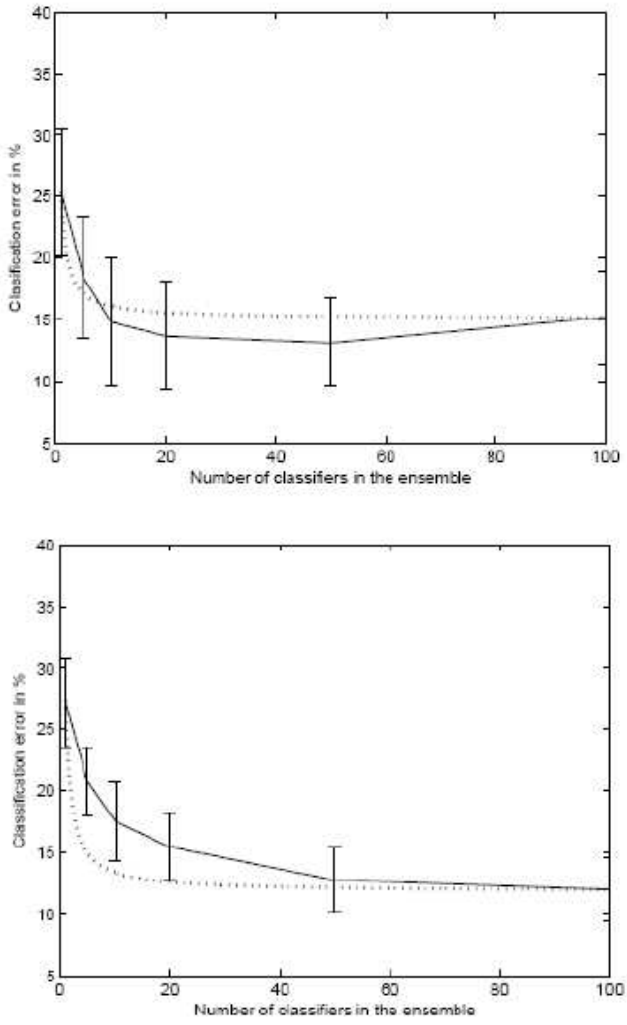


Figure 6.16: Classification error (with 95% confidence interval) of RO ensembles (top fig. ROE with J48 and bottom fig. ROE with RT, solid line) for the Promoter dataset with expected classification error (dotted line) using Fumera et al. [42] framework. The Y-axis of the graph represents testing error in % of the ensemble, and the X- axis represents the number of classifiers in the ensemble.

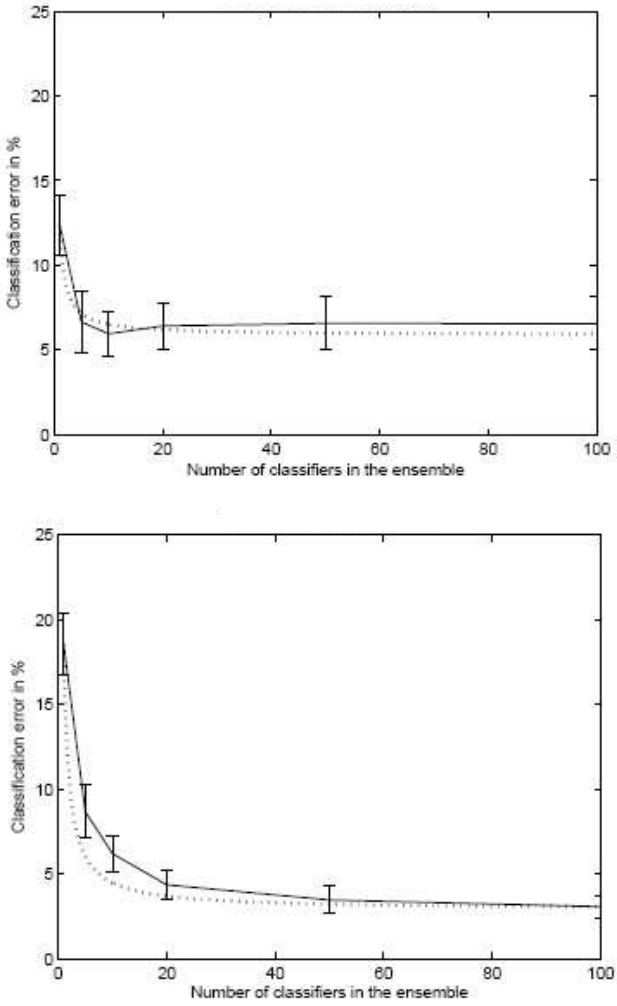


Figure 6.17: Classification error (with 95% confidence interval) of RO ensembles (top fig. ROE with J48 and bottom fig. ROE with RT, solid line) for the Tic-tac-toe dataset with expected classification error (dotted line) using Fumera et al. [42] framework. The Y-axis of the graph represents testing error in % of the ensemble, and the X- axis represents the number of classifiers in the ensemble.

and E_V, we need experiment values of $E(M)$ (classification error of the ensemble of size M) at two different values of M. Once, we know the values of E_B and E_V, we can predict the expected performance of ensembles of different sizes without doing any experiment. We used the experimental values of E(1) and E(100) to compute the values of E_B and E_V.

Fig. 6.14, Fig. 6.15, Fig. 6.16 and Fig. 6.17 show the experimental error of the RO with J48 ensembles for different ensemble sizes. We have also provided the errors calculated using the Fumera et al. analytical relationship [42]. This model fits well with experimental data for all the datasets. We carried out the same exercise for the RO with RS ensembles. The results are presented in Fig. 6.14, Fig. 6.15, Fig. 6.16 and Fig. 6.17 show the experimental error of the RO with J48 ensembles for different ensemble sizes.. For Car and DNA datasets, this model predicts correctly the performance of ensembles. For Promoter and Tic-Tac-Toe, empirical errors of smaller size ensembles are slightly more than the predicted errors. However, for rest of the plots, it fits well with the experiment values. It suggests that theoretical framework proposed by Fumera et al. [42] is a useful tool for choosing the size of the RO ensembles.

6.7.5 Combinations of RO with the Other Ensemble Methods

We study how two popular ensemble methods, Bagging and AdaBoost.M1 get affected by RO. We follow the same strategy as discussed in chapter 5 to combine RO with Bagging and AdaBoost.M1. We created 100 trees with the original data using Bagging process. In the second process we created 10 different datasets using RO and 10 trees using Bagging are created for each dataset. Hence in both cases 100 trees are trained. However in the second process diversity of RO has been combined with the Bagging. The same procedure was followed with AdaBoost.M1. Experiments were carried out with the same 5x2 cross validations methodology as suggested in section 3.2.2. Results suggest (Table 6.17 and Table 6.18) that Bagging and AdaBoost.M1 both have similar or better performances (except Monks 1 where Bagging has better performance) with RO. This indicates that RO can be combined with other ensemble methods.

We summarize RO ensembles as follows;

1. RO trees are generally more accurate as compared to normal decision trees.

2. RO ensembles reduce the data fragmentation problem, and provide performance improvements over several standard ensemble methods.

Dataset	Bagging + RO	Bagging
Promoter	15.9	16.8
Hayes-Roth	20.1	25.2 (+)
Breast Cancer	30.4	31.4
Monks1	9.6	3.9 (-)
Monks2	30.7	45.5 (+)
Monks3	0	0
Balance	18.3	28.1 (+)
Soyalarge	7.9	8.1
Tic-tac-toe	4.9	10.3 (+)
Car	4.4	8.3 (+)
DNA	5.1	6.0 (+)
Mushroom	0	0
Nursery	1.2	2.7 (+)
win/draw/lose		7/5/1

Table 6.17: Comparative Study of Bagging against RO + Bagging. '+/-' shows that performance of RO + Bagging is statistically better/worse than Bagging for that dataset. Results suggest that RO can be combined with Bagging to improve the performance of Bagging.

Dataset	AdaBoost.M1 + RO	AdaBoost.M1
Promoter	13.7	14.2
Hayes-Roth	18.6	23.4 (+)
Breast Cancer	32.4	35.6
Monks1	3.8	3.2
Monks2	30.7	45.5 (+)
Monks3	0	0
Balance	21.3	29.4 (+)
Soyalarge	7.7	7.8
Tic-tac-toe	1.8	2.3
Car	3.2	5.8 (+)
DNA	4.8	4.9
Mushroom	0	0
Nursery	0.6	1.0 (+)
win/draw/lose		6/7/0

Table 6.18: Comparative Study of AdaBoost.M1 against RO + AdaBoost.M1. '+/-' shows that performance of RO + AdaBoost.M1 is statistically better/worse than AdaBoost.M1 for that dataset. Results suggest that RO can be combined with AdaBoost.M1 to improve the performance of AdaBoost.M1.

3. Kappa-diversity plots suggest that RO ensembles have accurate classifiers with reasonable diversity.

4. RO trees are smaller than normal multi-split decision trees.

5. We can predict the performance of RO ensembles using the theoretical formalism for ensembles proposed by Fumera et al. [42].

6. This suggests that if we have to create an ensemble with only a small number of classifiers RO with J48 is more useful. For an ensemble with a large number of classifiers, RO with attribute randomization of Random Subspaces is a better choice.

7. ROE is easy to implement. Parallel implementation of RO ensembles is also possible.

6.8 Conclusion

In this chapter, we presented a data transformation scheme, random ordinality (RO), that projects multi-valued categorical data to a continuous space. The motivation for this approach is that categorical attribute values have no intrinsic order, hence, random order can be imposed on these vales. This technique is used to create diverse binary decision trees. Decision tree ensembles created by using RO are quite robust to the data fragmentation problem. The information theoretic framework suggest that RO attributes are good for classification. The comparative study of RO ensembles against other popular ensemble methods show the effectiveness of this approach.

Chapter 7

Conclusion and Future work

7.1 Contributions of the Thesis

Our main contribution in this thesis is to study and develop various data transformation techniques that can be used to create classifiers ensembles. These data transformation techniques can be combined with the existing ensemble techniques to improve them. We also showed that random linear oracle ensemble [68] technique can be studied using random projections. We summarize our contributions in the following points.

- Linear multivariate decision trees have better representational power than univariate decision trees, however, it is computationally expensive to create linear multivariate decision trees. We presented a computationally efficient technique to create ensembles of linear multivariate decision trees. This technique uses random projections. We then showed that this technique is a generalization of the random linear oracle framework [68] that is proposed to improve the performance of various ensemble methods. We presented the experimental results that indicate that the improvement in Bagging by using the proposed method is more than that achieved by using the random linear oracle framework.

- We showed that the discretization technique can be used to create ensembles. We proposed a novel disretization technique, Random Discretization (RD), to produce diverse discretized training datasets from a given continuous dataset. It creates random discretization boundaries. We then showed that RD ensembles have good representational power and can approximate any decision surface. We studied the performance of RD ensembles against other popular ensemble techniques. Results suggest that performance of RD ensembles is better than

Bagging and Random Forests and comparable with AdaBoost.M1. Experiments on the noisy data, suggest that the performance of RD ensembles is quite robust to the class noise.

- We showed how the proposed method to create linear multivariate decision trees can be employed to improve the performance of RD ensembles. Extensive experimental studies were carried out to show that the proposed ensemble method, Random Projection Random Discretization Ensembles (RPRDE), performs similar to or better than other ensemble methods. However, its competitive advantage is more for smaller ensembles. Experiments with the noisy data suggest that RPRDE is quite robust to the class noise. We also showed that RPRD can be combined with the other ensemble methods to improve them.

- We proposed a novel data transformation scheme, Random Ordinality (RO), for multi-valued categorical datasets. This technique is based on the data manipulation by imposing random ordinality on categorical attribute values. Trees, created by using RO, are binary, therefore, they are less affected by the data fragmentation problem. We further showed that RO ensembles reduce the data fragmentation problem, and provide significantly improved accuracies over current ensemble methods. We also showed that RO can be combined with the other popular ensemble methods to improve these methods. We presented the theoretical study of RO attributes to show that RO attributes are good for classification.

7.1.1 Conclusion

The learning of a classifier depends on the representations of datasets. If we know the properties of a classifier, we may change the representations of datasets such that the learning avoids the weaknesses of the classifier. Different data transformation schemes like discretization, PCA and random projections etc. are quite popular in the machine learning field for various reasons. Decision trees are very popular classifiers, however, they have two major weaknesses; the representational power and the data fragmentation problem. Ensembles of decision trees are quite useful because they generally give better accuracy than a single decision tree. The creation of diverse decision trees is the key to the success of an ensemble. Different data representations of a problem can be used to create diverse decision trees and can be combined to create ensembles. We showed that random projections and the discretization process are useful for the representational problem of decision trees. We also presented a novel data transformation

scheme, Random Ordinalty, that is useful for the data fragmentation problem. All the data transformation schemes that is studied or proposed in this thesis have random elements, in other words, these schemes are capable of producing diverse datasets. Hence, these schemes are useful in creating decision tree ensembles.

In summary, we showed in the thesis that data transformation techniques can be applied to generate ensembles of decision trees, and can be combined with existing ensemble techniques to improve their performance.

7.2 Future Work

- In Random Ordinality (RO) ensembles, we imposed random ordinality to each attribute independently. However, in future we will take into interdependencies of attributes into consideration while imposing random ordinality. As RO is based on data manipulation, experiments with other classifiers will also be one of the directions of the future research. We used a simulated dataset with known properties to study RO attributes by using the information theoretic framework. In future, we will try to develop a general framework for all kinds of multi-valued categorical datasets.

  RO does not need the information about the complete dataset as it only needs the information about the attribute values of the dataset. If we have the information about the attribute values of the dataset, the RO process can create diverse online decision trees. This will be an interesting future research field.

- For 5 out of 16 datasets, Random Discretization (RD) ensembles and Extreme Random Discretization (ERD) ensembles are statistically different. This is quite interesting as there is not much difference in the structure of a RD tree and a ERD tree. This point needs further investigation as it shows that different kinds of random discretization methods have different strengths. The study of combination of RD trees and ERD trees could be one direction of the future research. The study of ensembles of naive Bayes classifiers will be an interesting exercise to attempt in future as the discretization is effective for naive Bayes classifiers [97].

- ERD does not need the information about all the data points to create the bin boundaries. If we know the range of different attributes, ERD can be used to

create diverse online decision trees. ERD for online ensembles will be a research direction. Discretization techniques [25, 70] are useful for time series domains. In futures, we will employ the philosophy of the random dicretization for creating ensembles for the classification of time series.

- Regression by classification [89] is an interesting idea, this is done by transforming the range of continuous goal variable values into a set of intervals that will be used as discrete classes. ERD can be used to create diverse datasets, that in turn will create diverse classifiers. These will be used to create ensembles.

- RO ensembles work for multi-valued categorical datasets whereas RD ensembles work for pure continuous datasets. Some datasets have both kinds of attributes (multi-valued categorical attributes and continuous attributes), a combination of the RO process and the RD process for these kinds of datasets will be one of the future research work.

- In this thesis, we showed that Multi-RLE can be used to improve the performance of Bagging. In future, we will study the effect of Multi-RLE on other ensemble methods. As RLO improves the performance of different ensemble methods, we expect that Multi-RLE should be useful in improving different ensemble methods.

- In Random Projection Random Discretization (RPRDE), we use random projections to create new attributes. Different random matrices have been proposed to create random projections. In this thesis, we carried out experiments with one random matrix. Experiments with other random matrices [1] will be one of the future research directions.

- In RPRDE, we use random projections to create new attributes. There can be different methods to create these new attributes, for example kernel functions. Kernel machines have been very popular because of their excellent representational power. Balcan and Blum [5, 6] propose kernel attributes and similarity function attributes. The interesting point about these attributes is that the mapping, used to create these attributes, is random. In other words, this mapping creates different attributes in different runs. These attributes may be used to create ensembles. This will combine the ensemble philosophy with the kernel machines. It will be interesting research field as it may combine the robustness of ensemble methods with the representational power of kernel machines.

Appendix A

A.1 Datasets

Dataset Name	No. of Data Points	No. of Classes	No. of multi-valued attributes	No. of Binary attributes
Promoter	106	2	57	-
Hayes-Roth	160	3	4	-
Breast Cancer	286	2	7	3
Monks-1	432	2	4	2
Monks-2	432	2	4	2
Monks-3	432	2	4	2
Balance	625	3	4	-
Soyalarge	683	19	19	16
Tic-tac-toe	958	2	9	-
Car	1728	4	6	-
DNA	3190	3	60	-
Mushroom	8124	2	18	4
Nursery	12960	2	7	1

Table A.1: Datasets used in experiments. All datasets are categorical.

A.2 The Kappa measure

The Kappa measure is defined as follows: let us consider a problem with K classes with n data points, and let C be a $K \times K$ matrix such that C_{ij} contains the number of instances assigned to class *i* by the first classifier and to class *j* by the second classifier.

Dataset Name	Size	No. of Classes	No. of cont. attributes
Balance	625	3	4
Breast Cancer	699	2	9
Ecoli	336	8	7
Glass	215	7	9
Ionosphere	351	2	34
Letter	20000	26	16
Optical	5620	10	64
Pendigit	10992	10	16
Pima-diabetes	768	2	8
Phoneme	5404	2	5
RingNorm	7400	2	20
Satimage	6435	6	36
Segment	2310	7	19
Sonar	208	2	60
Spam	4601	2	57
TwoNorm	7400	2	20
Vehicle	846	4	18
Vowel	990	11	10
Waveform21	5000	3	21
Waveform40	5000	3	40
Yeast	1484	10	8

Table A.2: Datasets used in experiments. These datasets are pure continuous datasets.

We define two quantities,

$$\theta_1 = \frac{\sum_{i=1}^{K} C_{ii}}{n}, \tag{A.1}$$

$$\theta_2 = \sum_{i=1}^{K} \{ \sum_{j=1}^{K} \frac{C_{ij}}{n} \sum_{j=1}^{K} \frac{C_{ji}}{n} \}, \tag{A.2}$$

where θ_1 is the observed agreement between the classifiers and θ_2 is "agreement-by-chance".

The κ value is defined as

$$\kappa = \frac{\theta_1 - \theta_2}{1 - \theta_2}. \tag{A.3}$$

A.3 Results for RPRDE

In this section present a comparative study of RPRDE against the other popular ensemble methods. We also present a comparative study on the noisy data.

Dataset	Bagging	AdaBoost.M1	MultiBoosting	Random Forests	Single Tree
Balance	+	+	+	+	+
Breast Cancer	Δ	Δ	Δ	Δ	+
Ecoli	Δ	Δ	Δ	Δ	Δ
Glass	Δ	Δ	Δ	Δ	+
Iono	Δ	Δ	Δ	Δ	+
Letter	+	Δ	+	+	+
Optical	+	Δ	Δ	Δ	+
Pendigit	+	+	+	+	+
Pima-Dia	Δ	Δ	Δ	Δ	Δ
Phoneme	+	Δ	Δ	Δ	+
RingNorm	+	+	+	+	+
Satimage	Δ	Δ	Δ	Δ	+
Segment	+	Δ	+	+	+
Sonar	Δ	Δ	Δ	Δ	+
Spam	Δ	Δ	Δ	Δ	+
TwoNorm	+	+	+	+	+
Vehicle	Δ	Δ	Δ	Δ	Δ
Vowel	+	+	+	+	+
Waveform21	+	+	+	+	+
Waveform40	+	+	+	+	+
Yeast	Δ	+	Δ	Δ	+
win/draw/loss	11/10/0	8/13/0	9/12/0	9/12/0	18/3/0

Table A.3: Comparison Table - The ensembles size 10, '+' shows that performance of RPRDE is statistically better than that algorithm for that dataset, '-' shows that RPRDE is statistically worse for that dataset than this algorithm, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RPRDE and that algorithm.

Dataset	Bagging	AdaBoost.M1	MultiBoosting	Random Forests	Single Tree
Balance	+	+	+	+	+
Breast Cancer	Δ	Δ	Δ	Δ	+
Ecoli	Δ	Δ	Δ	Δ	Δ
Glass	Δ	Δ	Δ	Δ	+
Iono	Δ	Δ	Δ	Δ	+
Letter	+	Δ	Δ	+	+
Optical	+	-	Δ	Δ	+
Pendigit	+	Δ	Δ	+	+
Pima-Dia	Δ	+	Δ	Δ	Δ
Phoneme	+	Δ	Δ	Δ	+
RingNorm	+	+	+	+	+
Satimage	+	Δ	Δ	Δ	+
Segment	+	Δ	Δ	Δ	+
Sonar	Δ	Δ	Δ	Δ	+
Spam	+	Δ	-	Δ	+
TwoNorm	+	+	+	+	+
Vehicle	Δ	Δ	Δ	Δ	Δ
Vowel	+	+	+	Δ	+
Waveform21	+	+	+	Δ	+
Waveform40	+	Δ	Δ	Δ	+
Yeast	Δ	+	+	Δ	+
win/draw/loss	13/8/0	7/13/1	6/14/1	5/16/0	18/3/0

Table A.4: Comparison Table - The ensemble size 100, '+' shows that performance of *RPRD* is statistically better than that algorithm for that dataset, '-' shows that RPRDE is statistically worse for that dataset than this algorithm, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RPRDE and that algorithm.

Dataset	Bagging	AdaBoost.M1	MultiBoosting	Random Forests	Single Tree
Balance	+	+	+	+	+
Breast Cancer	Δ	Δ	Δ	Δ	+
Ecoli	Δ	+	+	+	+
Glass	Δ	Δ	Δ	Δ	+
Iono	Δ	+	+	Δ	+
Letter	+	+	+	+	+
Optical	+	+	+	+	+
Pendigit	+	+	+	+	+
Pima-Dia	Δ	Δ	Δ	Δ	Δ
Phoneme	+	+	+	+	+
RingNorm	+	+	+	+	+
Satimage	+	+	+	Δ	+
Segment	+	+	+	+	+
Sonar	Δ	Δ	Δ	Δ	+
Spam	-	+	Δ	Δ	+
TwoNorm	+	+	+	+	+
Vehicle	Δ	Δ	Δ	Δ	Δ
Vowel	+	+	+	+	+
Waveform21	+	+	+	+	+
Waveform40	+	+	+	+	+
Yeast	Δ	+	+	+	+
win/draw/loss	12/8/1	16/5/0	15/6/0	13/8/0	19/2/0

Table A.5: Comparison Table - The ensembles size 10, '+' shows that performance of *RPRD* is statistically better than that algorithm for that dataset, '-' shows that *RPRD* is statistically worse for that dataset than this algorithm, 'Δ' shows that there is no statistically significant difference in performance for this dataset between *RPRD* and that algorithm. The class noise is 10%.

Dataset	Bagging	AdaBoost.M1	MultiBoosting	Random Forests	Single Tree
Balance	+	+	+	+	+
Breast Cancer	Δ	+	Δ	Δ	+
Ecoli	Δ	+	+	Δ	+
Glass	Δ	Δ	Δ	Δ	+
Iono	Δ	+	+	Δ	+
Letter	+	+	+	+	+
Optical	+	Δ	Δ	Δ	+
Pendigit	+	+	+	+	+
Pima-Dia	Δ	Δ	Δ	Δ	Δ
Phoneme	+	+	+	+	+
RingNorm	+	+	+	+	+
Satimage	+	Δ	Δ	Δ	+
Segment	+	+	+	+	+
Sonar	Δ	Δ	Δ	Δ	+
Spam	Δ	+	Δ	Δ	+
TwoNorm	+	+	+	+	+
Vehicle	Δ	Δ	Δ	Δ	+
Vowel	+	+	+	+	+
Waveform21	+	+	Δ	+	+
Waveform40	+	Δ	Δ	Δ	+
Yeast	+	+	+	+	+
win/draw/loss	13/8/0	14/7/0	11/10/0	10/11/0	20/1/0

Table A.6: Comparison Table - The ensembles Size 100, '+' shows that performance of *RPRD* is statistically better than that algorithm for that dataset, '-' shows that RPRDE is statistically worse for that dataset than this algorithm, 'Δ' shows that there is no statistically significant difference in performance for this dataset between RPRDE and that algorithm. The class noise is 10%.

Bibliography

[1] D. Achlioptas, *Database-friendly Random Projections*, In Proc. ACM Symp. on the Principles of Database Systems, 2001, p. 274281.

[2] D. Achliptas, F. Mcsherry, and B. Scholkopf, *Sampling Techniques for Kernel Methods*, In Annual Advances in Neural Information Processing Systems, 2001, pp. 335–342.

[3] A. Agresti, *An Introduction to Categorical Data Analysis*, WileyBlackwell, 2007.

[4] E. Alpaydin, *Combined 5 x 2 cv f Test Comparing Supervised Classification Learning Algorithms*, Neural Computation **11** (1999), no. 8, 1885–1892.

[5] M. F. Balcan and A. Blum, *On a Theory of Learning with Similarity Functions*, Proceedings of the 23rd International Conference on Machine Learning, 2006.

[6] M. F. Balcan, A. Blum, and S. Vempala, *Kernels as Features: On Kernels, Margins, and Low-dimensional Mappings*, Machine Learning **65** (2006), 79–94.

[7] A. Bertoni and G. Valentini, *Ensembles Based on Random Projections to Improve the Accuracy of Clustering Algorithms*, Neural Nets, WIRN 2005,, 2006, pp. 31–37.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag New York Inc, 2008.

[9] A. Blum and R. L. Rivest, *Training a 3-node neural network is np-complete*, In proceeding of the 1988 Workshop on Computational Learning Theory, 1988, pp. 9–18.

[10] I. Bratko and I. Kononenko, *Learning Diagnostic rules from incomplete and noisy data*, Seminar on AI Methods in Statistics, London, 1986.

[11] L. Breiman, *Bagging Predictors*, Machine Learning **24** (1996), no. 2, 123–140.

[12] ———, *Randomizing Outputs to Increase Prediction Accuracy*, Machine Learning **40** (2000), no. 3, 229–242.

[13] ———, *Random Forests*, Machine Learning **45** (2001), no. 1, 5–32.

[14] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, CA: Wadsworth International Group, 1985.

[15] L. Brieman, *Arching classifiers*, The Annals of Statistics **26** (1998), no. 3, 801–824.

[16] C. E. Brodley and P. E. Utgoff, *Multivariate Decision Trees*, Machine Learning **19** (1995), no. 1, 45 – 77.

[17] C. J. C. Burges, *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery **2** (1998), 121167.

[18] T. Cai and X. Wu, *Research on Ensemble Learning based on Discretization Method*, 9th International Conference on Signal Processing, 2008, pp. 1528–1531.

[19] E. Cantu-Paz and C. Kamath, *Inducing Oblique Decision Trees with Evolutionary Algorithms*, IEEE Transaction on Evolutionary Computation **7** (2003), no. 1, 54–68.

[20] J. Catlett, *Megainduction: Machine learning on very large databases*, Ph.D. thesis, Basser Department of Computer Science, University of Sydney, 1991.

[21] C. C. Chan and C. Batur A. Srinivasan, *Determination of Quantization Intervals in Rule Based Model for Dynamic Systems*, Proceedings of the IEEE Conference on Systems, Man, and Cybernetics, 1991, pp. 1719–1723.

[22] S. Dasgupta, *Learning Mixtures of Gaussians*, In 40th Annual IEEE Symp. on Foundations of Computer Science, 1999, pp. 634–644.

[23] ———, *Experiments with Random Projection*, In Proc. Uncertainty in Artificial Intelligence, 2000.

[24] S. Dasgupta and A. Gupta, *An Elementary Proof of the Johnson-lindenstrauss Lemma*, Tech. Report TR-99-006, International Computer Science Institute, Berkeley, California, USA, 1999.

[25] C. S. Daw, C. E. A. Finney, and E. R. Tracy, *A Review of Symbolic Analysis of Experimental Data*, Review of Scientific Instruments. **74** (2003), no. 2, 915–930.

[26] T. Van de Merckt, *Decision Trees in Numerical Attribute Spaces*, Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1993, pp. 1016–1021.

[27] J. DePasquale and R. Polikar, *Random feature subset selection for ensemble based classification of data with missing features*, MCS 2007, Springer Berlin / Heidelberg, 2007, pp. 251–260.

[28] T. G. Dietterich, *Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms*, Neural Computation **10** (1998), 1895–1923.

[29] ———, *Ensemble Methods in Machine Learning*, Proc. of Conf. Multiple Classifier Systems, vol. 1857, 2000, pp. 1–15.

[30] ———, *An Experimental Comparison of Three Methods for Constructing Ensembles of Decision trees: Bagging, Boosting, and randomization*, Machine Learning **40** (2000), no. 2, 1–22.

[31] T. G. Dietterich and G. Bakiri, *Solving multiclass learning problems via error-correcting output codes*, J. Artificial Intelligence Research **2** (1995), 263–286.

[32] J. Dougherty, R. Kahavi, and M. Sahami, *Supervised and unsupervised discretization of continuous features.*, In Machine Learning: Proceedings of the Twelth International Conference, 1995.

[33] W. Fan, J. McCloskey, and P. S. Yu, *A General Framework for Accurate and Fast Regression by Data Summarization in Random Decision Trees*, In proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006, pp. 136 – 146.

[34] U. M. Fayyad and K. B. Irani, *Multi-interval Discretization of Continuous valued Attributes for Classification Learning*, In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 1993, pp. 1022–1027.

[35] U. M. Fayyad and K. B. Irani, *The Attribute Selection Problem in Decision Tree Generation*, Proc. AAAI-92, MIT Press, July 1992.

[36] X. Z. Fern and C. E. Brodley, *Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach*, ICML, 2003, pp. 186–193.

[37] X.Z. Fern and C.E. Brodley, *Random projection for high dimensional data clustering: A cluster ensemble approach*, Proc. of ICML, 2003.

[38] D. Fradkin and D. Madigan, *Experiments with Random Projections for Machine Learning*, In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 517–522.

[39] Y. Freud and R. E. Schpire, *Experiment with a new boosting algorithm*, Proceeding of 13th Conference on Machine learning, 1996.

[40] Y. Freund, *Boosting a Weak Learning Algorithm By Majority*, Information and Computation **121** (1995), no. 2, 256–285.

[41] Y. Freund and R. E. Schapire, *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, Journal of Computer and System Sciences **55** (1997), no. 1, 119–139.

[42] G. Fumera, R. Fabio, and S. Alessandra, *A Theoretical Analysis of Bagging as a Linear Combination of Classifiers*, IEEE Transactions on PAMI **30** (2008), no. 7, 1293–1299.

[43] G. Fumera, F. Roli, and A. Serrau, *Dynamics of Variance Reduction in Bagging and Other Techniques Based on Randomisation*, In Proc. of Conf. Multiple Classifier Systems MCS2005, 2005, pp. 316–325.

[44] J. Gama, *Oblique Linear Tree*, In Second International Symposium on Advances in Intelligent Data Analysis, (X. Liu and P. Cohen, eds.), Springer-Verlag, 1997, pp. 187–198.

[45] ———, *Functional Trees*, Machine Learning **55** (June 2004), no. 3, 219–250.

[46] N. Garca-Pedrajas, C. Garca-Osorio, and C. Fyfe, *Nonlinear Boosting Projections for Ensemble Construction*, Journal of Machine Learning Research **8** (2007), 1–33.

[47] P. Geurts, D. Ernst, and L. Wehenkel, *Extremely Randomized Trees*, Machine Learning **63** (2006), no. 1, 3–42.

[48] L. K. Hansen and P. Salamon, *Neural Network Ensembles*, IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1992), no. 10, 993–1001.

[49] D. G. Heath, S. Kasif, and S. Salzberg, *Induction of Oblique Decision Trees*, in Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1993, pp. 1002–1007.

[50] ———, *Committees of Decision Trees*, Cognitive Technology: In Search of a Human Interface (Amsterdam, The Netherlands) (B. Gorayska and J. Mey, eds.), Elsevier Science, 1996, pp. 305–317.

[51] R. Hecht-Nielsen, *Context Vectors: General Purpose Approximate Meaning Representations Self-organized from Raw Data*, Computational Intelligence (1994), 4356.

[52] C. Hegde, M.B. Wakin, and R.G. Baraniuk, *Random Projections for Manifold Learning*, in Neural Information Processing Systems (NIPS), 2007.

[53] K. M. Ho and P. D. Scott, *Overcoming Fragmentation in Decision Trees Through Attribute Value Grouping*, In Proceedings of Second European Symposium, PKDD 98 Nantes, France., 1998, pp. 337–344.

[54] T. K. Ho, *The Random Subspace Method for Constructing Decision Forests*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 8, 832–844.

[55] R. C. Holte, *Very Simple Classification Rules Perform Well on Most Commonly used Datasets*, Machine Learning **11** (1993), 63–90.

[56] E. Hunt, J. Martin, and P. Stone, *Experiments in Induction*, Academic Press, New York, 1966.

[57] L. Hyafil and R. L. Rivest, *Constructing optimal binary decision trees is np-complete*, Information Processing Letter **5** (1976), no. 1, 15–17.

[58] P. Indyk and R. Motwani, *Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality*, In Proceedings of the 30th Annual ACM STOC, 1998, pp. 604–613.

[59] V. S. Iyengar, *Hot: Heuristics for Oblique Trees*, In Eleventh International Conference on Tools with Artificial Intelligence., IEEE Press, 1999, pp. 91–98.

[60] W.B. Johnson and J. Lindenstrauss, *Extensions of Lipshitz Mapping into Hilbert Space*, In Conference in modern analysis and probability, volume 26 of Contemporary Mathematics, Amer. Math. Soc., 1984, pp. 189–206.

[61] I. T. Jolliffe, *Principal component analysis*, Springer, 2002.

[62] C. Kamath, E. Cantu-Paz, and D. Littau, *Approximate splitting for ensembles of decision trees using histrograms*, In Proceedings of the 2nd SIAM International Conference on Data Mining, 2002.

[63] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas, *On Combining Classifiers*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 3, 226–239.

[64] R. Kohavi and M. Sahami, *Error-based and Entropy-based Discretization of Continuous Features*, In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 114–119.

[65] T. Kohonen, *Selforganization and Associative Memory*, SpringerVerlag, Berlin, Germany, 1989.

[66] I. Kononenko, *A Counter Example to the Stronger Version of the Binary Tree Hypothesis*, In ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases, 1995, pp. 31–36.

[67] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-Interscience, 2004.

[68] L.I. Kuncheva and J.J. Rodriguez, *Classifier ensembles with a random linear oracle*, IEEE Trans. on Knowledge and Data Engineering **19** (2007), no. 4, 500–508.

[69] W. Kwedlo and M. Kretowski, *An Evolutionary Algorithm using Mulivariate Discretization for Decision Rule Induction*, In Principles of Data Mining and Knowledge Discovery, 1999, pp. 392–397.

[70] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, *A Symbolic Representation of Time Series, with Implications for Streaming Algorithms*, In proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. San Diego, CA, 2003, pp. 2–11.

[71] W. Maass, *Efficient Agnostic Pac-learning with Simple Hypotheses*, In Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory, 1994, pp. 67–75.

[72] D. D. Margineantu and T. G. Dietterich, *Pruning Adaptive Boosting*, Proc. 14th International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 211–218.

[73] J. Maudes, J. J. Rodriguez, and C. G. Osorio, *Disturbing Neighbors Diversity for Decision Forests*, Supervised and Unsupervised Ensemble Methods and Their Applications (SUEMA 2008), 2008, pp. 67–71.

[74] T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[75] F. Moosmann, E. Nowak, and F. Jurie, *Randomized Clustering Forests for Image Classification*, IEEE Transaction on Pattern Analysis and Machine Intelligence **30** (September 2008), no. 9, 1632–1646.

[76] S. K. Murthy, S. Kasif, S. Salzberg, and R. Beigel, *Oc1: A Randomized Induction of Oblique Decision Trees*, In Proceedings of the Eleventh National Conference on Artificial Intelligence, 1993, pp. 322–327.

[77] T. Oates and D. Jensen, *The Effects of Training Set Size on Decision Tree Complexity*, Proc. 14th International Conference on Machine Learning, 1997, pp. 254–262.

[78] L. E. Peterson and M. A. Coleman, *Principal Direction Linear Oracle for Gene Expression Ensemble Classification.*, Workshop on Computational Intelligence Approaches for the Analysis of Bioinformatics Data (CIBIO07); 2007 Int. Joint Conference on Neural Networks (IJCNN07)., 2007.

[79] R. Polikar, *Ensemble Based Systems in Decision Making*, IEEE Circuits and Systems Magazine (2006), 21–45.

[80] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[81] J. R. Quinlan, *Improved Use of Continuous Attributes in C4.5*, J. Artif. Intell. Res. **4** (1996), 7790.

[82] G. Rätsch, T. Onoda, and K.-R. Müller, *Soft margins for AdaBoost*, Machine Learning **42** (2001), no. 3, 287–320.

[83] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, *Rotation Forest: A New Classifier Ensemble Method*, IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006), no. 10, 1619–1630.

[84] J.J. Rodriguez and L.I. Kuncheva, *Naive bayes ensembles with a random oracle*, In Proc. 7th International Workshop on Multiple Classifier Systems, MCS07, LNCS 4472, 2007, pp. 450–458.

[85] A. Schclar and L. Rokach, *Random Projection Ensemble Classifiers*, In Proc. of 11th International Conference on Enterprise Information Systems, 2009, pp. 309–316.

[86] B. Scholkopf, A. J. Smola, and K. Muller, *Nonlinear Component Analysis as a kernel Eigenvalue problem.*, Neural Computation **10** (1998), 12991319.

[87] J. Shlens, *A Tutorial on Components Analysis*, 2005.

[88] L. Smith, *A Tutorial on Components Analysis*, Tech. report, University of Otago (New Zealand), 2002.

[89] L. Torgo and J. Gama, *Regression by Classification*, Advances in Artificial Intelligence, 1996, pp. 51–60.

[90] K. Tumer and J. Ghosh, *Error Correlation and Error Reduction in Ensemble Classifiers*, Connect. Sci. **8** (1996), no. 3, 385–404.

[91] R. Avogadri G. Valentini:, *Fuzzy Ensemble Clustering based on Random Projections for dna Microarray Data Analysis.*, Artificial Intelligence in Medicine, 2009, pp. 173–183.

[92] L. A. van der Ark, M. A. Croon, and K. Sijtsma, *New Developments in Categorical Data Analysis for the Social and Behavioral Sciences*, Psychology Press, 2004.

[93] V. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, New York, 1998.

[94] R. Vilalta, G. Blix, and L. Rendell, *Global Data Analysis and the Fragmentation Problem in Decision Tree Induction*, In Pro. of the 9th ECML, 1997, pp. 312–328.

[95] G. I. Webb, *Multiboosting: A Technique for Combining Boosting and Wagging*, Machine Learning **40** (2000), no. 2, 159–196.

[96] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques.*, 2 ed., Morgan Kaufmann San Francisco, CA, 2005.

[97] Y. Yang and I.W. Webb, *A comparative study of discretization methods for naive-bayes classifiers*, In Proc. of the 2002 Pacific Rim Knowledge Acquisition Workshop (PKAW'02), 2002, pp. 159–173.

[98] O. T. Yldz and E. Alpaydn, *Omnivariate Decision Trees*, IEEE Transactions on Neural Networks, VOL. 12, NO. 6, NOVEMBER 2001 **12** (2001), no. 6, 1539–1546.

[99] Li Yuanhong, D. Ming, and R. Kothari, *Classifiability-based Omnivariate Decision Trees*, IEEE Transactions on Neural Networks **16** (2005), no. 6, 1547–1560.