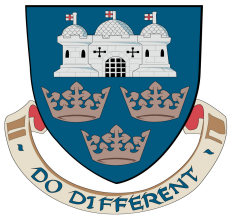
Empirical Evaluation of Semi-Supervised Naïve Bayes for Active Learning



Awat Abdulla Saeed

School of Computing Science

University of East Anglia

A thesis is submitted for the degree of

Doctor of Philosophy

May 2018

This thesis is proudly dedicated to:

All my beloved family, the symbol of love and giving. My great parents, who never stop giving of themselves in countless ways. My dearest wife, Kazhan Misri, who leads me through the valley of darkness with light of hope and support. My beloved sons: Arya, and Asa, whom I can't force myself to stop loving. My beloved brothers and sisters, and my parents in law. Thanks for your endless love, sacrifices, prayers, supports and advice.

Acknowledgements

I am grateful to many people, who worked hard with me from the beginning to the completion of the present research, particularly my supervisor Dr. Gavin Cawley, who has always been generous during all phases of the research. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. Without his support, this study would not have been possible. I would also like to thank Dr. Tony Bagnall and Dr. Nicola L. C. Talbot for the suggestions and support they provided.

I would like to take this opportunity to say warm thanks to all my beloved friends and colleagues at UEA, who have been so supportive along the way of doing my thesis, to name few: Alex, Andris, Andrada, Andrei, Bogdan, Claudia, David, Guillaume, Haval, Hussain, James, Joshua, Matt, Omar, Rekar, Ruth, Salma, Sarah, Zanist, and Zherou.

I also would like to express my wholehearted thanks to my family for the generous support they provided me throughout my entire life and particularly through the process of pursuing the PhD. Because of their unconditional love and prayers, I have the chance to complete this thesis.

I owe profound gratitude to my wife, Kazhan Misri, whose constant encouragement, limitless giving and great sacrifice, helped me accomplish my degree.

I am especially thankful to my two lovely sons, Aray and Asa who are the pride and joy of my life. I love you more than anything and I appreciate all your patience and support during Daddy's Ph.D studies.

I would like to take this opportunity to thank the Kurdistan Regional Government (KRG) for their generosity funding of many Kurdish students. I am very honoured to be the recipient of this award.

Finally I thank my God, for letting me through all the difficulties. You are the one who let me finish my degree. I will keep on trusting You for my future.

Last but not least, my deepest thanks go to all people who took part in making this thesis real.

Abstract

This thesis describes an empirical evaluation of semi-supervised and active learning individually, and in combination for the naïve Bayes classifier. Active learning aims to minimise the amount of labelled data required to train the classifier by using the model to direct the labelling of the most informative unlabelled examples. The key difficulty with active learning is that the initial model often gives a poor direction for labelling the unlabelled data in the early stages. However, using both labelled and unlabelled data with semi-supervised learning might be achieve a better initial model because the limited labelled data are augmented by the information in the unlabelled data. In this thesis, a suite of benchmark datasets is used to evaluate the benefit of semi-supervised learning and presents the learning curves for experiments to compare the performance of each approach. First, we will show that the semi-supervised naïve Bayes does not significantly improve the performance of the naïve Bayes classifier. Subsequently, a down-weighting technique is used to control the influence of the unlabelled data, but again this does not improve performance. In the next experiment, a novel algorithm is proposed by using a sigmoid transformation to recalibrate the overly confident naïve Bayes classifier. This algorithm does not significantly improve on the naïve Bayes classifier, but at least does improve the semi-supervised naïve Bayes classifier. In the final experiment we investigate the effectiveness of the combination of active and semi-supervised learning and empirically illustrate when the combination does work, and when does not.

Table of contents

List of figures **xiii**

List of tables **xxiii**

1 Introduction **1**

1.1 Introduction 1

1.2 Motivation 5

1.3 Outline of the thesis 11

1.4 Contributions of the thesis 12

2 Technical background and Literature Review **15**

2.1 Supervised learning 16

2.1.1 Maximum likelihood estimation for categorical features 18

2.1.2 Maximum likelihood estimation for continuous features 20

2.2 Semi-Supervised learning 21

2.2.1 Self-training: 27

2.2.2 Semi-supervised learning with generative mixture models 34

2.2.3 The semi-supervised naïve Bayes classifier 36

2.2.4 Co-training 47

2.2.5 Low density separation 50

2.2.6 Graph-based semi-supervised learning methods 62

2.3 Active learning 67
2.4 Active learning scenarios 68
2.4.1 Membership query synthesis 68
2.4.2 Stream-based (Selective Sampling) 69
2.4.3 Pool-based selection 69
2.5 Active learning query strategies 70
2.5.1 Uncertainty sampling 70
2.5.2 Query-By-Committee 72
2.5.3 Expected error reduction 75
2.5.4 Density-Weighted methods 75
2.6 Performance Evaluation 77
2.7 Statistical comparisons of classifiers 80
2.8 Model selection 84

3 Benchmarking the Semi-supervised naïve Bayes classifier 87
3.1 Experiments 88
3.1.1 UCI benchmark datasets 88
3.1.2 Why is the naïve Bayes classifier significantly better on average than the semi-supervised naïve Bayes classifier? 93
3.1.3 Synthetic benchmark datasets 95
3.1.4 Exploratory data analysis 97
3.2 Conclusions 100

4 Differential weighting of labelled and unlabelled examples 101
4.1 Down-weighting of the unlabelled data 102
4.2 Technical background 103
4.3 Preliminary experiment for contribution of labelled and unlabelled data . . 107

4.3.1 Discussion 111
4.4 Using a validation set to determine the contribution of labelled and unlabelled data 113
4.4.1 Design of experiments 113
4.4.2 Results 113
4.5 Final experiments for contribution of labelled and unlabelled data 122
4.5.1 Design of experiments 122
4.5.2 Results 123
4.6 Conclusions 131

5 The problem of class frequency bias in the predicted labels for unlabelled examples 133

5.1 Overconfidence generated by uncalibrated probability classes 134
5.2 Effect of overconfident classification on EM based SSNB 136
5.3 Logit transformation function for predicted class probabilities 137
5.4 Logistic rank transformation function 141
5.4.1 Spreading out and fixing the average of the predicted class probabilities 143
5.4.2 Spreading out without fixing the average of the predicted class probabilities 149
5.4.3 Spreading out the predicted class probabilities using Transfer Learning 152
5.4.4 Spreading out the predicted class probabilities using cross-validation method 154
5.4.5 Does the logistic transformation improve the SSNB classifier? . . . 156
5.5 Conclusions 157

6 Investigation of active learning 159

6.1 Introduction to active learning 160

6.2 Comparing passive and active learning classifier 163
6.3 Investigation of anomalous learning curves 179
6.4 Combining active and semi-supervised learning 182
6.5 When does combining active and semi-supervised learning work? 197
6.6 Summary 201

7 Conclusions and Future Work **203**
7.1 Conclusions 203
7.2 Future Work 205

References **207**

List of figures

1.1 Passive learning framework [72] 2

1.2 Illustration of the machine learning example where unlabelled patterns can improve classification performance for the two moons synthetic benchmark dataset 3

1.3 Semi-supervised learning framework [72] 4

1.4 Active learning framework [72] 5

1.5 Illustrative example to show the active learning process to select informative unlabelled patterns using uncertainty sampling strategy for the two-moon synthetic dataset [76] 6

1.6 (a) The blue line represents the decision boundary for the naïve Bayes model trained on all training patterns and the red line represents the decision boundary for the naïve Bayes model trained with four good labelled patterns, two in each class. Illustration of picking the most informative patterns (b). (c) The decision boundary for active learning after being trained with one actively queried example is represented by a green line. 8

1.7 (a) The blue line represents the decision boundary for the naïve Bayes model trained on all training patterns and the red line represents the decision boundary for the naïve Bayes model trained with four poor labelled patterns, two in each class. (b) Illustrates picking the most informative patterns. (c) The decision boundary for active learning after being trained with one actively queried example is represented by a green line. 9

1.8 Comparison between active learning with a good initial model, pink-line, and active learning with a bad initial model, blue-line. The passive learning plotted by green dash-line. 10

1.9 Comparison between the learning curves for random (RL: blue-line), semi-supervised (SSL: red-line), passive (PL: green dash-line), and active learning with bad initial model (AL: pink-line) 11

2.1 (a) Gaussian Mixture Models using only a small amount of labelled data (b) labelled and unlabelled data for both components (c) Gaussian Mixture Models using labelled and labelled data that impact the decision boundary Zhu et al. [94] 23

2.2 (a) Synthetic dataset with binary classes in four clusters (b) the model has a good fit, but the assumptions of the model are wrong [93] 24

2.3 (a) The decision boundary lies in a low density region. (b) The decision boundary lies in a high density region 25

2.4 (a)The two moon synthetic dataset with 200 patterns in a two-dimensional space, 198 unlabelled patterns and two labelled patterns belong to two classes. Each of the blue and red classes contain one labelled pattern and the green unlabelled pattern is the selected pattern to be labelled. (b) Constructing a graph by propagating labels over the graph using both labelled and unlabelled patterns. Similar patterns of the labelled and unlabelled data are connected and they have similar labels. 26
2.5 Clusters in the data set of concentric clusters. 27
2.6 Self-training without outliers and the assumption is correct. 29
2.7 Self-training with outliers then the model assumption would be incorrect [93] 31
2.8 The $\log(x)$ function is a concave on $[x_1, x_2]$ if $\lambda \log(x_1) + (1-\lambda)\log(x_2) \leq \log(\lambda x_1 + (1-\lambda)x_2))$ where $\lambda \in [0,1]$ Ng et al. [56] 43
2.9 (a) Linear SVM decision boundary for six labelled patterns belonging to two classes. (b) TSVM decision boundary for both labelled and unlabelled patterns [after Zhu and Goldberg [93]]. 51
2.10 (a) Linear decision boundary for the hard-margin SVM. (b) Linear decision boundary for the soft-margin SVM. 52
2.11 (a) The hinge loss as a function of $yf(x)$. (b) The hat loss as a function of $f(x)$. 57
2.12 Heatmaps illustrating the query behaviour of common uncertainty measures in a three-label classification problem, (from [72]). 72
2.13 Version space examples for (a) linear and (b) axis-parallel box classifiers. (from [72]) 73

2.14 An illustration of selecting the green pattern, which is closest pattern from decision boundary, using uncertainty sampling query strategy. Uncertainty sampling is a poor strategy and does not improve the performance of the classifier, because distribution of green patterns is not illustrative of the distribution of the other patterns. [After Settles [72]] 76
2.15 Illustrative example for critical difference diagram 83
2.16 The cross-validation model selection method. 85
2.17 The leave-one-out cross-validation model selection method. 86

3.1 Two-class classification problem for the synthetic dataset. 94
3.2 The learning curve for a two-class classification problem in Gaussian distribution for synthetic dataset 94
3.3 The average learning curve for NB and SSNB of the UCI and synthetic (splice, newthyroid) datasets 98
3.4 The average learning curve for the NB and SSNB of the UCI and synthetic (nursery, waveform) dataset 99
3.5 The average learning curve for the NB and SSNB of the UCI and synthetic audiology datasets 100

4.1 Critical difference diagram for NB, SSNB, and SSNB-λ over 28 continuous benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar 109
4.2 Critical difference diagram for NB, SSNB, and SSNB-λ over 28 continuous benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar 111

4.3 Comparison of the error rate learning curve for the NB and SSNB-λ-cvs classifiers for the banknote benchmark dataset. 112
4.4 Critical difference diagram for NB, SSNB, and SSNB-λ-cvs over 28 continuous benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar 116
4.5 Critical difference diagram for NB, SSNB, and SSNB-λ-cvs over 36 discrete benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar 116
4.6 Comparison of the error rate learning curve for NB, SSNB and SSNB-λ-cvs classifiers for nine continuous benchmark datasets. 118
4.7 Comparison of the error rate learning curve for NB, SSNB and SSNB-λ-cvs classifiers for 19 discrete benchmark datasets. 121
4.8 Critical difference diagram for the NB, SSNB, and SSNB-λ-mcv over 28 continuous benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar 126
4.9 Critical difference diagram for the NB, SSNB, and SSNB-λ-mcv over 36 discrete benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar 126
4.10 Comparison of the error rate learning curve for NB, SSNB and SSNB-λ-cvs classifiers for eight continuous benchmark datasets. 128
4.11 Comparison of the error rate learning curve for NB, SSNB and SSNB-λ-cvs classifiers for 14 discrete benchmark datasets. 130

5.1 The average value of the predicted probabilities of the positive class for the unlabelled data for 100 replications of the Mushroom benchmark dataset. . 137

5.2 The average value of predicted probabilities for the positive class, for the unlabelled data for 100 replications of the Mushroom benchmark dataset, using Platt scaling [63] for rank transformation 141

5.3 Histogram of the predicted probabilities of the positive class for the unlabelled data for 100 replications of the Mushroom benchmark dataset for positive and negative classes. 142

5.4 (a) The average value of predicted probabilities for the positive class for unlabelled patterns, q_{ic_+}, using the logistic transformation. (b) The area under the error rate learning curve for 100 replications of the Mushroom benchmark dataset using the logistic transformation. 144

5.5 Critical difference diagram for the NB, SSNB, and SSNB-sigmoid-fb over 12 discrete and 16 continues benchmark datasets. It shows that there are statistically significant differences between the means ranks for the SSNB classifier and both NB and SSNB-sigmoid-fb classifiers 148

5.6 Critical difference diagram for the NB, SSNB, and SSNB-sigmoid over 18 discrete and 16 continues benchmark data sets. It shows that there are statistically significant differences between the means ranks for the SSNB classifier and both NB and SSNB-sigmoid classifiers, however there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar 151

5.7 Critical difference diagram for the NB, SSNB, and SSNB-sigmoid-LODO over 12 discrete and 16 continues benchmark datasets. It shows that there are statistically significant differences between the means ranks for the SSNB classifier and both NB and SSNB-sigmoid-LODO classifiers, however there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar 154

5.8 Critical difference diagram for the NB, SSNB, and SSNB-sigmoid-cv over 12 discrete and 16 continues benchmark datasets. It shows that there are statistically significant differences between the means ranks for the SSNB classifier and both NB and SSNB-sigmoid classifiers, however there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar 156

5.9 Critical difference diagram for the NB, SSNB, SSNB-sigmoid, SSNB-sigmoid-fb, SSNB-sigmoid-LODO and SSNB-sigmoid-cv over 12 discrete and 16 continues benchmark datasets. It shows that there are statistically significant differences between the means ranks for the SSNB, SSNB-sigmoid-LODO and SSNB-sigmoid-cv classifiers and NB, SSNB-sigmoid and SSNB-sigmoid-fb classifiers, however there are no statistically significant differences between the mean ranks for the NB, SSNB-sigmoid and SSNB-sigmoid-fb classifiers which are linked by the bar 157

6.1 Average test set recall learning curve for the `Hiva` benchmark dataset using the uncertainty sampling strategy of ALNB classifier over 5-run 5-fold cross validation 161

6.2 The average accuracy of the ALNB classifier for (a)`diabetes` and (b) `iris` 163

6.3 Comparison of the error rate learning curve for the NB and ALNB classifiers. In these cases, the ALNB better than the NB classifier and the difference between these two classifiers are statistically significant for 13 discrete benchmark datasets. 167
6.4 Comparison of the error rate learning curve for NB and ALNB classifiers. In these cases, the ALNB classifier performs slightly better than NB classifier over 14 discrete benchmark datasets. 169
6.5 Comparison of the error rate learning curve for NB and ALNB classifiers. In these cases, both classifiers have equivalent performance on four discrete benchmark datasets. 170
6.6 Comparison of the error rate learning curves for NB and ALNB classifiers. The ALNB classifier is inferior to the NB classifier and the differences between these two classifiers are statistically significant for five discrete benchmark datasets. 171
6.7 Comparison of the error rate learning curve for NB and ALNB classifiers. In these cases, the ALNB performs better than NB classifier and the differences between these two classified are statistically significant for 12 continuous benchmark datasets. 174
6.8 Comparison of the error rate learning curve for the NB and ALNB classifiers on `waveform-noise`. In these cases, the ALNB performs slightly better than NB classifier. 175
6.9 Comparison of the error rate learning curve for NB and ALNB classifiers. In these cases, both classifiers has equivalent performance on eight continuous benchmark datasets. 176

6.10 Comparison of the error rate learning curves for NB and ALNB classifiers. The NB better than ALNB classifier and the differences between these two classified are statistically significant for seven continuous benchmark datasets. 177

6.11 The average of the error rate learning curve for 1000 replications for both active (alNB) and passive (NB) learning for the liver-disorder benchmark dataset 180

6.12 Scatter plot between Liver-disorder benchmark datasets feature patterns. The seven removed patterns are shown inside circles. 181

6.13 The average of 1000 area under error rate learning curve for both active (alNB) and passive (NB) learning for the liver-disorder benchmark dataset after removal of outliers. 182

6.14 Critical difference diagram for NB, SSNB, ALNB, and ALSSNB over 36 discrete benchmark datasets. It shows that the ALNB classifiers are statistically superior to the other classifiers. 184

6.15 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. In these cases, the ALSSNB is statistically superior to the NB classifier for 10 discrete benchmark datasets. 186

6.16 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. The ALSSNB classifier have equivalent performance over 13 discrete benchmark datasets. 188

6.17 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. In these cases, the ALSSNB is inferior to the other classifiers for 12 discrete benchmark datasets. 190

6.18 Critical difference diagram for the NB, SSNB, ALNB, and ALSSNB over 28 continuous benchmark datasets. It shows that there are statistically significant differences between the mean ranks for the ALNB and both SSNB and ALSSNB classifiers and that ALNB classifier is superior. Additionally, there is no statistically significant difference between ALNB and NB classifier . . 192

6.19 Comparison of the error rate learning curve for NB, SSNB, ALNB, and ALSSNB classifiers. In these cases, the ALSSNB is superior to the NB and ALNB classifiers but have equivalent classification performance as SSNB for six continuous benchmark datasets 193

6.20 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. All classifiers have equivalent classification performance for the `Climate-Model-Simulation-Crashes` benchmark dataset. 194

6.21 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. In these cases, the ALSSNB is inferior to the other classifiers for the 21 continuous benchmark datasets 197

6.22 Critical difference diagram for the NB, SSNB, ALNB, and ALSSNB over 36 discrete benchmark datasets. It shows that the ALSSNB is statistically superior compared to the other classifier 199

6.23 Critical difference diagram for the NB, SSNB, ALNB, and ALSSNB over 28 continuous benchmark datasets. It shows that the ALSSNB classifier is statistically superior compared to the NB and ALNB classifiers. In addition, it shows that SSNB is rather better than ALSSNB but the difference between their mean ranks is not statistically significant 201

List of tables

2.1 Confusion matrix for a binary classification task, the actual class is given by the columns while the predicted class is given by the rows 77

3.1 Attributes of the UCI datasets with discrete input features 89

3.2 Attributes of the UCI datasets with continuous input features 90

3.3 AULC for the NB and SSNB over 36 discrete datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 92

3.4 AULC for the NB and SSNB over 28 continuous datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 93

3.5 AULC for the NB and SSNB over 36 synthetic discrete datasets from the UCI. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 96

3.6 AULC for the NB and SSNB on 28 synthetic continuous datasets from the UCI. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 97

4.1 AULC for the NB, SSNB and SSNB-λ classifiers over 28 continuous benchmark datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ classifier is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier. 108

4.2 AULC for the NB, SSNB, and SSNB-λ classifier over 36 discrete datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ classifiers is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier. 110

4.3 AULC for the NB, SSNB and SSNB-λ-cvs classifiers over 28 continuous benchmark datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ-cvs classifier is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier. 114

4.4 AULC for the NB, SSNB, and SSNB-λ-cvs classifier over 36 discrete datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ-cvs classifiers is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier. 115

4.5 AULC for the NB, SSNB and SSNB-λ-mcv classifiers over 28 continuous benchmark datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ-mcv classifier is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier. 124

4.6 AULC for the NB, SSNB, and SSNB-λ-mcv classifier over 36 discrete datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ-mcv classifiers is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier. 125

5.1 The twenty matches of a chess player used as a training data for naive Bayes classifier. . . 135

5.2 Detials of the datasets used in experiemnts 146

5.3 AULC of the NB, SSNB, and SSNB-sigmoid-fb classifier with the best value of α over 12 discrete and 16 continues benchmark datasets. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier between NB and SSNB-sigmoid-fb classifiers. The results that are statistically equivalent between NB and SSNB-sigmoid-fb classifiers (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifiers is better or equivalent to the AULC for the SSNB-sigmoid-fb classifier. 147

5.4 AULC of the NB, SSNB, and SSNB-sigmoid classifier with the best value of α over 12 discrete and 16 continues benchmark datasets. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier between NB and SSNB-sigmoid classifiers. The results that are statistically equivalent between NB and SSNB-sigmoid classifiers (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifiers is better or equivalent to the AULC for the SSNB-sigmoid classifier. 150

5.5 AULC of the NB, SSNB, and SSNB-sigmoid-LODO classifier with the best value of α over 12 discrete and 16 continues benchmark datasets. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier between NB and SSNB-sigmoid-LODO classifiers. The results that are statistically equivalent between NB and SSNB-sigmoid-LODO classifiers (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifiers is better or equivalent to the AULC for the SSNB-sigmoid-LODO classifier. 153

5.6 AULC of the NB, SSNB, and SSNB-sigmoid-cv classifier over 12 discrete and 16 continues benchmark datasets. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent between NB and SSNB-sigmoid-cv classifiers (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 155

6.1 AULC of the NB, and ALNB classifiers with the uncertainty sampling strategy over 36 discrete benchmark datasets. The boldface font indicates that the AULC for one of the classifiers is better than the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 165

6.2 AULC of the NB and ALNB classifiers with uncertainty sampling strategies over 28 continuous benchmark datasets. The boldface font indicates that the AULC for one of the classifiers between the NB and ALNB is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 172

6.3 AULC of the NB, and ALSSNB classifiers with uncertainty sampling strategy over 36 discrete benchmark datasets. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 184

6.4 AULC of the NB, and ALSSNB classifiers with uncertainty sampling strategies over 28 continuous benchmark datasets. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 191

6.5 AULC of the NB, and ALSSNB classifiers with uncertainty sampling strategies over 36 discrete synthetic benchmark datasets. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. 199

6.6 AULC of the NB, and ALSSNB classifiers with uncertainty sampling over 28 continuous synthetic benchmark datasets. The boldface font indicates that the AULC for one of the classifiers between the NB and ALNB is better than the other classifier. The result that statistically not different according to the Wilcoxon signed rank test at 0.95% confidence level are shown in italics 200

Chapter 1

Introduction

1.1 Introduction

The primary goal of machine learning is the development of algorithms for the extraction of general patterns from a finite training set. Generally, machine learning tasks can be divided into two common types, namely supervised and unsupervised learning. In unsupervised learning, the learner is given a collection of unlabelled training patterns, represented by input features alone, without an indication of the desired output. The goal of unsupervised learning is to determine the internal structure of the training set. A very common form of unsupervised learning is cluster analysis [36]. Clustering tries to group similar patterns within the training set, such that examples in the same cluster are similar to each other and different from patterns in the other clusters. The other major form of machine learning task is supervised learning. In supervised learning, the learner is given a collection of training patterns with the corresponding *target* labels, where the labels indicate the desired output. Having additional target labels in supervised learning is the key distinction between supervised and unsupervised learning. In general, supervised learning first attempts to use the training data to learn a mapping from the input features to the desired outputs. These training patterns are generally assumed to be an independent and identically distributed (iid.) sample

from some fixed underlying distribution. The goal of the learner is to predict the labels for additional new (test) patterns, based on the set of training patterns. Supervised learning tasks can be categorised into two types, based on the target labels, which are classification and regression. In classification problems, the labels consist of discrete values indicating the true class of the example, whereas in regression the output can take on continuous values. The work reported in this thesis focuses on supervised learning for classification tasks. Figure 1.1 shows the framework of the supervised learning process which is known as passive learning. In passive learning, we typically submit all training patterns to be labelled by a human, or we iteratively select training patterns at random to be labelled, and then construct a classifier.

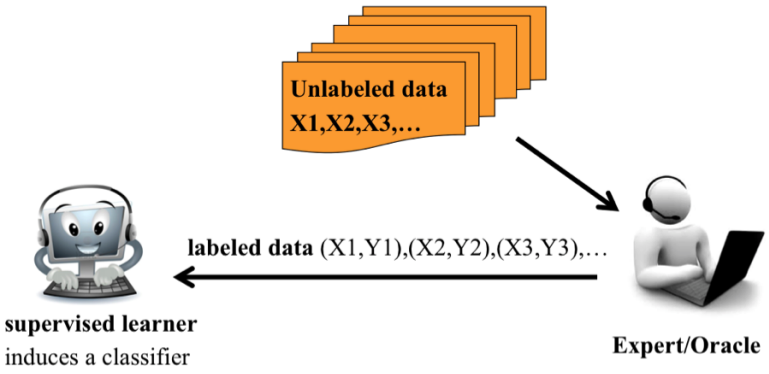


Fig. 1.1 Passive learning framework [72]

In practice, classification performance usually depends on the size of the labelled dataset. Hence, good classification performance relies on the availability of a substantial amount of labelled data, and poor classification performance is often obtained with a small amount of labelled training patterns. In real-world applications, manual labelling of a large amount of training data is often time-consuming and expensive. Therefore, building effective classification models with a minimum of labelled training patterns motivates us to improve classification algorithms, but an important question which arises here is what other sources of information can reduce the need for labelled patterns?

In many modern classification problems, large quantities of unlabelled patterns can be ob-

tained plentifully and cheaply. Consequently, combining this large quantity of unlabelled patterns together with a small amount of the labelled patterns in the learning process might construct better classification models and improve classification performance. To demonstrate how unlabelled examples might help and achieve better classification performance in more detail, we begin with an illustrative example. The example is based on the two moons benchmark dataset [76], which is a two-class classification problem with two labelled training patterns, only one labelled pattern in each class, and the rest are unlabelled.

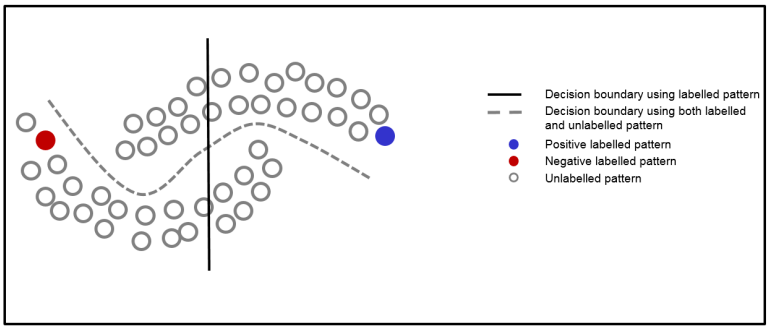


Fig. 1.2 Illustration of the machine learning example where unlabelled patterns can improve classification performance for the two moons synthetic benchmark dataset

Figure 1.2 shows a decision boundary (solid line) that is estimated by supervised learning, using one labelled training pattern in each class. However, a more reliable decision boundary can be obtained by combining both labelled and unlabelled training patterns (dashed line). In such situations, the two labelled training patterns give an indication of which class belongs to which cluster and then unlabelled data can help to identify the region. This an intuitive example of machine learning where unlabelled data can improve classification performance. However, traditional supervised learning methods cannot use unlabelled patterns to train models. In general, there are two common paradigms for exploiting unlabelled data.

The first paradigm is semi-supervised learning, lying between supervised and unsupervised learning as can be seen in Figure 1.3. Semi-supervised learning can potentially use a combination of a small set of labelled patterns with a large amount of unlabelled data in order to achieve better performance than supervised learning as shown in Figure 1.2. Thus, the

semi-supervised learning paradigm seems to address the drawbacks of supervised learning, which leads to reduced human effort, time and cost of manual labelling especially where labels cannot be generated automatically.

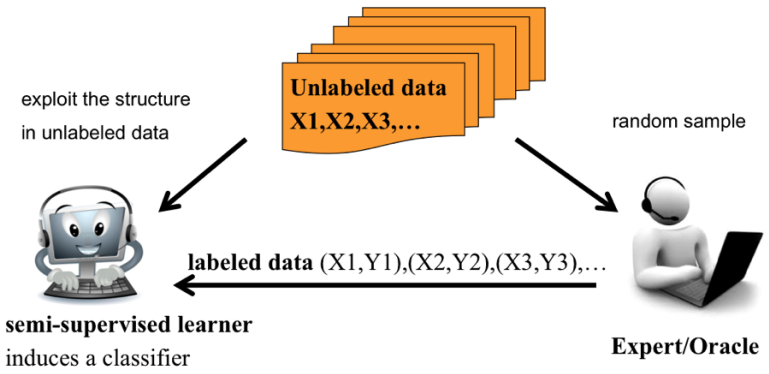


Fig. 1.3 Semi-supervised learning framework [72]

The second paradigm is active learning. In contrast to passive learning, the active learner requests the human expert (oracle) to obtain the true labels for unlabelled training patterns by asking queries. The active learner selects only a small number of unlabelled patterns to be labelled by the oracle, instead of labelling all unlabelled training patterns as in the case of passive learning. Therefore, the patterns selected are those the algorithm considers most likely to improve the performance of the classifier if the labels were known. Figure 1.4 shows the framework of active learning. A feature of active learning is that the active learner iteratively selects one or more unlabelled patterns and adds them to the previous labelled set, it then uses them to retrain the classifier. There are different scenarios and strategies to select unlabelled training patterns that we discuss in more detail in Chapter 2. For example in the case of pool-based scenario with an uncertainty sampling strategy the active learner selects a pattern closest to the decision boundary, which is considered the most informative one. Under these circumstances, active learning might achieve better performance compared to passive learning by labelling fewer unlabelled training patterns. In such cases, the oracle might not spend effort on labelling uninformative patterns, which do not influence the decision

boundary. Thus, active learning can significantly reduce the human effort and labelling cost in the process of manually labelling [72].

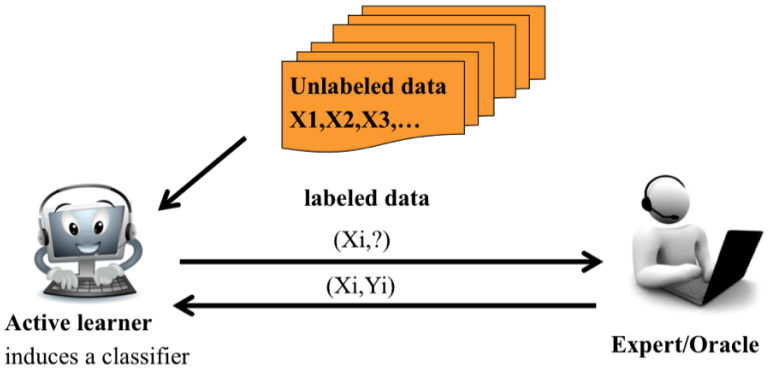
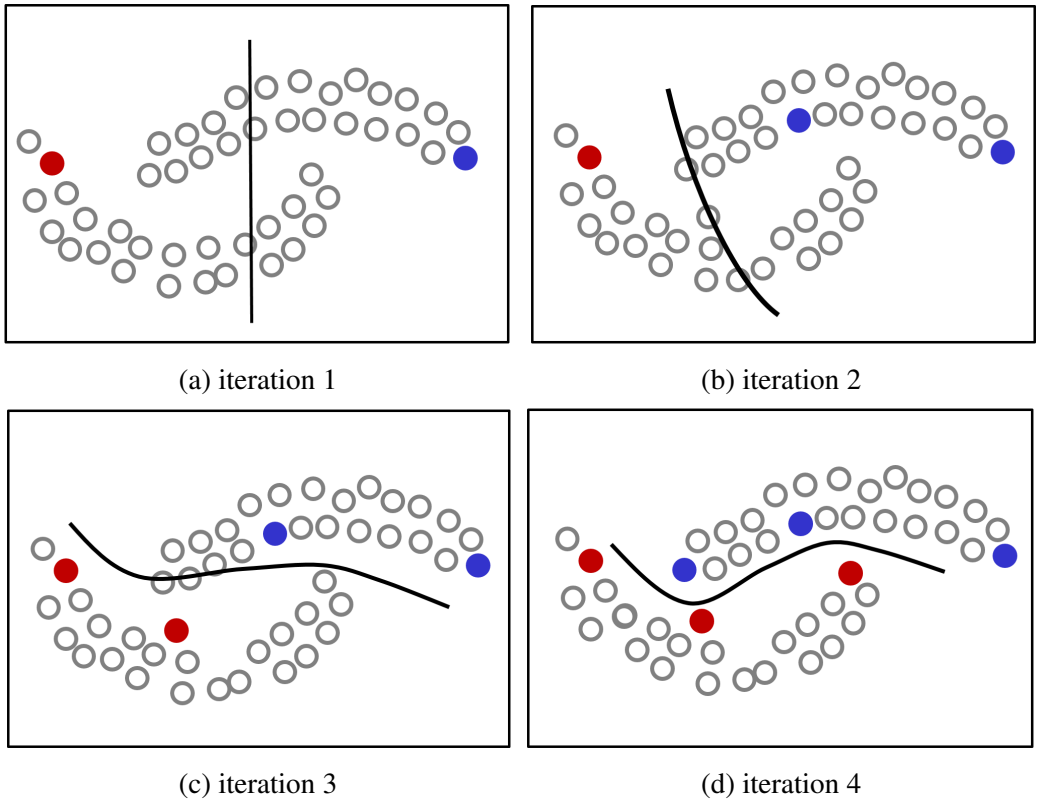


Fig. 1.4 Active learning framework [72]

Figure 1.5 shows how active learning can iteratively select unlabelled patterns to be labelled for the previous example, two moons benchmark dataset. As you can see after it labelled a few patterns the active learner can estimate a reliable decision boundary.

1.2 Motivation

In the previous section, we demonstrate that active learning algorithms are a potential paradigm for exploiting unlabelled data. However, when active learning is used in the early stages the initial model is often poor, and so can provide a poor direction for the selection of further unlabelled training data to be labelled by the oracle. The reason for generally providing a poor initial model in the early stages is that the initial model parameters are estimated from only a few labelled training patterns. Active learning is then likely to collect labels for uninformative examples in the training data which will do little to improve performance. As a result, it may be better to perform some passive learning first where labels are sampled randomly, as this is more likely to generate useful labels than may be



(a) iteration 1 (b) iteration 2

(c) iteration 3 (d) iteration 4

Fig. 1.5 Illustrative example to show the active learning process to select informative unlabelled patterns using uncertainty sampling strategy for the two-moon synthetic dataset [76]

obtained by using a poor initial model. This process involves exploration-exploitation trade-off. Exploitation builds a classifier using the patterns that were labelled by the oracle in the active learning process to direct the selection of the patterns that remain in the unlabelled patterns pool. However, exploration can pick patterns from the wider feature space, as it is not only focusing on examples closest to the classification boundary. Exploitation is a risky choice, especially when the poor initial model provides poor direction for the search in the early stages. In addition, exploitation can easily present uninformative patterns to be labelled by the oracle. Semi-supervised learning may help because a better initial model may be achieved where both labelled and unlabelled data are used, and active learning may then prove beneficial immediately. There is also a secondary issue that it may be true that with very few labels it is better to use unsupervised learning rather than semi-supervised

learning. In practice, this requires managing a transition between unsupervised learning through semi-supervised learning, to fully supervised learning according to the amount of labelled data available. The aim of the research described in this thesis is to develop algorithms to reliably control the transition from unsupervised, semi-supervised to supervised learning in active learning.

This thesis focuses on the naïve Bayes classifier as a baseline for semi-supervised and active learning. Although the naïve Bayes classifier is a simple algorithm it is still a useful and effective algorithm in many real-world applications. For example, [58] show a successful application of semi-supervised naïve Bayes for multi-class text classification. Mathematically the naïve Bayes classifier is also simple to analyse. More importantly, understanding the estimation of model parameters in the case of the partially labelled dataset using Expectation Maximisation (EM) algorithm Dempster et al. [24] is relatively straightforward. It would be better to understand a simple model rather than try to understand a more complicated classifier.

To illustrate the nature of the difficulties involved in active learning, a simple two-dimensional synthetic dataset is generated that has been shown in Figure 1.6. The data are drawn from two spherical bivariate Gaussian distributions where each Gaussian represents a different class, with equal numbers of patterns, 2200 in each class. The data are randomly partitioned into training and test sets, 400 patterns were used for training a classifier and 4000 were separated as a test set to evaluate the classification performance during the experiments. The positive class examples are drawn from a Gaussian centred on [1, 3]; the negative examples are drawn from a Gaussian centred on [-1, 1] where both Gaussian have identical variance, σ^2 =1.

In Figure 1.6a the green line shows the decision boundary for the problem after the model is trained on all training patterns and the red line shows the decision boundary for the problem after it is trained on two labelled patterns from each class (red and blue) with the rest of the

synthetic benchmark dataset unlabelled. Figure 1.6b shows the most informative pattern selected, in this case, the most informative pattern is the closest pattern to the decision boundary. Notice that after labelling one pattern using active learning, the generalisation performance is better than passive learning Figure 1.6c.

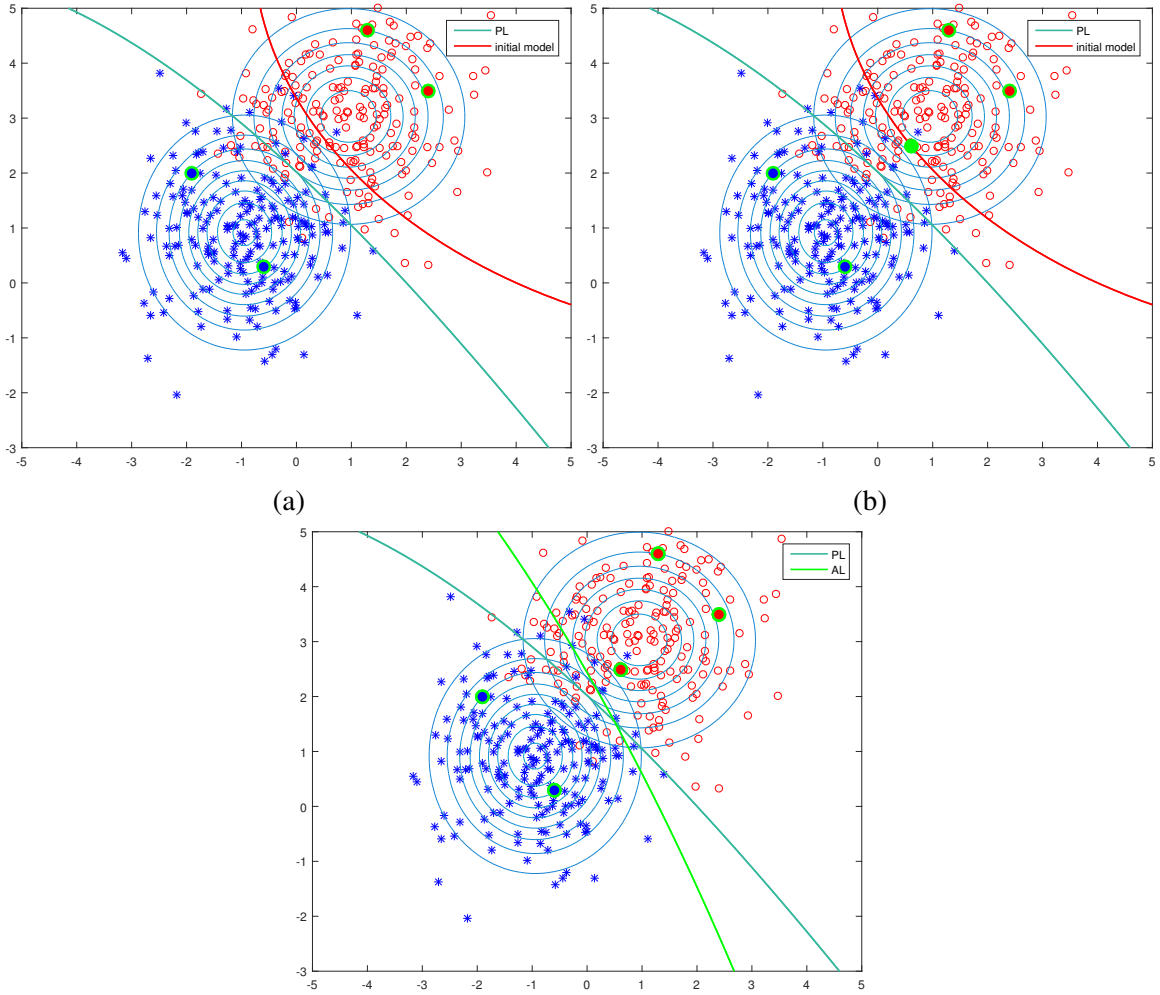


Fig. 1.6 (a) The blue line represents the decision boundary for the naïve Bayes model trained on all training patterns and the red line represents the decision boundary for the naïve Bayes model trained with four good labelled patterns, two in each class. Illustration of picking the most informative patterns (b). (c) The decision boundary for active learning after being trained with one actively queried example is represented by a green line.

Figure 1.7 shows the same synthetic benchmarks, but with a bad initial model. In contrast, the active learning generalisation performance is worse compared to the passive learning.

The reason for this, is that generally we selected uninformative patterns to be labelled using the initial model.

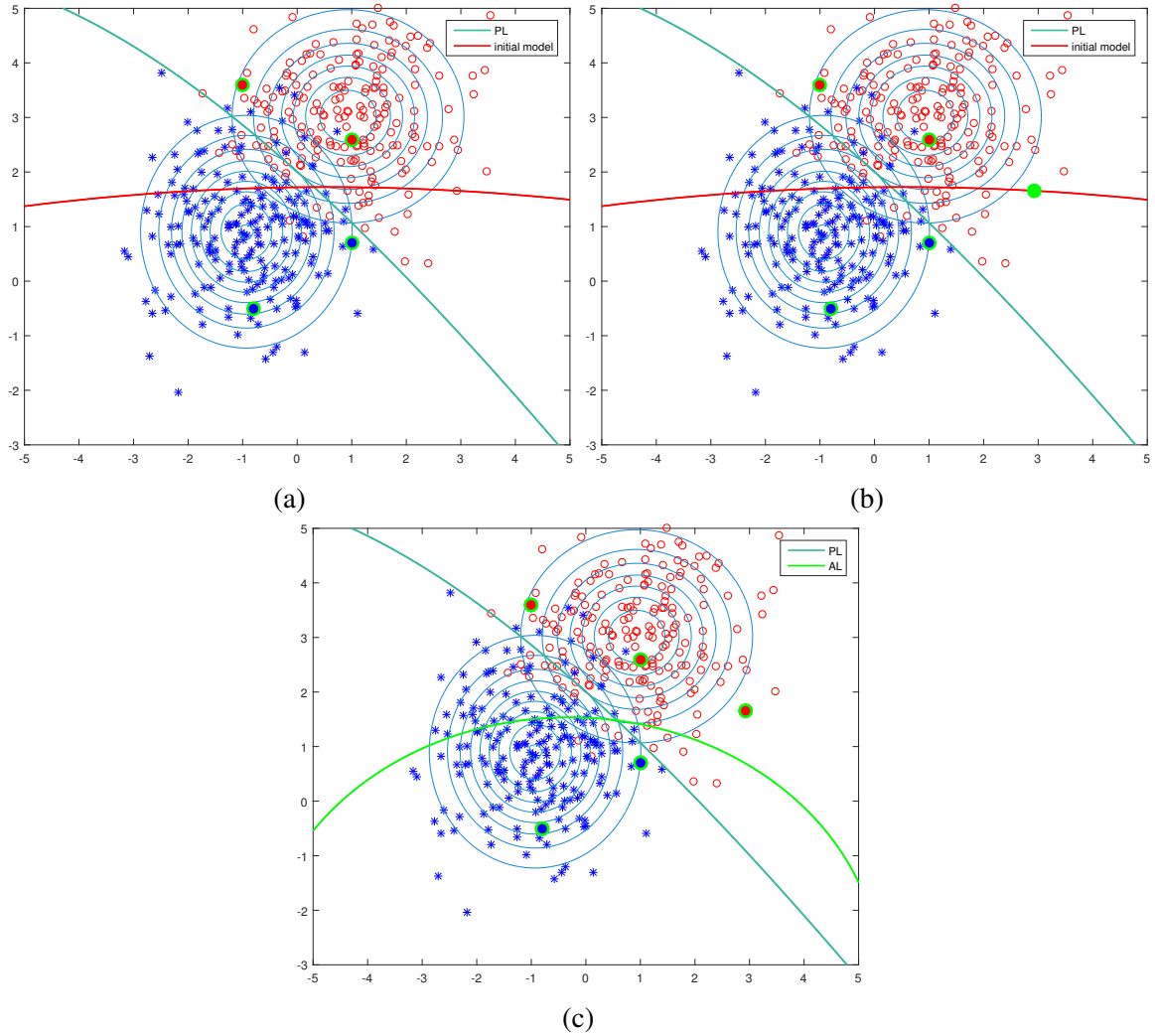


Fig. 1.7 (a) The blue line represents the decision boundary for the naïve Bayes model trained on all training patterns and the red line represents the decision boundary for the naïve Bayes model trained with four poor labelled patterns, two in each class. (b) Illustrates picking the most informative patterns. (c) The decision boundary for active learning after being trained with one actively queried example is represented by a green line.

To illustrate the classification performance for these classifiers, we show the area under error rate learning curve. The error rate of the classifier performance measures the percentage of the predicted class that differs from the actual class, and the area under the error rate learning curve is a curve that shows the error rate with an increasing number of labelled

patterns. We take the average of 100 iterations to generate the area under the error rate learning curve of the synthetic benchmark. This experiment consisted of 100 trials to generate synthetic benchmark with good and bad initial models, and random partitioning into training and test sets. 400 patterns were used for training and 4000 patterns were held-out as a test set, used to evaluate the classification error rate performance during the experiments. Figure 1.8 illustrates that the error rate learning curve of active learning with a good initial model is much lower than the error rate learning curve of active learning with a bad initial model.

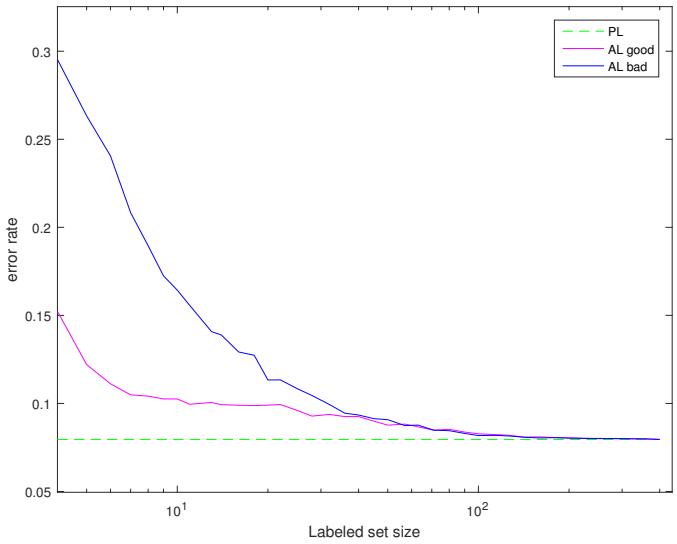


Fig. 1.8 Comparison between active learning with a good initial model, pink-line, and active learning with a bad initial model, blue-line. The passive learning plotted by green dash-line.

From the above example, it is clearly seen that the generalisation performance of active learning from a good initial model can be better than active learning from a bad initial model. However, if we use semi-supervised learning in the case of the bad initial model, the two labelled patterns in each class might identify their cluster and then start the best solution immediately as shown in Figure 1.9.

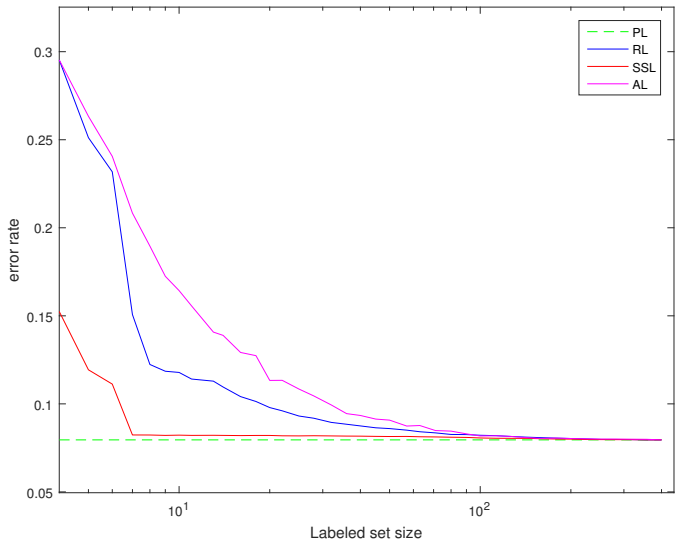


Fig. 1.9 Comparison between the learning curves for random (RL: blue-line), semi-supervised (SSL: red-line), passive (PL: green dash-line), and active learning with bad initial model (AL: pink-line)

1.3 Outline of the thesis

The remainder of this thesis is organised as follows:

Chapter 2 we first provide a technical background on active and semi-supervised learning, and their common applications in general, focusing on naïve Bayes methods in particular as a base classifier. The details of the derivations of the Expectation-Maximization (EM) algorithm for semi-supervised learning are provided in this chapter. Then, we look at the implementation of different model selection criteria, model selection being the general approach for finding optimal parameter values where the model is defined by a set of model parameters and hyper-parameters. Finally, different measures to evaluate the performance of the classifier are presented, highlighting the importance of using statistical tests to compare classifiers.

Chapter 3 provides an experimental evaluation of naïve Bayes and semi-supervised naïve Bayes classifier on benchmark datasets in order to investigate whether using unlabelled

patterns through semi-supervised learning improves supervised classification performance overall. We began by introducing the benchmark datasets that are used throughout this thesis. Additionally, we set out the experimental design, including evaluation measures and statistical significance tests.

Chapter 4 discusses down-weighting approaches for semi-supervised learning to control the influence of unlabelled patterns relative to labelled patterns because the training data consist of a large amount of unlabelled patterns with few labelled patterns.

Chapter 5 discusses the over-confidence of the naïve Bayes classifier, which can produce a bias in the predicted class probabilities in both the naïve Bayes classifier and the EM based semi-supervised naïve Bayes classifier. Then, the design, implementation, and evaluation of a logistic transformation are presented to correct the un-calibrated probability estimates in the E-step for EM-based semi-supervised learning. Finally, the benefit of a logistic transformation for the semi-supervised naïve Bayes classifier is shown, then is compared to supervised naïve Bayes classifier.

Chapter 6 first presents results for active learning used to reduce the number of unlabelled patterns presented to the oracle for a naïve Bayes classifier. Visualisation techniques are then used to demonstrate the exploration guided active learning selection strategy. Finally, semi-supervised and active learning are combined to improve the performance of the naïve Bayes classifier as a baseline classifier in the case of small amount of labelled data and large amount of unlabelled data.

Chapter 7 presents conclusions about the implications of the experimental results and suggestions for possible future directions.

1.4 Contributions of the thesis

Benchmarking semi-supervised naïve Bayes: An extensive study was performed using 36 discrete and 20 continuous benchmark datasets to evaluate whether using unlabelled

data through semi-supervised learning can increase the performance of the naïve Bayes classifier. Previously, Nigam et al. [58] showed that using unlabelled patterns can improve the naïve Bayes classifier for the 20 newsgroup benchmark dataset. Self-training is a common technique for semi-supervised learning which iteratively predicts labels for unlabelled patterns. We investigate whether a self-training scheme, that uses the expectation maximization algorithm, improves the naïve Bayes classifier. Initially, this investigation focused on a comparative study between standard naïve Bayes and semi-supervised naïve Bayes with numerous benchmark datasets. After performing this study we found that, surprisingly, using unlabelled data generally makes the classifier worse. To determine the cause of this, we investigate further experiments with categorical and continuous features sets. We found that the violation of the independence assumption of naïve Bayes may degrade classification performance. There had been no large-scale empirical evaluation of this issue already in the literature.

List of Publications:

Awat A Saeed, Gavin C Cawley, and Anthony Bagnall. Benchmarking the semi-supervised naïve Bayes classifier. In *Neural Networks (IJCNN), 2015 International Joint Conference on,* pages 1–8. IEEE, 2015.

Chapter 2

Technical background and Literature Review

In this chapter we will introduce some background material that is used throughout the thesis to ensure that experiments are unambiguously described by precisely defining the methods used. In general, we introduce the ideas behind two fields of machine learning that reduce the amount of labelled patterns required to learn a classifier. These areas are: active [72] and semi-supervised [94] learning. We start by introducing semi-supervised learning, followed by the description of simple supervised learning techniques for the Naïve Bayes classifier. We then provide a brief overview of common approaches and query strategies for active learning. Subsequently, we present the different performance criteria used to estimate the generalisation performance of classifier and test whether there is a statistically significant difference between estimates. Finally, we briefly describe model selection methods used to choose good hyper-parameter values, which is difficult where there are few labelled examples.

2.1 Supervised learning

In a supervised classification setting, we are given labelled training data, $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{l}$, where $x^{(i)} \in \mathscr{X} \subseteq \mathscr{R}^d$ is a feature vector describing the i^{th} example, with corresponding class label $y^{(i)} \in \{1, 2, \ldots, C\}$. The set of training examples, D, that is assumed to be an independent, identically distributed (iid) sample drawn from a fixed distribution, is used to obtain an estimate of the model parameters, denoted by $\hat{\theta}$. The model parameters of the naïve Bayes (NB) classifier are the class probabilities $p(y^i|\theta)$, which are also called prior probabilities, and class conditional probabilities for input features given corresponding class label $p(x^i|y^i;\theta)$:

$$p(x^i|y^i;\theta) \;=\; p(x_1^{(i)}, x_2^{(i)}, ..., x_d^{(i)}|y^{(i)};\theta),$$

where $x_j^{(i)}$ refers to the value of feature j^{th} for example i and $y^{(i)}$ is the class label for the same example.

To reduce the complexity of estimating the model parameters, the NB classifier makes the assumption that the input features are conditionally independent of each other given the class labels of the example,

$$\begin{aligned} p(x^i|y^i;\theta) \;&=\; p(x_1^{(i)}|y^{(i)};\theta)\,.\,p(x_2^{(i)}|y^{(i)};\theta)\,...\,p(x_d^{(i)}|y^{(i)};\theta), \\ &=\; \prod_{j=1}^{d} p(x_j^{(i)}|y;\theta). \end{aligned}$$

For example, if the input feature vectors are comprised of d binary attributes and the class labels have binary value. Then, $2 * (2^d - 1)$ parameters must be estimated for learning Bayesian classifier. This might be impossible when the number of features becomes large or given a limited amount of the training examples. In this case the NB model assumption reduces the amount of estimated parameter to $(2d)$.

The independence assumption is the strong assumptions, because it is valid if the occurrence of features are not dependent on each other which is not true for many states. Therefore, it

is often unrealistic in the real world, but it simplifies the estimation of $p(x|y;\theta)$ from the training samples. Therefore, it is particularly suitable when the dimensionality of the input features is so high that a large number of parameters must be estimated [45].

One common way to estimate the model parameters, $\hat{\theta}$, is via the maximum likelihood estimate (MLE):

$$\hat{\theta} \;=\; \arg\max_{\theta}\; p(\mathscr{D};\theta).$$

As we assumed the training data comprise an iid sample, the likelihood given by

$$\begin{aligned} p(\mathscr{D};\theta) &= \prod_{i=1}^{l} p\left(x_j^{(i)}, y^{(i)};\theta\right), \\ &= \prod_{i=1}^{l}\left(p\left(y^{(i)};\theta\right)\prod_{j=1}^{d} p\left(x_j^{(i)}|y^{(i)};\theta\right)\right). \end{aligned} \tag{2.1}$$

The likelihood function is a product of the conditional probability given class labels, $p(x_j^i|y^i;\theta)$. In a practical application, the likelihood function (2.1) can be used with low dimension of the input features to estimate model parameters. In contrast, when the dimension of the input features vector is high, the product of many of the conditional probability given class labels may be underflow and tends to zero. In this case, the product of conditional probability given class is a very small to represent in a floating point number and may produce an infinite or NaN result [35]. To address this, taking the *log* of the likelihood function and using the maximum log-likelihood instead of maximum likelihood to estimate the model parameters is a common solution,

$$\begin{aligned} \log\, p(\mathscr{D};\theta) &= \sum_{i=1}^{l}\log\left(p(y^{(i)};\theta)\prod_{j=1}^{d} p(x_j^{(i)}|y^{(i)};\theta)\right), \\ &= \sum_{i=1}^{l}\log\, p(y^{(i)};\theta) + \sum_{i=1}^{l}\sum_{j=1}^{d}\log\, p(x_j^{(i)}|y^{(i)};\theta). \end{aligned} \tag{2.2}$$

Finally, the derivative of (2.2) wrt $\hat{\theta}$ can be set to zero to estimate the model parameters.

The task of the naïve Bayes (NB) classifier is the prediction of class labels (y) for a new

pattern (x) by modelling the class conditional probability $p(x|y;\theta)$ and the prior probability $p(y;\theta)$, where θ are the model parameters, and then using Bayes' rule to estimate the posterior probability of class membership for all classes, $p(y|x;\theta)$, after parameter estimation [46, 38].

$$p(y=c|x;\theta) \;=\; \frac{p(x|y;\theta)\; p(y;\theta)}{\sum_{k=1}^{C} p(x|y=k;\theta)\; p(y=k;\theta)}. \tag{2.3}$$

The summation in the denominator is over all class labels k and it is a constant that is used to normalise the nominator term to one. The test pattern (x) is classified belonging to a single class by selecting the maximum posterior probability of class membership according to the maximum a posterior (MAP) classification rule,

$$\hat{y} \;=\; \arg\max_{c}\; p(y=c|x;\theta). \tag{2.4}$$

2.1.1 Maximum likelihood estimation for categorical features

The categorical distribution is the appropriate discrete distribution for handling nominal data. Suppose the features come from a categorical distribution where the j^{th} feature, x_j, has (S_j) possible values (states), $x_j \in \{1,2,\ldots,S_j\}$. The j^{th} feature of the i^{th} example indicates one of the (S_j) values, i.e. $x_j^{(i)} = s$. The likelihood of observing a state $x_j^{(i)} = s$ is denoted by $\theta_{sc}^{j} = p(x_j^{(i)} = s|y^{(i)} = c)$ which is the probability of feature value $x_j^{(i)} = s$ in class c where $\sum_{s=1}^{S} \theta_{sc}^{j} = 1$ and $\pi_c = p(y=c)$ is the class prior probability for class c where $\sum_{c=1}^{C} \pi_c = 1$. If $x_j \sim cat(\theta)$, x_j has a categorical distribution which is a discrete probability distribution, then (2.2) can be written more explicitly in terms of the parameters.

$$\begin{aligned}
\log p(\mathscr{D};\pi,\theta) &= \sum_{i=1}^{l}\sum_{c=1}^{C}\phi(y^{(i)}=c)\log\pi_c \\
&\quad + \sum_{i=1}^{l}\sum_{j=1}^{d}\sum_{s=1}^{S}\sum_{c=1}^{C}\log cat(x_j^{(i)}|y^{(i)};\theta_{sc}^{j}) \,, \\
&= \sum_{i=1}^{l}\sum_{c=1}^{C}\phi(y^{(i)}=c)\log\pi_c \\
&\quad + \sum_{i=1}^{l}\sum_{j=1}^{d}\sum_{s=1}^{S}\sum_{c=1}^{C}\phi(x_j^{(i)}=s\wedge y^{(i)}=c)\log\theta_{sc}^{j} \,, \qquad (2.5)
\end{aligned}$$

where

$$\phi(z)=\begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \textit{otherwise} \,. \end{cases}$$

The log-likelihood can be maximised with respect to the parameters (θ_{sc}^{j},π_c) using the method of Lagrange multipliers (α,β_c^{j}) to enforce the constraint that the class prior and class-conditional probabilities must sum to one [42]. The log-likelihood with Lagrangian terms is given as follows:

$$\begin{aligned}
\Lambda(\pi,\theta,\alpha,\beta) &= \sum_{i=1}^{l}\sum_{c=1}^{C}\phi(y^{(i)}=c)\log\pi_c \\
&\quad + \sum_{i=1}^{l}\sum_{j=1}^{d}\sum_{s=1}^{S}\sum_{c=1}^{C}\phi(x_j^{(i)}=s\wedge y^{(i)}=c)\log\theta_{sc}^{j} \\
&\quad - \alpha\left(\sum_{c=1}^{C}\pi_c-1\right)-\sum_{c=1}^{C}\sum_{j=1}^{d}\beta_c^{j}\left(\sum_{s=1}^{S}\theta_{sc}^{j}-1\right). \qquad (2.6)
\end{aligned}$$

In order to obtain the maximum likelihood solution for the parameters, the partial derivatives can be computed for (2.6) with respect to all the parameters, and each partial derivative set to

zero:

$$\begin{aligned}
\frac{\partial \Lambda}{\partial \alpha} &= 0 \Rightarrow \sum_{c=1}^{C} \pi_c = 1, \\
\frac{\partial \Lambda}{\partial \beta_c^j} &= 0 \Rightarrow \sum_{s=1}^{S} \theta_{sc}^j = 1, \\
\frac{\partial \Lambda}{\partial \pi_c} &= 0 \Rightarrow \pi_c = \frac{\sum_{i=1}^{l} \phi(y^{(i)} = c)}{\sum_{k=1}^{C} \sum_{i=1}^{l} \phi(y^{(i)} = k)}, \\
\frac{\partial \Lambda}{\partial \theta_{sc}^j} &= 0 \Rightarrow \theta_{sc}^j = \frac{\sum_{i=1}^{l} \phi(x_j^{(i)} = s \wedge y^{(i)} = c)}{\sum_{m=1}^{S} \sum_{i=1}^{l} \phi(x_j^{(i)} = m \wedge y^{(i)} = c)}.
\end{aligned} \tag{2.7}$$

In some cases the probability estimation suffers from zero probability values when there are not enough training samples. If a zero conditional probability or zero prior probability estimate is used while predicting the class labels for the test data, the whole product becomes zero. Therefore, a small-sample correction can be added into all probabilities to prevent zero probability values. This technique is known as the Laplace correction [86].

$$\begin{aligned}
\pi_c &= \frac{\sum_{i=1}^{l} \phi(y^{(i)} = c) \; + \; 1}{\sum_{k=1}^{C} \sum_{i=1}^{l} \phi(y^{(i)} = k) \; + \; C}, \\
\theta_{sc}^j &= \frac{\sum_{i=1}^{l} \phi(x_j^{(i)} = s \wedge y^{(i)} = c) \; + \; 1}{\sum_{m=1}^{S} \sum_{i=1}^{l} \phi(x_j^{(i)} = m \wedge y^{(i)} = c) \; + \; S_j}.
\end{aligned} \tag{2.8}$$

where π_c is the class prior probability, and $\sum_{i=1}^{l} \phi(y^{(i)} = c)$ is number of patterns in class (c). θ_{sc}^j is a class-conditional probabilities, and $\sum_{i=1}^{l} \phi(x_j^{(i)} = s \wedge y^{(i)} = c)$ is number of (s) value for the feature x_j in class (c).

2.1.2 Maximum likelihood estimation for continuous features

Suppose x_j is the j^{th} numeric input feature drawn from a Gaussian distribution conditioned on class labels, $x_j \sim \mathcal{N}(\mu_c, \sigma_c^2)$, with unknown model parameters (mean μ and variance σ^2). The difference between the Gaussian log-likelihood and the categorical log-likelihood

distribution lies in the $p(x_j^{(i)} \mid y^{(i)}; \theta)$, because the estimation of the class prior for all distributions is the same. Then the log-likelihood (2.2) without class prior probability for Gaussian distribution can be written as follows.

$$\begin{aligned} \log p(\mathscr{D}; \mu, \sigma^2) &= \sum_{i=1}^{l} \sum_{c=1}^{C} \sum_{j=1}^{d} \log \mathscr{N}\left(x_j^{(i)} | y^{(i)}; \mu_{jc}, \sigma_{jc}^2\right), \\ &= \sum_{i=1}^{l} \sum_{c=1}^{C} \sum_{j=1}^{d} \log \left(\frac{1}{(2\pi)^{\frac{1}{2}} |\sigma_{jc}^2|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \frac{(x_j^{(i)} - \mu_{jc})^2}{(\sigma_{jc}^2)} \right) \right). \end{aligned} \tag{2.9}$$

To obtain the maximum likelihood estimate in closed form, the partial derivatives can be computed for (2.9), with respect to all the parameters $(\mu_{jc}, \sigma_{jc}^2)$, and then each partial derivative is set to zero:

$$\begin{aligned} \frac{\partial \Lambda}{\partial \mu_{jc}} &= 0 \Rightarrow \mu_{jc} = \frac{\sum_{i=1}^{l} x_j^{(i)}}{l_c}, \\ \frac{\partial \Lambda}{\partial \sigma_{jc}^2} &= 0 \Rightarrow \sigma_{jc}^2 = \frac{\sum_{i=1}^{l} (x_j^{(i)} - \mu_{jc})^2}{l_c}, \end{aligned} \tag{2.10}$$

where $l_c = \sum_{i=1}^{l} \phi(y^{(i)} = c)$ is number of patterns in class (c).

2.2 Semi-Supervised learning

The goal of semi-supervised learning is to improve the performance of supervised learning methods by utilising unlabelled data. The labelled data can be used to learn a mapping from the input features to the desired outputs in supervised learning. However, supervised learning cannot learn a mapping from unlabelled data because unlabelled data does not provide corresponding target labels. Normally, unlabelled data provide knowledge about the unconditional distribution of the input features, x, denoted by $p(x)$, i.e. if we have an infinite amount of the unlabelled data then the density of the unlabelled data can be known, which is the data distribution, $p(x)$, and should be beneficial for the inference of the posterior

probability of class membership, $p(y|x)$. If two points x_i and x_j in a high density region are close, then so should be the corresponding outputs y_i and y_j.

The essential assumption of semi-supervised learning is that the distribution of the features of labelled and unlabelled data are relevant for the classification problem. If this assumption about the distribution holds, then the unlabelled data can improve the performance of supervised learning. Thus, learning from unlabelled data is achieved by matching the assumptions in the classifier with the actual class structure. If this assumption does not hold the unlabelled data will not improve the classification performance or may even degrade classification performance.

Beside this fundamental assumption, semi-supervised algorithms typically also require three other assumptions in order to use the unlabelled data effectively and improve classification performance. Therefore, semi-supervised algorithms use one or more of the following assumptions. In addition, these three fundamental assumptions are generally grouped under the main assumption known as the semi-supervised smoothness assumption [15]:

Cluster Assumption: the cluster assumption states that if the patterns are located in the same cluster, they are likely to have similar labels, and hence the value of the predicted class probabilities for unlabelled data from the same cluster should be close. In general, a cluster is a set of patterns inside a group such that each two similar (close) patterns, in terms of similarity measure such as Euclidean distance measure, passes through a high density region of patterns. The semi-supervised classifiers used in this thesis are based on the cluster assumption.

In order to show how the cluster assumption can be beneficial, we illustrate the simple Gaussian Mixture Models in Figure 2.1 where it appears that there are two clusters of data. First, a small amount of labelled data are used to estimate a model parameters such that each component contains three patterns as shown in Figure 2.1a. As can be seen, the small amount of patterns is confusing the classifier because they are far from the centre of the

components. Then, we add the large amount of unlabelled data for both components, as shown in Figure 2.1b. The unlabelled data can adjust both Gaussian components because, with the unlabelled data we can optimise model parameters such that the means of each Gaussian are equivalent to the centre of the unlabelled data, shown in Figure 2.1c. Therefore, the decision boundary is drawn in the middle to separate the two classes.

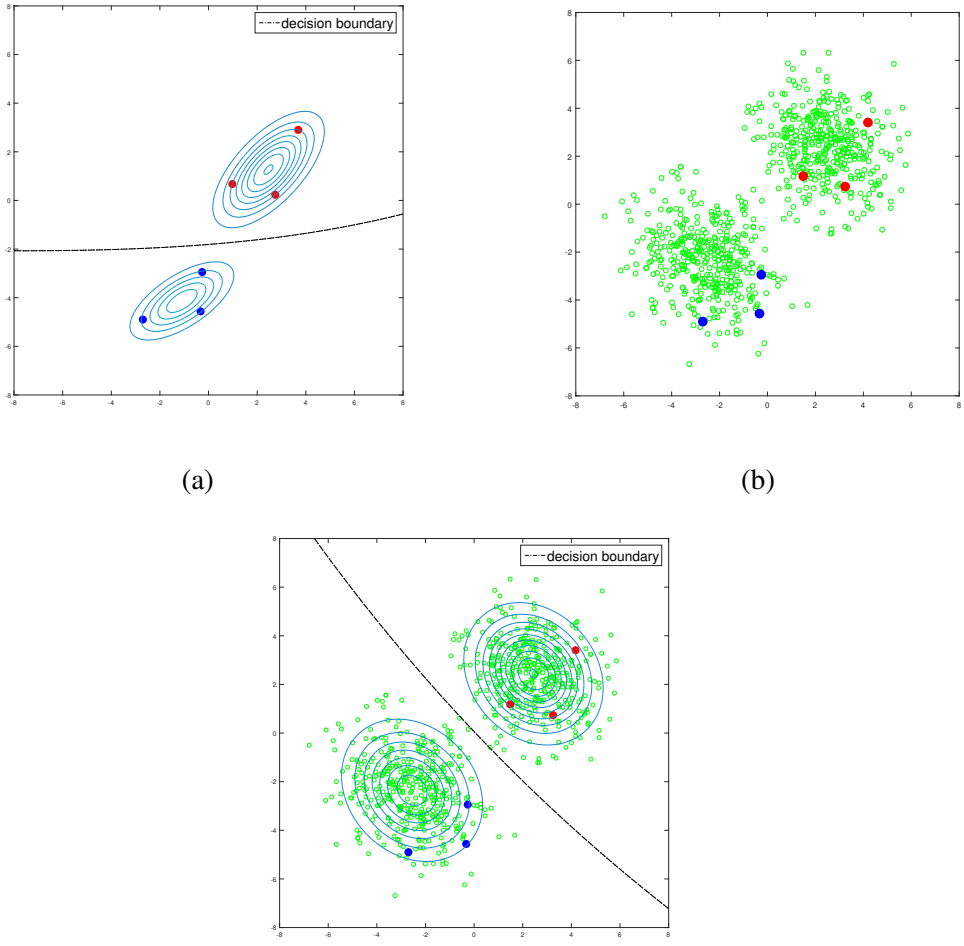
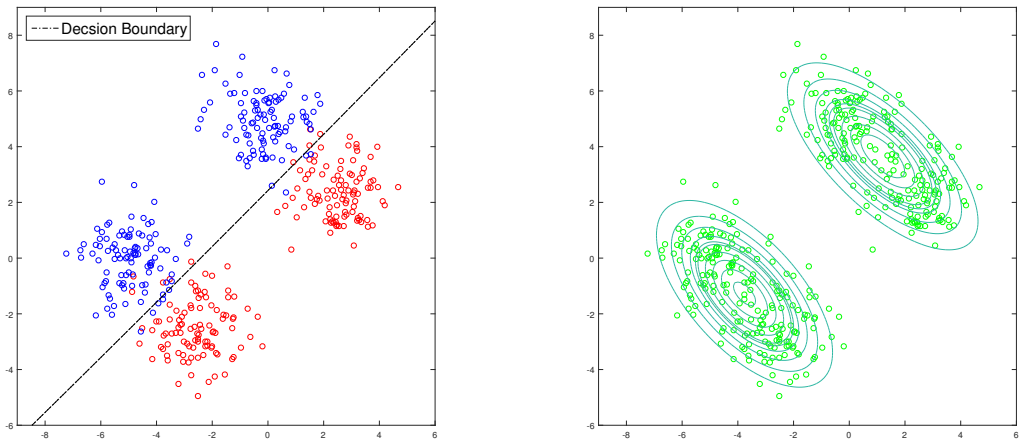


Fig. 2.1 (a) Gaussian Mixture Models using only a small amount of labelled data (b) labelled and unlabelled data for both components (c) Gaussian Mixture Models using labelled and labelled data that impact the decision boundary Zhu et al. [94]

This clear example shows that the unlabelled data give information to the classifier about the cluster structures of the data and then labels the patterns inside each group as the same

class according to the cluster assumption. Therefore, the unlabelled data can improve the classification performance.

In contrast, if the assumption does not hold the unlabelled data will not improve classification performance or using unlabelled data might degrade the classification performance. For example, Figure 2.2 shows a binary classification problem for Gaussian synthetic datasets such that each Gaussian consists of two clusters of data. This dataset is not generated from two Gaussian in reality, therefore, the decision boundary is a horizontal dashed line, as can be seen in Figure 2.2a. If we assume each cluster represented one of the class then the model gets good fit, but the assumption is wrong, as can be shown from Figure 2.2b.

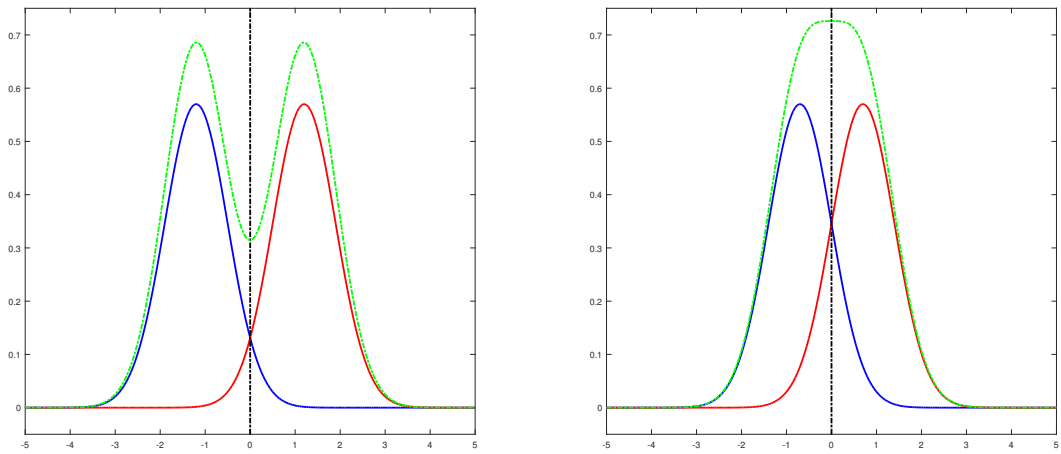


(a) (b)

Fig. 2.2 (a) Synthetic dataset with binary classes in four clusters (b) the model has a good fit, but the assumptions of the model are wrong [93]

Low Density Separation Assumption: this assumption states that the class decision boundary should pass through a low density region of the input space. Both the cluster and low density separation assumption are closely related to each other. Figure 2.3 illustrates this relation. The cluster assumption is an equivalent assumption to the low density separation assumption if the class separation boundaries should lie in a low density region as shown in Figure 2.3a. In this case, clearly if one assumption is holding for a semi-supervised classifier

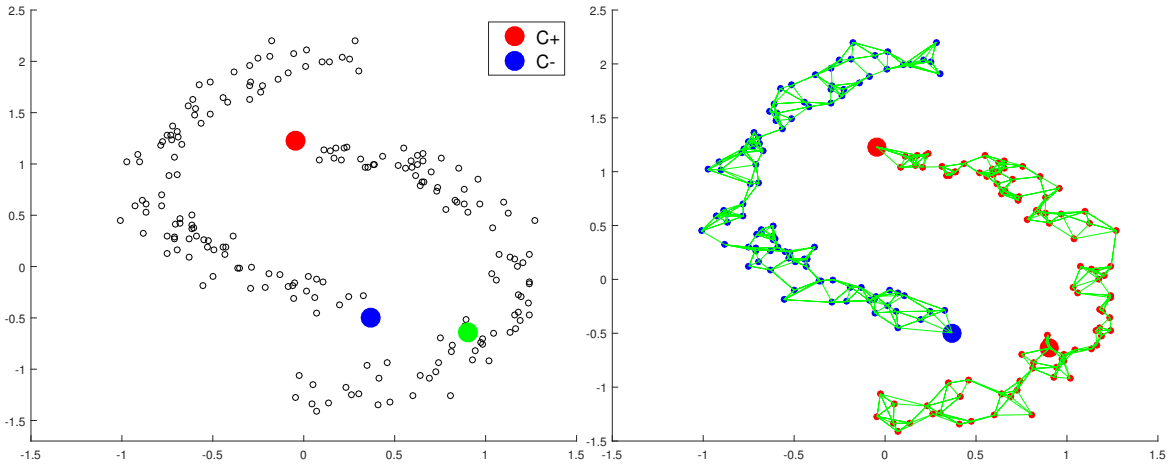
then the other assumption will also hold. However, the decision boundary in Figure 2.3b passes through a high-density region in which the low density separation assumption does not hold but the cluster assumption does still hold.



(a) (b)

Fig. 2.3 (a) The decision boundary lies in a low density region. (b) The decision boundary lies in a high density region

Manifold assumption: This assumption states that high-dimensional data lie on a low-dimensional manifold in the feature space. Using the manifold (low dimension) representation instead of the original structure of the data (high dimension) is useful to reduce the number of dimensions and may improve system performance. Therefore, this assumption is beneficial for a dimensionality reduction. However, if the manifold is the same as the high-dimensional region regarding dimensionality, then the manifold assumption is equivalent to the smoothness assumption, Figure 2.4.

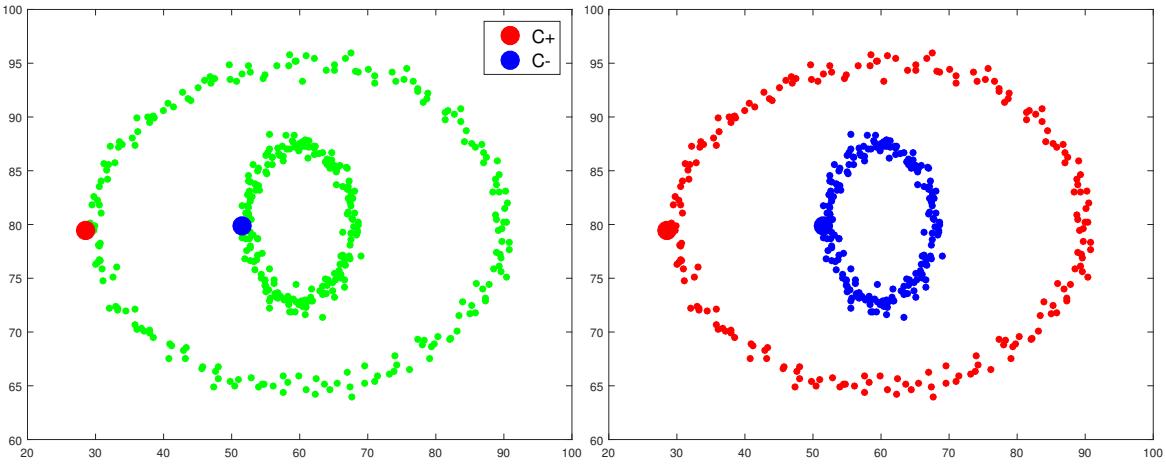


(a) (b)

Fig. 2.4 (a)The two moon synthetic dataset with 200 patterns in a two-dimensional space, 198 unlabelled patterns and two labelled patterns belong to two classes. Each of the blue and red classes contain one labelled pattern and the green unlabelled pattern is the selected pattern to be labelled. (b) Constructing a graph by propagating labels over the graph using both labelled and unlabelled patterns. Similar patterns of the labelled and unlabelled data are connected and they have similar labels.

Consider the two moon synthetic benchmark problem in Figure 2.4 that has one labelled pattern in each class and the rest are unlabelled. in Figure 2.4a, without the graph, the unlabelled pattern that is represented by green dot, would be put on the side of the blue class rather than red class according to Euclidean distance measure. However, with the graph methods in Figure 2.4b that both the labelled and unlabelled data are taken to construct a graph and they represented as nodes in a graph. Then, a similarity matrix is used to connect the graph and the information should propagate from labelled through the unlabelled patterns using indirect graph methods. In this case, clearly a graph would put the unlabelled green pattern on the side of the red class because the unlabelled patterns are taken into account. There are many unlabelled patterns that connect the unlabelled green pattern to the red labelled patterns but there is a big gap that disconnect the unlabelled green pattern from the blue class. The intuition this graph captures is similar that patterns on the manifold should have similar labels.

In some cases, both manifold and cluster assumption are correct but it is not necessary for the cluster assumption to be Gaussian, Figure 2.5.



(a) the unlabelled data with two labelled patterns (b) the two clusters shown by the contours.

Fig. 2.5 Clusters in the data set of concentric clusters.

A number of semi-supervised learning algorithms, which typically make one or more of the above assumptions, have been developed in order to help in learning from a small amount of labelled data and potentially a large amount of unlabelled data. In this chapter, several popular semi-supervised learning methods are described.

2.2.1 Self-training:

Self-training [71, 28, 1] is an iterative and single view algorithm, which consists of one classifier. A single-view means that the algorithm use the whole set of input features. It starts by using the labelled training set to build an initial classifier. Afterwards, the classifier is used to classify (predict class labels for) a larger set of unlabelled patterns. The most confidently classified unlabelled patterns, together with their predicted labels, are added to the training set. The classifier is then re-trained and the procedure repeated until no more unlabelled patterns are available or stopping criteria are satisfied [93]. The self-training procedure is summarised in Algorithm 1.

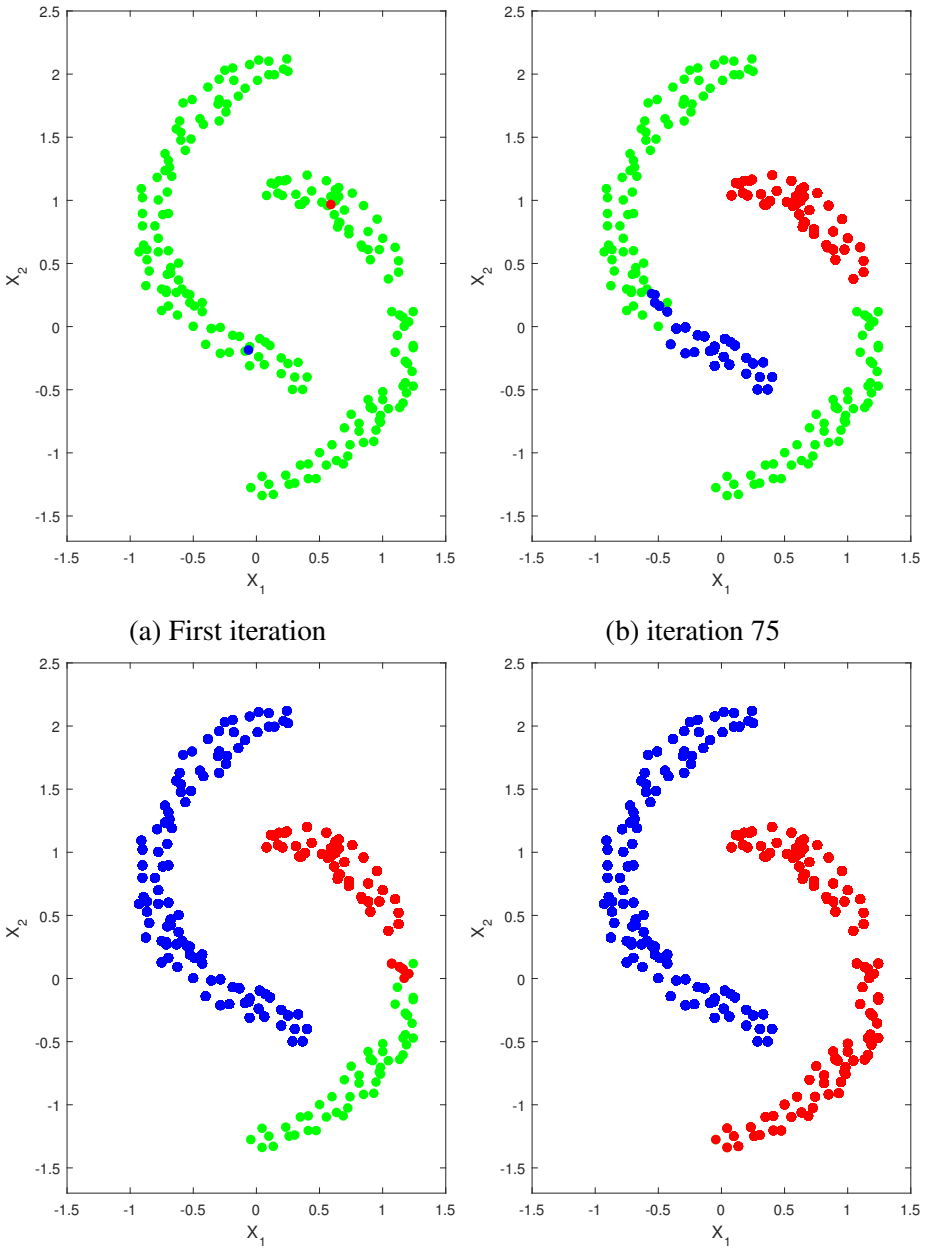
Predicting labels for unlabelled patterns in the self-training process is based on confidence

Algorithm 1 Self-training

Inputs:
$X_l \leftarrow \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{l}$
$X_u \leftarrow \left\{x^{(i)}\right\}_{i=l+1}^{l+u}$
L : learning algorithm
Onputs:
learn hypothesis f
Methods:
$t \leftarrow 0$
$f_t \leftarrow L(X_l)$
while $X_u \neq \phi$ **do**
  $X_s \leftarrow \phi$
  for all i such that $X^{(i)} \in X_u$ **do**
    $Z^{(i)} \leftarrow \psi\left(f_t(X^{(i)})\right)$ where $\psi(.)$ is step function
    $\psi(v) \leftarrow \begin{cases} 1 & v \quad true \\ 0 & v \quad false \end{cases}$
    $X_s \leftarrow \left\{(X^{(i)}, Z^{(i)})\right\}_{i=l+1}^{l+u}$
  end for
  $X_s \leftarrow$ top K most confident (Rank (X_s))
  $X_u \leftarrow X_u \,|\, X'_s$
  $X_l \leftarrow X_l \cup X'_s$
  $t \leftarrow t+1$
  $f_t \leftarrow L(X_l)$
end while

predictions therefore the selection confidence measure is important to the performance of self-training. The way in which the confidence of predictions is measured, is dependent on the type of classifier used. Probabilistic classifiers such as naïve Bayes are based on the output probability in prediction, i.e. the most confident patterns are the patterns whose posterior probability of class membership are closest to 1 or 0 for a binary classification problem. However, non-Probabilistic classifiers such as K nearest neighbour use distance metric as the measure of confidence. Triguero et al. [81] used the nearest neighbour classifier as the base learner, so they used a distance metric as the measure of confidence. The most confident unlabelled instance is defined as the closest unlabelled pattern to any labelled one.

Self-training is also known as self-teaching because the classifier uses its own predictions to teach itself based on the high confidence prediction. Thus, the self-training algorithm assumption is that the classifier's own predictions tend to be correct at each iteration of the training procedure to lead itself to better results [26].



(a) First iteration

(b) iteration 75

(c) iteration 150

(d) Final iteration

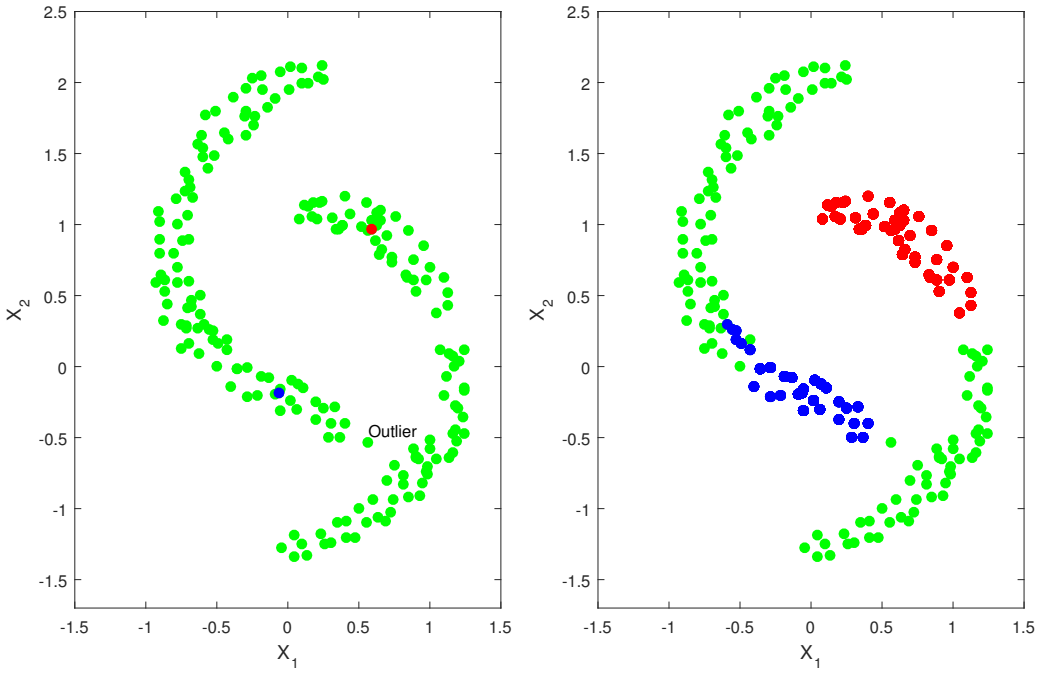
Fig. 2.6 Self-training without outliers and the assumption is correct.

Figure 2.6 is an illustrative example to show the overall view of the self-training algorithm and explain their steps in Algorithm 1. The self-training algorithm is run on a two-dimensional binary classification problem, which is the two moon synthetic dataset, with the k-nearest-neighbour KNN classifier as the base learner, where $k = 1$. Figure 2.6 illustrates the way labels propagate 1-nearest-neighbour starting from one labelled pattern in each class. Figure 2.6a shows the original dataset consisting of 200 patterns, two patterns are labelled and the rest are unlabelled. The labelled training patterns are depicted as a red and blue classes, and the unlabelled patterns as green. In each iteration, one unlabelled pattern is assigned to the class of labelled pattern that is closest (in each iteration, self-training can predict the labels for one nearest pattern) and then add it to the labelled training patterns to retrain the classifier. Figure 2.6b to 2.6d shows three iterations where self-training can predict the labels for all unlabelled patterns successfully. This is because the model assumption is valid for this dataset [93].

In contrast, wrongly labelled patterns during self-training process can lead to a degradation in performance. This shows that in the case of the outliers, the self-training would be sensitive [75]. Figure 2.7 shows a different sequence of iterations that a self-training approach based on the 1-nearest-neighbour classifier leads to propagating incorrect information after introducing a single outlier pattern for the previous dataset shown in Figure 2.6. This outlier breaks the well-separated cluster assumption, in that it falls between the two classes and far from the centre of any of the clusters [93].

The self-training algorithm is very sensitive to the performance of the initial classifier that may produce an early poor initial model due to limited training data. Then, the classifier can reinforce the poor initial model by generating incorrect labels for the unlabelled data, i.e. if we add a misclassified unlabelled pattern in an iteration during the self-labelling process, it makes the algorithm worse in performance rather than better. Therefore, in some cases, self-training may try to avoid this by ignoring these unlabelled patterns if the prediction

confidence drops below a threshold.



(a) First iteration

(b) iteration 75

(c) iteration 150

(d) Final iteration

Fig. 2.7 Self-training with outliers then the model assumption would be incorrect [93]

The first step towards improving classification performance using unlabelled data is the self-training algorithm that uses both labelled and unlabelled data [71, 28, 1]. Although these methods have been used for several decades, they still remain effective in machine learning research, especially when collecting a large amount of labelled data is difficult. Self-training has been used in several practical applications such as natural language processing tasks. For example, Yarowsky [87] applied the self-training technique to word sense disambiguation, and Riloff et al. [67] applied it to identify subjective sentences and distinguish then from objective sentences. Yarowsky [87] is an early self-training application, which addressed the word sense disambiguation problem in the context. More specifically, Yarowsky [87] applied self-training to translate the context of written words when they have different meanings. For example, whether “crane" can mean a bird or a machine. Furthermore, using only two labelled patterns with unlabelled patterns to train a classifier can lead to higher classification performance because words around the target word provide a strong sense as to its meaning. In order to identify the subjective nouns automatically, Riloff et al. [67] applied the self-training algorithm to improve the naïve Bayes classifier. The main working principle is the use of unlabelled patterns to learn sets of subjective nouns through two bootstrapping algorithms. The first algorithm was Meta-bootstrapping and the second was the Basilisk algorithm. These two algorithms were designed for learning semantic words such as apple is a fruit. The Meta-Bootstrapping and Basilisk algorithms start with the non-annotated texts and seed words that are semantic in nature. Both start with the extraction of patterns as follows; the former extracts patterns using the syntactic templates and saves the top-five best nouns phrases only. The latter algorithm extracts patterns to build a semantic dictionary. Additionally, Basilisk can learn more than a 1000 subjective nouns. Thus, self-training paved the way to identifying subjective nouns.

The self-training algorithm was brought to the machine translation application by Rosenberg et al. [68]. They applied self-training to object detection systems from images that specifically

targeted human eye detection. In this application, the authors use an object detector based on the Nearest Neighbour classifier. The results obtained showed that using unlabelled patterns can improve the performance of the Nearest Neighbour classifier where the unlabelled data were selected by an independent measure rather than the classifier confidence. This suggests that the low density separation assumption is not satisfied for object detection and other approaches may work better.

One of the drawbacks of the self-training is apparent when noisy patterns are classified as the most confident patterns. Thus, the performance of the classifier would be worse when these noisy patterns are added to the labelled training patterns and then used them for training model. Li and Zhou [48] modified the self-training approach and they showed that the quality of the initial labelled patterns is important. If the algorithm starts with poor quality labelled data then the final prediction will also be distorted, in this case the self-training algorithm may degrade the performance of the classifier rather than improving it. Therefore, Li and Zhou [48] added some statistical filters to the self-training algorithm so that a new algorithm provides better classification performance. A new derived algorithm from self-training is self-training with Editing (SETRED). Nevertheless, this method does not perform well in many domains. The aim of adding the statistical filters is generally to remove the noisy patterns. However, with removal of potentially noisy patterns may remove informative patterns as well.

Guo et al. [33] illustrate an intensive benchmarking study of self-training techniques based on a range of different classifiers. In this study, the author considers 26 UCI Benchmark datasets and applied six different Bayesian classifiers, NB, NBTree, DNB, HNB, TAN, and HGC. They also used TSVM and graph-based semi-supervised learning methods as well. However, they obtained results that show that the performance of self-training under-performs that of the supervised classifier that use only the labelled data.

Tanha et al. [79] reached a similar result and concluded that, using unlabelled data through

a self-training algorithm, does not improve the performance of a base classifier when the decision tree is used as base classifier. The best explanation for the poor results obtained by self-training algorithm is that the decision tree classifier does not compute reliable probability estimates for their predictions. However Tanha et al. [79] shows that, a sequence of modifications to the decision tree classifier such as the Laplace correction, grafting and NBTree leads to an improvement in the probability estimates. Thus, using unlabelled data through the self-training, based on a new version of the decision tree classifier, can improve the classification performance.

2.2.2 Semi-supervised learning with generative mixture models

Semi-supervised learning of mixture models is another early semi-supervised method that uses probabilistic generative models, $p(x,y) = p(x|y)p(y)$, when both labelled and unlabelled patterns are available. This method attempts to estimate the class conditional densities $p(x|y)$ by making the assumption that both labelled and unlabelled data are drawn from a mixture distribution.

In semi-supervised learning there are no true class labels for unlabelled data, so the generative model treats them as a hidden variable and employs the Expectation-Maximization (EM) algorithm to maximize the likelihood of the model parameters on both labelled and unlabelled data [24]. The procedure of labelling the unlabelled data in semi-supervised learning for mixture models is unlike self-training 2.2.1, which assigns discrete classification label (hard self-training) to the most confident unlabelled data. In semi-supervised learning for mixture models, the EM algorithm assigns probabilistic labels to the unlabelled data. Therefore, the EM algorithm is recognised as a form of soft self-learning [93]. The detailed procedure of semi-supervised learning with generative models is given in Algorithm 2.

Algorithm 2 The EM Algorithm for Semi-Supervised Classifier

1: **Inputs:**
   $X_l \leftarrow \left\{(x^{(i)}, y^{(i)})\right\} l_{i=1}$
   $X_u \leftarrow \left\{x^{(i)}\right\} \; {}^{l+u}_{i=l+1}$
   $t \leftarrow 0$
2: **Initialise:**
   $\hat{\theta}^{(0)} \leftarrow argmax_{\theta}\, P(X_l, Y_l \mid \theta)P(\theta)$
3: **while** classifier parameters improve as measured by the change in $l(\theta \mid X_l, Y_l, X_u)$: **do**
4: **E-Step** use the current classifier, $\hat{\theta}^{(t)}$, to find $\gamma_{ik} \leftarrow P(y_i = k \mid x_u; \theta)$, equation 2.14
5: **M-Step** re-estimate the classifier, $\hat{\theta}^{(t+1)} \leftarrow argmax_{\theta}\, P(X_l, Y_l, X_u \mid \theta^{(t)})P(\theta^{(t)})$
6: $t \leftarrow t + 1$
7: **Output** a classifier, $\hat{\theta}^{(t)}$, that takes unlabelled data and predicts a class label.
8: **end while**

The EM algorithm starts with an estimate for the initial vector of model parameters, using the labelled data only, and then iterates over the following two steps until it converges to a stable solution and set of predicted labels for the unlabelled data. The EM algorithm first estimates the expectations of the missing labels (latent variables) for the unlabelled instances in the E-step and assigns probabilistic labels to the unlabelled data. The M-step estimates the new model parameters using all of the labelled and unlabelled data, and treats the expected values of the latent variable that were calculated in the E-step as the true class labels for the unlabelled data. If the model assumptions are correct, using a large amount of unlabelled data might improve the classifier's performance. Otherwise, unlabelled data may not help in improving the classifier's performance [13, 66], as it is shown in Figure 2.1 and 2.2.

Mixture models use different methods to fit the data with the EM algorithm. For example, in the image classification problem investigated by Shahshahani et al. [75], the authors assume that the data come from a Gaussian mixture model, so, they assume that $p(x|y)$ is a Gaussian distribution. Similarly, Inoue and Ueda [37] used Hidden Markov Models for the speech recognition and then Nigam et al. [59] applied the mixture multinomial naïve Bayes model to the text classification problem, because $p(x|y)$ is then distributed according to the mixture multinomial. Nigam et al. [59] describes a successful application of the EM algorithm,

applied to maximise the likelihood for both the labelled and unlabelled patterns. Similarly, both Fergus et al. [27] and Baluja [7] used the EM algorithm for learning of object categories and face orientation discrimination, respectively. If the model assumptions are violated, then using the unlabelled data might degrade the performance of a classifier [18, 21]. Therefore, valid model assumptions are required in order to apply soft self-training successfully. This point was experimentally shown by Saeed et al. [70], using large scale real and synthetic benchmark datasets. When only a few labelled patterns with considerable unlabelled data are available the majority of the data determining EM's parameter estimates come from the unlabelled set. A possible solution for this issue is down-weight the unlabelled data, which is used by Nigam [60] and Callison-Burch et al. [12].

2.2.3 The semi-supervised naïve Bayes classifier

In Section 2.1, the supervised naïve Bayes (NB) classifier with fully labelled data was described. However, in some cases the training data D consists of both labelled, D_l, and unlabelled, D_u instances, $D = D_l \cup D_u$. Applying the NB classifier with both types of data is called semi-supervised naïve Bayes (SSNB). Considering the labelled data $D_l = \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{l}$ and unlabelled data is $D_u = \left\{(x^{(i)})\right\}_{i=l+1}^{l+u}$. Then, the likelihood function is defined as:

$$\begin{aligned} p(\mathscr{D};\theta) &= p(\mathscr{D}_l;\theta) \times p(\mathscr{D}_u;\theta), \\ &= \prod_{i=1}^{l}\left(p(y^{(i)};\theta)\prod_{j=1}^{d}p(x_j^{(i)}|y^{(i)};\theta)\right), \\ &\quad \times \prod_{i=l+1}^{l+u}\prod_{j=1}^{d}p(x_j^{(i)};\theta). \end{aligned} \tag{2.11}$$

The likelihood for the unlabelled data is the essential difference between the supervised and semi-supervised log likelihoods. The likelihood for unlabelled data is the marginal

probability $p(x_j^{(i)};\theta)$, as we do not know to which class they belong. To address this problem we add the latent variables $z^{(i)}$, where $i = l+1, l+2, ..., l+u$, for the unlabelled data and try to maximise the semi-supervised likelihood.

$$\begin{aligned} p(\mathscr{D};\theta) &= \prod_{i=1}^{l}\left(p(y^{(i)}|\theta)\prod_{j=1}^{d}p(x_j^{(i)}|y^{(i)};\theta)\right) \\ &\quad \times \prod_{i=l+1}^{l+u}\left(\sum_{c=1}^{C}p(z^{(i)}=c;\theta)\prod_{j=1}^{d}p(x_j^{(i)}|z^{(i)}=c;\theta)\right). \end{aligned} \tag{2.12}$$

Instead of maximising the likelihood, $p(\mathscr{D};\theta)$, we work with log-likelihood log $p(\mathscr{D};\theta)$.

$$\begin{aligned} \log p(\mathscr{D};\theta) &= \sum_{i=1}^{l}\log\left(p(y^{(i)};\theta)\prod_{j=1}^{d}p(x_j^{(i)}|y^{(i)};\theta)\right), \\ &\quad + \sum_{i=l+1}^{l+u}\log\left(\sum_{c=1}^{C}p(z^{(i)}=c;\theta)\prod_{j=1}^{d}p(x_j^{(i)}|z^{(i)}=c;\theta)\right). \end{aligned} \tag{2.13}$$

When there are latent variables (no labels for unlabelled data) in the model, it is no longer possible to find a closed form solution for the MLE, because the summation inside the *log* is hard to maximise by setting it partial derivatives to zero. Therefore, we use an iterative statistical technique known as Expectation Maximisation (EM). This algorithm overcomes this problem; it can find a local maximum or saddle point of the likelihood by maximizing a lower bound on the likelihood for unlabelled data instead of maximizing likelihood itself.

The EM algorithm starts with an estimate for the initial vector of parameters, using the labelled data only, via the standard NB, and then iterates over the following two steps until it converges to a stable solution and set of labels for the data. The EM algorithm first estimates the expectations of the missing labels (latent variables) for the unlabelled instances in the E-step, $q_{ic} = p(z^{(i)} = c|x_j^{(i)};\theta)$, where $i = (l+1, ..., l+u)$ and $0 \leq q_{ic} \leq 1$ and assigns probabilistic labels to the unlabelled data.

$$q_{ic} = \frac{p(z^{(i)}=c;\theta)\prod_{j=1}^{d}p(x_j^{(i)}|z^{(i)}=c;\theta)}{\sum_{k=1}^{C}p(z^{(i)}=k;\theta)\prod_{j=1}^{d}p(x_j^{(i)}|z^{(i)}=k;\theta)}. \tag{2.14}$$

Note that for the labelled data we already know to which class each pattern belongs then $q_{ic} = 1$ if $y^{(i)} = c$ and $q_{ic} = 0$ otherwise. In addition, the q_{ic} satisfy the summation constraint $\sum_{c=1}^{C} q_{ic} = 1$. In order to obtain the lower bound for unlabelled data we multiply and divide $\log p(\mathscr{D}_u;\theta)$ by q_{ic},

$$\begin{aligned}
\log p(\mathscr{D}_u;\theta) &= \sum_{i=l+1}^{l+u} \log\left(p(z^{(i)}=c;\theta) \prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta)\right)\frac{q_{ic}}{q_{ic}}, \\
&= \sum_{i=l+1}^{l+u} \log \sum_{c=1}^{C} q_{ic}\left(\frac{p(z^{(i)}=c;\theta)\ \prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta)}{q_{ic}}\right), \\
&= \sum_{i=l+1}^{l+u} \log E_{q_{ic}}\left\{\frac{p(z^{(i)}=c;\theta)\ \prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta)}{q_{ic}}\right\}. \qquad (2.15)
\end{aligned}$$

The lower bound for unlabelled data is obtained via Jensen's inequality [59] $E[\log(X)] \leq \log(E[X])$,

$$\begin{aligned}
&\sum_{i=l+1}^{l+u} \log E_{q_{ic}}\left\{\frac{p(z^{(i)}=c;\theta)\ \prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta)}{q_{ic}}\right\}, \\
&\geq \sum_{i=l+1}^{l+u} E_{q_{ic}}\left\{\log\frac{p(z^{(i)}=c;\theta)\ \prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta)}{q_{ic}}\right\}. \qquad (2.16)
\end{aligned}$$

we substitute the right hand side of the expression (2.16), instead of the second term in (2.13) and denote by $\psi(\theta)$,

$$
\begin{aligned}
\psi(\theta) &= \sum_{i=1}^{l} \log\Big(p(y^{(i)};\theta) \prod_{j=1}^{d} p(x_j^{(i)}|y^{(i)};\theta)) \Big) \\
&\quad + \sum_{i=l+1}^{l+u} E_{q_{ic}} \left\{ \log \frac{p(z^{(i)}=c;\theta)\ \prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta)}{q_{ic}} \right\}, \\
&= \sum_{i=1}^{l} q_{ic} \log\Big(p(y^{(i)};\theta) \prod_{j=1}^{d} p(x_j^{(i)}|y^{(i)};\theta) \Big) \\
&\quad + \sum_{i=l+1}^{l+u} \sum_{c=1}^{C} q_{ic} \log\left(\frac{p(z^{(i)}=c;\theta)\ \prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta)}{q_{ic}} \right), \\
&= \sum_{i=1}^{l} q_{ic} \log\Big(p(y^{(i)};\theta) \prod_{j=1}^{d} p(x_j^{(i)}|y^{(i)};\theta) \Big) \\
&\quad + \sum_{i=l+1}^{l+u} \sum_{c=1}^{C} q_{ic} \log\Big(p(z^{(i)}=c;\theta)\ \prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta) \Big) \\
&\quad - \sum_{i=l+1}^{l+u} \sum_{c=1}^{C} q_{ic} \log q_{ic}, \\
&= \sum_{i=1}^{l+u} \sum_{c=1}^{C} q_{ic} \log\Big(p(y^{(i)}=c;\theta)\ \prod_{j=1}^{d} p(x_j^{(i)}|y^{(i)}=c;\theta) \Big) \\
&\quad - \sum_{i=l+1}^{l+u} \sum_{c=1}^{C} q_{ic} \log q_{ic}, \qquad (2.17)
\end{aligned}
$$

where

$$
y^{(i)} = \begin{cases} y^{(i)} & : i = 1,...,l \\ z^{(i)} & : i = l+1,...,l+u\ . \end{cases}
$$

The M-step estimates the new model parameters via the partial derivatives for (2.17) using all of the labelled and unlabelled data. In this step, the expected values of the latent variable that is calculated in the E-step treats as the true class labels for the unlabelled data. We can show how to estimate the new model parameters as follows.

Maximum likelihood estimation for the categorical distribution

If $x_j \sim cat(\theta)$ then (2.17) can be written in terms of the parameters with a Lagrangian term,

$$\begin{aligned}\Lambda(\pi,\theta,\alpha,\beta) &= \sum_{i=1}^{l+u}\sum_{c=1}^{C} q_{ic}\log\pi_c \\ &+ \sum_{i=1}^{l+u}\sum_{j=1}^{d}\sum_{s=1}^{S}\sum_{c=1}^{C} q_{ic}\phi(x_j^{(i)}=s)\log\theta_{sc}^{j} \\ &- \sum_{i=l+1}^{l+u}\sum_{c=1}^{C} q_{ic}\log q_{ic} - \alpha\left(\sum_{c=1}^{C}\pi_c - 1\right) \\ &- \sum_{c=1}^{C}\sum_{j=1}^{d}\beta_c^j\left(\sum_{s=1}^{S}\theta_{sc}^{j} - 1\right). \end{aligned} \tag{2.18}$$

To obtain the maximum likelihood estimate the partial derivatives can be computed for (2.18) with respect to all the parameters $(\pi_c, \theta_{sc}^j, \alpha, \beta_c^j)$ and set to zero. The maximum likelihood estimate for α and β_c^j is same as supervised NB (2.7):

$$\begin{aligned}\frac{\partial\Lambda}{\partial\pi_c} &= 0 \Rightarrow \pi_c = \frac{\sum_{i=1}^{l+u} q_{ic}}{\sum_{k=1}^{C}\sum_{i=1}^{l+u} q_{ik}}, \\ \frac{\partial\Lambda}{\partial\theta_{sc}^{j}} &= 0 \Rightarrow \theta_{sc}^{j} = \frac{\sum_{i=1}^{l+u} q_{ic}\phi(x_j^{(i)}=s)}{\sum_{m=1}^{S}\sum_{i=1}^{l+u} q_{ic}\phi(x_j^{(i)}=m)}, \end{aligned} \tag{2.19}$$

where the summation in the denominator is over all possible values (states) of m, for each feature x_j. The Laplace correction for the parameters, (θ_{sc}^j, π_c), is as follows:

$$\begin{aligned}\pi_c &= \frac{\sum_{i=1}^{l+u} q_{ic} \;+\; 1}{\sum_{k=1}^{C}\sum_{i=1}^{l+u} q_{ik} \;+\; C}, \\ \theta_{sc}^{j} &= \frac{\sum_{i=1}^{l+u} q_{ic}\phi(x_j^{(i)}=s) \;+\; 1}{\sum_{m=1}^{S}\sum_{i=1}^{l+u} q_{ic}\phi(x_j^{(i)}=m) \;+\; S_j}. \end{aligned} \tag{2.20}$$

Maximum likelihood estimation for the Gaussian distribution

The log-likelihood (2.17) without class prior probability for Gaussian distribution in SSNB can be written as follows if $\mathscr{X} \sim \mathscr{N}(\mu, \sigma^2)$, because the only difference with Categorical

log-likelihood in $p(x_j^{(i)}|y^{(i)};\theta)$:

$$\begin{aligned}\log p(\mathscr{D};\mu,\sigma^2) &= \sum_{i=l+1}^{l+u}\sum_{c=1}^{C}\sum_{j=1}^{d} q_{ic}\log\left(\frac{1}{(2\pi)^{\frac{1}{2}}|\sigma_{jc}^2|^{\frac{1}{2}}}\exp\big(-\frac{1}{2}(x_j^{(i)}-\mu_{jc})^2(\sigma_{jc}^2)^{-1}\big)\right)\\ &\quad -\sum_{i=l+1}^{l+u}\sum_{c=1}^{C} q_{ic}\log q_{ic}. \end{aligned} \tag{2.21}$$

The closed form maximum likelihood estimate can be obtained by computing the partial derivatives for (2.21) with respect to all the parameters (μ_{jc},σ_{jc}^2), and then setting each partial derivative to zero:

$$\begin{aligned}\frac{\partial\Lambda}{\partial\mu_{jc}} &= 0\Rightarrow\mu_{jc}=\frac{\sum_{i=1}^{l+u} q_{ic}\,x_j^{(i)}}{\sum_{i=1}^{l+u} q_{ic}},\\ \frac{\partial\Lambda}{\partial\sigma_{jc}^2} &= 0\Rightarrow\sigma_{jc}^2=\frac{\sum_{i=1}^{l+u} q_{ic}(x_j^{(i)}-\mu_{jc})^2}{\sum_{i=1}^{l+u} q_{ic}}.\end{aligned} \tag{2.22}$$

The general theory of the expectation-Maximization algorithm

In the previous Section, the derivation of semi-supervised learning for the NB classifier, based on the EM algorithm, was presented. The EM algorithm was applied with Gaussian and categorical mixture model. In this section we give the derivation of the general form of the EM algorithm and we show how it is applied to parameter estimation by maximum likelihood with latent variables. In this section we discuss *Jensen's inequality* as well, which is important part for this section.

The EM Algorithm

The Expectation-Maximization (EM) Algorithm was introduced by Dempster et al. [24]. The EM Algorithm is an efficient iterative statistical procedure for finding a local maximum likelihood estimate (MLE) or maximum a posteriori (MAP) estimate of the parameters of an underlying distribution in the presence of latent (missing or hidden) variables. The algorithm

starts by initializing the parameters, $\hat{\theta}^0$, and it then iteratively alternates between two steps, called the expectation step (E-step) and the maximization step (M-step), respectively.
In the E-step, the values of the latent variables are estimated given the complete data and current estimate of the model parameters, i.e. the expected value of the complete log likelihood function is computed, which is called the q-function, using the complete data where the expectation is taken w.r.t. the computed conditional distribution of the latent variables given the current settings of parameters $\theta, \hat{\theta}$. The M-step, re-estimates all the parameters $\hat{\theta}^{t+1}$ to maximize the new, q-function. In this step, the estimates of the latent variables from the E-step are treated as the actual values, so it is assumed that the values of the latent variables are known.
These two updates are iterated until the log likelihood converges. The EM algorithm is a hill-climbing approach, thus it cannot guarantee to reach global maxima. When there are multiple maxima and it starts close to the right hill, it might reach global maxima. However, it is often hard to start with right hill if there are multiple local maxima.

Derivation of the EM Algorithm

Assume we are given the unlabelled data an $D_u = \left\{(x^i)\right\}_{i=l+1}^{l+u}$ where $x^i \in \mathscr{X} \subseteq \mathscr{R}^d$ represents a feature vector describing the i^th pattern and the log-likelihood for incomplete data is the marginal probability $p(x^i;\theta)$. Here, to address this problem we add the latent (hidden) variable z^i which represents a corresponding unobserved variables for x^i. Then, the log-likelihood function is given by:

$$\begin{aligned} L(\theta) &= \log p(\mathscr{D}_u;\theta) \\ &= \sum_{i=l+1}^{l+u} \log p(x^i;\theta) \\ &= \sum_{i=l+1}^{l+u} \log p(x^i,z^i;\theta) \end{aligned} \tag{2.23}$$

We would like to choose the parameters, θ, to maximise the log-likelihood function of the (2.23). Since z^i is a hidden variable, finding the maximum log-likelihood estimate of the parameters θ directly is difficult. However, if z^i were known then it would be easy to maximise the log-likelihood. It attempts to find a lower bound on the log-likelihood function (E-step) and then maximise that lower bound (M-step) iteratively as follows:

Suppose for each i, we have some probability distribution $p(z^i|x^i;\theta)$ for the latent variable z^i, where $p(z^i|x^i;\theta) > 0$ and $\sum_z p(z^i|x^i;\theta) = 1$. Then multiplying and dividing inside the summation of the right-hand side (2.23) by $p(z^i|x^i;\theta)$, gives

$$L(\theta) \quad = \quad \sum_{i=l+1}^{l+u} \log \sum_{z^i} p(z^i|x^i;\theta) \frac{p(x^i,z^i;\theta)}{p(z^i|x^i;\theta)}. \tag{2.24}$$

It is difficult to deal with this expression which involves the logarithm of a sum. So, Jensen's inequality is applied to this expression to replace with a sum of the logarithm.

Jensen's inequality

Let $f(x) = \log(x)$ be a real valued function defined on an interval I=[x_1,x_2]. the function f is said to be a concave function on I $\forall x \in \mathscr{R}$ if f"$(x) \leq 0$. Similarly, f is said to be strictly concave if f"$(x) < 0$. Figure 2.8 shows a example of a concave function.

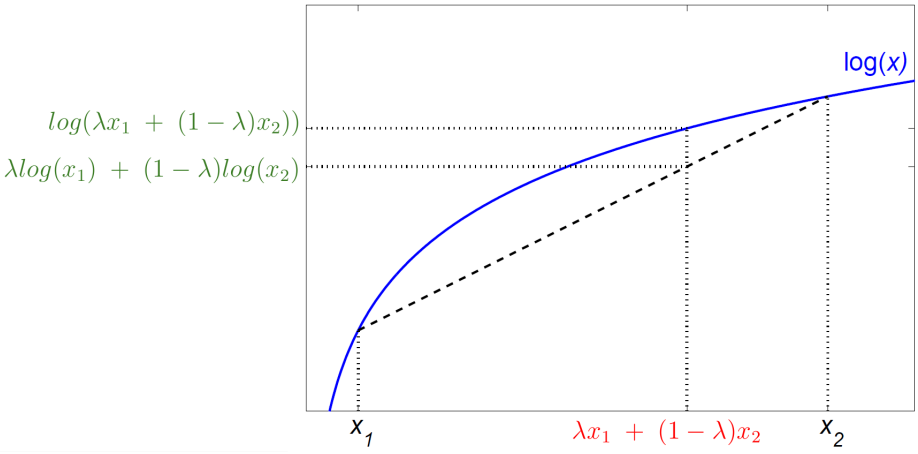


Fig. 2.8 The $\log(x)$ function is a concave on $[x_1,x_2]$ if $\lambda \log(x_1) + (1-\lambda)\log(x_2) \leq \log(\lambda x_1 + (1-\lambda)x_2))$ where $\lambda \in [0,1]$ Ng et al. [56]

Theorem 1: Let f be a concave function defined on an interval I. If $x_1, x_2, ..., x_m \in I$ and $\lambda_1, \lambda_2, ..., \lambda_m \geq 0$ then;

$$\sum_{i=1}^{m} \lambda_i f(x_i) \leq f\left(\sum_{i=1}^{m} \lambda_i x_i\right) \quad where \quad \sum_{i=1}^{m} \lambda_i = 1 \tag{2.25}$$

The result above can be adapted to our purposes by defining $f(x) = \log(x)$, which is a concave function as shown in Figure 2.8, to obtain

$$\sum_{i=1}^{m} \lambda_i \log(x_i) \leq \log\left(\sum_{i=1}^{m} \lambda_i x_i\right) \tag{2.26}$$

Now Jensen's inequality can be applied to (2.24) by introducing λ_i, with $p(z^i|x^i;\theta)$, to replace the logarithm of a sum with a sum of logarithms, This gives;

$$L(\theta) \geq \sum_{i=l+1}^{l+u} \sum_{z^i} p(z^i|x^i;\theta) \log \frac{p(x^i,z^i;\theta)}{p(z^i|x^i;\theta)}, \tag{2.27}$$

Equation (2.27) gives the lower bound on the log-likelihood function, denoted by $\mathscr{L}(\theta)$. The EM algorithm in this step could maximise the expected log-likelihood function by maximising the $\mathscr{L}(\theta)$.

$$\begin{aligned} \mathscr{L}(\theta) &= \sum_{i=l+1}^{l+u} \sum_{z^i} p(z^i|x^i;\theta) \log \frac{p(x^i,z^i;\theta)}{p(z^i|x^i;\theta)}, \\ &= \sum_{i=l+1}^{l+u} [\sum_{z^i} p(z^i|x^i;\theta) \log p(x^i,z^i;\theta) - \sum_{z^i} p(z^i|x^i;\theta) \log p(z^i|x^i;\theta)]. \end{aligned} \tag{2.28}$$

The second term has not affected the optimisation because it is constant with respect to θ. Thus, we deal only with first part of $\mathscr{L}(\theta)$, which is the expected log-likelihood function.

Using the definition $\mathbb{E}[f(x)] = \sum_x p(x)f(x)]$,

$$\begin{aligned}\mathscr{L}(\theta) &= \sum_{i=l+1}^{l+u}[\sum_{z^i} p(z^i|x^i;\theta)\log p(x^i,z^i;\theta)], \\ &= \sum_{i=l+1}^{l+u}[\mathbb{E}_{z^i|x^i;\theta}\log p(x^i,z^i;\theta)]. \end{aligned} \tag{2.29}$$

To make the inequality hold with equality, the EM algorithm uses the initial parameter θ^t to fit the lower bound and iteratively maximise the lower bound $\mathscr{L}(\theta)$. The process of maximizing $L(\theta)$ is the same as maximizing the difference in the log likelihood between iterations,

$$\begin{aligned} L(\theta) - L(\theta^t) &= \sum_{i=l+1}^{l+u}[\log\sum_{z^i} p(z^i|x^i;\theta^t)\frac{p(x^i,z^i;\theta)}{p(z^i|x^i;\theta^t)} - \log p(x^i;\theta^t)], \\ &\geq \sum_{i=l+1}^{l+u}[\sum_{z^i} p(z^i|x^i;\theta^t)\log\left(\frac{p(x^i,z^i;\theta)}{p(z^i|x^i;\theta^t)}\right) - \log p(x^i;\theta^t)], \\ &= \sum_{i=l+1}^{l+u}\sum_{z^i} p(z^i|x^i;\theta^t)\log\left(\frac{p(x^i,z^i;\theta)}{p(z^i|x^i;\theta^t)p(x^i;\theta^t)}\right), \\ &\equiv \triangle(\theta|\theta^t). \end{aligned} \tag{2.30}$$

$\triangle(\theta|\theta^t)$ is the Kullback-Leibler divergence which is a measure of difference between two probability distribution. Moving $L(\theta^t)$ to the other side, then we get

$$L(\theta) \geqslant L(\theta^t) + \triangle(\theta|\theta^t) = l(\theta|\theta^t). \tag{2.31}$$

The $l(\theta|\theta^t)$ is the lower bound of the log-likelihood function everywhere but at that value of θ^t, is equal to $L(\theta)$ if we show that $\triangle(\theta|\theta^t) = 0$. This is normally what would happen if the

two function were exactly equal at $\theta = \theta^t$,

$$\begin{aligned}
\triangle(\theta^t|\theta^t) &= \sum_{i=l+1}^{l+u}\sum_{z^i} p(z^i|x^i;\theta^t)\log\left(\frac{p(x^i,z^i;\theta^t)}{p(z^i|x^i;\theta^t)p(x^i;\theta^t)}\right), \\
&= \sum_{i=l+1}^{l+u}\sum_{z^i} p(z^i|x^i;\theta^t)\log\left(\frac{p(x^i,z^i;\theta^t)}{p(x^i,z^i;\theta^t)}\right), \\
&= \sum_{i=l+1}^{l+u}\sum_{z^i} p(z^i|x^i;\theta^t)\log(1), \\
&= 0. \qquad (2.32)
\end{aligned}$$

Now the goal is finding the value of θ that maximises $L(\theta)$. Increasing the value of $l(\theta|\theta^t)$ would guarantee to an improvement in the $L(\theta)$, as a result, at each iteration the EM algorithm tries to select the θ that maximises $l(\theta|\theta^t)$. The next update of the parameter can be computed as follows;

$$\begin{aligned}
\theta^{t+1} &= \arg\max_{\theta}\left\{l(\theta|\theta^t)\right\}, \\
&= \arg\max_{\theta}\left\{L(\theta^t) + \triangle(\theta|\theta^t)\right\}, \\
&= \arg\max_{\theta}\left\{L(\theta^t) + \sum_{i=l+1}^{l+u}\sum_{z^i} p(z^i|x^i;\theta^t)\log\left(\frac{p(x^i,z^i;\theta^t)}{p(z^i|x^i;\theta^t)p(x^i;\theta^t)}\right)\right\} \qquad (2.33)
\end{aligned}$$

The constant term could be dropped because it does not have any affect the procedure of maximisation which is respect to the θ,

$$\begin{aligned}
\theta^{t+1} &= \arg\max_{\theta}\left\{\sum_{i=l+1}^{l+u}\sum_{z^i} p(z^i|x^i;\theta)\log p(x^i,z^i;\theta)\right\}, \\
&= \arg\max_{\theta}\left\{\sum_{i=l+1}^{l+u}\mathbb{E}_{z^i|x^i;\theta}\log p(x^i,z^i;\theta)\right\}. \qquad (2.34)
\end{aligned}$$

The EM algorithm tries to control the lower bound of log-likelihood function with respect to $p(z^i|x^i;\theta)$ in the M-step and maximise it with respect to θ in the E-step. For this reason it can be viewed as a coordinate ascent [55].

2.2.4 Co-training

Another popular algorithm for semi-supervised learning is co-training, introduced by Blum and Mitchell [11]. In the Co-training algorithm, the features in the training set are divided into two different sets (views). Co-training starts with training two separate classifiers, with the labelled data from its respective view. Then, each classifier labels the unlabelled data of its own view and the most confident predictions of each classifier on the unlabelled data are used to expand the training set of the other classifier. Afterwards, both classifiers are retrained with the newly training labelled data given by the other classifier, and the process repeats. For the co-training algorithm, there are two assumptions on the feature sets:

Algorithm 3 Co-training

1: **Inputs:**
   $X_l \leftarrow \left\{(x^{(i)}, y^{(i)})\right\}\ l_{i=1}$
   $X_u \leftarrow \left\{x^{(i)}\right\}\ _{i=l+1}^{l+u}$
   $f^{(1)}$, $f^{(2)}$: two classifiers
2: **Initialise:**
   learn hypothesis f
   split X_l to $X_z^{(1)}$ and $X_z^{(2)}$; $X_l = (X_z^{(1)}, X_z^{(2)})$
3: **repeat**
4: **until** $X_u \neq \phi$
5: Classify X_u with $f^{(1)}$ and $f^{(2)}$ separately
6: Select ${f^{(1)}}'s$ top k most confident predictions into $X_s^{(1)}$, select ${f^{(2)}}'s$ top k most confident predictions into $X_s^{(2)}$
7: $X_u = X_u - X_s^{(1)} - X_s^{(2)}$
8: $X_z^{(1)} = X_z^{(1)} + X_s^{(2)}$
9: $X_z^{(2)} = X_z^{(2)} + X_s^{(1)}$

- Features can be split into two views and each view is sufficient to train a good classifier.
- The two views must satisfy the conditional independence given the class label.

The first assumption on the quality of the views is essential to the generalisation of both classifiers. If both views are sufficiently good, then we can trust the labels of each classifier

on the unlabelled data. The second assumption of conditional independence between the views is necessary for adding most confident data with predicted label by one classifier for the other classifier. If the conditional independence between the views holds, then each view can add the most informative unlabelled patterns to other view. However, co-training makes strong assumptions on the conditional independence between the views that are unlikely to be satisfied in a real world applications.

Dasgupta et al. [22] introduced a new theoretical study based on the Blum and Mitchell [11] paper. They proved that a lower generalization error for the co-training algorithm can be obtained by maximising the agreement with unlabelled patterns when the assumptions of the co-training algorithm are true. Blum and Mitchell [11] empirically investigated the possibility of the co-training algorithm working well. For this purpose, the original input features, which consist of the small amount of the labelled patterns and large amount of the unlabelled patterns, are artificially divided for the two views in order to achieve better result via the co-training algorithm. The results obtained show that using the unlabelled patterns to improve the base classifier through the co-training algorithm is difficult when a few labelled patterns are available. The best explanation for this conclusion is that the co-training assumptions (finding the best separation of the features for two views) is not valid with a small amount labelled data.

Due to the success of co-training but its relatively limited application, many works have proposed the improvement of standard co-training by eliminating the required conditions. Nigam and Ghani [57] proposed the Co-EM algorithm which is the combination of the co-training and expectation maximization (EM) and can probabilistically label the unlabelled patterns. The EM algorithm is used as a base for the new version of the co-training while in the basic co-training algorithm naïve Bayes classifiers are used. The Co-EM algorithm was applied to web page datasets and their results were better than the co-training algorithm.

Goldman and Zhou [30] relaxed the co-training split views assumption that does not require

splitting the input feature for two views. In addition, the co-training algorithm used two different classifiers instead of just a single classifier. Each classifier can be obtained via an equivalence classes set by divide the input space.

In order to relax the conditional independence assumption between views, Zhou and Li [90] proposed the Tri-training algorithm, which uses three classifiers. In order to train one classifier, the remaining two classifiers should agree on the labelling of the unlabelled data and then it will be used in the training set of the given (third) classifier. If the splitting of the feature set is not straightforward, both tri-training and co-training may over-fit with the use of the most confident instances. In addition, the over-fitting classifier can lead to the degradation of classification accuracy because both methods depend on the quality of the subsets of features. More generally, we can define learning paradigms that utilize the agreement among different classifiers. Subsequently, they expanded this idea by proposing a new algorithm named the co-forest algorithm, that involved ensemble techniques to include a large number of base classifiers [49]. Multi-view learning models do not require the particular assumptions of co-training and it has access to separate classifiers. The classifiers might be of different types (e.g., naïve Bayes, decision tree, neural network, etc.) but they are trained on the same labelled data, and are necessary to make similar predictions on any given unlabelled data [93].

Similarly, the co-training algorithm was modified fro another semi-supervised method [89] called democratic co-learning. The current algorithm does not use multiple views but it uses multiple classifiers. In addition, democratic co-learning uses a weighted majority voting procedure for labelling the unlabelled patterns. The ensemble method is applied to the training of each classifier separately, using only the labelled patterns, then each classifier uses the unlabelled patterns to obtain predictions separately. Finally, majority voting is applied between the classifiers for labelling the unlabelled patterns. In the labelling procedures, the cross-validation was used over the labelled patterns to select the confidence of the unlabelled

patterns and also to evaluate the performance of the classifier. However, cross-validation might give poor estimates when the amount of labelled patterns is small Zhou and Goldman [89] and Zhou and Li [90].

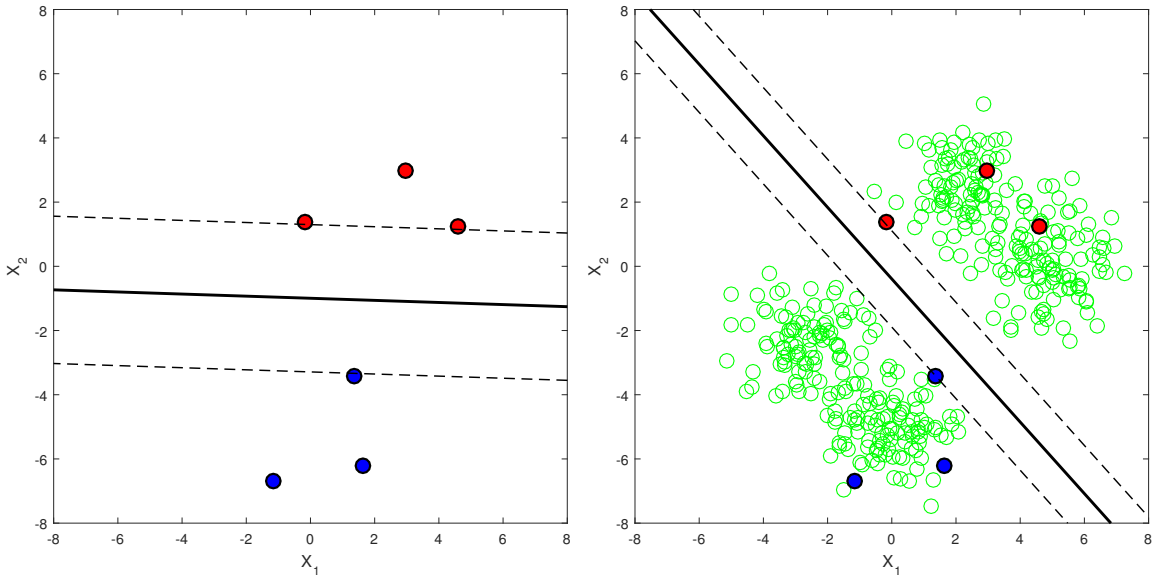
2.2.5 Low density separation

Low density separation methods are another family of learning algorithms based on the low density separation assumption that the decision boundary (hyperplane) should fall in low density regions. Thus, low density separation methods utilise the unlabelled data to regularise the decision boundary. In this section we highlight the most common low density separation approaches, which include transductive SVM and entropy regularization.

Transductive support vector machine (TSVM)

The support vector machine (SVM) Cortes and Vapnik [20], Vapnik [84] is the base classifier for transductive support vector machine (TSVM) Joachims [41], Demiriz et al. [23] that deal with labelled training data only. However, TSVM extended the SVM to exploit both labelled and unlabelled training data. In this section, we explain both the SVM and TSVM through an illustrative example. Figure 2.9a shows that the linear decision boundary, which is found by the SVM classifier, falls in the middle separating three patterns in each of a positive and negative class. The distance between the decision boundary and the two dashed lines that go through the nearest positive and negative examples is called the margin.

The TSVM attempts to find labels for the unlabelled patterns in order to identify the largest margin between the decision boundary and both labelled and unlabelled patterns by implementing low density separation assumption. From Figure 2.9b, it can be seen that the unlabelled data attempt to push the decision boundary into low density regions of unlabelled data in order to be far away from the both labelled and unlabelled patterns.



(a) SVM (b) S3VM

Fig. 2.9 (a) Linear SVM decision boundary for six labelled patterns belonging to two classes. (b) TSVM decision boundary for both labelled and unlabelled patterns [after Zhu and Goldberg [93]].

In order to understand the TSVM classifier, we first briefly review a standard SVM classifier before introducing the TSVM. Given labelled training patterns $X_l = \{(x_i, y_i)\}_{i=1}^{l}$, we assume a binary classification problem for simplicity where $x_i \in \mathscr{X} \subseteq \mathbb{R}^d$ is a feature vector of a training patterns describing the i^{th} example with class label $y_i \in \{-1, +1\}$. Then the goal of the SVM classifier is to construct a linear decision boundary given by $f(x) = w^T x_i + b$ in order to classify new patterns. As can be seen from the toy example in Figure 2.10a the decision boundary separates the feature space into two parts, where the decision boundary is defined by $f(x) = 0$, such that:

$x \in y_i = +1 \;\; if \;\; f(x) > 0,$

$x \in y_i = -1 \;\; if \;\; f(x) < 0.$

The distance between the decision boundary and the training pattens X_l either near or far from the decision boundary is greater than 0 such that $|w^T x_i + b| > 0$, but we are only interested in nearest patterns, that are called the support vectors $\acute{x}_i$ of the relative class.

$w^T \acute{x}_i + b = +1,$

$w^T \acute{x}_i + b = -1.$

If the labelled training patterns are linearly separable they also satisfy the following constraint:

$w^T x_i + b \geq +1 \;\; if \;\; y_i = +1,$

$w^T x_i + b \leq -1 \;\; if \;\; y_i = -1.$

Both of the above inequalities into single inequality:

$y_i(w^T x_i + b) \geq 1 \quad i = 1, \ldots, l.$

The margin is the perpendicular distance between the decision boundary and the support

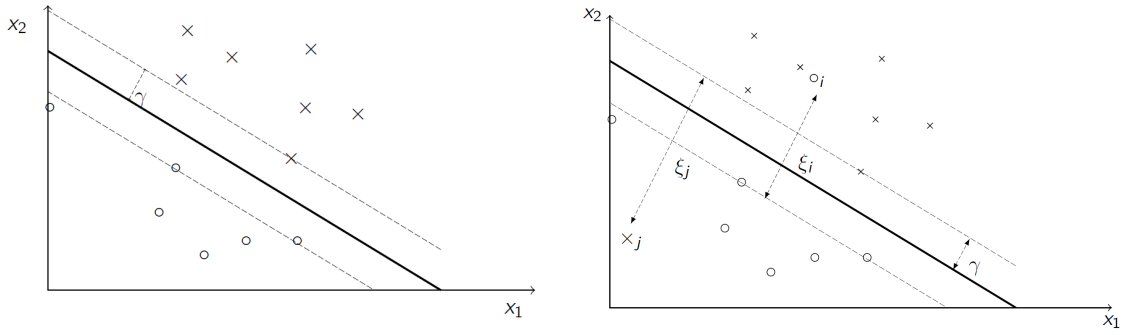


Fig. 2.10 (a) Linear decision boundary for the hard-margin SVM. (b) Linear decision boundary for the soft-margin SVM.

vectors, defined by γ, as illustrated in Figure 2.10a. The margin can be found for any patterns in the support vectors $\acute{x}_i$ as follows:

$$\begin{aligned} \gamma &= \frac{1}{\| w \|} w^T (\acute{x} - x), \\ &= \frac{1}{\| w \|} (w^T \acute{x} - w^T x), \\ &= \frac{1}{\| w \|} (w^T \acute{x} + b - w^T x - b), \\ &= \frac{1}{\| w \|} (1 + 0), \\ &= \frac{1}{\| w \|}. \end{aligned} \tag{2.35}$$

where $w^T \acute{x} + b = 1$ and $w^T x - b = 0$.

Now, the margin $\gamma = \frac{1}{\|w\|}$ can be maximised subject to the constrains $y_i(w^T x_i + b) \geq 1$.

$$\begin{aligned} &\text{minimize} && \tfrac{1}{\|w\|}, \\ &\text{subject to} && y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, l. \end{aligned} \tag{2.36}$$

Instead of maximising the margin, equivalently we can minimise $\frac{1}{2} \parallel w \parallel^2$, which is the quadratic constrained convex optimization problem and will be easier to optimise by considering the boundaries:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2} \parallel w \parallel^2, \\ &\text{subject to} && y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, l. \end{aligned} \tag{2.37}$$

This is the case, when the linear SVM classifier can separate all labelled training patterns correctly Figure **??**, this type of SVM is called hard-margin SVM, while in reality almost all datasets have noise where the labelled training patterns may not be linearly separable. In this case, the soft-margin SVM can be used, which introduces *slack variables*, which allow some patterns to lie in the incorrect side of the margin. Thus, the constraint of the optimization problem can be represented as follows:

$y_i(w^T x_i + b) \geq 1 - \xi_i \quad i = 1, \ldots, l.$

The total of violation patterns is $\sum_{i=1}^{l} \xi_i$, where $\xi_i \geq 0$. If the patterns fall into the incorrect side of the decision boundary then $\xi_i > 1$. In the case where the patterns do not fall on the wrong side, but fall between the decision boundary and the margin boundaries, then $0 \leq \xi_i \leq 1$. Thus, in order to penalise the objective function by the amount of violation, the penalty term $C\left(\sum_{i=1}^{l} \xi_i\right)^k$ is introduced into the optimization problem with parameter C and hyper-parameter k. In this case, for simplicity we assumed $k = 1$. Some methods should

be used for choosing the regularisation parameter C such as cross-validation. If C is too small, under-fitting may occur, while large C may result in over-fitting. After modifying both constraint and objective function, the optimization problem for soft-margin becomes

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2} \parallel w \parallel^2 + C\textstyle\sum_{i=1}^{l} \xi_i, \\ &\text{subject to} && y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l, \\ & && \xi_i \geq 0, \quad i = 1, \ldots, l. \end{aligned} \tag{2.38}$$

The constrained optimization problem can be solved by adding the new constraints to the objective function through a set of Lagrange multipliers. The new optimization problem is then:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2} w^T w \; - \; \textstyle\sum_{j=1}^{l} \alpha_i (y_i(w^T x_i + b) - 1), \\ &\text{subject to} && 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l. \end{aligned} \tag{2.39}$$

The inequality constraint in the optimization problem can be replaced by an equality according to the Karushkun-Tucker conditions (KKT). The derivative of the objective function regarding to the w and b parameters is set to zero:

$$\begin{aligned} \frac{\partial}{\partial w} &= 0 \Rightarrow w = \sum_{i=1}^{l} \alpha_i y_i x_i, \\ \frac{\partial}{\partial b} &= 0 \Rightarrow \sum_{i=1}^{l} \alpha_i y_i = 0. \end{aligned} \tag{2.40}$$

We substitute both w and $\sum_{i=1}^{l} \alpha_i y_i$ in to the objective function (2.39), and maximises the new objective function with respect to α:

$$\begin{aligned} &\text{maximize} && \textstyle\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j, \\ &\text{subject to} && \textstyle\sum_{i=1}^{l} \alpha_i y_i = 0 \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l. \end{aligned} \tag{2.41}$$

where $\alpha = (\alpha_1, ..., \alpha_l)$ is the vector of non-negative Lagrange multipliers. This new form of optimization problem is called the dual form. The value of $\alpha \neq 0$ is only for the support vectors. Given the optimal value of the parameters α, the prediction for the unseen (test) datasets can be made.

$$f(x) \quad = \quad \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i x_i \cdot x_{test} + b \right). \tag{2.42}$$

where for the support vectors patterns $y_i(w^T x_i + b) = 1$, so the value of parameter b can be found as follows:

$$b \quad = \quad y_i - \sum_{j=1}^{l} \alpha_j y_j x_j x_i. \tag{2.43}$$

The soft margin SVM does not help in the case of complex data structure, so we try to transform the feature space into a new feature space by a fixed non-linear transformation $\phi(x)$, this transformation method is known as a *Kernel* denoted by $K(x_i, x_j)$. We re-write the soft-margin objective function including Kernel function with prediction equation:

$$\begin{aligned} &\text{maximize} && \textstyle\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j), \\ &\text{subject to} && \textstyle\sum_{i=1}^{l} \alpha_i y_i = 0 \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l. \end{aligned} \tag{2.44}$$

where $K(x_i, x_j) = \phi(x) \cdot \phi(x)$. There are many types of kernel functions, but the most popular kernel functions are:

$K(x_i, x_j) = x_i^T \cdot x_j$ (linear kernel function)

$K(x_i, x_j) = exp\big(-\gamma(x_i - x_j)^T(x_i - x_j)\big)$ (Gaussian kernel function)

$K(x_i, x_j) = (1 + x_i^T \cdot x_j)^\gamma$ (polynomial kernel function)

Given the optimal value of the parameters α, the prediction for the unscreened datasets can be made.

$f(x) = \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x_{test}) + b\right)$.

Return back to (2.38), the inequality constraints on ξ_i can be written as $\xi_i \geq 1 - y_i(w^T x_i + b)$ thus the optimization problem can be written as follows:

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{l} max(1 - y_i(w^T x_i + b), 0), \\ &\text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l, \\ &\qquad\qquad \xi_i \geq 0, \quad i = 1, \ldots, l. \end{aligned} \tag{2.45}$$

where the first term is the regularisation term and the second is the loss function for labelled data known as hinge loss. The linear function $w^T x_i + b$ separator attempts to pass through a region that separates labelled patterns correctly in order to minimise hinge losses. However, in the case of the existing unlabelled data the separating hyperplane is far from the supervised decision boundaries that attempt to pass through low density region over both labelled and unlabelled data. The predicted labels are $\hat{y}_i = sign(f(x))$ because the true labels do not exist for the unlabelled patterns. Then the loss function for the unlabelled data, which is known as hat or symmetric function, can be presented as follows:

$$\begin{aligned} L_{hat} &= max(1 - \hat{y}_i(w^T x_i + b), 0), \\ &= max(1 - sign(w^T x_i + b)(w^T x_i + b), 0), \\ &= max(1 - \mid w^T x_i + b \mid, 0). \end{aligned} \tag{2.46}$$

In order to show the difference between the hinge and hat function, Zhu and Goldberg [93] illustrated the two diagrams below. As can be seen from Figure 2.11a hinge loss function for

$yf(x)$ but the hat loss function differs from the hinge loss. The true labels do not exist for the unlabelled patterns, thus, the hat loss function is just the function of $f(x)$ as can be seen in Figure 2.11b.

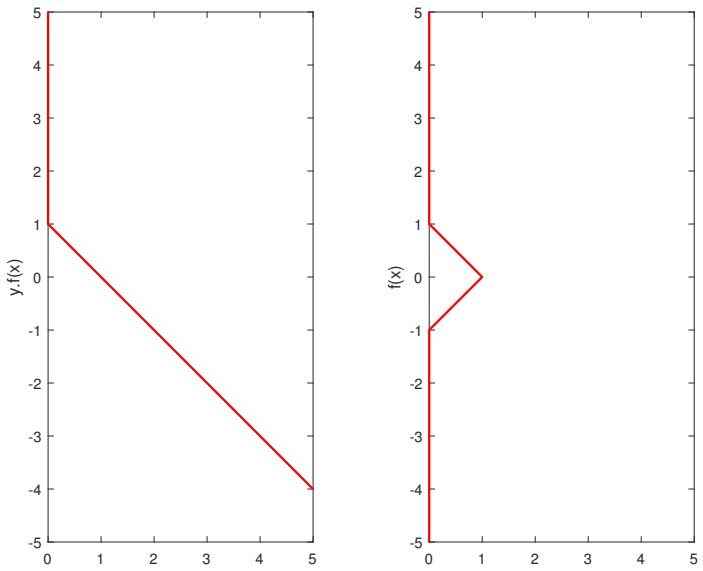


Fig. 2.11 (a) The hinge loss as a function of $yf(x)$. (b) The hat loss as a function of $f(x)$.

The only change for TSVM is to add a new term, L_{hat}, to the objective function in (2.45), which is called the hat function. Thus, in the case of existing labelled and unlabelled patterns, the optimization problem including objective function with constrained can be shown as follows

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2} \parallel w \parallel^2 + C_1 \sum_{i=1}^{l} max(1 - y_i(w^T x_i + b), 0) \\
& + C_2 \sum_{i=l+1}^{l+u} max(1 - \mid w^T x_i + b \mid, 0), \\
\text{subject to} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l, \\
& \mid w^T x_i + b \mid \geq 1 - \xi_i, \quad i = l+1, \ldots, l+u, \\
& \xi_i \geq 0, \quad i = 1, \ldots, l.
\end{aligned} \tag{2.47}$$

The last constraint added to the minimisation problem (2.47) is a class balance constraint. This constraint helps to avoid the imbalanced solution assigning all unlabelled patterns to only one of the classes.

$$\frac{1}{u}\sum_{i=l+1}^{l+u}\hat{y}_i = \frac{1}{l}\sum_{i=1}^{l} y_i. \tag{2.48}$$

In the case of the $\hat{y_j}$, which is a discontinuous function (2.48), enforcing the constraint would be hard. The minimisation problem can be solved by a continuous optimisation technique. Thus, the imbalance constraint changes to:

$$\frac{1}{u}\sum_{i=l+1}^{l+u} f(x_i) = \frac{1}{l}\sum_{i=1}^{l} y_i. \tag{2.49}$$

The complete objective function for the TSVM includes three terms: A regularization term, the loss function for labelled training data which is called hinge loss function, and hat loss function for unlabelled training data.

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2} \parallel w \parallel^2 + C_1 \sum_{i=1}^{l} max(1 - y_i(w^T x_i + b), 0) \\
& \qquad + C_2 \sum_{i=l+1}^{l+u} max(1 - \mid w^T x_i + b \mid, 0), \\
\text{subject to} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l, \\
& \mid w^T x_i + b \mid \geq 1 - \xi_i, \quad i = l+1, \ldots, l+u, \\
& \xi_i \geq 0, \quad i = 1, \ldots, l, \\
& \frac{1}{u}\sum_{i=l+1}^{l+u} f(x_i) = \frac{1}{l}\sum_{i=1}^{l} y_i.
\end{aligned} \tag{2.50}$$

Unlike the SVM which has a convex optimisation problem, the TSVM objective is non-convex with multiple local minima. A learning algorithm can be stuck in a sub-optimal local minimum, and not find the global minimum solution. In order to avoid the TSVM getting

stuck in local minimum, many algorithms have been developed.

Vapnik [83] proposed the earliest semi-supervised Support Vector Machine approach, denoted by TSVM. The goal of the TSVM approach is to use the unlabelled patterns to find a better decision boundary that maximises the margin between the classes, compared to SVM. However, the solution for the objective function of TSVM is computationally difficult, because it is a non-convex function. Therefore, to reduce this computational cost, several other methods have been proposed. A novel heuristic approach was proposed by Joachims [40]. This method can iteratively solve the convex function problem for TSVM, but it can deal with just a few thousand patterns. Another method to fix the TSVM objective function is Laplacian SVM. This method was first introduced by Chapelle and Zien [16] and adds a regularization term to the objective function for both labelled and unlabelled patterns while Tsang and Kwok [82] proposed a new version of the Laplacian SVM, applying a sparse manifold regularisation. Chakraborty [14] introduced a Bayesian approach for semi-supervised Support Vector Machine training. The large margin space was found for binary classification problems by adding the regularisation term for the unlabelled patterns. In spite of the review many algorithms to the TSVM as a low density separation methods but these methods was very sensitive in the case of overlapping classifier as explained in Section 2.2.

Entropy regularization

In the previous section, the semi-supervised methods for non-probabilistic classification model based on the SVM were discussed. However, there are other alternative semi-supervised learning frameworks for probabilistic models that can compute the posterior probability of class membership $p(y|x)$, such as entropy regularization. In Section 2.2.2 we discussed semi-supervised learning with generative mixture models. One drawback of this approach is estimating the joint probability $p(x,y)$ which needs to estimate more parameters than discriminative model. Thus, in this section we discuss a particular discriminative model,

logistic regression, and then extend this to a semi-supervised learning framework via entropy regularization [32]. Given labelled training patterns, $X_l = \{(x_i, y_i)\}_{i=1}^{l}$, we assume a binary classification problem for simplicity where $x_i \in \mathscr{X} \subseteq \mathbb{R}^d$ is feature vector of a training patterns describing the *i*th example with class label $y_i \in \{0,1\}$. Using the logistic regression model the posterior probability of class membership $p(y|x)$ can be computed directly from the test data as follows:

$$\begin{aligned} p(y=1|x) &= \frac{1}{1+\exp(-f(x))}, \\ p(y=0|x) &= \frac{\exp(-f(x))}{1+\exp(-f(x))}. \end{aligned} \tag{2.51}$$

where $f(x) = w^T x_i + b$ is a linear function. If $f(x) \geqslant 0$ then $p(y|x)$ with the positive class otherwise $p(y|x)$ fall in the negative class. In addition, $p(y=0|x) = 1 - p(y=1|x)$ because the sum of the $p(y=1|x)$ and $p(y=0|x)$ must equal to 1. Each w vector and b are model parameters that can be learned by maximising the log-likelihood of the training data:

$$\log\ p(\mathscr{D}; w, b) = \sum_{i=1}^{l} \log\ p(y_i|x_i, w, b). \tag{2.52}$$

Typically the objective function contains the log-likelihood with a regularization penalty to prevent over-fitting, that is commonly referred to as L2 regularisation :

$$\log\ p(\mathscr{D}; w, b) = \sum_{i=1}^{l} \log\ p(y_i|x_i, w, b) - \lambda \sum_{i=1}^{l} w_i^2, \tag{2.53}$$

where the regularisation parameter, λ, is normally tuned through cross validation. The Logistic regression objective function can be maximized using gradient descent methods, where the gradient with respect to the θ_i parameter is given by:

$$\frac{\partial \log\ p(\mathscr{D};w,b)}{\partial w} = \sum_{i=1}^{l}(\phi(y_i - p(y_i|x_i,w,b))x_i - \lambda w_i. \tag{2.54}$$

The logistic regression classifier does not deal with the unlabelled data, so to make use of the unlabelled data, semi-supervised methods can be used. The entropy regularization semi-supervised algorithm can be used to exploit unlabelled data. However, semi-supervised learning cannot be used without an assumption. The entropy regularization can maximise posterior probability confidently if the decision boundaries lie in low density regions of the feature space i.e. both classes are well separated so that a posterior probability is either close to one or to zero which is most certain. It is obvious that the entropy is a measure of uncertainty, the most certain pattern can be find when entropy reaches zero. This happens in the case of the entropy regularization if the probability is either close to 1, or to zero. Thus, given both labelled and unlabelled training patterns entropy regularization can find parameters by maximising the following objective function:

$$\begin{aligned}\log\ p(\mathscr{D};w,b)) \quad = \quad & \sum_{i=1}^{l}\log\ p(y_i|x_i,w,b)) - \lambda_1\sum_{i=1}^{l}w_i^2 + \\ & \lambda_2\sum_{i=l+1}^{l+u}\Big(p(y_i|x_i,w,b)\Big)\log\Big(p(y_i|x_i,w,b))\Big).\end{aligned} \tag{2.55}$$

where the first two terms are the regularised log conditional likelihood of the labelled training patterns and the third term is the conditional entropy of the unlabelled training patterns. In this case, λ_1 and λ_2 can be tuned using cross-validation. Unfortunately, entropy minimization is a non-convex optimization problem and has local minima, and so it does not have a unique solution. Thus, some algorithms have been proposed to solve this problem.

Grandvalet and Bengio [31] proposed a novel semi-supervised approach for discriminative entropy regularization classification. The objective function can be optimised to find the parameters that maximise a regularised conditional log-likelihood of the labelled patterns

and the negative conditional entropy over the unlabelled patterns. The conditional entropy is optimised with a trade-off parameter to control the influence of the unlabelled patterns. Whereas the proposed methods experimentally obtain high classification performance, entropy regularization has some drawbacks. For example, difficulty in tuning the parameters and in the case of a small amount of labelled data, the entropy regularization assigns the same label to all labelled patterns. This method was extended to provide a new semi-supervised framework by Jiao et al. [39]. In this paper, the objective function consists of the conditional random field log-likelihood over labelled patterns that is penalised by the regularisation parameter, and the negative conditional entropy over the unlabelled patterns. This method introduced a trade-off parameter to control the influence of unlabelled patterns.

Mann and McCallum [50] introduced a novel algorithm, Expectation Regularization, for multi class classification problems. The objective function for this method consists of the conditional log-likelihood of the labelled patterns, a regularisation parameter, and the KL divergence for the predicted prior probability for each class. Therefore, the estimation of prior probability for each class should be known. The experimental results obtained suggest that Expectation Regularization can achieve improved performance compared to naïve Bayes, semi-supervised naïve Bayes, logistic regression, and entropy regularisation, even with only a small amount of labelled patterns.

2.2.6 Graph-based semi-supervised learning methods

Graph-based semi-supervised learning [9] usually employ undirected graph methods to construct graphs that connect similar data patterns of the labelled and unlabelled data set. Both labelled and unlabelled patterns are represented as nodes and edges the distances among the patterns, respectively. The edges are assigned with weights corresponding to their pairwise similarities. Thus, the graph can be represented by the weight matrix, W, which is symmetric.

If there is a connection between both x_i and x_j patterns then $W_{ij} > 0$ otherwise $W_{ij} = 0$. In addition, for these patterns w_{ij} is non-zero, each pair of patterns should be in the same class if they are connected by a strong edge (highly similarity). Given such a graph, the smoothness assumption is the main assumption in the semi-supervised learning graph-based methods. The common used similarity graphs are:

- k-nearest neighbourhood graph where $w_{ij} = 1$ if x_i is among the k-nearest neighbours of x_j or vice-versa and $w_{ij} = 0$ otherwise.
- ε-nearest neighbourhood graph x_i are connected by an edge with x_j if the distance $d(x_i, x_j) \leq \varepsilon$.
- The similarity graph with respect to the popular weight matrix is the Gaussian kernel or radial basis function (RBF) kernel,

$$W_{ij} \;=\; \exp\left(-\frac{\|x_i \,-\, x_j\|^2}{2\sigma^2}\right), \tag{2.56}$$

  where σ is the kernel bandwidth.

Given labelled, $X_l = \{(x_i, y_i)\}_{i=1}^{l}$, and unlabelled, $X_u = \{(x_i)\}_{i=l+1}^{l+u}$, training patterns for a binary classification problem, $y_i \in \{-1, +1\}$, where $x_i \in \mathscr{X} \subseteq \mathbb{R}^d$ is a feature vector of training patterns that describing the i^{th} example. Let $G = (V, E)$ be an image of the weighted graphs, where E is the set of edges and V is a set of image nodes, $V = V_L \cup V_U$. The goal of the graph based semi-supervised learning is to propagate the label information from V_L to the V_U of the graph. In this section, different graph-based methods are introduced that some of these discussions a based on that from Zhu and Goldberg [93].

Min-cut algorithm The first graph-based semi-supervised learning method is the Min-cuts algorithm proposed by Blum and Chawla [9]. Based on graphs, the min-cut algorithm

attempt to find minimum cuts in graphs that minimise the number of edges that are given different labels in order to learn from both labelled and unlabelled data. Mathematically min-cut algorithm minimises:

$$\begin{aligned} \min_{\text{f:f(x)}\in\{-1,1\}} \quad & \sum_{i,j=1}^{l+u} w_{ij} \left(f(x_i) - f(x_j)\right)^2 \\ \text{s.t.} \quad & f(x_i) = y_i \;\; for \;\; i = 1,\ldots,l. \end{aligned} \tag{2.57}$$

The min-cut algorithm is subject to fixing y_i for labelled patterns and results in hard labels for unlabelled patterns $y_i \in \{0,1\}$ where $i = l+1,\ldots,l+u$. The min-cut optimisation problem can be solved using max-flow algorithm for undirected graphs Blum and Chawla [9]. After the graph is built, the two special nodes $(v+, v-)$ called classification vertices are connected by edges of infinite weight to the labelled patterns, $w(v, v+) = \infty$, v is a positive pattern and $w(v, v-) = \infty$, v is a negative pattern. Finding the minimum cut in the graph is the main step of the min-cut algorithm. The graph can be cut in two parts that contain $V+, v+ \in V+$ and $V-, v- \in V-$ by finding and removing a set of edges with minimum total weight. We assign label $+1$ to unlabelled patterns from the set of the $V+$ classification vertices, and label -1 from the set of the $V-$ classification vertices. The drawback of the min-cut algorithm is that a single pattern may be left out in a partition after the cut the graph. In this case, a highly unbalanced partitioning can be obtained. Blum et al. [10] employed bagging to fix min-cut graph issue. Later, Zhu et al. [94] applied an iterative algorithms for graph based semi-supervised learning known as Label propagation. The intuition behind Label propagation is for each node to iteratively pass their label to the neighbour's nodes until convergence.

Harmonic Function

In order to relax the binary constraints $y_i \in \{-1,1\}$ for $i \in U$ in the min-cut algorithm to continuous labels, Zhu et al. [92] introduced a new method based on Gaussian random fields and harmonic functions. As a first step of relaxation, a harmonic function is a function that

was has given the weighted average of the value on the unlabelled data, however, the value y_i for labelled data is still fixed, $y_i \in -1,1$, $f(x_i) = y_i, \quad i = 1,\ldots,l$.
$f(x_j) = \frac{\sum_{k=1}^{l+u} w_{jk} f(x_k)}{\sum_{k=1}^{l+u} w_{jk}}, \quad j = l+1,\ldots,l+u.$
The harmonic function simply computes a continuous prediction function f on a given graph $G = (V,E)$ and assigns f to the unlabelled patterns which is the weighted average of its neighbours' value. Thus, it is known as as soft version of min-cut algorithm and it is the solution to the same optimisation problem in (2.57).

$$\begin{aligned} \min_{\text{f:f(x)}\in\mathbb{R}} \quad & \sum_{i,j=1}^{l+u} w_{ij} \big(f(x_i) - f(x_j)\big)^2 \\ \text{s.t.} \quad & f(x_i) = y_i \ \ for \ \ i = 1,\ldots,l \end{aligned} \tag{2.58}$$

Equation (2.58) can be minimized to find the optimal value for f in a continuous space which means $f(x)$ values fall between -1 and 1 by solving a linear equation. The unique value for $f(x)$ does not correspond to a label, therefore, it can be converted to a label by applying the thresholds which is a drawback for this methods:

$$\begin{aligned} y_i &= +1 \quad if \quad f(x) >= 0, \\ y_i &= -1 \quad if \quad f(x) < 0. \end{aligned} \tag{2.59}$$

Zhu et al. [92] used a random walk to interpret the harmonic function. A transition probability matrix was used to propagate labels, which probability random move from vertex i to j. $P(i/j) = \frac{w_{ij}}{\sum_k w_{ik}}$.
The easier way to obtain the closed-form solution is the harmonic function with a Laplacian matrix which basically is a matrix notation for the Laplacian matrix. W is a weight matrix for both labelled and unlabelled data, D is a diagonal matrix,

$$D_{ij} = \begin{cases} \sum_{k=1}^{l+u} w_{jk} & : i = j \\ 0 & otherwise. \end{cases}$$

Then the graph Laplacian matrix L is given as follows:

$$L = D - W \tag{2.60}$$

Now, the regularisation term (2.58) can be written as follows:

$$\sum_{i,j=1}^{l+u} w_{ij} \big(f(x_i) - f(x_j)\big)^2 = f^T L f, \tag{2.61}$$

where $f = (f(x_1), \ldots, f(x_{l+u}))$. In order to find labels for unlabelled patterns, we can partition the f vector into (f_l, f_u) and Laplacian matrix into sub-matrix respectively:

$$\begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix}$$

Let $y_i = (y_1, \ldots, y_l)^T$ then using Lagrange multipliers with matrix algebra to solve the optimisation problem,

$$\begin{aligned} f_l &= y_i, \\ f_u &= -L_{uu}^{-1} L_{ul} y_l, \end{aligned} \tag{2.62}$$

Zhou et al. [88] used the normalized graph Laplacian to propagate labels.

$$\tilde{L} = I - D^{-\frac{1}{2}} W D^{\frac{1}{2}} \tag{2.63}$$

Then, the close form solution for (2.61) can be obtain by setting to zero the partial derivative w.r.t to the regularisation matrix L.

These methods are transductive that normally use the structure of the graph to propagate labels from the labelled data to the unlabelled in the graph. Belkin et al. [8] proposed a

manifold regularisation method called the Laplacian Support Vector Machine (LapSVM). This approach is the inductive methods that has able to predict labels for the unseen patterns into the graph, f is defined over the whole feature space, $f : X \rightarrow R$. The optimisation manifold regularisation problem can reprsent as follows:

$$\min_{\mathrm{f:f(x)}\in\mathbb{R}} \qquad \lambda \cdot f^T L f + \lambda_2 \parallel f \parallel^2 \tag{2.64}$$

where $\lambda_1, \lambda_2 \geq 0$. λ_1 is a regularisation parameter to f in order to be smooth with respect to the graph Laplacian. The second regularised term is λ_2 which enforces smoothness in order to improve generalization classification performance. The complexity of the Laplacian Support Vector Machine was reduced by Melacci and Belkin [53].

2.3 Active learning

Supervised learning often provides poor classification performance with a small amount of labelled data, $X_l = \{(x^{(i)}, y^{(i)})\}_{i=1}^{l}$, but it is possible to achieve better performance by adding more labelled data. However, labelling of a large amount of unlabelled data by an expert (oracle) can be time-consuming and expensive. In this case, involving the unlabelled data, $X_U = \{x^{(i)}\}_{i=l+1}^{l+u}$, in the learning process might help, because the unlabelled data can obtained cheaply. One possible method that we can rely on is active learning [72] that requests the human expert to obtain the true labels, y^*, for most informative unlabelled training patterns, $x^* \in U$, by asking queries. Thus, the labelled training patterns, X_l, can be expand and may improve the generalisation performance.

In this framework, the active learner selects only the most informative few unlabelled patterns and ask the oracle to label the new patterns instead of labelling all unlabelled data. Figure 1.6 in chapter 1, shows a simple two-dimensional illustrative example where the active learning

can improve generalisation performance by selecting the most informative pattern to be labelled by the expert. The choice of the unlabelled patterns for labelling is the important step in active learning, known as query selection, and this selection is normally done through query strategies. We explain the general methods for both query selection and strategies in the following sections.

2.4 Active learning scenarios

The first critical part in the active learning algorithms is the query selection part that are explained in this section. In active learning generally, three main query selection scenarios exist to choose unlabelled patterns: membership query synthesis, stream-based, and pool-based scenarios.

2.4.1 Membership query synthesis

Membership query synthesis is an early query selection scenario proposed by Angluin [2]. The goal behind this scenario is for the leaner to generate queries artificially based on the input space setting rather than drawn from the underlying distribution, and to request their labels from the oracle. This form of query synthesis has been used in different practical real world applications, such as robotics [19] and handwritten character recognition [44]. This query strategy is an effective scenario for the finite unlabelled patterns set Angluin [3], but in some cases it has serious problems. For example, Lang and Baum [44] applied the query synthesis scenario with a human expert to learn a neural network classifier to classify handwritten digits. They illustrated that the images generated by the query synthesis might not be recognised by human experts. The queries synthetic were artificial combinations of digits images that were meaningless. Thus, the experts have difficulty labelling the synthetic

patterns. In addition, applying the query synthesis is impossible for text classification because the patterns would meaningless if generated by the query synthesis [47, 80, 52].

2.4.2 Stream-based (Selective Sampling)

Atlas et al. [5] introduced the stream-based active learning scenario, which is an alternative to the query synthesis approach. In this scenario, rather than constructing the synthetic queries, on each trial the leaner selects one unlabelled pattern from the stream of unlabelled patterns. The data stream is a sequential data that arrives at a system continuously, therefore, an infinite amount of unlabelled data is made available. Then, the leaner request labels for this pattern or it is discarded. In general, the capacity of this selective sampling is better than membership query synthesis because the queries are drawn from the natural distribution, $P(x)$, and it might have natural meaning, which does not confusing the human experts to labelling patterns.

2.4.3 Pool-based selection

Pool-based active learning was proposed first by Lewis and Gale [47]. In this scenario, a large pool is available, which consists of a small set of labelled data, L, and a large set of unlabelled data, U, for training. Essentially, the active learning algorithm attempts to rank the unlabelled patterns based on informativeness and selects the top k patterns. Then the classifier is trained after adding additional labelled patterns on each iteration. The drawback of the Pool-based scenario is that it selects patterns from a dataset, which has finite number of patterns. In this case, the Pool-based scenario does not deal with dynamically datasets, such as stream datasets. Lewis and Gale [47], Tong and Koller [80], McCallumzy and Nigamy [52] used a pool-based approach for the text classification problem, because for text classification, a large pool of documents already exists.

Both stream-based and pool-based query selection methods have access to a large number

of patterns. The pool-based method can eventuate the entire pool and select k top most informative patterns, whilst the stream-based method can sequentially go through the dataset and select only one pattern each time. On the other hand, both synthetic query and pool-based query selection methods rank the unlabelled data according to the most informative one and select best patterns, but the sources of selecting patterns are different. In general, the most common query selection method is pool-based query approach. In this thesis the pool-based sampling is investigated.

2.5 Active learning query strategies

In the previous section, three different query selection scenarios have been explained which form a critical part in the active learning process. In this section, we introduce a different query criteria, denoted by, $\psi(.)$, for the evaluation most informative patterns in pool-based active learning.

2.5.1 Uncertainty sampling

Uncertainty sampling Lewis and Gale [47], is the most commonly used active learning query strategy where the learner selects the closest patterns to the current classification decision boundary, which is the most uncertain pattern in the unlabelled data. Typically, a single classifier is trained with labelled patterns, X_l, and subsequently this classifier is used to classify the unlabelled patterns, X_u. Then, the output of the classifier can be used as a measure of uncertainty. More details of the uncertainty sampling can be found in Figure 1.5. The uncertainty sampling query strategy requires the classifier to produce a confidence score for the prediction. For example, Lewis and Gale [47] used a probabilistic classifier to produce a confidence score $P(y|x)$ for a pattern x when it is predicted as class y. In the case of a probabilistic classifier for a binary classification problem, the uncertainty sampling strategy

queries the pattern with predicted posterior probability nearest 0.5. In general, the uncertainty sampling query strategy can use different uncertainty measures such as **least confidence**:

$$\psi_{LC}(x) \;=\; \arg\max_{x} \left(1 \;-\; P(\hat{y}|x;\theta)\right). \tag{2.65}$$

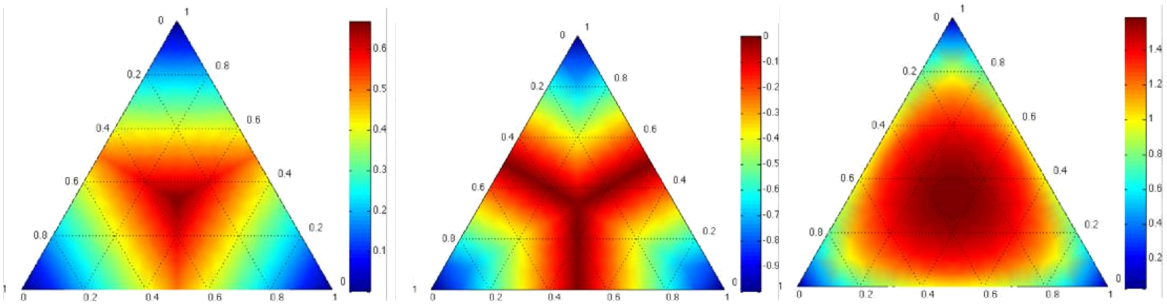
where $\hat{y} \;=\; argmax_y \, P_\theta(y|x)$, is the predicted class that is equivalent to the highest posterior probability class membership. Another uncertainty measure for selecting unlabelled patterns is the **smallest margin** between the two highest posterior probabilities of class membership, which is called margin sampling:

$$\psi_{M}(x) \;=\; \arg\min_{x} \left(P_\theta(\hat{y_1}|x) \;-\; P(\hat{y_2}|x;\theta)\right). \tag{2.66}$$

where $\hat{y}_1$ and $\hat{y}_2$ are class labels that have the highest posterior probability class membership. However, the most common uncertainty measure that is generally used by uncertainty sampling strategy is the **entropy measure**:

$$\psi_{H}(x) \;=\; \arg\max_{x} \;-\; \sum_{i} P(y_i|x;\theta) \;\log\; P(y_i|x;\theta). \tag{2.67}$$

where y_i is calculated over all class label through the posterior probability class membership. For a binary classification problem, These uncertainty measures are equivalent to selecting the unlabelled patterns. However, the have different result for multi-class classification problem, as shown in Figure 2.12.



(a) least confidence (b) smallest margin (c) entropy measure

Fig. 2.12 Heatmaps illustrating the query behaviour of common uncertainty measures in a three-label classification problem, (from [72]).

Settles [72] shows the relation between all uncertainty measures for a three classes problem in Figure 2.12. As can be seen, the least informative patterns are at the corners of the triangle, where the posterior probability class membership one of the classes is approximately one and zero for other classes. While, the centre of the triangle is a place for those patterns that the posterior probability class membership is uniformly distributed which are the most informative patterns. These uncertainty measures, are similar only in the centre and corner of triangle, but they are differ in the rest part of the triangle. For example, the least confident and margin measures select these patterns if the classifier cannot distinguish between the remaining two classes. In contract, the entropy measure, does not select these patterns where only one of the labels is highly unlikely.

2.5.2 Query-By-Committee

A second active learning query strategy was proposed by Seung et al. [74], which is known as Query-By-Committee (QBC). In this method a leaner creates a "committee" from the classifier, $C = \theta^{(1)}, \theta^{(2)}, ..., \theta^{(c)}$, based on the disagreement measure in order to select the unlabelled patterns. Then, these unlabelled patterns will be select for the labelling if they have the biggest classification disagreement among the committee based the **vote entropy** or **Kullback-Leibler** divergence that are more common methods for estimating the

disagreement.

Settles [72] illustrates the version space in the Query-By-Committee strategies, as shown in Figure 2.13. The intuition behind query-by-committee is that all committee member participate to vote on the selected patterns for labelling such that the uninformative query which has most disagree, where the version space is consist of the set of hypotheses from the training labelled patterns.

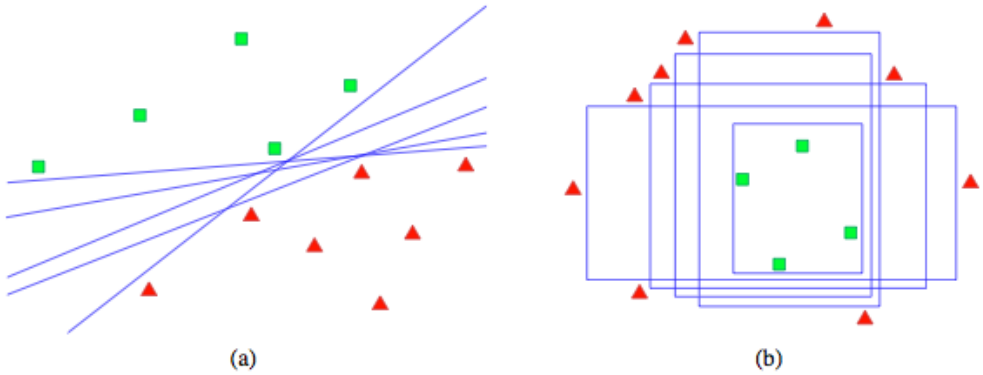


Fig. 2.13 Version space examples for (a) linear and (b) axis-parallel box classifiers. (from [72])

The goal of the Query-By-Committee strategy to reduce the size of the version space through labelling of an few patterns is possible. This is dependent on the properties of the version space and the committee member. In general, the two most common ways of measuring the disagreement have been proposed that are **vote entropy** and **Kullback-Leibler (KL)** divergence. The **vote entropy** can be defined as:

$$\psi_{VE}(x) = argmax_x - \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}. \tag{2.68}$$

The second measure for the level of disagreement is the average **Kullback-Leibler (KL) divergence**:

$$\psi_{KL}(x) = argmax_x \frac{1}{C} \sum_{c=1}^{C} D(P_{\theta^{(c)}} \,||\, P_C). \tag{2.69}$$

where:

$$D(P_{\theta^{(c)}} \,||\, P_C) \;=\; \sum_i P(y_i|x;\theta^{(c)}) \,\log\, \frac{P(y_i|x;\theta^{(c)})}{P(y_i|x;C)} \tag{2.70}$$

$\theta^{(c)}$ can be recognised as one of the model in the committee member and C is the committee as a whole. $P_C(y_i \setminus x) \;=\; \frac{1}{C} \sum_{c=1}^{C} P(y_i|x;\theta^{(c)})$ is the consensus probability that y_i is the correct label.

Outliers are the major problem for the query-by-committee strategy that prevent it reducing the version space. However, there some other uncertainty measure can fix this issue.
Ramirez-Loaiza et al. [65] presented an empirical evaluation of different active learning strategies. They attempted to find a suitable active learning strategy for two probabilistic classification models, naïve Bayes and logistic regression, with different performance measures. They perform a large scale benchmark on ten datasets, with large dimensional features and number of patterns. In that paper two sample-based query strategies, uncertainty sampling (UNC), and query-by-committee (QBC), were compared for the selection of the most informative unlabelled instances. Then, each of the UNC and QBC query strategy were compared with the baseline query strategy, random sampling (RND), which selects patterns randomly from unlabelled. The paper concludes that using a single classifier or single performance measure can be misleading in an active learning strategies.
McCallum et al. [51] was first proposed the idea of combining active learning with semi-supervised learning, which interesting topic in this thesis. The author combining the Query-By-Committee with EM algorithm under name CO-EM to assign new label for unlabelled patterns, using Bayesian classifier as a base. However, this idea was extended to the CO-EMT by the Muslea et al. [54]. The deference between this idea with McCallum et al. [51], they used multi-view learning for both active and semi-supervised learning. Zhu et al. [92] used different semi-supervised learning. They combining active learning with graph-based method. However, rather than these method are effect methods by combining active learning with

semi-supervised learning but almost the implementation on the artificial data set or 20 news group datasets.

2.5.3 Expected error reduction

The expected error reduction is another query strategy proposed by Roy and McCallum [69] for evaluating which unlabelled patterns to select for labelling in active learning algorithms. This method attempts to reduce expected error, which reduce the overall uncertainty of the model by directly minimising error on the available data to select the unlabelled patterns confidently. If the most informative pattern is labelled by the oracle and added to the training set, how much its generalization error would be reduced? Technically, this strategy is started by using either log loss or zero-one loss of a model trained on $L(x_i, y_i)$, where, y_i is a true label for a pattern, x_i. The pattern is selected for labelling with minimal expected error. In the case of the expected zero-one loss:

$$x^*_{0/1} \;=\; \arg\min_{x} \sum_{i} P(y_i|x;\theta) \left(\sum_{u=1}^{U} 1 \;-\; P(\hat{y}|x^{(u)};\theta^{+\langle x,y_i\rangle}) \right) \tag{2.71}$$

After training the labelled patterns with new selected unlabelled patterns then the model $\theta^{+\langle x,y_i\rangle}$ would be achieved.

2.5.4 Density-Weighted methods

The previous method, expected error reduction, utilises all the available data in order to reduce the generalization error. In this section another query strategy approach is shown that focuses on the individual patterns rather than all the data, which is known as the density-weighted approach, Zhu et al. [91]. Both active learning query strategies, US and QBC, may query outliers as uncertain pattern for US and controversial for QBC. An outlier is a pattern that behaviour is far from the other patterns distribution. Figure 2.14 shows the linear

classifier separate the binary classification problem, which red and blue classes, where the data consist of both labelled and unlabelled patterns. As we can see, the green pattern is the most informative unlabelled pattern regarding uncertainty sampling, as it is the closest pattern from the decision boundary. Selecting the green pattern however provides a poor direction for the selection of further unlabelled training data to be labelled by the oracle and does not improve the classier because it is an outlier. In this case a density estimator might help because it is a modelling the input distribution while selecting unlabelled patterns. Thus, the most informative patterns are the patterns selected from the same underlying distribution rather than the patterns that are most uncertain.

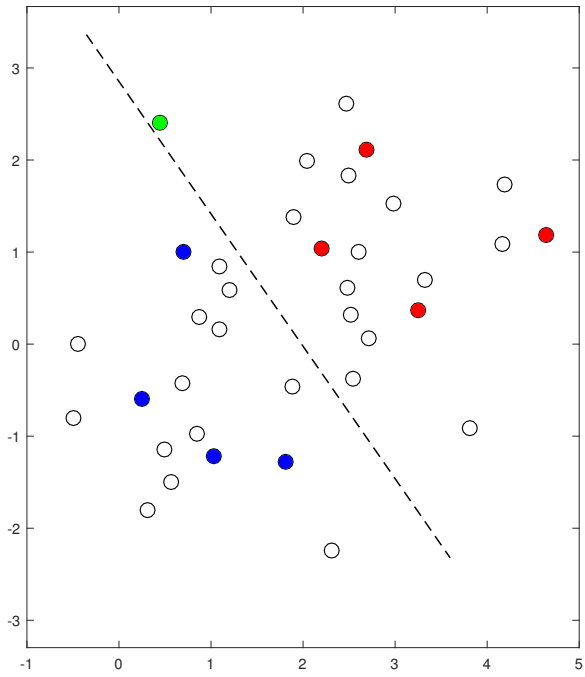


Fig. 2.14 An illustration of selecting the green pattern, which is closest pattern from decision boundary, using uncertainty sampling query strategy. Uncertainty sampling is a poor strategy and does not improve the performance of the classifier, because distribution of green patterns is not illustrative of the distribution of the other patterns. [After Settles [72]]

The general density-Weighted method is proposed by Settles and Craven [73], which is known as information density (ID) technique. In this technique, the informativeness pattern x_H^* is weighted by its avenge similarity to the unlabelled patterns U, where the relative importance of the density can be controlled by parameter β,

$$x_{ID}^* = \arg\min_x x_H^* \left(\frac{1}{U} \sum_{u=1}^{U} sim(x^*, x^{(u)}) \right)^{\beta}. \tag{2.72}$$

2.6 Performance Evaluation

In this section, we describe metrics and methods used to evaluate the performance of classification algorithms. A classifier usually learns from a set of labelled training patterns and the performance of a classifier is evaluated on a set of unseen (test) labelled patterns, which are never used during training.

The outcome of a classifier on a test data are stored in a confusion matrix, which is a base for calculating various performance measures. Suppose we have a binary classification problem where one class is known as positive and the other as negative. Then the actual and predicted label outcomes for this particular classifier can be illustrated by plotting a confusion matrix. Table 2.1 shows a confusion matrix for a binary classification problem.

If the outcome from a prediction is positive and the actual label is also positive, then it is

Table 2.1 Confusion matrix for a binary classification task, the actual class is given by the columns while the predicted class is given by the rows

		actual class		
		p	n	total
predicted class	p'	TP	FP	P'
	n'	FN	TN	N'
	total	P	N	

called a true positive (TP); however if the actual label is negative then it is said to be a false positive (FP). Conversely, a true negative (TN) has occurred when negative examples are

correctly classified as negative, and a false negative (FN) is when a positive examples is incorrectly classified as negative. These four values can be combined to evaluate a classifiers performance. Generally, high performing classifiers will have relatively few false positives and false negatives.

Given the numbers from the confusion matrix, several performance measures can be calculated, such as classification accuracy, error rate, and recall. In this thesis, classification performance was measured using the error rate. It is the fraction of the total number that are incorrectly classified (2.73). For a particular pattern, if the predicted class is equal to the actual class then the incorrect classification is 0, otherwise is 1.

$$Error\ rate\ =\ \frac{FP+FN}{TP+FP+FN+TN} \tag{2.73}$$

The error rate classification measure typically is most useful as a measure in the case of balanced data that a number of patterns from each class are approximately equal compare the other classes. While, the error rate might not be appropriate when the data is imbalanced, when the data consists of a large percentage of a single class. It has a poor evaluation measure in this case because a classifier may assign the label of the majority class found in the training patterns which obtain low error rate[17].

A Classification performance can be representation via a learning curve Perlich [62]. The learning curve generally refers to a plot of the generalization performance on a training and testing set of a machine learning model over a varying number of training patterns. A particular learning curve is the error rate learning curve would be considered in this thesis. Error rate learning curves ware a plot of the test set error rate of a classifier on a testing set over a function of number of training patterns.

One technique is used for partitioning data is called hold out. This method randomly partitions a dataset into two disjoint sets where the first is used as training data and the second as test data. In each partitions, the error rate with varying numbers of labelled training patterns

is recorded on approximately equal intervals on a logarithmic scale. Using a logarithmic scale x-axis instead of linear scale can be helpful since it covers a large range of quantities data and we want to focus on scenario with few labelled patterns. Then the learning curve is plotted for the error rate as a function of the number of labelled training patterns on a logarithmic x-axis that the patterns is added in a random way. Then, the area under the error rate learning curve (AULC) was computed to compare the error rate learning curve of different classifiers.

Algorithm 4 shows the steps of finding area under the error rate learning curve in each partition. The Algorithm start by introducing the training and test set in step1. Then the varying numbers of labelled training patterns is initialised using a logarithmic scale x-axis in step2. Later, step 3-9 shows the learning and evaluating a classifier. In step10, the error rate as a function of the number of labelled training patterns is calculated. Then, the trapezoid method is used to find approximating the area under the error rate learning curve. As can be seen in step13-16, the area under the error rate learning curve between two labelled set size is approximately equal to the average of the error rate as a function for this two labelled set size multiply by difference of $\log_2$ for this two labelled set size. The reason for using $\log_2$ to make approximately equal interval between labelled set size. Assuming x_1 and x_2 are two labelled set size respectively and $f(x_1)$ and $f(x_2)$ are their error rate as a function for x_1 and x_2. Then, the AULC between x_1 and x_2 can be calculate as follows:

$$AULC \approx \left(\frac{f(x_1)+f(x_2)}{2}\right) \times \Delta(x) \tag{2.74}$$

where $\Delta(x) = \log_2(x_2) - \log_2(x_1)$. Finally in step17, the $AULC$ for whole learning curve in this trail is approximately calculated by integrating the result obtained in step17, AUC.

In order to reduce the possible bias introduced by the partitioning of training set and test set, usually multiple trials of the experiments are used with a different random partition of the training set and the test set according to the held-out method. I this thesis, 100 trials was

Algorithm 4 Find area under the error rate learning curve

Inputs:
Let X: dataset
Split X into training set (X_{tr}) and test set (X_{ts})
n: number of training pattern
m: number of test pattern
n_0: number of initial labelled training pattern
L: learning algorithm
Initialise:
$l = \text{unique}([\text{round}(10^{[\log_{10}(n_0):\, 0.05:\, \log_{10}(n)\,]})\ n])$
for t from 1 to length(l) **do**
$X_l \leftarrow \left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{l_t}$
$\hat{\theta}^{(t)} \leftarrow argmax_\theta\ P(X_l, Y_l \mid \theta)P(\theta)$
    for j from 1 to m **do**
        $\hat{y_j} \leftarrow \arg\max_c\ p(y = c|X_{ts}; \theta)$
        $e_j \leftarrow \phi(\hat{y_j} \neq y_j)$
    end for
$err_t \leftarrow \frac{1}{m}\sum_{j=1}^{m} e_j$
$t \leftarrow t + 1$
end for
for k from 2 to length(l) **do**
$AUC_{k-1} \leftarrow \left(\frac{err_{t-1} + err_t}{2}\right) \times (\log_2(l_{t-1}) - \log_2(l_t))$
$t \leftarrow t + 1$
end for
$AULC \leftarrow \sum_{k=1}^{t-1} AUC_k$

used for each dataset. Finally, the test statistic is provided by averaging the AULC over all trials associate with standard error.

2.7 Statistical comparisons of classifiers

To test the generalization performance of classifiers involved in the experiments, two procedures have been used to compare the classification algorithms. The former is whether there is a statistically significant difference in the performance of a pair of classifiers over various benchmark datasets or between the performance of two classifiers on a single benchmark

dataset, where multiple training/test partitions are used. Later whether there is a statistically significance difference between the k classifiers over multiple benchmark datasets.

Comparison of a pair of Classifiers: The Wilcoxon signed-ranks test [85] is a non-parametric approach used to determine whether there is a statistically significant difference between the performance of two classifiers over various multiple benchmark datasets or over a single benchmark dataset that using results obtained from independent test sets. This approach is based on the ranks of the differences in performance of two classifiers for each dataset.

Suppose we have C-matrix classification performance, which is k by n where $k = 2$ is the number of classifiers and n is the number of trials from independent test sets. Let C_{i1} and C_{i2} be classification performances on the i^{th} trial for a single dataset. Then the idea behind the Wilcoxon signed-ranks test is to rank the absolute values of D_i, when D_i is the difference between two classifiers performance on the i^{th} trial, $D_i = |C_{i1} - C_{i2}|$. The rank starting from smallest to largest rank, then calculate the sum of ranks for positive R^+ and negative R^- differences separately. Then the smallest sum of the ranks is considered as the test statistic. It is approximately normally distributed For large number of benchmark datasets. Later we will find a critical value at the level of significance $\alpha = 0.05$. If the test statistic is less than or equal to the critical value, that means the null hypothesis H_0 can be rejected, where H_0: that the two set of classifier results have equal median ranks. The alternative hypothesis test is H_1, where H_1: that the two set of classifier results have different median ranks. In order to compare two classifiers over multiple benchmark datasets, the Wilcoxon signed-ranks test is used that instead of ith trails we have ith benchmark datasets.

The Wilcoxon signed-rank test is a more appropriate test for comparing two classifiers as it assumes independence between the performance measures. In addition, it does not require that the difference in the performance of a pair of classifiers are commensurable, because communicability of the differences is difficult across multiple benchmark datasets. Moreover,

the test does not assume that the difference in the performance of a pair of classifiers is normally distributed, which is more useful when the number of benchmark datasets are small. The Wilcoxon signed-rank test is robust to outliers, that skew the performance measures have less affect on this test. The Wilcoxon signed-rank test assumes the distributions of the differences must be symmetrical. In other words each side of the median must have a similar shape. If this assumption is violated, it can affect the power of Wilcoxon signed-rank test.

Comparisons of Multiple Classifier The Friedman test [29] is a non-parametric alternative to analysis of variance ANOVA. This test is to determine whether there is a statistically significant difference between the average ranks of k classifiers, where $k > 2$. The null hypothesis H_0 assumes that the average ranks R_i over multiple datasets will be equal against the alternative hypothesis H_1 that at least one of the classifiers has different average ranks. Given two matrices, C-matrix classification performance, which is k by n where $k > 2$ is a number of classifier and n is the number of benchmark datasets and R, which is a matrix of average ranks, which is k by n as well. Then the Friedman statistic Q can be calculated:

$$Q = \frac{12n}{k(k+1)} \cdot \left[\sum_{j=1}^{k} \bar{R}_j^2 - \frac{k(k+1)^2}{4} \right],$$

The Q statistics is approximately distributed according to a Chi-squared distribution with $(k-1)$ degrees of freedom. The Q statistics is sufficiently use when the number of benchmark datasets, n, and classifiers, k, are large enough (as a rule of a thumb, $n > 10$ and $k > 5$). However, Demšar [25] notes that this calculation is often conservative for small number of benchmark datasets, and proposes using the following statistic:

$$F = \frac{(n-1)Q}{n(k-1)-Q},$$

which follows an F distribution with $(k-1)(n-1)$ degrees of freedom. The null hypothesis, H_0, will be rejected if the value of this statistic greater than critical value

that means there is a statistically significant difference at least between two classifiers. If a significant difference is found then post-hoc test is applied to determine statistical significance between pairs of classifiers. The Nemenyi test is used to calculate the critical difference, *CD*,

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}}$$

where q_a depends on both α and k. The Nemenyi test result can be visually demonstrated by critical difference datagram. Figure 2.15 is an illustrative example for representing The Nemenyi test result for three classifiers. The two classifiers are significantly different if their average ranks is differ by more than the critical difference. While, these classifiers are linked by the bar indicate that there are no statistically significant differences between the means ranks for these classifiers and the differences between the means ranks is less than the critical difference. For example, in Figure 2.15 the Classifier 1 is statistically significantly different compare to the Classifier 3. In addition, the statistical different between the Classifier 1 and Classifier 2, and the statistical different between the Classifier 2 and Classifier 3 is not significantly difference.

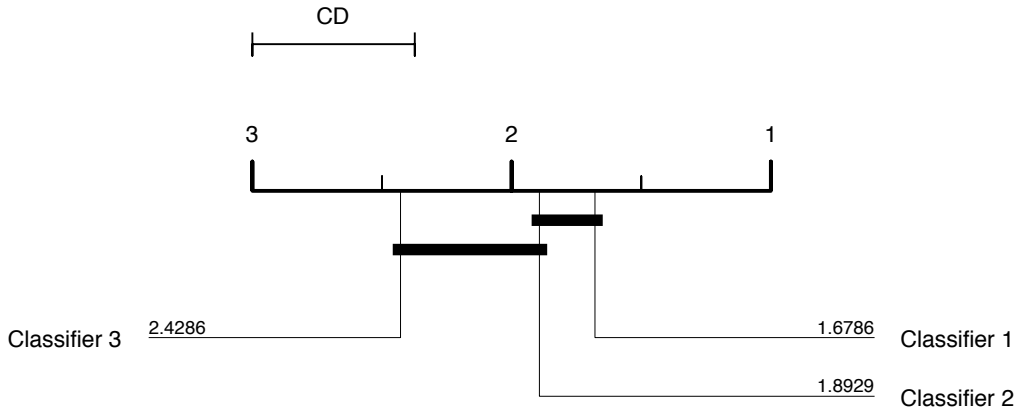


Fig. 2.15 Illustrative example for critical difference diagram

2.8 Model selection

Model selection is the search for optimal hyper-parameter values where the model is defined by a set of model parameters and hyper-parameters such as finding parameters in chapter 3. The simplest approach to model selection criterion is a validation-set approach that randomly partitions the data in to three sets ; the training set C_{train} is used to estimate model parameters, an appropriate performance statistic is evaluated on the validation set C_{val} to determine the optimal hyper-parameter values, finally, test set C_{test} is used to estimate the generalisation performance of the model. If y_i is the actual class for a particular pattern, x_i and $f(x_i)$ is their predicted class, then the error rate measure for the validation set C_{val}, which consist of n_{val} patterns, can be calculate as follows:

$$Err_{val}(h) = \frac{1}{n_{val}} \sum_{i \in C_{val}} \phi(h(x_i) \neq y_i). \tag{2.75}$$

where $\phi(z) = 1$ if z is true, and $\phi(z) = 0$ otherwise.

However, for the semi-supervised and active learning problem, the amount of labelled data is limited, separating train and validation set is wistful. Therefore, the cross-validation approach is better.

Cross-validation is the most commonly used approach to model selection, [78]. In k-fold cross-validation, the labelled training data are randomly partitioned into k approximately equal sized non-overlapping subsets $C_1, C_2, ..., C_k$, where C_k denotes the indices of the observations in part k, As can be seen in Figure 2.16. There are nk observations in part k, then $nk = \frac{n}{K}$. The k - 1 sets are used for training each model and the remaining k^{th}fold set, which is a validation set, is used for evaluating performance.

$$Err_k(h) = \frac{1}{nk} \sum_{i \in C_K} \phi(h(x_i) \neq y_i). \tag{2.76}$$

The average performance on the test sets is used to estimate the model performance, and this can be repeated and averaged over multiple random partitions. The main advantage of the cross-validation approach is use all available labelled patterns as both training and test patterns, however the achieved cross-validation can suffer when only small amounts of labelled patterns are available [77].

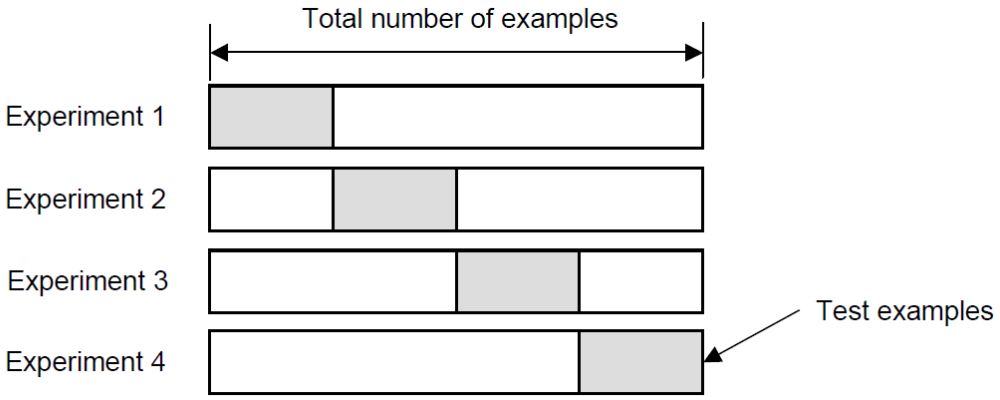


Fig. 2.16 The cross-validation model selection method.

The extreme case of k-fold cross-validation is leave-one-out cross-validation [43] that is normally used for extremely small datasets, in which each subset contain only one pattern i.e. $k = l$ where l is the labelled training patterns, Figure 2.17. The result obtained from leave-one-out cross-validation procedure almost unbiased estimator but it is computational expensive and it has high variance which can lead to over-fitting. In general, the available model selection approach for estimating model performance suffer when there are not many labelled patterns.

$$Err_{loocv}(h) = \frac{1}{l}\sum_{i \in C_K} \phi(h(x_i) \neq y_i). \tag{2.77}$$

$\phi(z) = 1$ where z is true, and $\phi(z) = 0$ otherwise.

So, in this thesis we focused on leave-one-out cross-validation when only a few labelled

patterns is available, but when the amount of labelled patterns is sufficiently large we used k-fold cross-validation for model selection.

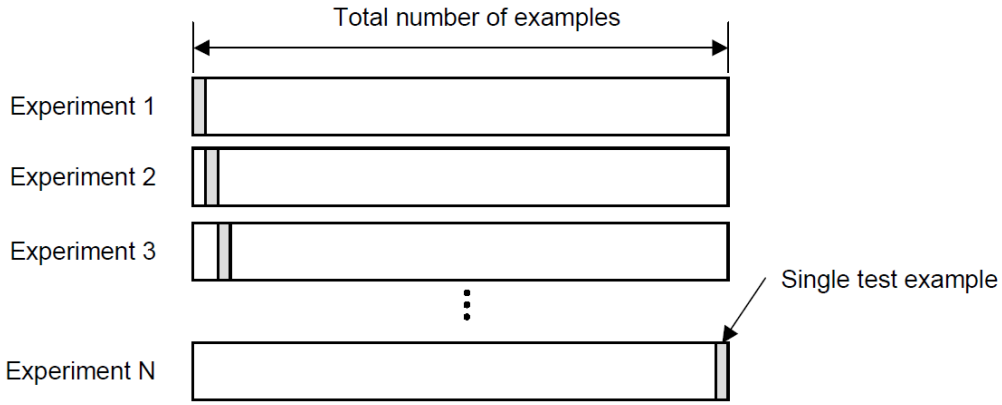


Fig. 2.17 The leave-one-out cross-validation model selection method.

Chapter 3

Benchmarking the Semi-supervised naïve Bayes classifier

In this chapter, we investigate whether an expectation maximization scheme improves a naïve Bayes classifier in a semi-supervised learning context, through experimentation with 36 discrete and 20 continuous real world benchmark UCI datasets. Rather surprisingly, we find that in practice self training generally makes the classifier worse. The cause for this detrimental effect on performance could either be with the self training scheme itself, or how self training works in conjunction with the classifier. Our hypothesis is that it is the latter cause, and the violation of the naïve Bayes model assumption of independence of features means predictive errors propagate through the self training scheme. To test whether this is the case, we generate simulated data with the same feature distribution as the UCI data, but where the features are conditionally independent. Experiments with this data demonstrate that semi-supervised learning does improve performance, leading to significantly more accurate classifiers. These results demonstrate that semi-supervised learning cannot be applied blindly without considering the nature of the classifier, because the assumptions implicit in the classifier may result in a degradation in performance. When the assumption of the classifier is not valid the self training iteratively makes the classifier worse.

3.1 Experiments

3.1.1 UCI benchmark datasets

Datasets and experimental design

To evaluate the performance of NB compared to the SSNB classifier, we performed two sets of experiments for discrete and continuous features respectively. The first experiment is based on 36 discrete benchmark datasets and a summary of the properties of these datasets is shown in Table 3.1. The second experiment used the 28 continuous benchmark datasets. Table 3.2 shows the information about the selected datasets for the second experiment. All datasets were taken from the UCI machine-learning repository [6], excluding (`monk1-corrupt`, `monk1-cross`, `monk1-local`, `monk3-cross`, `monk3-local`) that are available in the SGI[1] repository. Both Table 3.1 and Table 3.2 show the information about the number of features, patterns, and frequency of each class for each dataset. As we can see, they differ in the number of features, patterns, and classes. The smallest datasets had 32 patterns and the largest had 105908, and the number of features ranges from 3 to 180. Both experiments include binary and multi-class classification problems where the number of classes varies between two 2 to 26.

#	Dataset	Features	Patterns	Classes
1	audiology	69	226	24
2	balance scale	4	625	3
3	blogger	5	100	5
4	breast cancer	9	286	2
5	breastw	9	699	2
6	car	6	1728	4
7	DNA	180	3186	3
8	flare1	12	323	6
9	flare2	12	1066	6
10	hayes roth	4	160	3
11	house votes	16	435	2

Continued on next page

[1] https://www.sgi.com/tech/mlc/db/

#	Dataset	Features	Patterns	Classes
12	kr-vs-kp(chess)	36	3196	2
13	led7	7	3200	10
14	led24	24	3200	10
15	lung-cancer	56	32	3
16	lymphography	18	148	4
17	marketing	13	8993	9
18	monk1-corrupt	6	432	2
19	monk1-cross	5	432	2
20	monk1-local	17	432	2
21	monk1	6	432	2
22	monk3-cross	7	432	2
23	monk3-local	17	432	2
24	monk3	6	432	2
25	mushroom	22	8124	2
26	nursery	8	12960	5
27	primary-tumor	17	339	21
28	promoters	57	106	2
29	shuttle-landing-control	6	253	2
30	soybean-small	35	47	4
31	soybean-large	35	683	19
32	splice	60	3190	3
33	threeOf9	9	512	2
34	titanic	3	2201	2
35	xd6	9	512	2
36	zoo	16	101	7

Table 3.1 Attributes of the UCI datasets with discrete input features

#	Dataset	Features	Patterns	Classes
1	banknote	4	1372	2
2	blood-transfusion	4	748	2
3	breast-cancer-continuous	9	683	2
4	climate model simulation crashes	18	540	2
5	glass	9	214	6
6	haberman	3	306	2
7	ionosphere	34	351	2
8	iris	4	150	3
9	letter	16	20000	26
10	liver-disorder	6	345	2
11	magic04	10	19020	2
12	magic04	166	476	2

Continued on next page

#	Dataset	Features	Patterns	Classes
13	new-thyroid	5	215	3
14	pendigits	16	10992	10
15	sleep	13	105908	5
16	vehicle	18	846	4
17	vowel	10	528	11
18	waveform-noise	40	5000	3
19	waveform	21	5000	3
20	wine	13	178	3
21	arcene	10000	200	2
22	gisette	5000	7000	2
23	madelon	500	2600	2
24	sonar	60	208	2
25	spambase	57	4601	2
26	Synthetic	2	1250	2
27	vertebral	6	310	2
28	diabetes	8	768	2

Table 3.2 Attributes of the UCI datasets with continuous input features

Across both of the experiments, the following steps were taken for all datasets at the pre-processing stage: The categorical and ordinal variables were encoded using discrete values from 1-to-n. For each feature, whether discrete or continuous, the instances where any feature value is missing are discarded.

All experiments consisted of 100 trials, with random partitioning of the datasets without replacement into training and test sets in each trial. For each dataset, 75% was used for training and 25% was held-out as a test set, used only to evaluate the classification error rate during the experiments. The error rate learning curve is plotted for each dataset, then the area under error rate learning curve (AULC) was computed in each replication to evaluate the error rate performance. The learning curve plot started with one labelled training pattern for discrete benchmark datasets during training stage, while at least two labelled training patterns were required for each class for continuous benchmark datasets. For example, in the case of binary classification problem, the learning curve started with four labelled training patterns, two patterns for each class, in order to estimate the variance for each features. For

a particular feature, if the two labelled training patterns for each class has the same value then this feature is omitted from the analysis in order to avoid having zero variance. Finally, the Wilcoxon signed rank test [25] was used to determine the statistical significance of the difference between the SSNB and the NB over multiple datasets in terms of the AULC.

Results for UCI benchmark datasets

Our first experiments found that the use of the unlabelled dataset does not generally reduce the classification error rate. Table 3.3 shows the AULC results for 36 discrete benchmark datasets. NB was best on 24 out of 36 benchmark datasets, the SSNB best on only 12. The result for the Wilcoxon signed rank test shows that the NB is not statistically superior over all datasets at the 95% level of significance.

From Table 3.4 it can be seen that the AULC of the SSNB is statistically better than the NB only for the eight continuous datasets. However, the AULC for NB was best on most of the datasets. There is statistical significant difference according to Wilcoxon signed rank test at the 95% level of significance over all datasets.

#	Dataset	NB	SSNB
1	audiology	**4.891±0.0339**	5.292±0.0317
2	balance-scale	**2.830±0.0312**	3.224±0.0247
3	blogger	**2.173±0.0422**	2.521±0.0376
4	breast-cancer	**2.656±0.0432**	2.965±0.0392
5	breastw	0.874±0.0225	**0.325±0.0293**
6	car	**2.965±0.0308**	3.208±0.0474
7	DNA	3.181±0.0231	**2.103±0.0642**
8	flare1	*4.185±0.0354*	***4.149±0.0336***
9	flare2	4.264±0.0299	**4.151±0.0315**
10	hayes-roth	**3.140±0.0396**	3.587±0.0304
11	house-votes	**1.043±0.0234**	1.119±0.0266
12	kr-vs-kp	**3.136±0.0200**	4.771±0.0249
13	led7	5.200±0.0234	**5.095±0.0234**
14	led24	6.193±0.0141	**5.676±0.0210**
15	lung-cancer	2.743±0.0442	**2.637±0.0462**
16	lymphography	**2.190±0.0365**	2.373±0.0379

Continued on next page

#	Dataset	NB	SSNB
17	marketing	**9.488±0.0200**	9.768±0.0188
18	monk1-corrupt	**3.465±0.0245**	3.553±0.0284
19	monk1-cross	**1.803±0.0237**	2.106±0.0250
20	monk1-local	**3.218±0.0294**	3.644±0.0296
21	monk1	***3.129±0.0259***	*3.158±0.0300*
22	monk3-cross	**1.681±0.0275**	2.207±0.0474
23	monk3-local	**1.735±0.0223**	2.737±0.0368
24	monk3	**1.695±0.0237**	2.029±0.0231
25	mushroom	1.771±0.0226	**1.670±0.0313**
26	nursery	**3.662±0.0172**	3.767±0.0236
27	primary-tumor	**5.806±0.0301**	5.890±0.0259
28	promoters	1.694±0.0360	**1.336±0.0361**
29	shuttle-landing-control	**1.365±0.0275**	1.440±0.0386
30	soybean-small	1.362±0.0513	**1.010±0.0483**
31	soybean-large	**6.563±0.0191**	6.770±0.0198
32	splice	3.332±0.0169	**2.652±0.0522**
33	threeOf9	**2.711±0.0294**	2.967±0.0267
34	titanic	**3.033±0.0451**	3.211±0.0411
35	xd6	**2.670±0.0292**	2.789±0.0275
36	zoo	1.696±0.0446	**1.577±0.0469**

Table 3.3 AULC for the NB and SSNB over 36 discrete datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

#	Dataset	NB	SSNB
1	banknote	**1.541±0.0225**	2.931±0.0197
2	Blood-transfusion	**2.070±0.0262**	2.564±0.0357
3	breast-cancerw-continuous	0.500±0.0167	**0.343±0.0120**
4	Climate-Model-Simulation-Crashes	**0.603±0.0140**	0.631±0.0154
5	glass	**2.006±0.0192**	2.289±0.0170
6	haberman	**1.778±0.0301**	2.045±0.0527
7	ionosphere	**1.415±0.0278**	1.742±0.0298
8	iris	0.418±0.0187	**0.349±0.0170**
9	letter	**4.201±0.0111**	5.647±0.0074
10	liver-disorder	**2.719±0.0210**	2.989±0.0207
11	magic04	**3.506±0.0182**	4.832±0.0136
12	musk1	**2.207±0.0225**	2.716±0.0240
13	new-thyroid	0.352±0.0150	**0.183±0.0122**
14	pendigits	**2.104±0.0139**	2.652±0.0164

Continued on next page

#	Dataset	NB	SSNB
15	sleep	**4.956±0.0145**	7.175±0.0296
16	vehicle	**3.517±0.0166**	3.948±0.0168
17	vowel	**2.051±0.0138**	2.481±0.0112
18	waveform-noise	**2.581±0.0145**	2.913±0.0254
19	waveform	**2.917±0.0121**	3.451±0.0205
20	wine	0.640±0.0187	**0.311±0.0185**
21	arcene	2.160±0.0223	**2.051±0.0255**
22	gisette	**2.099±0.0117**	3.057±0.0285
23	madelon	4.116±0.0091	**3.935±0.0160**
24	sonar	**1.928±0.0239**	2.247±0.0210
25	spambase	**2.263±0.0195**	2.409±0.0288
26	Synthetic	1.182±0.0217	**1.116±0.0269**
27	vertebral	*1.421±0.0200*	***1.417±0.0231***
28	diabetes	**2.137±0.0207**	2.525±0.0249

Table 3.4 AULC for the NB and SSNB over 28 continuous datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

In these experiments, we concluded that the performance of the SSNB was inferior overall to that the NB for both discrete and continuous input features.

3.1.2 Why is the naïve Bayes classifier significantly better on average than the semi-supervised naïve Bayes classifier?

The most obvious explanation for this result is that NB is unable to utilise the unlabelled data correctly. The key characteristic of NB is that it makes the assumption of independence between features. This assumption is usually false and NB often produces inaccurate probability estimates, but fairly good classifications [61]. EM relies on the probability estimates, so may be overcompensating. To test this hypothesis, we generate simulated data that satisfies the NB assumption. A simple synthetic dataset is generated from two classes with univariate Gaussian distributions when an infinite amount of labelled and unlabelled

data is available for training and testing. The model parameters mean and variance for the two Gaussian is (μ_1=-1, μ_2=+1, $\sigma_1 = \sigma_2$ =1) respectively as shown in Figure 3.1.

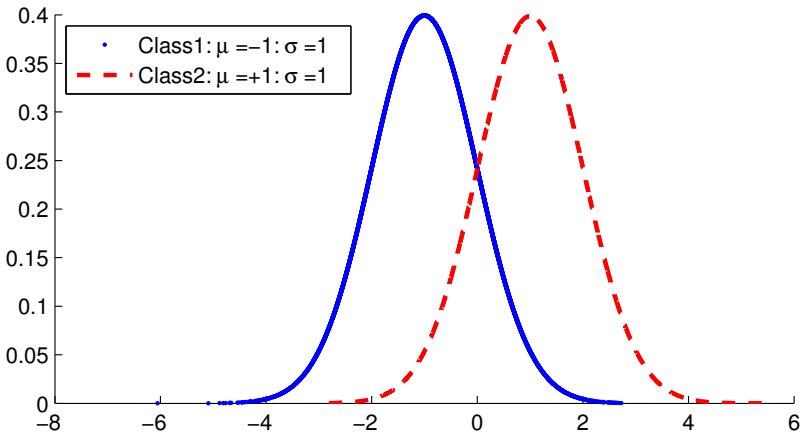


Fig. 3.1 Two-class classification problem for the synthetic dataset.

This experiment consisted of 10,000 trials of random partitioning of the datasets (67584 patterns) into training and test sets, that 2048 patterns were used for training and 65536 patterns were held-out as a test set used to evaluate the classification error rate performance during the experiments. The experimental design was exactly the same as Section 3.1.1.

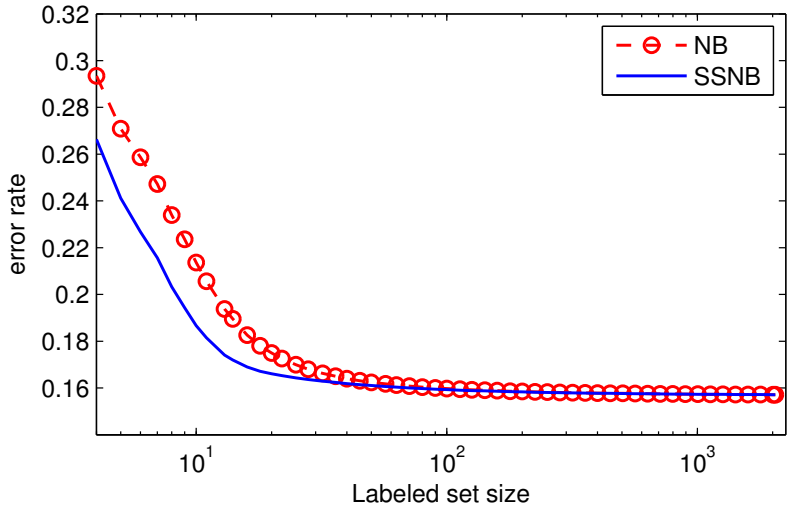


Fig. 3.2 The learning curve for a two-class classification problem in Gaussian distribution for synthetic dataset

Figure 3.2 shows the error rate learning curve for both the semi-supervised Gaussian classifier and the Gaussian classifier. It is clearly seen that the semi-supervised Gaussian classifier performs better than the Gaussian classifier, especially when very few labelled data were used for training, and rapidly converges in the number of labelled samples.

3.1.3 Synthetic benchmark datasets

Generating synthetic dataset and experimental design

The results from the previous Section 3.1.2 indicate that violation of the independence assumption of the NB classifier model might be a reason for the lack of increased performance for SSNB classifier. It is possible that the SSNB is sensitive to the correctness of the model's assumptions. To investigate this further, we generate simulated data from the UCI sets that will satisfy the independence assumption. To do this, we first fit a NB model to each data set, then use the model estimates of the feature distributions to generate simulated data with independent features. The synthetic datasets are similar in character to the original datasets but the model's assumption of the independence of features is valid.

Result for synthetic benchmark datasets

Table 3.5 shows that the SSNB performs well compared to the NB for the 33 synthetic benchmark datasets. The SSNB also was best for all the continuous synthetic datasets, shown in Table 3.6, thus the SSNB performed better than the NB in the current experiment. There is a statistical significant difference between the average rank of the AULC for the SSNB and NB classifier according to the Wilcoxon signed rank test at the 95% level of confidence over multiple benchmark datasets in both experiments. This suggests that SSNB is sensitive to conformance to its assumption of independence between features.

#	Dataset	NB	SSNB
1	audiology	**4.683±0.0361**	4.777±0.0334

Continued on next page

#	Dataset	NB	SSNB
2	balance-scale	3.359±0.0316	**3.182±0.0417**
3	blogger	2.350±0.0430	**2.248±0.0529**
4	breast-cancer	**2.553±0.0381**	2.631±0.0336
5	breastw	0.705±0.0221	**0.098±0.0274**
6	car	3.286±0.0324	**3.146±0.0407**
7	DNA	2.724±0.0242	**1.624±0.0616**
8	flare1	4.005±0.0292	**3.637±0.0346**
9	flare2	4.037±0.0278	**3.796±0.0359**
10	hayes-roth	3.188±0.0417	**3.008±0.0395**
11	house-votes	0.135±0.0099	**0.011±0.0071**
12	kr-vs-kp	3.265±0.0201	**2.310±0.0471**
13	led7	5.199±0.0229	**5.123±0.0210**
14	led24	6.163±0.0155	**5.542±0.0224**
15	lung-cancer	1.988±0.0558	**1.824±0.0565**
16	lymphography	1.832±0.0372	**1.230±0.0385**
17	marketing	8.816±0.0195	**8.305±0.0219**
18	monk1-corrupt	***3.096±0.0258***	*3.105±0.0387*
19	monk1-cross	2.137±0.0278	**1.238±0.0549**
20	monk1-local	3.205±0.0263	**2.942±0.0433**
21	monk1	3.265±0.0273	**3.191±0.0399**
22	monk3-cross	1.704±0.0267	**0.868±0.0386**
23	monk3-local	1.950±0.0281	**1.083±0.0309**
24	monk3	1.985±0.0288	**1.530±0.0478**
25	mushroom	1.003±0.0309	**0.016±0.0008**
26	nursery	4.022±0.0194	**3.422±0.0472**
27	primary-tumor	5.561±0.0321	**5.509±0.0319**
28	promoters	1.215±0.0311	**0.245±0.0240**
29	shuttle-landing-control	1.569±0.0315	**1.417±0.0412**
30	soybean-small	1.235±0.0458	**0.902±0.0482**
31	soybean-large	6.462±0.0220	**6.215±0.0214**
32	splice	3.167±0.0161	**1.505±0.0781**
33	threeOf9	2.953±0.0327	**2.656±0.0408**
34	titanic	2.978±0.0391	**2.772±0.0379**
35	xd6	2.713±0.0289	**2.219±0.0423**
36	zoo	1.897±0.0466	**1.716±0.0515**

Table 3.5 AULC for the NB and SSNB over 36 synthetic discrete datasets from the UCI. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

#	Dataset	NB	SSNB
1	banknote	2.033±0.0168	**1.738±0.0254**
2	Blood-transfusion	1.022±0.0180	**0.713±0.0235**
3	breast-cancerw-continuous	0.220±0.0160	**0.018±0.0082**
4	Climate-Model-Simulation-Crashes	0.280±0.0110	**0.122±0.0114**
5	glass	0.735±0.0178	**0.525±0.0165**
6	haberman	1.417±0.0272	**1.242±0.0387**
7	ionosphere	0.418±0.0207	**0.127±0.0168**
8	iris	0.067±0.0118	**0.014±0.0077**
9	letter	3.086±0.0060	**2.188±0.0095**
10	liver-disorder	1.714±0.0205	**1.371±0.0238**
11	magic04	0.538±0.0219	**0.083±0.0038**
12	musk1	0.760±0.0285	**0.441±0.0320**
13	new-thyroid	0.130±0.0110	**0.024±0.0080**
14	pendigits	1.623±0.0093	**0.730±0.0128**
15	sleep	2.461±0.0107	**1.845±0.0225**
16	vehicle	2.118±0.0141	**1.776±0.0168**
17	vowel	0.945±0.0123	**0.545±0.0149**
18	waveform-noise	1.205±0.0165	**0.516±0.0216**
19	waveform	1.386±0.0157	**0.645±0.0192**
20	wine	0.199±0.0160	**0.053±0.0119**
21	arcene	0.138±0.0173	**0.035±0.0110**
22	gisette	0.427±0.0190	**0.318±0.0213**
23	madelon	3.306±0.0109	**2.855±0.0171**
24	sonar	0.086±0.0146	**0.023±0.0143**
25	spambase	0.004±0.0023	**0.002±0.0019**
26	Synthetic	0.047±0.0091	**0.032±0.0057**
27	vertebral	0.491±0.0239	**0.181±0.0224**
28	diabetes	1.639±0.0218	**1.185±0.0239**

Table 3.6 AULC for the NB and SSNB on 28 synthetic continuous datasets from the UCI. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

3.1.4 Exploratory data analysis

The results for UCI benchmark datasets experiments suggest that a few datasets are always likely to have better performance for SSNB, such as (`breastw`, `DNA`, `led7`, `led24`,

lung-cancer, promoters, shuttle-landing-control, splice, soybean-small) in discrete and (iris, new-thyroid, wine) in continuous benchmark datasets. The details of this performance can be seen by examining the learning curve. We can show the learning curve only for a few datasets due to space limitation. Figure 3.3 shows the learning curve for one of the discrete datasets splice and new-thyroid which is a continuous dataset.

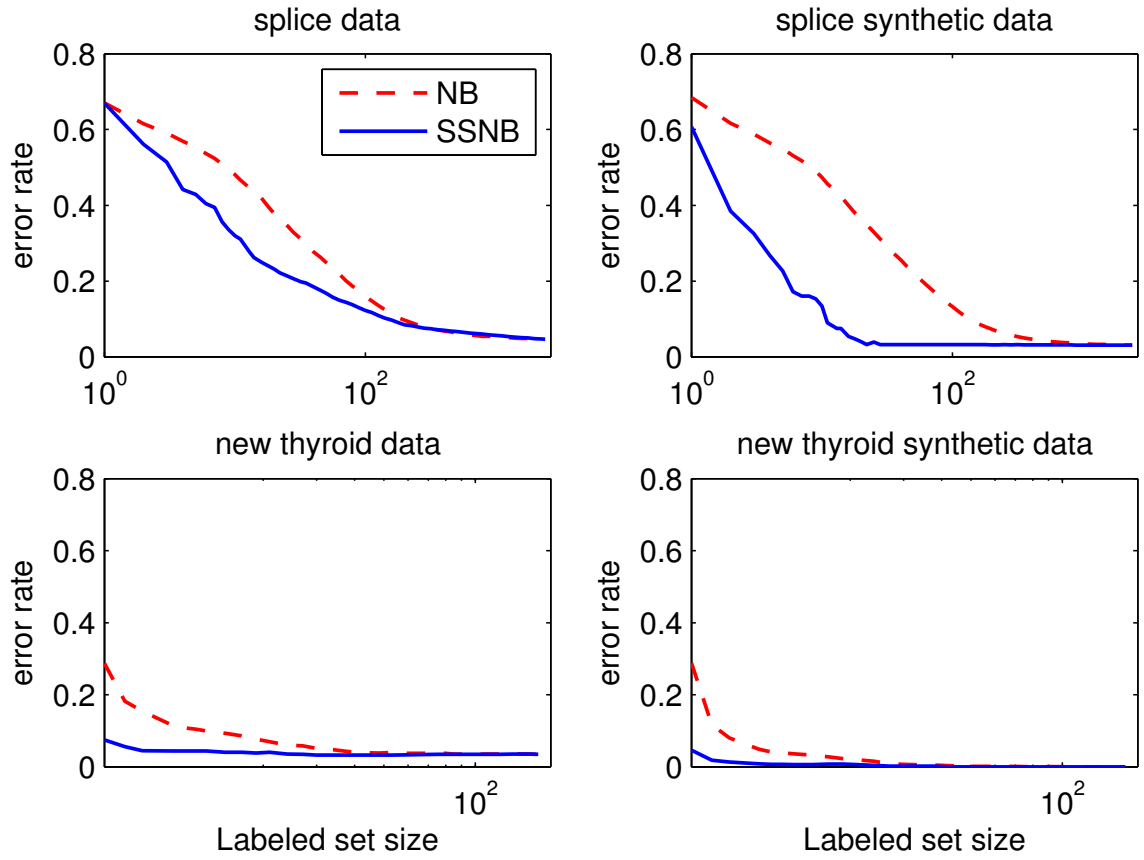


Fig. 3.3 The average learning curve for NB and SSNB of the UCI and synthetic (splice, newthyroid) datasets

Interestingly, most of the synthetic discrete and continuous benchmark datasets show improved classification performance for the SSNB. Figure 3.4 shows the learning curve for nursery and waveform datasets which are discrete and continuous respectively. However, the learning curves for (audiology, breast-cancer, monk1, mushroom, primary-tumor,

`soybean-large`) in discrete datasets and (`magic04`) in continuous datasets show that SSNB does not help in all experiments, as we can see from the learning curve for one of them in Figure 3.5.

The learning curve results across all experiments show that if the model assumption is correct the unlabelled data might help to improve performance, especially when a few labelled data are used as a training set; but if the model assumption is violated, the classification performance could degrade when adding more unlabelled data to the training set.

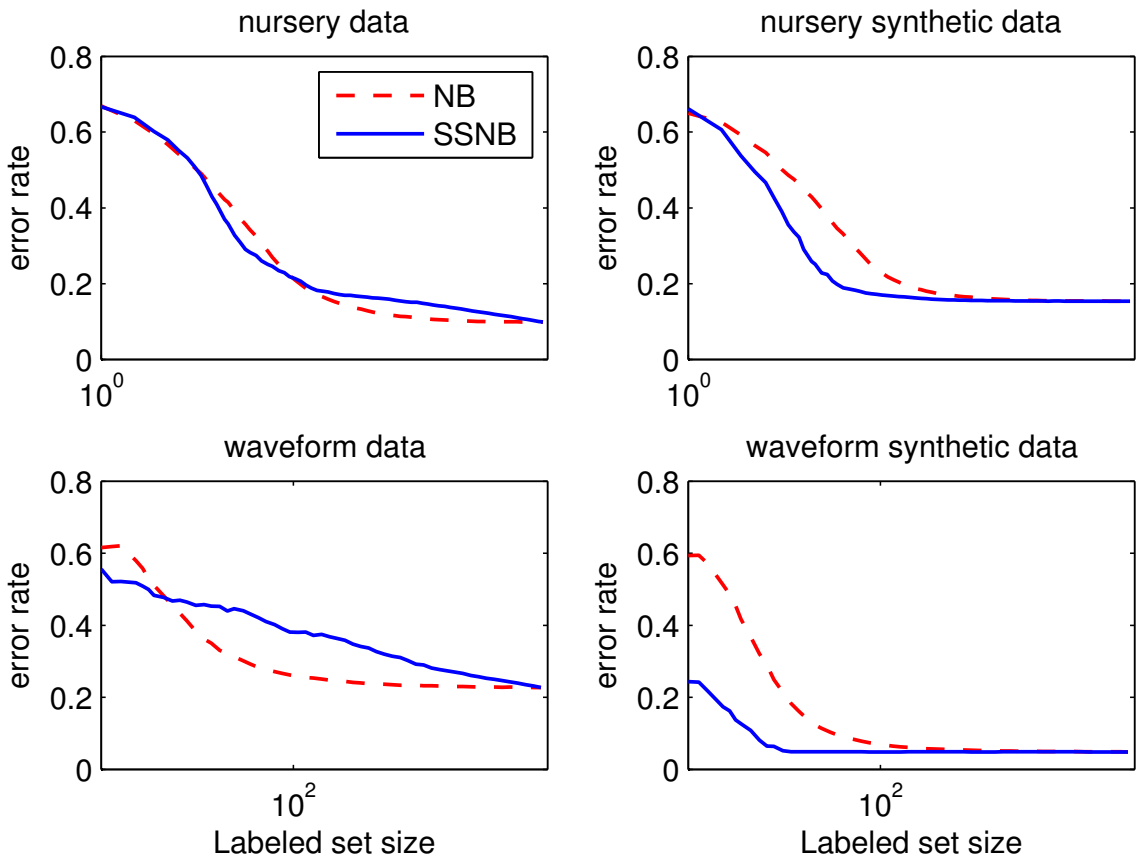


Fig. 3.4 The average learning curve for the NB and SSNB of the UCI and synthetic (nursery, waveform) dataset

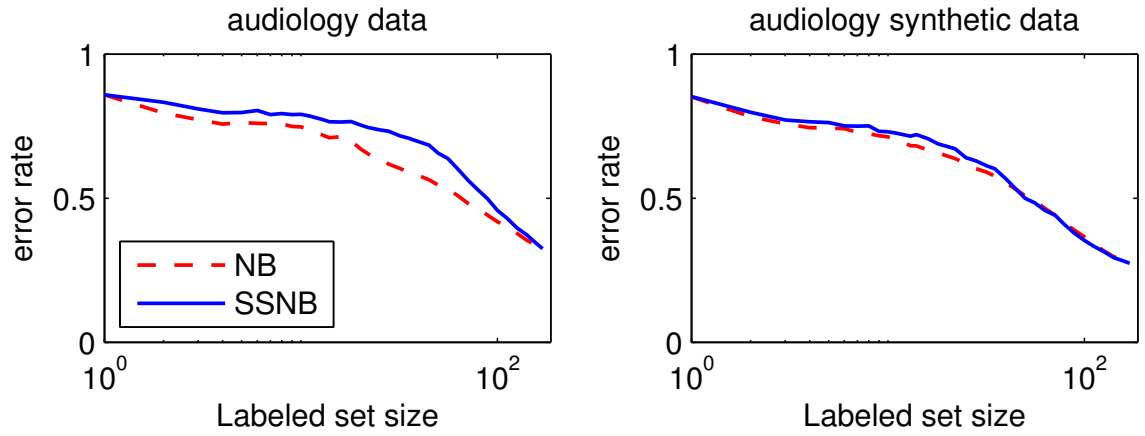


Fig. 3.5 The average learning curve for the NB and SSNB of the UCI and synthetic audiology datasets

3.2 Conclusions

The contribution of this chapter is an empirical evaluation of NB and SSNB on binary and multi-class classification problems with continuous and discrete features. We wish to address the question of whether using unlabelled data will improve classification accuracy. This will clearly be dictated by our choice of classifier and semi-supervised learning scheme. We evaluated a naïve Bayes classifier used in conjunction with an Expectation-Maximization algorithm that iteratively uses NB to predict the unlabelled instances. We found that using the unlabelled data made the classifier significantly less accurate. To understand why this may be so, we assessed the performance of NB and SSNB on synthetic data for which the NB assumption of independent features was valid. We found that SSNB was significantly more accurate on these data. We conclude that if a classifier is not suitable for a data set, then using unlabelled data in a self training scheme is likely to make it worse. This implies that effort should be applied in finding a classifier suitable for a problem before using unlabelled data to self train.

Chapter 4

Differential weighting of labelled and unlabelled examples

As demonstrated by the benchmarking in Chapter 3, the use of semi-supervised learning with the naïve Bayes classifier generally degrades rather than improves classification performance. Experiments show that one of the reasons for this degradation is that the assumption of independence between input features is often invalid in practice. Another reason probably is the size of data, where the amount of labelled patterns is usually much smaller than the amount of unlabelled patterns. Thus, it is possible that a large amount of uninformative unlabelled data is swamping the more reliable information in the labelled data.

In this chapter, we employ down-weighting of the unlabelled data to test whether this reduces the influence of the unlabelled data and improves the performance of the naïve Bayes classifier. Furthermore, we investigate the use of a hyper-parameter, λ, to down-weight the contribution of the unlabelled data, and some model selection methods which have been used to tune λ. A preliminary study, as expected, shows that down-weighting the influence of the unlabelled data improves the baseline classifier somewhat. The cause for this improvement is tuned to maximise test set performance, which is a biased protocol. Then, an unbiased model selection procedure has been investigated but then the down-weighting was less successful.

Investigating other model selection procedures such as k fold cross-validation and leave-one-out-cross-validation, may give unreliable indicate for selecting a hyper-parameter, λ. The k fold cross-validation procedure needs a large amount of labelled patterns to obtain a reliable result and using leave-one-out-cross-validation provides high variance. Thus, a different value of λ, might be utilised if the experiment is repeated with different a sample of dataset. Therefore, we used a new method between the leave-one-out-cross-validation and k folds cross-validation section 4.5 and again the unlabelled data does not improve classification performance because it is difficult to tune the value of λ.

4.1 Down-weighting of the unlabelled data

The standard EM based semi-supervised NB algorithm procedure works well to estimate the model parameters with unlabelled patterns in the case of semi-supervised learning, when the data conform to the assumptions of the model [59]. However this assumption of independence is generally invalid in practice, thus there exists the possibility that the EM algorithm would degrade rather than improve classification error [70]. As described in the chapter 3, a common scenario in semi-supervised learning is that the majority of the data is unlabelled, but unlabelled data participates in estimating the model parameters in the M-step of the EM algorithm. Thus, it is possible that a large amount of unlabelled data may swamp the more reliable information in the labelled data.

In order to reduce the influence of unlabelled data, we investigate the inclusion of a hyper-parameter, λ, to down-weight the contribution of the unlabelled data in the M-step of the EM algorithm, which is denoted by EM-λ.

Nigam et al. [59], show that down-weighting the influence of the unlabelled data in this way can improve the performance of the naïve Bayes classifier for the `WebKB` dataset. The experiments in this chapter, use a large number of benchmark datasets from the UCI repository to test whether implementing down-weighting of the contribution of the unlabelled data can

improve the performance of naïve Bayes SSNB-λ classifier, especially for cases with a small number of labelled examples. In addition, while running the experiments, another research question is raised which is how to choose the value of weighting factor λ.

The contributions of this chapters are summarised as follows:

- The main contribution is that down weighting the influence of the unlabelled data does not generally improve the classifier. In fact, our experiments show that for the majority of the benchmark datasets it is preferable not to use the unlabelled data.
- Tuning the value of λ through the test set can improve the performance of the NB classifier, but it is a biased protocol giving over-optimistic estimates of performance. Therefore, it would be better to investigate other model section methods for tuning hyper-parameter λ.
- The results obtained with other model selection methods suggest that none of the model selection methods that we evaluate for choosing the value of the λ are a significant improvement over the naïve Bayes classifier.

4.2 Technical background

In this section, we provide a brief overview of the SSNB that uses the extended EM algorithm, EM-λ, for estimating the parameters in the M-step. The EM-λ algorithm, proposed by Nigam et al. [59], allows the contribution of the unlabelled data to be down-weighted in order to decrease the impact of the unlabelled data for the EM algorithm on the WebKB dataset when the ratio of labelled to unlabelled data is low. In this section, the EM-λ method is used to moderate the contribution the labelled and unlabelled data by adding a weighting

hyper-parameter λ, $0 \leq \lambda < 1$, into the log likelihood function (2.13), such that

$$\begin{aligned} \log p(\mathscr{D};\theta) &= (1-\lambda)\sum_{i=1}^{l}\log\left(\, p(y^{(i)};\theta)\prod_{j=1}^{d} p(x_j^{(i)}|y^{(i)};\theta)\right) \\ &+ (\lambda)\sum_{i=l+1}^{l+u}\log\left(\sum_{c=1}^{C} p(z^{(i)}=c;\theta)\prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta)\right). \end{aligned} \tag{4.1}$$

It is obvious that when $\lambda = 0$ the unlabelled data are ignored, such that the likelihood depends only on the labelled data. In contrast, when λ has largest value, $\lambda = 0.9$, the labelled data have little weight and the likelihood approximately depends only on the unlabelled data, but with $\lambda = 1$, the likelihood depends only on the unlabelled data (unsupervised learning). The EM-λ algorithm has the same E-step as standard EM algorithm, but maximizing of the M-step is different. The weighting hyper-parameter λ can be added into the M-step in (2.17),

$$\begin{aligned} \psi(\theta) &= (1-\lambda)\sum_{i=1}^{l} q_{ic}\log\left(\, p(y^{(i)};\theta)\prod_{j=1}^{d} p(x_j^{(i)}|y^{(i)};\theta)\right) \\ &+ (\lambda)\sum_{i=l+1}^{l+u}\sum_{c=1}^{C} q_{ic}\log\left(p(z^{(i)}=c;\theta)\prod_{j=1}^{d} p(x_j^{(i)}|z^{(i)}=c;\theta)\right) \\ &- (\lambda)\sum_{i=l+1}^{l+u}\sum_{c=1}^{C} q_{ic}\log q_{ic}. \end{aligned} \tag{4.2}$$

The M-step estimates the new model parameters by setting the partial derivatives to zero for (4.2), using the labelled and unlabelled data, the true class labels for labelled data is y and the expected values of the latent variable is z that EM-λ algorithm treated the expected values as a true class labels for the unlabelled data in the M-step. in addition, the expected values of the latent variable z are calculated in the E-step. We can show how to estimate the new model parameters as follows. If $x_j \sim cat(\theta)$ then (4.2) can be written in terms of the

parameters with Lagrangian term.

$$\begin{aligned}
\Lambda(\pi,\theta,\alpha,\beta) &= (1-\lambda)\sum_{i=1}^{l}\sum_{c=1}^{C} q_{ic}\log\pi_c \\
&+ (1-\lambda)\sum_{i=1}^{l}\sum_{j=1}^{d}\sum_{s=1}^{S}\sum_{c=1}^{C} q_{ic}\phi(x_j^{(i)}=s)\log\theta_{sc}^{j} \\
&+ (\lambda)\sum_{i=l+1}^{l+u}\sum_{c=1}^{C} q_{ic}\log\pi_c \\
&+ (\lambda)\sum_{i=l+1}^{l+u}\sum_{j=1}^{d}\sum_{s=1}^{S}\sum_{c=1}^{C} q_{ic}\phi(x_j^{(i)}=s)\log\theta_{sc}^{j} \\
&- (\lambda)\sum_{i=l+1}^{l+u}\sum_{c=1}^{C} q_{ic}\log q_{ic} - \alpha\left(\sum_{c=1}^{C}\pi_c - 1\right) \\
&- \sum_{c=1}^{C}\sum_{j=1}^{d}\beta_c^{j}\left(\sum_{s=1}^{S}\theta_{sc}^{j} - 1\right).
\end{aligned} \tag{4.3}$$

To obtain the maximum likelihood estimate, the partial derivatives can be computed for (4.3) with respect to all the parameters $(\alpha,\beta_c^j,\pi_c,\theta_{sc}^j)$ and set to zero. For α and β_c^j, it is the same as for supervised NB.

$$\begin{aligned}
\frac{\partial\Lambda}{\partial\pi_c} &= 0 \Rightarrow \\
\pi_c &= \frac{(1-\lambda)\sum_{i=1}^{l} q_{ic} + (\lambda)\sum_{i=l+1}^{l+u} q_{ic}}{(1-\lambda)\sum_{k=1}^{C}\sum_{i=1}^{l} q_{ik} + (\lambda)\sum_{k=1}^{C}\sum_{i=l+1}^{l+u} q_{ik}} \quad , \\
\frac{\partial\Lambda}{\partial\theta_{sc}^{j}} &= 0 \Rightarrow \\
\theta_{sc}^{j} &= \frac{(1-\lambda)\sum_{i=1}^{l} q_{ic}\phi(x_j^{(i)}=s) + (\lambda)\sum_{i=l+1}^{l+u} q_{ic}\phi(x_j^{(i)}=s)}{(1-\lambda)\sum_{m=1}^{S}\sum_{i=1}^{l} q_{ic}\phi(x_j^{(i)}=m) + (\lambda)\sum_{m=1}^{S}\sum_{i=l+1}^{l+u} q_{ic}\phi(x_j^{(i)}=m)},
\end{aligned} \tag{4.4}$$

where the summation in the denominator is over all possible values (states) m for each feature x_j. The Laplace correction for the parameters, (θ^j_{sc}, π_c), is as follows:

$$\begin{aligned}
\pi_c &= \frac{(1-\lambda)\sum_{i=1}^{l} q_{ic} + (\lambda)\sum_{i=l+1}^{l+u} q_{ic} \; + \; 1}{(1-\lambda)\sum_{k=1}^{C}\sum_{i=1}^{l} q_{ik} + (\lambda)\sum_{k=1}^{C}\sum_{i=l+1}^{l+u} q_{ik} \; + \; C} \\
\theta^j_{sc} &= \frac{(1-\lambda)\sum_{i=1}^{l} q_{ic}\phi(x_j^{(i)} = s) + (\lambda)\sum_{i=l+1}^{l+u} q_{ic}\phi(x_j^{(i)} = s) \; + \; 1}{(1-\lambda)\sum_{m=1}^{S}\sum_{i=1}^{l} q_{ic}\phi(x_j^{(i)} = m) + (\lambda)\sum_{m=1}^{S}\sum_{i=l+1}^{l+u} q_{ic}\phi(x_j^{(i)} = m) \; + \; S_j}.
\end{aligned} \tag{4.5}$$

Suppose x_j are drawn from a Gaussian distribution, $x_j \sim \mathcal{N}(\mu, \sigma^2)$, with unknown model parameters (mean μ and variance σ^2). Then we can illustrate (4.2) as follows:

$$\begin{aligned}
\Lambda(\pi_c, \mu_c^j, {\sigma^2}_c^j) &= (1-\lambda)\sum_{i=1}^{l} q_{ic}\log\pi_c \\
&+ (1-\lambda)\sum_{i=1}^{l}\sum_{j=1}^{d} q_{ic}\log\Bigg(\frac{1}{(2\pi)^{\frac{1}{2}}|{\sigma^2}_c^j|^{\frac{1}{2}}}\exp \\
&\quad \Big(-\frac{1}{2}(x_i^j - \mu_c^j)^2({\sigma^2}_c^j)^{-1}\Big)\Bigg) \\
&+ \lambda\sum_{i=l+1}^{l+u}\sum_{c=1}^{C} q_{ic}\log\pi_c \\
&+ \lambda\sum_{i=l+1}^{l+u}\sum_{c=1}^{C}\sum_{j=1}^{d} q_{ic}\log\Bigg(\frac{1}{(2\pi)^{\frac{1}{2}}|{\sigma^2}_c^j|^{\frac{1}{2}}}\exp \\
&\quad \Big(-\frac{1}{2}(x_i^j - \mu_c^j)^2({\sigma^2}_c^j)^{-1}\Big)\Bigg) \\
&- \lambda\sum_{i=l+1}^{l+u}\sum_{c=1}^{C} q_{ic}\log q_{ic} - \alpha\Bigg(\sum_{c=1}^{C}\pi_c - 1\Bigg).
\end{aligned} \tag{4.6}$$

Then, the closed form of the maximum likelihood estimate can be obtained by computing the partial derivatives of (4.6) with respect to all the parameters $(\alpha, \pi_c, \mu_c^j, {\sigma^2}_c^j)$, and then setting each partial derivative to zero. The result for (α, π_j) are the same as for naïve Bayes

classifier.

$$\begin{aligned}
\frac{\partial \Lambda}{\partial \mu_c^j} &= 0 \Rightarrow \mu_c^j = \frac{(1-\lambda)\sum_{i=1}^{l} q_{ic} x_i^j \; + \; (\lambda)\sum_{i=l+1}^{l+u} q_{ic}\, x_i^j}{(1-\lambda)\sum_{i=1}^{l} q_{ic} \; + \; (\lambda)\sum_{i=l+1}^{l+u} q_{ic}}. \\
\frac{\partial \Lambda}{\partial {\sigma^2}_c^j} &= 0 \Rightarrow \\
{\sigma^2}_c^j &= \frac{(1-\lambda)\sum_{i=1}^{l} q_{ic}(x_i^j - \mu_c^j)^2 \; + \; (\lambda)\sum_{i=l+1}^{l+u} q_{ic}(x_i^j - \mu_c^j)^2}{(1-\lambda)\sum_{i=1}^{l} q_{ic} \; + \; (\lambda)\sum_{i=l+1}^{l+u} q_{ic}}.
\end{aligned} \tag{4.7}$$

where for labelled data $q_{ic} = 1$ if $y_i = c$, $q_{ic} = 0$ otherwise, and for unlabelled data q_{ic} is the expected value.

4.3 Preliminary experiment for contribution of labelled and unlabelled data

In the preliminary experiments, we performed two sets of experiments in order to evaluate the performance of NB compared to the SSNB-λ classifier. The 28 continuous benchmark datasets in Table 3.2 used in the former experiment and the 36 discrete benchmark datasets in Table 3.1 used in the latter were taken from the UCI machine learning repository. The experimental design for both experiments is the same as the experimental design in Section 3.1.1. The experiments in this section investigate the impact of weight λ in the contribution of labelled and unlabelled data by varying λ in the Equation 2.13. In the EM-λ algorithm, the hyper-parameter λ is assigned values such as $\lambda \in [0,1)$ in increments of 0.1. If the optimal value of $\hat{\lambda}$ is found such that $\lambda > 0$ then we search for a better value of λ within a smaller interval in smaller increments such that λ ranges from optimal value of $\hat{\lambda}$-0.1 to optimal value of $\hat{\lambda}$+0.1 in increments of 0.01. On the other hand, if the optimal value of $\hat{\lambda} = 0$, then we look for the optimal value of $\hat{\lambda}$ between 0 and $\hat{\lambda}$+0.1 in increments of 0.01. Experimental results are summarised in Table 4.1 and 4.2

#	Dataset	NB	SSNB	SSNB-λ	Optimal λ
1	banknote	1.541±0.0225	2.931±0.0197	1.541±0.0225	0
2	Blood_transfusion	2.070±0.0262	2.564±0.0357	2.070±0.0262	0
3	breast_cancerw_continuous	0.500±0.0167	0.343±0.0120	**0.327±0.0118**	0.2
4	Climate_Model_Simulation_Crashes	0.603±0.0140	0.631±0.0154	**0.588±0.0148**	0.11
5	glass	2.006±0.0192	2.289±0.0170	2.006±0.0192	0
6	haberman	1.778±0.0301	2.045±0.0527	1.778±0.0301	0
7	ionosphere	1.415±0.0278	1.742±0.0298	1.415±0.0278	0
8	iris	0.418±0.0187	0.349±0.0170	**0.319±0.0168**	0.22
9	letter	4.201±0.0111	5.647±0.0074	4.201±0.0111	0
10	liver_disorder	2.719±0.0210	2.989±0.0207	2.719±0.0210	0
11	magic04	3.506±0.0182	4.832±0.0136	3.506±0.0182	0
12	musk1	2.207±0.0225	2.716±0.0240	2.207±0.0225	0
13	new_thyroid	0.352±0.0150	0.183±0.0122	**0.175±0.0116**	0.48
14	pendigits	2.104±0.0139	2.652±0.0164	2.104±0.0139	0
15	sleep	4.956±0.0145	7.175±0.0296	4.956±0.0145	0
16	vehicle	3.517±0.0166	3.948±0.0168	3.517±0.0166	0
17	vowel	2.051±0.0138	2.481±0.0112	2.051±0.0138	0
18	waveform_noise	2.581±0.0145	2.913±0.0254	2.581±0.0145	0
19	waveform	2.917±0.0121	3.451±0.0205	2.917±0.0121	0
20	wine	0.640±0.0187	0.311±0.0185	**0.255±0.0148**	0.49
21	arcene	2.141±0.0216	2.051±0.0255	**2.010±0.0257**	0.2
22	gisette	2.099±0.0117	3.057±0.0285	2.099±0.0117	0
23	madelon	4.116±0.0091	3.935±0.0160	**3.918±0.0160**	0.99
24	sonar	1.928±0.0239	2.247±0.0210	1.928±0.0239	0
25	spambase	2.263±0.0195	2.409±0.0288	2.263±0.0195	0
26	Synthetic	1.182±0.0217	1.116±0.0269	**1.111±0.0262**	0.34
27	vertebral	1.421±0.0200	1.417±0.0231	**1.385±0.0214**	0.03
28	diabetes	2.137±0.0207	2.525±0.0249	2.168±0.0217	0

Table 4.1 AULC for the NB, SSNB and SSNB-λ classifiers over 28 continuous benchmark datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ classifier is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier.

As can be seen from Table 4.1, down-weighting the unlabelled data makes a slight improvement for just nine benchmarks datasets out of 28. The mean rank of SSNB-λ classifier is not statistically different from that of the standard NB classifier. To illustrate the statistical difference between the NB, SSNB-λ, and SSNB classifiers, a critical difference

diagram is shown in Figure 4.1. Both NB and SSNB-λ classifier are grouped with a clique bar which means there is no statistically significant difference between their average ranks. However, both classifiers are statistically superior to the SSNB classifier, which suggests that down weighting the influence of unlabelled data is better than using all unlabelled data without down-weighting.

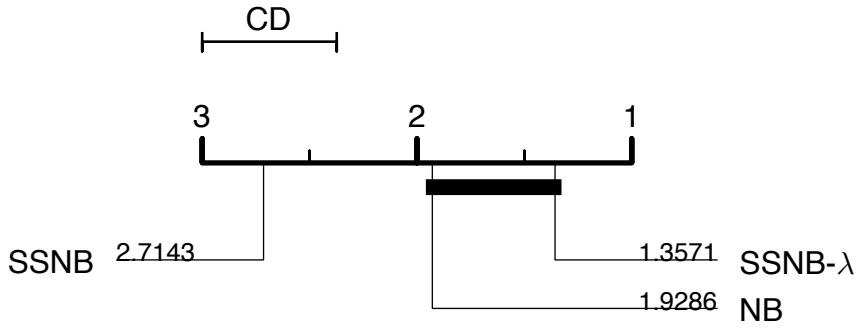


Fig. 4.1 Critical difference diagram for NB, SSNB, and SSNB-λ over 28 continuous benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar

In the second experiment, which evaluates the performance of down weighting the unlabelled data over 36 discrete benchmark datasets, the results obtained for the SSNB-λ classifier are statistically superior to the NB classifiers according to the Wilcoxon signed rank test at the 95% level of significance over all benchmark datasets. Table 4.2 presents the results for discrete benchmark datasets that the SSNB-λ classifier outperforms the NB classifier on 20 of the 36 benchmark datasets.

#	Dataset	NB	SSNB	SSNB-λ	Optimal λ
1	audiology	4.891±0.0339	5.292±0.0317	4.891±0.0339	0
2	balance_scale	2.830±0.0312	3.224±0.0247	2.830±0.0312	0
3	blogger	*2.173±0.0422*	2.521±0.0376	***2.168±0.0421***	0.03
4	breast_cancer	*2.656±0.0432*	2.965±0.0392	***2.629±0.0478***	0.11
5	breastw	0.874±0.0225	0.325±0.0293	**0.329±0.0294**	0.93
6	car	2.965±0.0308	3.208±0.0474	**2.858±0.0451**	0.03
7	DNA	3.181±0.0231	2.103±0.0642	**2.129±0.0613**	0.36

Continued on next page

#	Dataset	NB	SSNB	SSNB-λ	Optimal λ
8	flare1	4.185±0.0354	4.149±0.0336	4.185±0.0354	0
9	flare2	*4.264±0.0299*	4.151±0.0315	***4.222±0.0318***	0.32
10	hayes_roth	3.140±0.0396	3.587±0.0304	3.140±0.0396	0
11	house_votes	1.043±0.0234	1.119±0.0266	1.043±0.0234	0
12	kr_vs_kp	3.136±0.0200	4.771±0.0249	3.136±0.0200	0
13	led7	5.200±0.0234	5.095±0.0234	**5.137±0.0239**	0.33
14	led24	6.193±0.0141	5.676±0.0210	**5.548±0.0215**	0.18
15	lung_cancer	*2.743±0.0442*	2.637±0.0462	***2.726±0.0462***	0.79
16	lymphography	*2.190±0.0365*	2.373±0.0379	***2.166±0.0372***	0.19
17	marketing	9.488±0.0200	9.768±0.0188	9.488±0.0200	0
18	monk1_corrupt	3.465±0.0245	3.553±0.0284	3.465±0.0245	0
19	monk1_cross	1.803±0.0237	2.106±0.0250	1.803±0.0237	0
20	monk1_local	3.218±0.0294	3.644±0.0296	3.218±0.0294	0
21	monk1	3.129±0.0259	3.158±0.0300	**3.111±0.0266**	0.04
22	monk3_cross	1.681±0.0275	2.207±0.0474	1.681±0.0275	0
23	monk3_local	1.735±0.0223	2.737±0.0368	1.735±0.0223	0
24	monk3	1.695±0.0237	2.029±0.0231	1.695±0.0237	0
25	mushroom	1.771±0.0226	1.670±0.0313	**1.577±0.0309**	0.03
26	nursery	3.662±0.0172	3.767±0.0236	**3.527±0.0198**	0.05
27	primary_tumor	5.806±0.0301	5.890±0.0259	5.806±0.0301	0
28	promoters	1.694±0.0360	1.336±0.0361	**1.220±0.0366**	0.41
29	sls	1.365±0.0275	1.440±0.0386	**1.334±0.0282**	0.06
30	soybean_small	1.362±0.0513	1.010±0.0483	**1.108±0.0525**	0.81
31	soybean_large	6.563±0.0191	6.770±0.0198	6.563±0.0191	0
32	splice	3.332±0.0169	2.652±0.0522	**2.466±0.0439**	0.11
33	threeOf9	2.711±0.0294	2.967±0.0267	2.711±0.0294	0
34	titanic	3.033±0.0451	3.211±0.0411	**2.857±0.0558**	0.02
35	xd6	2.670±0.0292	2.789±0.0275	**2.643±0.0283**	0.08
36	zoo	1.696±0.0446	1.577±0.0469	**1.630±0.0455**	0.24

Table 4.2 AULC for the NB, SSNB, and SSNB-λ classifier over 36 discrete datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ classifiers is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier.

Figure 4.2 shows a critical difference diagram for the NB, SSNB, and SSNB-λ over the 36 discrete benchmark datasets. The diagram shows that the SSNB-λ is statistically superior to both the NB and SSNB classifiers.

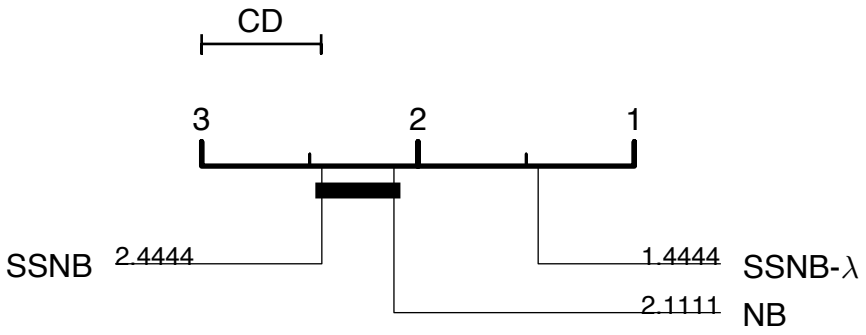


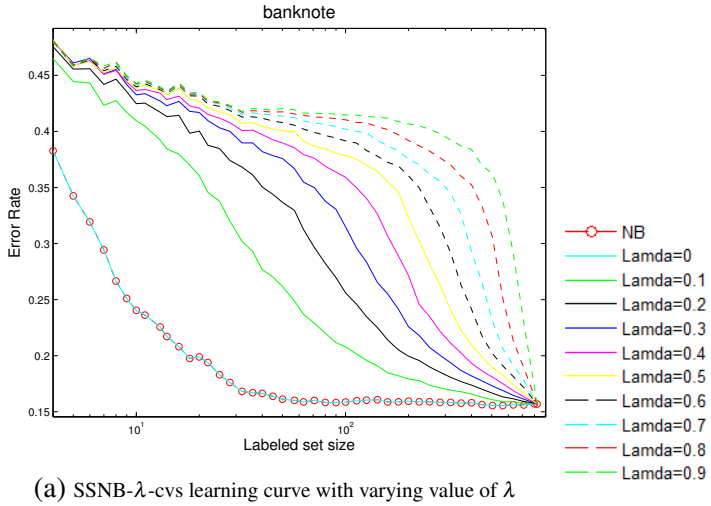
Fig. 4.2 Critical difference diagram for NB, SSNB, and SSNB-λ over 28 continuous benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar

4.3.1 Discussion

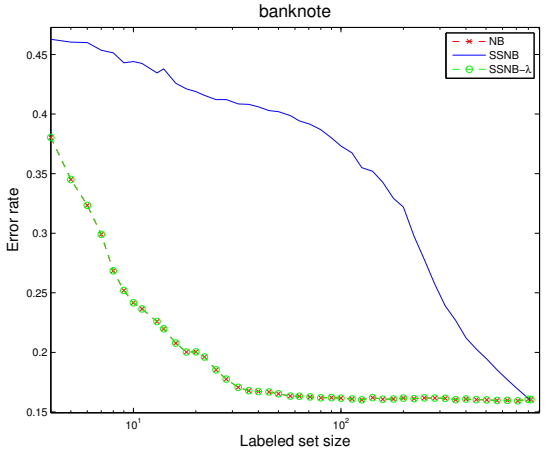
The experimental evaluation in the first experiment shows that using SSNB-λ does not significantly improve the performance of the semi-supervised naïve Bayes classifier. The question is, how much can λ make a better result compared to the previous chapter 3, results? In each trial, when the value of λ is increased, then a different result for a different value of λ is obtained but it is not statistically different. Figure 4.3 shows result from the `banknote` dataset that illustrate ten different lines, representing ten different values of λ between 0 to 0.9 in increments of 0.1. As we can see, tuning the extra hyper-parameter does not reduce the area under the error rate learning curve, but just decreases the classification performance, such that for both experiments 19 dataset out of the 28 continuous benchmark datasets and 16 dataset out of the 36 discrete benchmark datasets the optimal value is $\hat{\lambda} = 0$. In addition, the optimal value is $\hat{\lambda}$ for each of the `vertebral`, `blogger`, `car`, `monk1`, `mushroom`, `nursery`, `shuttle-landing-control`, `titanic`, `xd6` benchmarks is approximately zero.

These results implies that tuning λ does not help. In addition, for the rest of the datasets using SSNB-λ does not make significant improvements and using a different value of λ gives approximately the same result. However, in second experiment down-weighting the

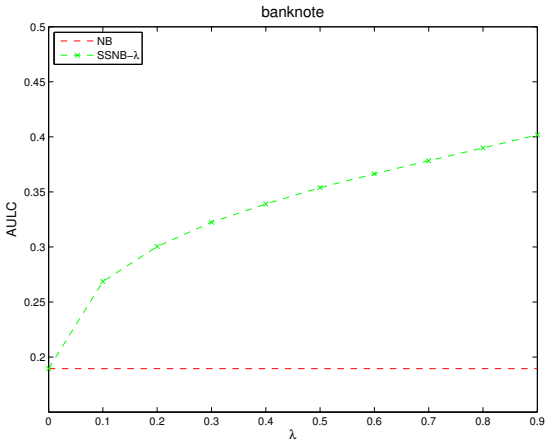
influence of unlabelled data does make some improvements as expected because of choosing the value of λ to maximise test set performance. The reason for that is that a biased model selection procedure for choosing λ is used because the benchmark datasets partition is just for training and testing set. Therefore, it is better to partition the benchmark datasets into training, validation, and test sets. Then the validation set is used to choose λ rather than having at the test set.



(a) SSNB-λ-cvs learning curve with varying value of λ



(b) SSNB-λ-cvs learning curve classifier with best value of λ



(c) The AULC result for the varying value of λ

Fig. 4.3 Comparison of the error rate learning curve for the NB and SSNB-λ-cvs classifiers for the banknote benchmark dataset.

4.4 Using a validation set to determine the contribution of labelled and unlabelled data

4.4.1 Design of experiments

In these experiments, we performed two sets of experiments to evaluate the performance of NB compared to the SSNB-λ-cvs classifier. A validation set is used to tune, λ, instead of minimizing the test set error rate. The former experiment used the 36 discrete datasets in Table 3.1 and the latter used 28 continuous datasets in Table 3.2 which were taken from UCI machine learning repository. The main purpose of these experiments, is to illustrate the nature of how to choose λ, which is a significant challenge.

The experimental design for both experiments is the same as Section 4.3. The only difference in these experiments is that instead of just randomly partitioning the datasets into training and test sets in each trial, the datasets was partitioned into training, validation, and test set. The purpose of adding a validation set is to avoid an over-fitting problem when tuning hyper-parameter λ. For each dataset, 60% was used for training the classifier with varying values of λ, 20% was used as a validation set to obtain optimal value of hyper-parameter, and 20% was held-out as a test set, used only to evaluate the classification error rate during the experiments.

4.4.2 Results

Table 4.3 and Table 4.4 show the results for 28 continuous and 36 discrete benchmark datasets respectively. The SSNB-λ-cvs classifier is run with different values of λ, which in some cases does help but not very much and was best on nine out of 28 continuous and 19 out of 36 discrete benchmark datasets. The result for the Wilcoxon signed rank test shows that there is no overall statistically significant difference between the NB and SSNB-λ-cvs classifiers.

#	Dataset	NB	SSNB	SSNB-λ-cvs	Optimal λ
1	banknote	1.514±0.0207	2.768±0.0193	1.514±0.0207	0
1	banknote	1.514±0.0207	2.768±0.0193	1.514±0.0207	0
2	Blood_transfusion	2.008±0.0268	2.500±0.0443	2.008±0.0268	0
3	bcc	0.522±0.0208	0.345±0.0169	**0.328±0.0169**	0.11
4	CMSC	*0.590±0.0158*	0.609±0.0165	***0.585±0.0177***	0.15
5	glass	1.841±0.0176	2.197±0.0168	1.841±0.0176	0
6	haberman	1.653±0.0295	1.866±0.0513	1.653±0.0295	0
7	ionosphere	1.331±0.0279	1.564±0.0290	1.331±0.0279	0
8	iris	0.400±0.0172	0.312±0.0141	**0.296±0.0148**	0.16
9	letter	4.087±0.0120	5.335±0.0070	4.087±0.0120	0
10	liver_disorder	2.622±0.0207	2.847±0.0221	2.622±0.0207	0
11	magic04	3.377±0.0174	4.697±0.0160	3.377±0.0174	0
12	musk1	2.127±0.0189	2.601±0.0239	2.127±0.0189	0
13	new_thyroid	0.339±0.0148	0.174±0.0150	**0.185±0.0197**	0.5
14	pendigits	2.073±0.0115	2.981±0.0256	2.073±0.0115	0
15	sleep	4.837±0.0153	7.046±0.0265	4.837±0.0153	0
16	vehicle	3.324±0.0181	3.727±0.0167	3.324±0.0181	0
17	vowel	1.901±0.0143	2.287±0.0130	1.901±0.0143	0
18	waveform_noise	2.498±0.0136	2.817±0.0252	2.498±0.0136	0
19	waveform	2.824±0.0141	3.359±0.0222	2.824±0.0141	0
20	wine	0.580±0.0214	0.286±0.0181	**0.220±0.0130**	0.49
21	arcene	1.950±0.0224	1.933±0.0215	**1.887±0.0241**	0.2
22	gisette	2.101±0.0126	4.518±0.0068	2.101±0.0126	0
23	madelon	4.005±0.0109	3.834±0.0159	**3.830±0.0159**	0.9
24	sonar	1.799±0.0237	2.095±0.0216	1.799±0.0237	0
25	spambase	2.180±0.0217	2.768±0.0256	2.180±0.0217	0
26	Synthetic	1.129±0.0181	1.097±0.0243	**1.086±0.0245**	0.6
27	vertebral	1.361±0.0229	1.348±0.0272	**1.323±0.0278**	0.1
28	diabetes	2.096±0.0220	2.417±0.0253	2.096±0.0220	0

Table 4.3 AULC for the NB, SSNB and SSNB-λ-cvs classifiers over 28 continuous benchmark datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ-cvs classifier is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier.

#	Dataset	NB	SSNB	SSNB-λ-cvs	Optimal λ
1	audiology	4.838±0.0354	5.191±0.0326	4.838±0.0354	0
2	balance_scale	2.828±0.0346	3.298±0.0288	2.828±0.0346	0
3	blogger	2.169±0.0495	2.443±0.0417	2.169±0.0495	0
4	breast_cancer	2.563±0.0427	2.885±0.0349	**2.503±0.0444**	0.07

Continued on next page

#	Dataset	NB	SSNB	SSNB-λ-cvs	Optimal λ
5	breastw	0.902±0.0280	0.302±0.0298	**0.399±0.0401**	0.86
6	car	2.930±0.0301	3.170±0.0451	**2.848±0.0470**	0.04
7	DNA	3.235±0.0242	2.205±0.0640	**2.371±0.0419**	0.45
8	flare1	4.068±0.0302	3.986±0.0335	4.068±0.0302	0
9	flare2	4.127±0.0332	4.048±0.0367	4.127±0.0332	0
10	hayes_roth	3.098±0.0349	3.482±0.0329	3.098±0.0349	0
11	house_votes	0.984±0.0296	1.067±0.0343	0.984±0.0296	0
12	kr_vs_kp	3.120±0.0213	4.645±0.0272	3.120±0.0213	0
13	led7	5.122±0.0227	5.028±0.0245	**5.084±0.0257**	0.31
14	led24	6.073±0.0149	5.560±0.0240	**5.480±0.0221**	0.18
15	lung_cancer	2.559±0.0505	2.536±0.0499	2.559±0.0505	0
16	lymphography	*2.096±0.0416*	2.267±0.0479	***2.095±0.0394***	0.25
17	marketing	9.229±0.0220	9.490±0.0199	9.229±0.0220	0
18	monk1_corrupt	*3.373±0.0244*	3.472±0.0285	***3.378±0.0259***	0.11
19	monk1_cross	*1.777±0.0219*	2.060±0.0306	***1.767±0.0224***	0.07
20	monk1_local	3.079±0.0341	3.521±0.0324	3.079±0.0341	0
21	monk1	3.035±0.0266	3.069±0.0313	**3.021±0.0271**	0.04
22	monk3_cross	1.707±0.0256	2.135±0.0380	1.707±0.0256	0
23	monk3_local	1.786±0.0292	2.683±0.0330	1.786±0.0292	0
24	monk3	1.720±0.0234	2.044±0.0353	1.720±0.0234	0
25	mushroom	1.754±0.0286	1.676±0.0391	**1.572±0.0354**	0.02
26	nursery	3.649±0.0183	3.858±0.0238	**3.526±0.0186**	0.04
27	primary_tumor	5.642±0.0293	5.719±0.0251	5.642±0.0293	0
28	promoters	1.688±0.0401	1.404±0.0395	**1.526±0.0369**	0.37
29	sls	*1.320±0.0322*	1.450±0.0389	***1.298±0.0338***	0.2
30	soybean_small	1.299±0.0459	0.910±0.0435	**1.158±0.0466**	0.77
31	soybean_large	6.411±0.0187	6.601±0.0195	6.411±0.0187	0
32	splice	3.323±0.0133	2.531±0.0518	**2.389±0.0400**	0.12
33	threeOf9	2.629±0.0330	2.895±0.0302	2.629±0.0330	0
34	titanic	2.939±0.0457	3.115±0.0480	**2.789±0.0586**	0.05
35	xd6	2.576±0.0284	2.673±0.0331	**2.554±0.0290**	0.1
36	zoo	*1.712±0.0446*	1.605±0.0476	***1.724±0.0447***	0.2

Table 4.4 AULC for the NB, SSNB, and SSNB-λ-cvs classifier over 36 discrete datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ-cvs classifiers is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier.

Figures 4.4 and 4.5 show the AULC for the NB, SSNB, and SSNB-λ-cvs over the 28 continuous and 36 discrete benchmark datasets respectively. Clearly, the SSNB-λ is

statistically superior compared to the SSNB, however, SSNB-λ is not statistically superior to NB.

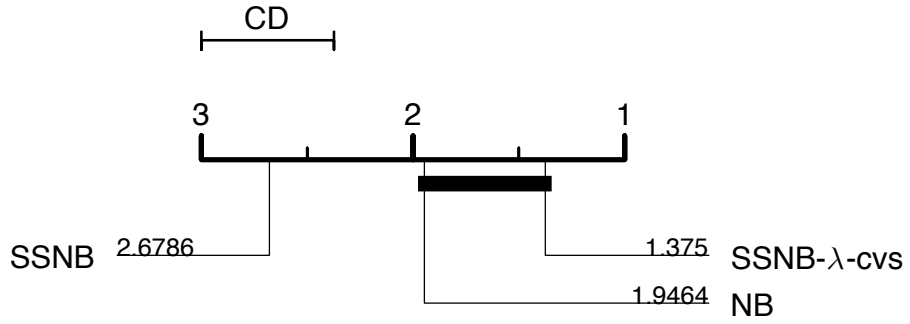


Fig. 4.4 Critical difference diagram for NB, SSNB, and SSNB-λ-cvs over 28 continuous benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar

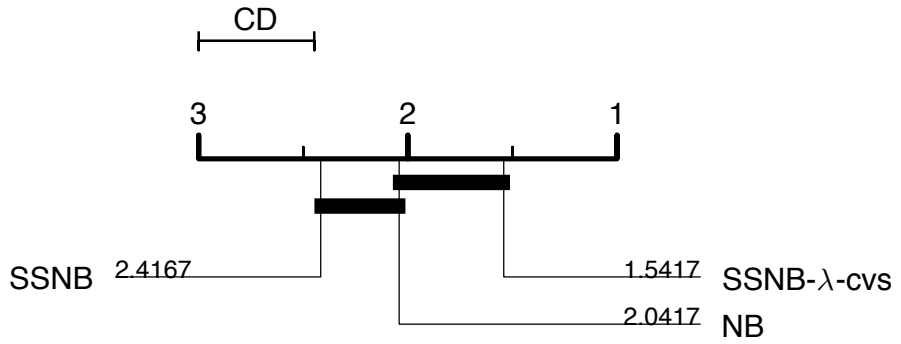
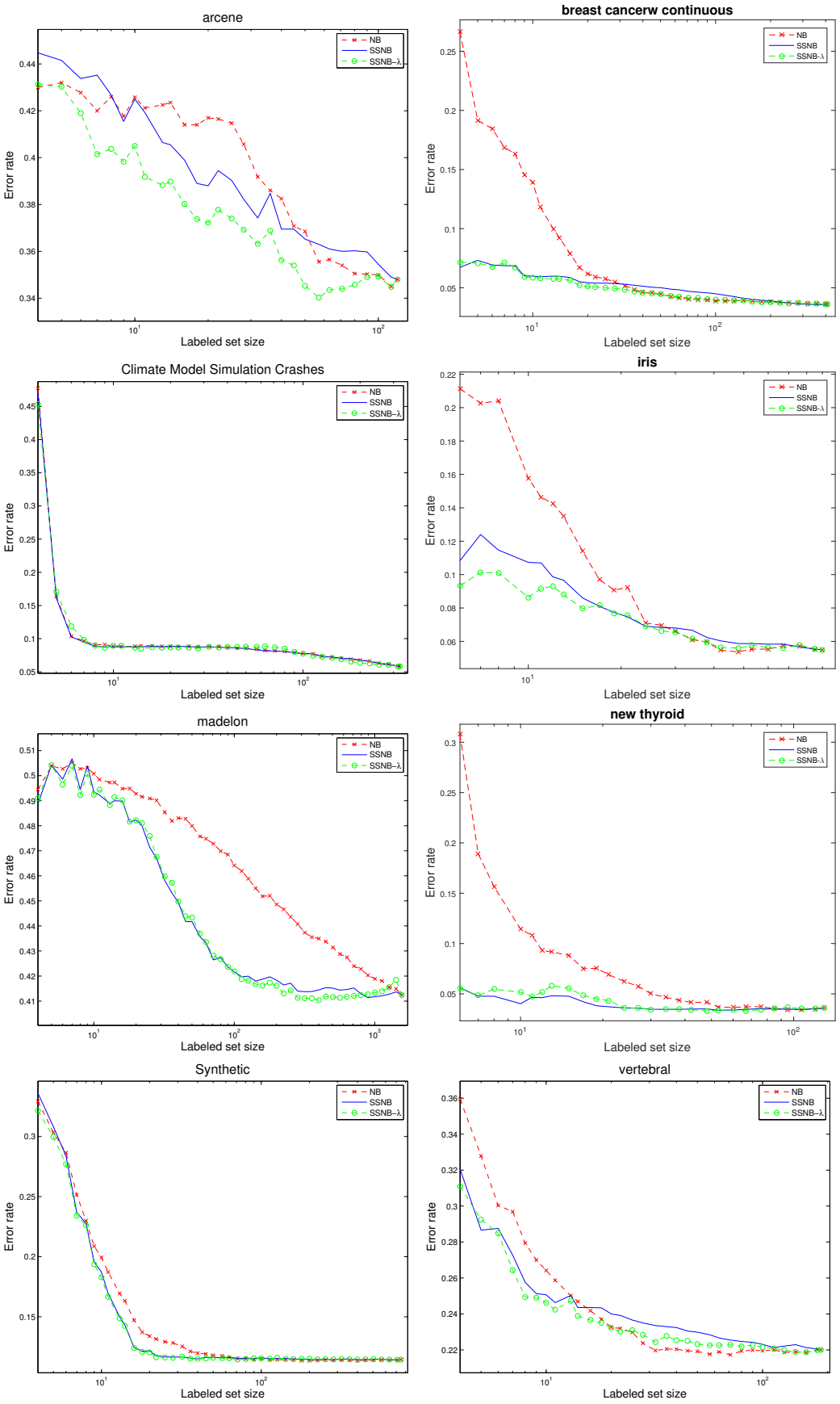


Fig. 4.5 Critical difference diagram for NB, SSNB, and SSNB-λ-cvs over 36 discrete benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar

According to the results from Table 4.3, the SSNB-λ-cvs classifier outperforms the NB classifier, for `breast-cancerw-continuous`, `iris`, `new-thyroid`, `wine`, `arcene`, `Climate-Model-Simulation-Crashes`, `madelon`, `Synthetic`, and `vertebral` benchmark datasets with continuous features. Figure 4.6 shows the average of the error rate learning curve results for the above nine datasets.


arcene
breast cancerw continuous
Climate Model Simulation Crashes
iris
madelon
new thyroid
Synthetic
vertebral
NB
SSNB
SSNB–λ
Error rate
Labeled set size

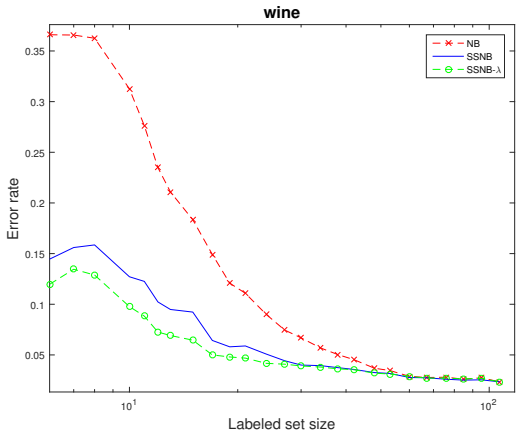
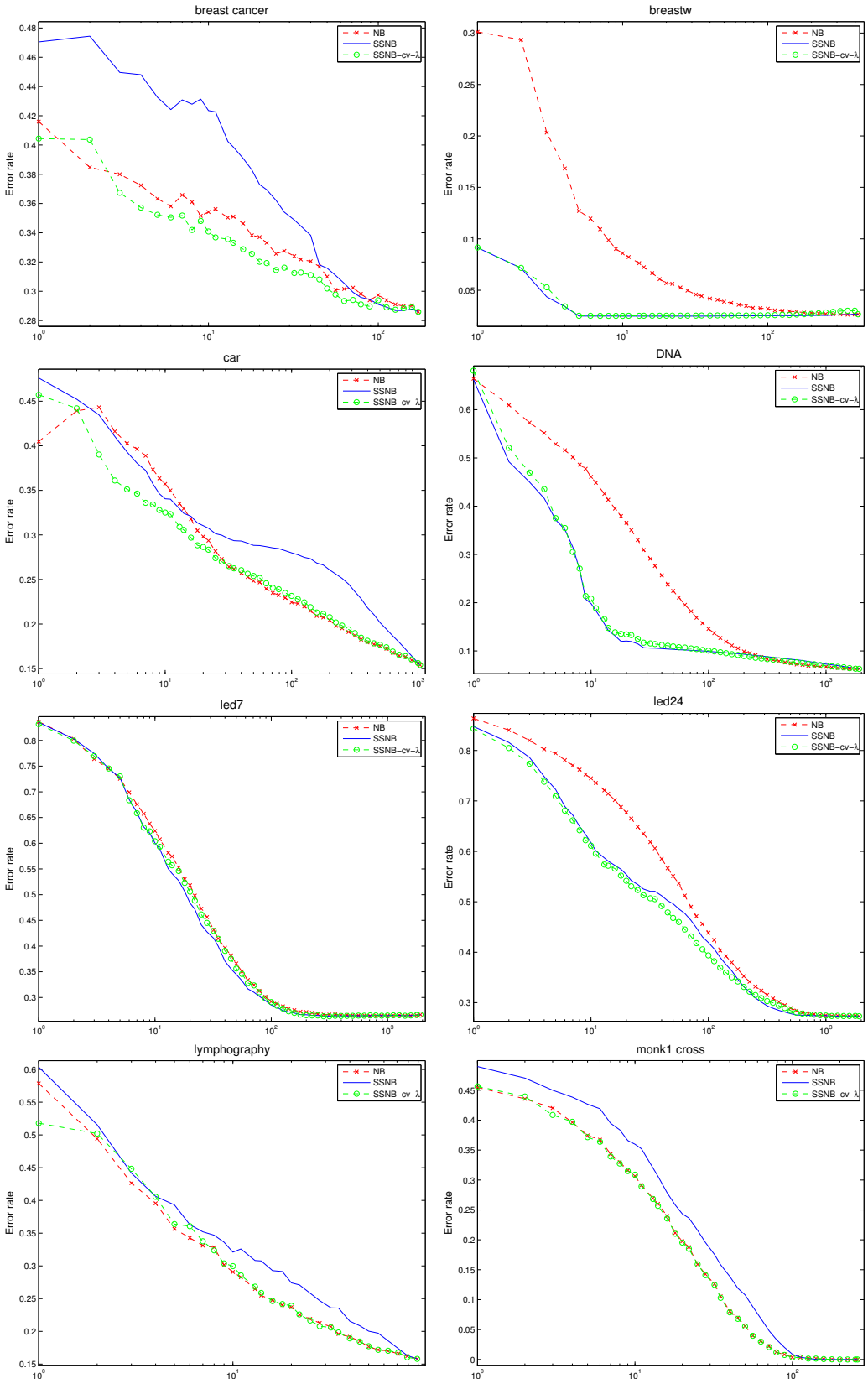
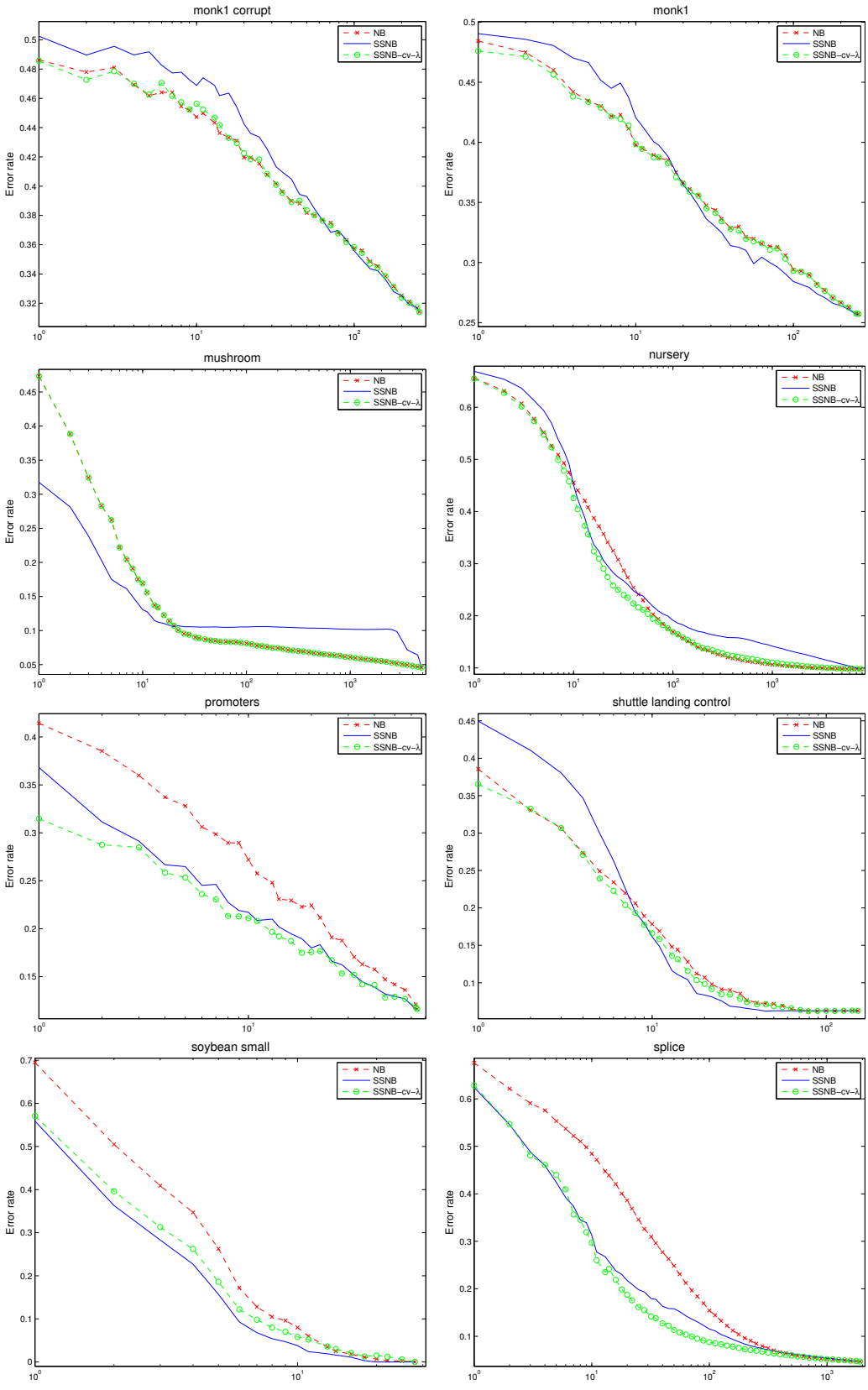


Fig. 4.6 Comparison of the error rate learning curve for NB, SSNB and SSNB-λ-cvs classifiers for nine continuous benchmark datasets.

As can be seen from Figure 4.6, the learning curve for the `Climate Model Simulation Crashes` benchmark dataset has nearly equivalent performance for all three classifiers. Furthermore, it is clear that the down weighting technique can improve NB classifier for the above nine benchmark datasets but the SSNB classifier without using down-weighting technique, already improved the NB classifier for these datasets. On the other hand, the benchmark datasets results for discrete features are presented in Table 4.4. The results show that the SSNB-λ-cvs classifier outperforms the NB classifier for 19 benchmark datasets, `breast-cancer`,`breastw`, `car`, `DNA`, `led7`, `led24`, `lymphography`, `monk1-cross`, `monk1-corrupt`, `monk1`, `monk1`, `nursery`, `promoters`, `shuttle-landing-control`, `xd6`, `soybean-small`, `splice`, `titanic`, and `zoo`. Figure 4.7 shows the average of the error rate learning curve results for the above 19 datasets.


breast cancer
breastw
car
DNA
led7
led24
lymphography
monk1 cross
NB
SSNB
SSNB−cv−λ
Error rate


monk1 corrupt
monk1
mushroom
nursery
promoters
shuttle landing control
soybean small
splice
NB
SSNB
SSNB−cv−λ
Error rate

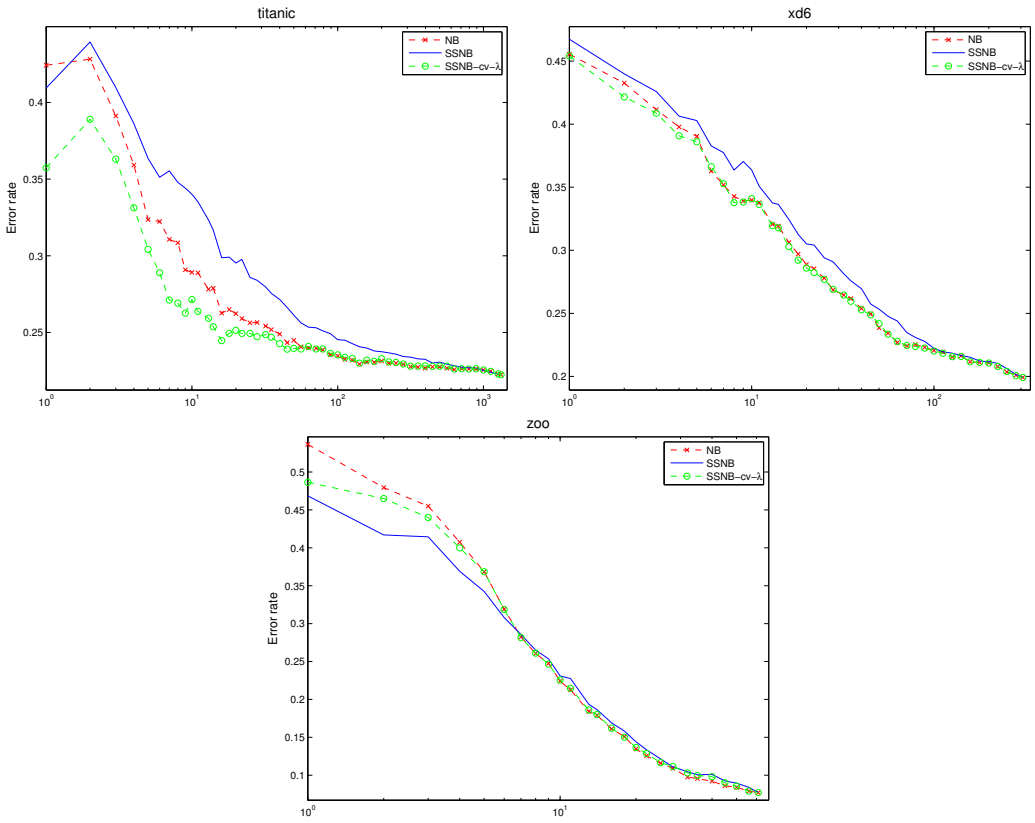


Fig. 4.7 Comparison of the error rate learning curve for NB, SSNB and SSNB-λ-cvs classifiers for 19 discrete benchmark datasets.

From the results of the learning curves shown, in Figure 4.7, the performance of the SSNB-λ-cvs classifier can be divided into two groups. The first group, where the SSNB-λ-cvs classifier can be slightly useful compared to the baseline which is NB classifier, on `lymphography`, `led7`, `monk1-cross`, `monk1-corrupt`, `monk1`, `mushroom`, `nursery`, `shuttle-landing-control`, `xd6`, and `zoo` benchmark datasets. In addition, there is no statistically significant difference between the AULC for both classifiers on the `lymphography`, `monk1-corrupt`, `monk1-cross`, `shuttle-landing-control`, and `zoo` benchmark datasets. The second group is where the performance of the SSNB-λ-cvs rather than that of the NB classifier is better on the rest of the learning curves in Figure 4.7, especially on `breast-cancer`, `breastw`, `car`, `DNA`, `led24`, `promoters`, `soybean-small`, `splice`, and `titanic`. However, the performance of the SSNB classifier is better than that of the NB classifier on these

datasets.

We expected down-weighting would make some improvement because the unbiased method was used, which partitions each dataset into training, validation, and testing set, but it appears the chosen values of λ give very similar results. The results of this experiment imply that SSNB-λ-cvs is not helpful on more than half of benchmarks datasets, suggesting zero for the optimal value of λ. The results for these benchmark datasets in Table 4.3 and Table 4.4 that suggests a non-zero value for the optimal value of λ, as the SSNB classifier is already better than NB classifier.

Only 20% of the original set is used for the validation set to get an unbiased estimator. This is the reason we get high variance compared to the biased estimator method as this is a relatively small amount of data. In practice, when there are very few labelled examples, it is hard to get a good estimate. However, these experiments show that in principle, using unlabelled data does help to reduce classification error rate slightly which is not statistically significant different compared to the based classifier. We used 20% of the original set as validation set, so the main question here is: if we have more labelled data, can we use any other better methods to choose λ?

4.5 Final experiments for contribution of labelled and unlabelled data

4.5.1 Design of experiments

The more data available for cross-validation, the better the estimate of the probability an error can be obtained. When you have a few labelled patterns, the cross-validation estimate is not a very reliable indicator. So, it would be better to use Leave-one-out-Cross-validation but unfortunately Leave-one-out-Cross-validation has a high variance. Repeating the experiment with different samples might give very different values, while the result with normal k fold

cross-validation tends to be a more reliable performance indicator. So, in the final experiment we use k-fold cross-validation method to choose λ in the semi-supervised learning, denoted by SSNB-λ-mcv classifier. The best way for using cross-validation adapts the number of folds according to the amount of data available. In order to have the minimum number of examples for each class in each training fold, but increasing k up to say 10 as the labelled data set gets bigger to reduce the bias and variance of the cross-validation estimate.

4.5.2 Results

Table 4.5 shows the outcome of a comparison of NB, SSNB, and SSNB-λ-mcv classifiers for 28 continuous benchmark datasets. The SSNB-λ-mcv classifiers outperform the NB classifier on eight of the 28 benchmark datasets, which are `breast-cancerw-continuous`, `iris`, `new-thyroid`, `wine`, `arcene`, `madelon`, `spambase`, and `Synthetic`. The result for the Wilcoxon signed rank test shows that there is no statistically significant difference between NB and SSNB-λ-mcv when using cross-validation to choose λ.

#	Dataset	NB	SSNB	SSNB-λ-mcv
1	banknote	1.184±0.0167	2.473±0.0166	1.222±0.0179
2	Blood_transfusion	1.701±0.0201	2.118±0.0307	1.734±0.0185
3	bcc	0.310±0.0108	0.284±0.0099	**0.284±0.0097**
4	CMSC	0.492±0.0105	0.466±0.0112	0.506±0.0121
5	glass	1.436±0.0152	1.663±0.0123	1.486±0.0152
6	haberman	1.397±0.0234	1.603±0.0432	1.413±0.0250
7	ionosphere	0.999±0.0232	1.352±0.0234	1.030±0.0235
8	iris	*0.216±0.0118*	0.212±0.0107	***0.210±0.0117***
9	letter	3.154±0.0095	4.697±0.0070	3.209±0.0098
10	liver_disorder	2.270±0.0181	2.499±0.0184	2.324±0.0177
11	magic04	3.090±0.0133	4.358±0.0099	3.117±0.0144
12	musk1	1.743±0.0200	2.262±0.0210	1.781±0.0206
13	new_thyroid	0.173±0.0090	0.133±0.0087	**0.147±0.0087**
14	pendigits	1.494±0.0093	2.064±0.0133	1.535±0.0097
15	sleep	4.366±0.0125	6.500±0.0268	4.406±0.0134
16	vehicle	2.880±0.0143	3.307±0.0133	2.934±0.0151
17	vowel	1.367±0.0105	1.763±0.0090	1.410±0.0103
18	waveform_noise	1.956±0.0109	2.410±0.0238	2.001±0.0127

Continued on next page

#	Dataset	NB	SSNB	SSNB-λ-mcv
19	waveform	2.297±0.0109	2.930±0.0191	2.337±0.0118
20	wine	0.232±0.0098	0.173±0.0108	**0.155±0.0086**
21	arcene	1.694±0.0181	1.600±0.0206	**1.626±0.0195**
22	gisette	1.607±0.0114	2.579±0.0263	1.649±0.0123
23	madelon	3.617±0.0086	3.439±0.0153	**3.477±0.0124**
24	sonar	1.439±0.0230	1.764±0.0192	1.504±0.0233
25	spambase	1.887±0.0154	2.044±0.0204	**1.870±0.0148**
26	Synthetic	0.877±0.0141	0.856±0.0164	**0.865±0.0147**
27	vertebral	1.094±0.0164	1.148±0.0187	1.109±0.0160
28	diabetes	1.757±0.0172	2.072±0.0191	1.800±0.0151

Table 4.5 AULC for the NB, SSNB and SSNB-λ-mcv classifiers over 28 continuous benchmark datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ-mcv classifier is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier.

Table 4.6 shows the result of a comparison of NB, SSNB, and SSNB-λ-mcv classifiers for 36 discrete benchmark datasets. The SSNB-λ-mcv classifiers outperforms the NB classifier on 14 of the 36 benchmark datasets, which are, `breastw`, `car`, `DNA`, `flare2`, `led7`, `led24`, `lung-cancer`, `mushroom`, `nursery`, `promoters`, `shuttle-landing-control`, `soybean-small`, `splice`, and `titanic`. The results of the Wilcoxon signed rank test shows that there is no statistically significant difference between NB and SSNB-λ-mcv when using cross-validation to choose λ.

#	Dataset	NB	SSNB	SSNB-λ-mcv
1	audiology	4.064±0.0298	4.447±0.0282	4.086±0.0290
2	balance_scale	2.329±0.0269	2.665±0.0211	2.390±0.0271
3	blogger	1.765±0.0349	2.056±0.0318	1.859±0.0326
4	breast_cancer	*2.251±0.0330*	2.501±0.0352	*2.261±0.0328*
5	breastw	0.585±0.0151	0.229±0.0131	**0.457±0.0189**
6	car	2.546±0.0247	2.775±0.0336	**2.517±0.0262**
7	DNA	2.573±0.0174	1.546±0.0527	**1.694±0.0362**
8	flare1	3.424±0.0325	3.393±0.0300	3.478±0.0313
9	flare2	3.523±0.0263	3.425±0.0273	**3.477±0.0245**
10	hayes_roth	2.567±0.0347	2.972±0.0273	2.699±0.0328
11	house_votes	0.849±0.0164	0.922±0.0158	0.863±0.0164
12	kr_vs_kp	2.652±0.0192	4.274±0.0241	2.745±0.0200

Continued on next page

#	Dataset	NB	SSNB	SSNB-λ-mcv
13	led7	4.377±0.0217	4.273±0.0212	**4.342±0.0207**
14	led24	5.344±0.0135	4.850±0.0191	**4.884±0.0185**
15	lung_cancer	*2.083±0.0369*	1.979±0.0378	***2.073±0.0355***
16	lymphography	1.662±0.0320	1.812±0.0316	1.687±0.0312
17	marketing	8.632±0.0175	8.957±0.0156	8.670±0.0169
18	monk1_corrupt	2.983±0.0224	3.058±0.0256	3.035±0.0228
19	monk1_cross	1.367±0.0212	1.614±0.0233	1.438±0.0195
20	monk1_local	2.736±0.0260	3.140±0.0277	2.808±0.0258
21	monk1	2.650±0.0237	2.664±0.0275	2.680±0.0247
22	monk3_cross	*1.266±0.0239*	1.795±0.0389	*1.285±0.0264*
23	monk3_local	1.312±0.0204	2.231±0.0352	1.405±0.0210
24	monk3	1.277±0.0211	1.554±0.0209	1.330±0.0194
25	mushroom	1.337±0.0188	1.399±0.0217	**1.275±0.0213**
26	nursery	3.012±0.0157	3.114±0.0224	**2.925±0.0181**
27	primary_tumor	4.921±0.0264	4.995±0.0236	4.961±0.0243
28	promoters	1.288±0.0298	0.983±0.0281	**1.117±0.0288**
29	sls	*0.998±0.0219*	1.008±0.0335	***0.979±0.0248***
30	soybean_small	0.742±0.0413	0.533±0.0377	**0.728±0.0411**
31	soybean_large	5.627±0.0181	5.825±0.0185	5.677±0.0177
32	splice	2.689±0.0153	2.037±0.0458	**2.000±0.0312**
33	threeOf9	2.279±0.0249	2.503±0.0226	2.349±0.0242
34	titanic	2.568±0.0358	2.769±0.0325	**2.527±0.0343**
35	xd6	2.211±0.0243	2.328±0.0243	2.254±0.0231
36	zoo	1.213±0.0369	1.155±0.0379	1.227±0.0367

Table 4.6 AULC for the NB, SSNB, and SSNB-λ-mcv classifier over 36 discrete datasets from the UCI repository. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for the SSNB-λ-mcv classifiers is better than for the NB classifiers. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifier is better or equivalent to the AULC for the SSNB-λ classifier.

Following the recommendation of Demšar [25], Friedman's test is used to determine whether there exist statistical significant differences in the mean ranks of the NB, SSNB, and SSNB-λ-mcv classifiers. There is no statistical difference between the mean ranks of the three classifiers for both discrete and continuous benchmark datasets. A graphical illustration is provided for these results which is a critical difference diagram as presented in Figure 4.8 and 4.9.

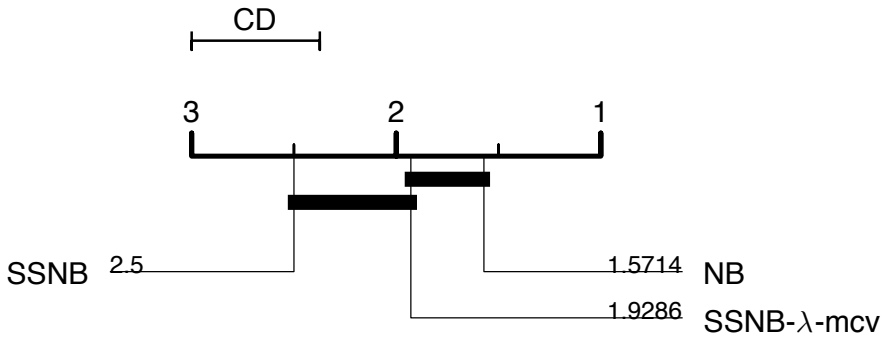


Fig. 4.8 Critical difference diagram for the NB, SSNB, and SSNB-λ-mcv over 28 continuous benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar

Figure 4.8 and 4.9 shows the mean rank for the NB, SSNB, and SSNB-λ-mcv classifiers over the 28 continuous and 36 discrete benchmark datasets respectively. Clearly, the SSNB-λ-mcv classifier does not statistically outperform the NB and SSNB classifiers.

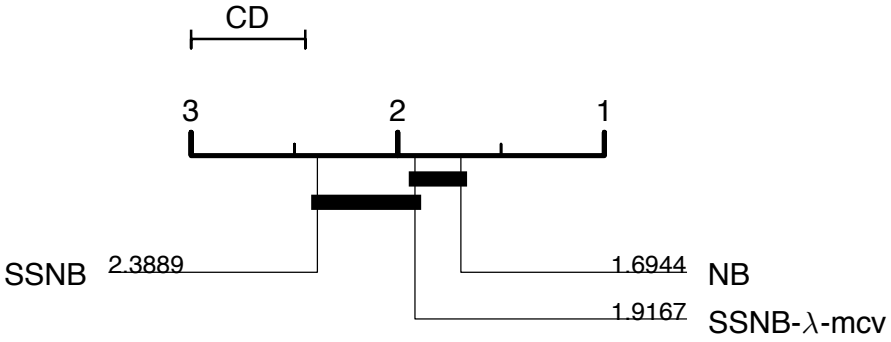
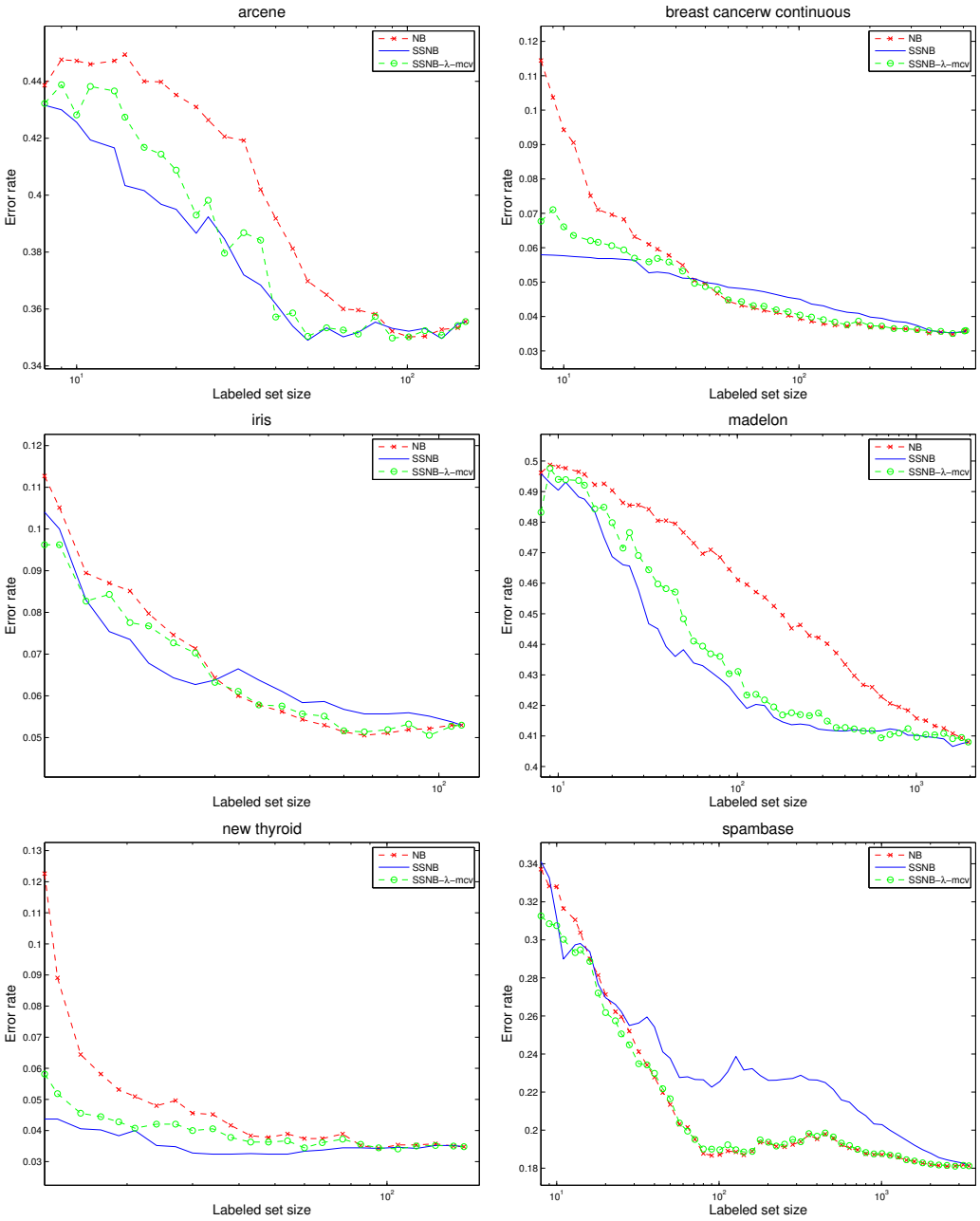


Fig. 4.9 Critical difference diagram for the NB, SSNB, and SSNB-λ-mcv over 36 discrete benchmark datasets. It shows that there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar

The results from Table 4.5 and 4.6, show that the tuning λ does help only for eight out of the 28 continuous benchmark datasets and 14 out of the 36 discrete benchmark datasets. It is useful if the average of the error rate learning curve can be shown for these datasets that choose non zero values for λ. Figures 4.10 and 4.11 show the average of the error rate

learning curves for these continuous and discrete benchmark datasets respectively. From Figure 4.10, it is clear that the SSNB-λ-cvs classifier performs slightly better than NB classifier on `breast-cancerw-continuous`, `iris`, `spambase`, and `Synthetic` benchmark dataset. However, the performance of the SSNB-λ-cvs classifier is better compared to NB classifier on `new-thyroid`, `wine`, `arcene`, and `madelon` benchmark datasets.



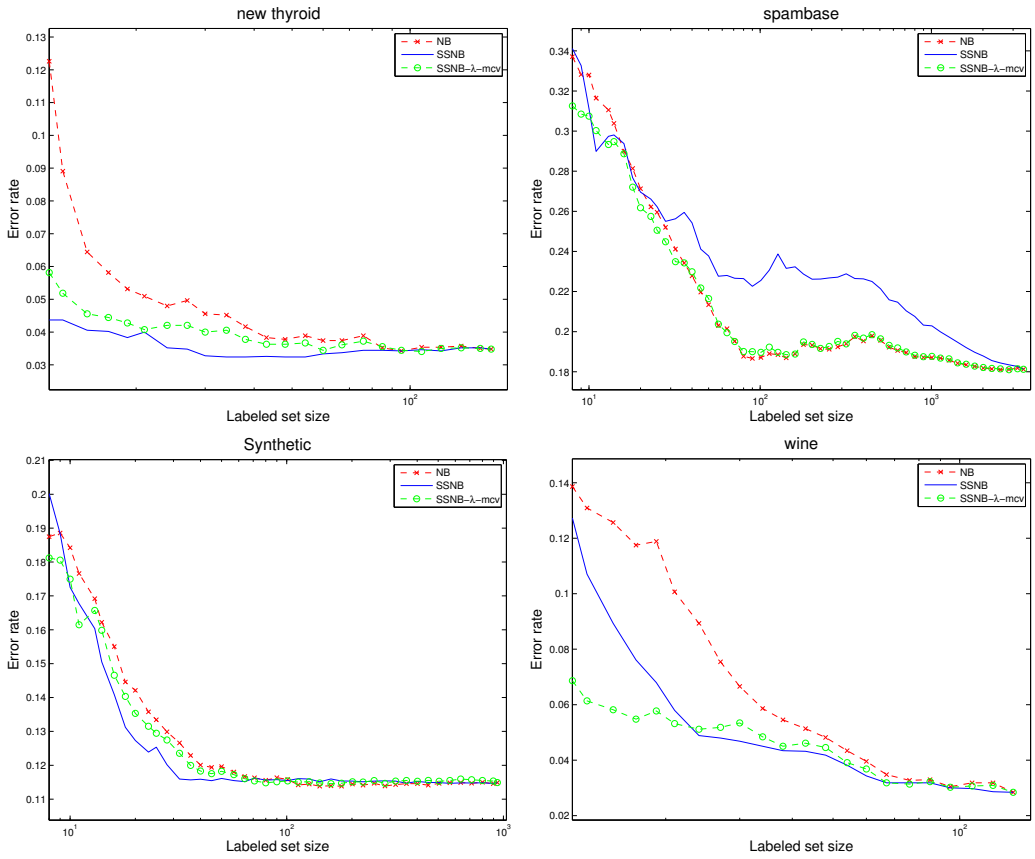
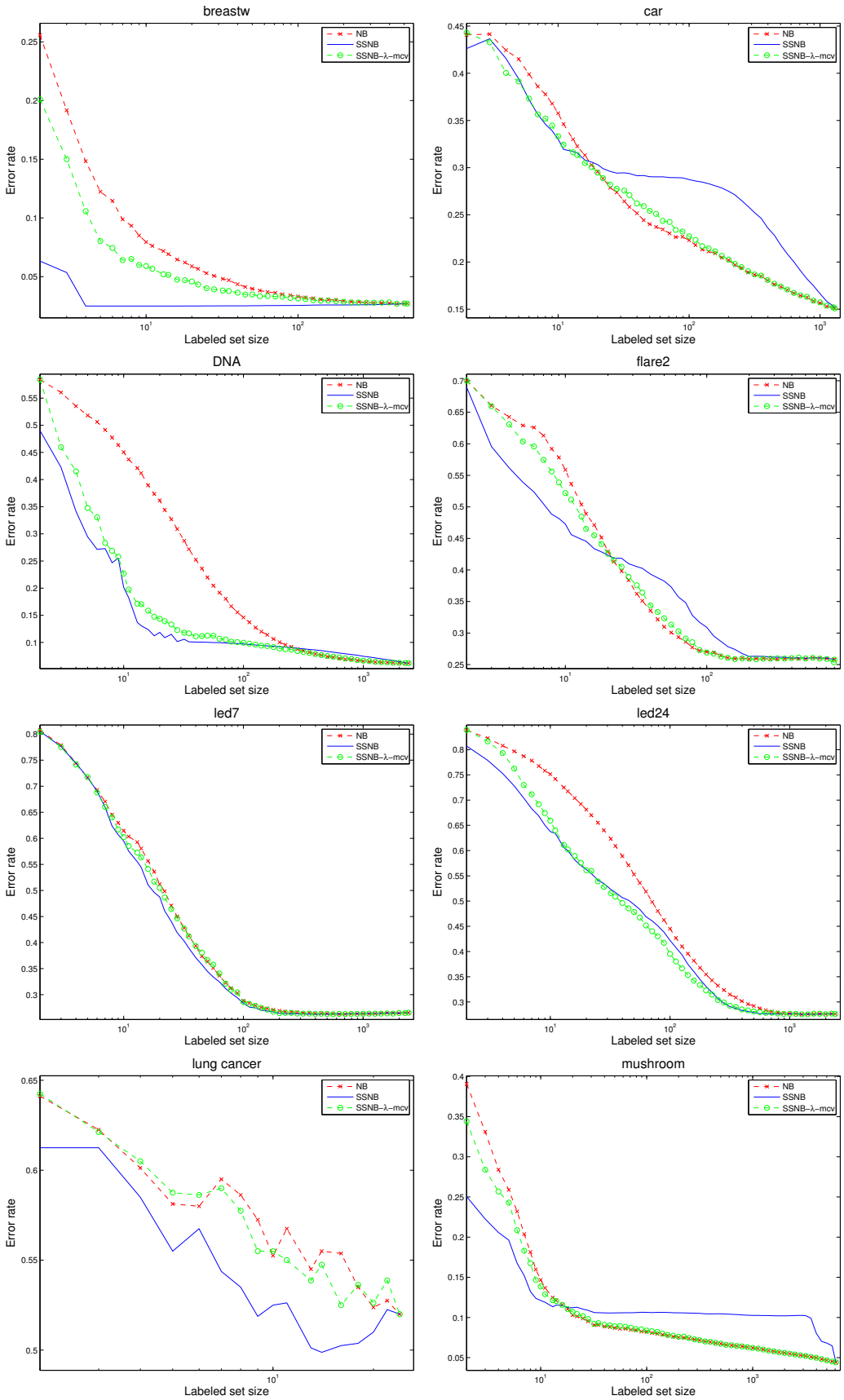


Fig. 4.10 Comparison of the error rate learning curve for NB, SSNB and SSNB-λ-cvs classifiers for eight continuous benchmark datasets.

The average learning curves shown in Figure 4.11 suggest that the SSNB-λ-cvs classifier can slightly improve on the NB classifier on `car`,`flare2`, `led7`,`lung-cancer`, `mushroom`, `nursery`, `shuttle-landing-control`, and `soybean-small` datasets. The SSNB-λ-cvs classifier can improve on the NB classifier just on `breastw`, `DNA`, `led24`, `promoters`, `splice`, and `titanic` datasets.


breastw
car
DNA
flare2
led7
led24
lung cancer
mushroom
NB
SSNB
SSNB–λ–mcv
Error rate
Labeled set size

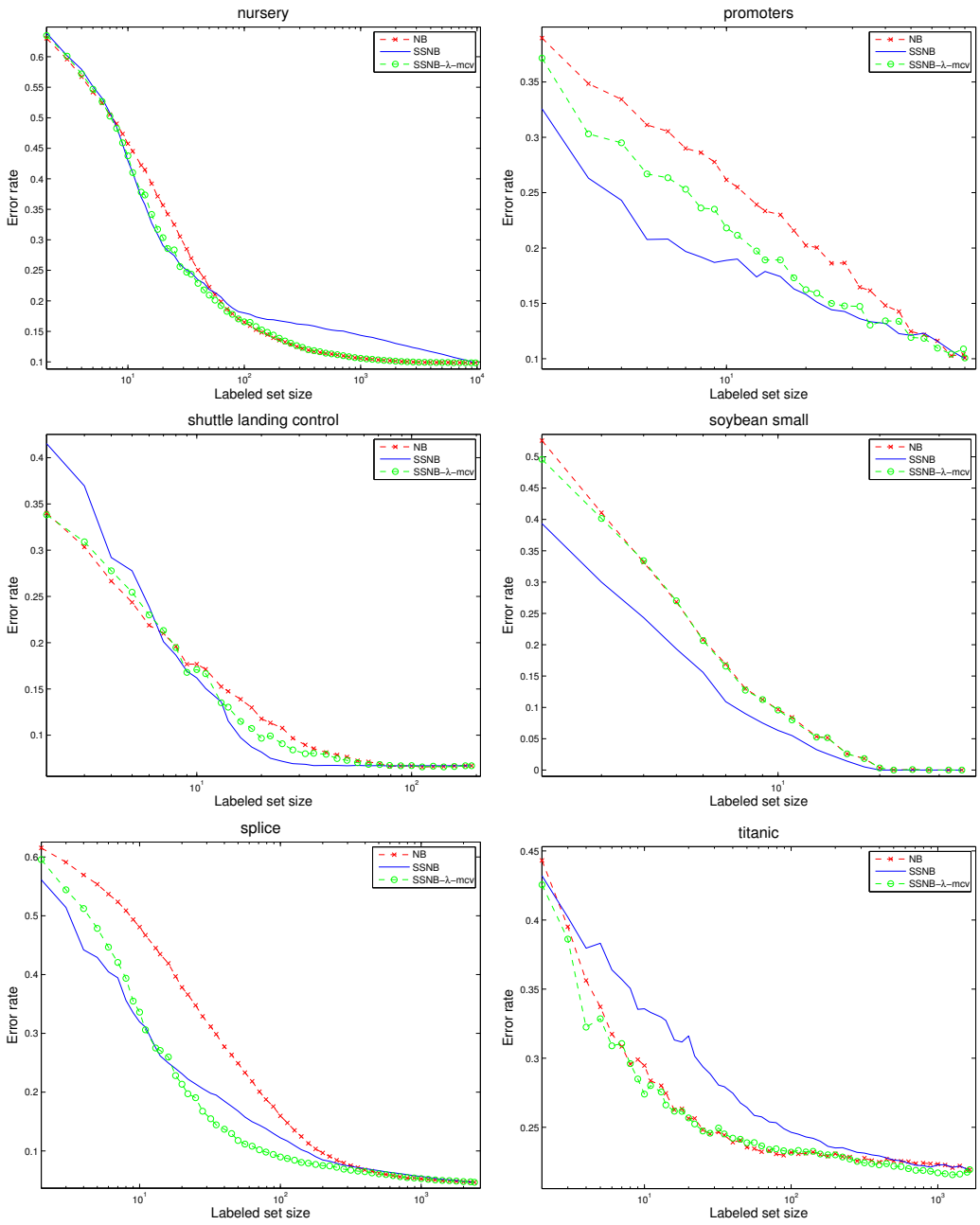


Fig. 4.11 Comparison of the error rate learning curve for NB, SSNB and SSNB-λ-cvs classifiers for 14 discrete benchmark datasets.

Tuning the value of λ does help in a few cases, but not very much and it quickly become worse. Thus we reach the conclusion that the SSNB-λ-mcv makes performance worse in general and tuning the different value of λ does not seem to really help.

4.6 Conclusions

In this chapter, we test whether down-weighting the unlabelled data can improve classification performance against 64 UCI benchmark datasets, 36 of these benchmark datasets have discrete features and the rest have continuous features. We investigated methods to choose the optimal value of λ, which is the main research question in this chapter. For a starting point, we used the train-test partition method. In this method for choosing the value of λ, using the test partition performance is an optimistically biased protocol. The weighting factor seems to decrease the sensitivity to unlabelled data. In order to choose the value of λ in an unbiased way, the λ factor can be chosen via a validation set. In this experiment, down-weighting the unlabelled data technique does not substantially improve the base classifier at all. In addition, instead of using part of the labelled data to choose the value of λ, we prefer to use it for training purposes. In practice there are two other ways of choosing λ: k-fold cross-validation and leave one out cross-validation. However, the k-fold cross-validation protocol does not reliably indicate the best value of λ because the amount of labelled patterns is very small. On the other hand, leave one out cross-validation protocol is unreliable because it has a high variance. If the experiment is repeated with different sample datasets, it might obtain very different values. Therefore, we used a method alternatively between k-folds cross-validation and leave one out cross-validation depending on the size of the labelled training. As mentioned in Section 4.5, in order to have the minimum number of examples of each class in each training fold, but increasing k up to say 10 as the labelled data gets bigger to reduce the bias and variance of the cross-validation estimate. Finally, we conclude that none of these model selection methods for choosing the value of λ can substantially improve the baseline classifier.

Chapter 5

The problem of class frequency bias in the predicted labels for unlabelled examples

As described in chapter 3, the use of unlabelled data may degrade the classification performance of semi-supervised learning methods when the assumptions of the baseline model are not valid. In practice, the naïve Bayes classifier often provides poor estimates of the probability of class membership due to the assumption of independence between input features. As a result, the naïve Bayes classifier tends to be overconfident, generating posterior probabilities with scores either very close to 0 or 1, tending to produce poorly calibrated probabilities of class membership. This is likely to be important for EM based semi-supervised learning as it relies on the estimated class probability for the unlabelled data, in the E-step.

A classifier is well-calibrated if the average of the predicted class probability, $p(y = c|x;\theta)$, is approximately equal to the true class proportion. Unfortunately, the naïve Bayes classifier does not generally provide well-calibrated predictions, due to overconfidence in its predicted class probabilities. Therefore, it is possible that this is one of the reasons why the unlabelled data does not improve the classification performance and the semi-supervised naïve Bayes

classifier may provide uncalibrated predicted class probabilities for unlabelled data too. In this case, recalibrating (correcting) the predicted class probabilities for unlabelled data may improve the performance of semi-supervised naïve Bayes classifier. Thus, the predicted class probabilities can be transformed to another domain which might discover better quality of predicted class.

In this chapter we begin by demonstrating overconfidence in the output of the naïve Bayes classifier such that it provides poorly calibrated predictive class probabilities for EM algorithm based semi-supervised naïve Bayes classifiers. Consequently, this problem leads to providing a bias between the averages of the predicted class probabilities and the true class proportion. Then, we introduce different transformation methods in order to provide better calibrated predictive class probabilities in the E-step stage of the EM algorithm. Finally, the benefit of these transformation techniques with semi-supervised naive Bayes classifier are demonstrated using benchmark datasets from the UCI repository.

5.1 Overconfidence generated by uncalibrated probability classes

In principle, the naïve Bayes classifier provides probability estimates for class membership that range between 0 and 1, but it can give poor estimates of the class probability membership, i.e. uncalibrated predicted class probabilities. In this section we show why the naïve Bayes classifier gives poorly calibrated estimates of the probability of class membership using a simple example. Assume Table 5.1 shows the results of twenty matches of a chess player, who was playing ten matches with black and the rest with white. In this example, the match colour is an feature and the target is the match result.

white				black			
1	won	6	won	11	lost	16	lost
2	lost	7	lost	12	lost	17	won
3	won	8	draw	13	draw	18	lost
4	lost	9	won	14	lost	19	draw
5	won	10	won	15	lost	20	lost

Table 5.1 The twenty matches of a chess player used as a training data for naive Bayes classifier.

We use the naïve Bayes classifier to predict whether the chess player won, drew or lost based on this data, via equation 5.1.

$$p(y=c|x;\theta) \;=\; \frac{p(x|y;\theta)\; p(y;\theta)}{\sum_{k=1}^{C} p(x|y=k;\theta)\; p(y=k;\theta)}. \tag{5.1}$$

The prior probabilities $p(y;\theta)$ for the illustrative example are;

$$p(won)=\tfrac{7}{20}=0.35, \quad p(draw)=\tfrac{3}{20}=0.15, \quad p(loss)=\tfrac{10}{20}=0.50$$

The condition probabilities $p(x|y;\theta)$ for the feature values;

$$p(white|won)=\tfrac{6}{7}=0.86, \qquad p(black|won)=\tfrac{1}{7}=0.14,$$

$$p(white|draw)=\tfrac{1}{3}=0.34, \qquad p(black|draw)=\tfrac{2}{3}=0.66,$$

$$p(white|loss)=\tfrac{3}{10}=0.30, \qquad p(black|loss)=\tfrac{7}{10}=0.70.$$

Suppose we are interested in finding the posterior probabilities $p(y=c|x;\theta)$ for class *won*;

$$p(won|white)=0.60, \quad p(won|black)=0.40.$$

Now, we duplicate the feature (matches), which include white and black;

$$p(won|white,white)=0.85, \quad p(won|black,black)=0.15.$$

Then, we triplicate the matches;

$$p(won|white,white,white)=0.93, \quad p(won|black,black,black)=0.07.$$

As we can see from the above example, duplicating and triplicating the same feature violates the naïve Bayes classifier assumption as a copy of an feature is perfectly correlated with the original, and the posterior probabilities of the class *won* tend to 0 or 1, which provides an overconfident output. The main reason for this overconfidence is the likelihood that gets the double and triple count. The likelihood is then maximised. Thus, the poorly calibrated predicted class probability is produced by the naïve Bayes classifier.

5.2 Effect of overconfident classification on EM based SSNB

In the previous section, poor calibration of predicted class probabilities for labelled data are generated by the naïve Bayes classifier due to the overconfidence problem. Then, the EM algorithm may place too much confidence in the predicted class probabilities for the unlabelled data. Hence, the semi-supervised naïve Bayes classifier degrades performance. The question is how the EM algorithm based semi-supervised naïve Bayes classifier is affected by the overconfidence problem with the naïve Bayes classifier.

In this section, the predicted class probabilities of the unlabelled data and true class proportion can be visualized in order to evaluate the predicted class probabilities for unlabelled data. The average value of the predicted probability for the positive class for the unlabelled data for 100 replications in the Mushroom benchmark dataset, is plotted as a function of the amount of labelled data, Figure 5.1. In this Figure, a clear difference (bias) between the true positive class proportion, which is a green line, and the average value of the predicted class probabilities of positive class for the unlabelled data can be seen. This bias is unstable until approximately training 30 labelled patterns are available to the naïve Bayes classifier, but later the bias remains stable.

This result suggest that the EM algorithm is affected by the overconfidence problem with the naïve Bayes classifier. Then, solving this bias issue in the Mushroom dataset might reduce the error rate. One of the possible solutions is to force the mean value of the predicted class

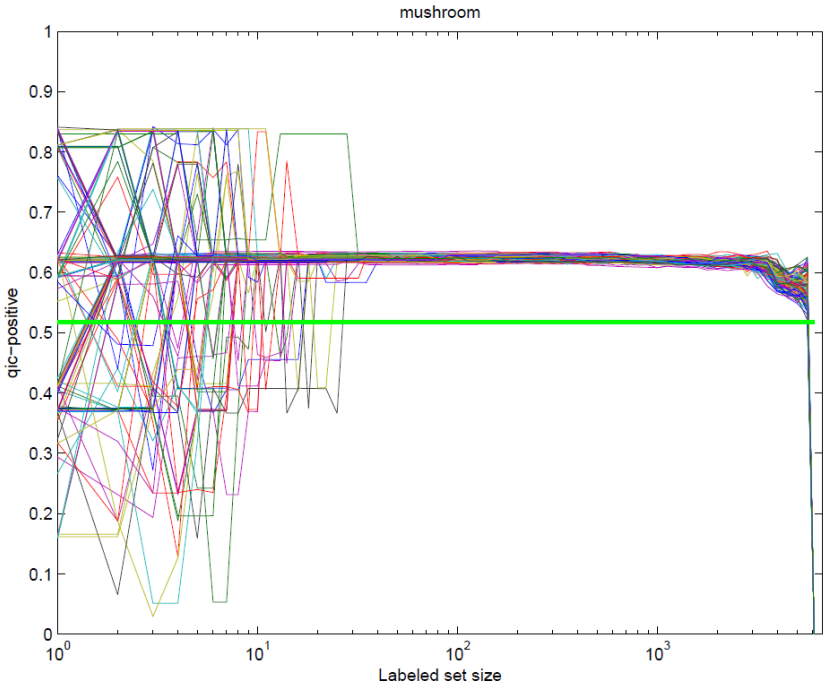


Fig. 5.1 The average value of the predicted probabilities of the positive class for the unlabelled data for 100 replications of the Mushroom benchmark dataset.

probabilities of the unlabelled data, q_{ic}, to the correct value (assuming the correct value is known). So, rather than trusting the initial values of the predicted probabilities of class distribution for the unlabelled data, q_{ic}, by adjusting them slightly we hope to make them better for the subsequent iteration.

5.3 Logit transformation function for predicted class probabilities

In the previous section, Figure 5.1 shows the discrepancy is produced between the predicted and true class, because the basic EM algorithm based semi-supervised learning is more likely to produce uncalibrated predicted probabilistic labels. Thus, the semi-supervised naïve Bayes classifier is more likely to give worse performance results rather than better. To overcome this

problem, we recalibrate the uncalibrated predicted labels for the unlabelled patterns between the E-step and the M-step during each iteration of the EM algorithm in order to obtain the correct mean value. The basic idea is to shift all the predicted probability labels of unlabelled patterns to the level approximately equal to that class distribution, whilst constraining them to be within the range 0 to 1.

The inverse sigmoid function is used to transform the predicted class probabilities of the unlabelled patterns in advance in order to shift the probability values. The inverse sigmoid function gives the log-odds ratio, r_{ic}, for binary classification so in this work, we consider only binary classification tasks, where $y_i \in \{-1,+1\}$ is an indicator variable such that $y_i = +1$ if the i^{th} pattern is drawn from the positive and $y_i = -1$ if drawn from the negative class.

Let the proportion of patterns in the positive class be denoted by θ and q_{ic_+} be the predicted probabilistic label of the ith unlabelled pattern belonging to the positive class. We believe that there is a *bias* between θ and the mean q_{ic_+} value. Correcting this bias by recalibrating the q_{ic} values might improve classification performance. The whole calibration process is given below:

The predicted class probabilities of the unlabelled patterns belonging to the positive class, q_{ic_+} where $i = l+1,...,l+u$, are transformed into other real value domain, r_{ic_+} where $i = l+1,...,l+u$, using the inverse sigmoid function.

$$r_{ic_+} \quad = \quad -log\left(\frac{1-q_{ic_+}}{q_{ic_+}}\right)$$

In the case, r_{ic_+} has a real value in the range $[-\infty,+\infty]$, for all $i = l+1,...,l+u$ such that;

$$r_{ic_+} = \begin{cases} (0,+\infty], & \text{if} \quad q_{ic_+} > 0.5, \\ 0, & \text{if} \quad q_{ic_+} = 0.5, \\ [-\infty,0), & \text{if} \quad q_{ic_+} < 0.5. \end{cases} \tag{5.2}$$

We then normalise the real values, r_{ic_+}, by dividing each value by the total number of unlabelled data, u ;

$$r_{ic_+} = -\frac{r_{ic_+}}{u}.$$

We apply the sigmoid function to convert the average value of the r_{ic_+} corresponding to the probability values q_{ic_+} and then plot against the true proportion of positive class to see whether this transformation can approximately give the correct mean values,

$$\begin{aligned} \tilde{q}_{ic_+} &= \frac{1}{1+exp(-\bar{r}_{ic_+})}; \quad \text{where} \quad \bar{r}_{ic_+} = \sum_{i=l+1}^{l+u} r_{ic_+}. \\ \tilde{q}_{ic_-} &= 1 - \tilde{q}_{ic_+}. \end{aligned}$$

Before the next iteration in the estimation of the model parameters, the sigmoid function is applied in order to convert the log-odds values r_{ic} back into probability values, q_{ic}, that will guarantee $0 \leq q_{ic} \leq 1$,

$$\begin{aligned} q_{ic_+} &= \frac{1}{1+exp(-r_{ic_+})}, \\ q_{ic_-} &= 1 - q_{ic_+}. \end{aligned}$$

The whole procedure of the semi-supervised naïve Bayes classifiers with the inverse sigmoid function transformation technique for predicting the probabilities of class membership for the unlabelled patterns can be seen in Algorithm 5.3 . The error rate results for the SSNB classifier by using the inverse sigmoid function transformation is the same as the normal SSNB classifier, because the log-odds ratio, r_{ic}, was converted back to the, q_{ic}. However, the mean value of the predicted class log-odds ratio of the unlabelled data was converted by the sigmoid function, $\tilde{q}_{ic}$, that corresponds to the mean value of the predicted class probabilities, q_{ic}. Therefore, the inverse sigmoid function transformation will give the correct mean value.

Algorithm 5 log-odds ratio transformation for uncalibrated predicted class probabilities for the EM based Semi-Supervised naïve Bayes classifier

- **Input** : $X = \{(x_1, y_1), ..., (x_l, y_l), x_{l+1}, ..., x_{l+u}\}$
- **Set** : $t = 0$
- **Set Initialise** : $\hat{\theta}^{(0)} = argmax_\theta \, P(X_l, Y_l \mid \theta) P(\theta)$
- **Loop while** classifier parameters improve as measured by the change in $l(\theta \mid X_l, Y_l, X_u)$:
  - **(E-Step)** : Use the current classifier, $\hat{\theta}^{(t)}$, to find $q_{ik} = P(y_i = k \mid x_u; \theta)$ as shown in Equation 2.14

$$
\begin{aligned}
r_{ik_+} &= -\log\left(\frac{1-q_{ik_+}}{q_{ik_+}}\right), \\
r_{ik_+} &= \frac{r_{ik_+}}{u}, \\
\tilde{q}_{ik_+} &= \frac{1}{1+exp(-\bar{r}_{ic_+})}, \quad \text{see Figure5.2; where} \quad \bar{r}_{ic_+} = \textstyle\sum_{i=l+1}^{l+u} r_{ic_+} \\
q_{ik_+} &= \frac{1}{1+exp(-r_{ik_+})}, \\
q_{ik_-} &= 1 - q_{ik_+}.
\end{aligned}
$$

  - **(M-Step)** : Re-estimate the classifier, $\hat{\theta}^{(t+1)}$, using
$$\hat{\theta}^{(t+1)} = argmax_\theta \, P(X_l, Y_l, X_u \mid \theta^{(t)}) P(\theta^{(t)})$$
- **Set** : $t = t + 1$
- **Output** : A classifier, $\hat{\theta}^{(t)}$, that takes unlabelled data and predicts a class label.

Figure 5.2 shows the plot of the $\tilde{q}_{ic_+}$ and the proportion of the positive examples, θ which is the green line, for 100 replications of the Mushroom benchmark dataset. The inverse sigmoid function transformation fits badly, effectively pushing the converted mean value of the r_{ic_+} to 0 or 1 as can be seen in the Figure 5.2.

The positive patterns are more clustered with less variability which makes the classifier overconfident when the positive patterns have been added. However, the average value of the predicted probability for the positive class for the unlabelled patterns seems to have a small difference with the proportion of the positive examples, θ, in Figure 5.1. In reality almost always the value of predicted probabilities for the positive class for the unlabelled patterns is close to 0 or 1, as can be seen in Figure 5.2, but when averaged it seems to have a small difference.

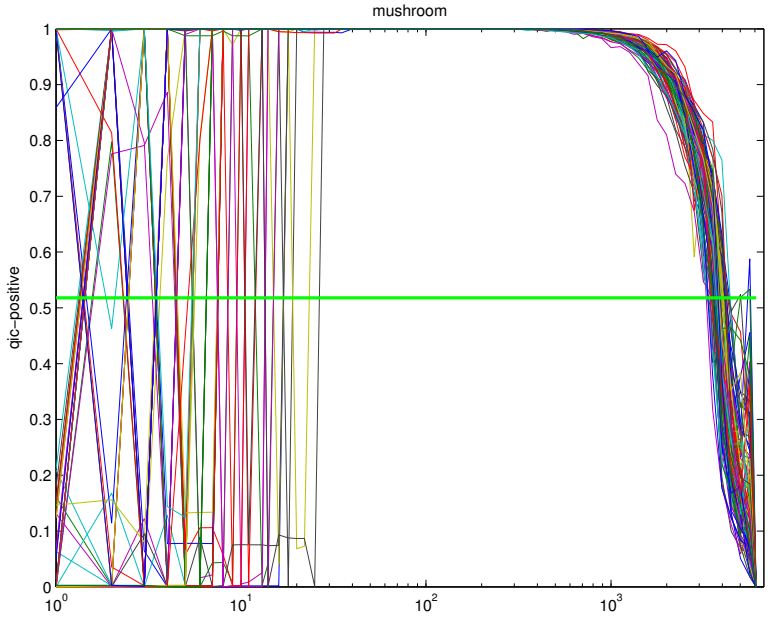


Fig. 5.2 The average value of predicted probabilities for the positive class, for the unlabelled data for 100 replications of the Mushroom benchmark dataset, using Platt scaling [63] for rank transformation

The log-odds ratio transformation does not fix the discrepancy problem between the predicted and actual class. In addition, this results suggest that the discrepancy problem is not the only issue problem. The overconfidence problem is also an issue. One way to visualize the overconfidence problem is by plotting a histogram of the predicted class probabilities. Figure 5.3 shows the percentage histogram of predicted class probabilities output, q_{ic_+}, for 100 replication for the UCI Mushroom benchmark dataset. It should be noted that the classifier output is overly confident, the output being almost always close to 0 or close to 1.

5.4 Logistic rank transformation function

The log-odds ratio transformation technique from the previous section clearly suggests that an attempt to fix the discrepancy problem between predicted and true class does not improve classification performance. There are two key problems with the EM algorithm that are consequences of the violations of the independence assumption of the naïve Bayes classifier.

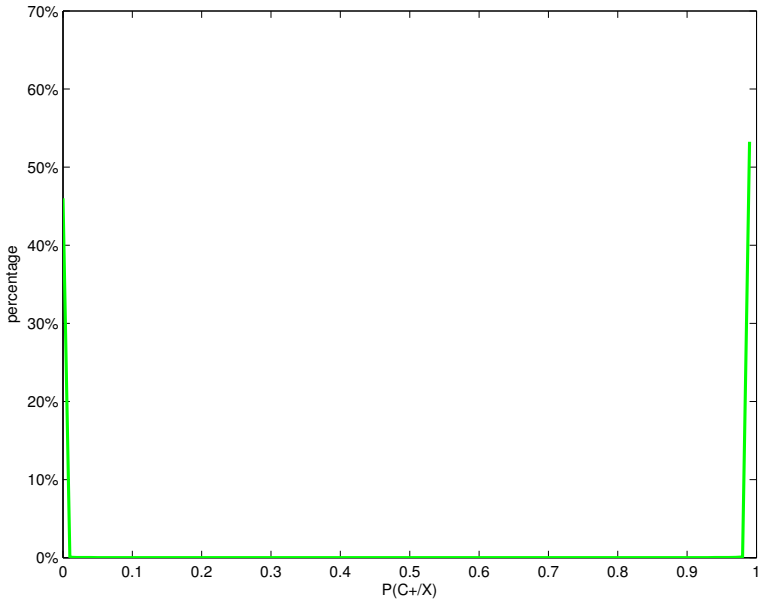


Fig. 5.3 Histogram of the predicted probabilities of the positive class for the unlabelled data for 100 replications of the Mushroom benchmark dataset for positive and negative classes.

Firstly, the values of estimated class probabilities for the unlabelled data tend to 0 or 1, and the second is the difference between the relative class frequency distribution and the mean of the predicted class probabilities.

Previously, the Platt scaling method [63] was proposed for transforming the SVM classifier outputs to provide estimates of the probability of class membership. Platt scaling is a parametric method, which was used for calibrating SVM predictions on $[-\infty,+\infty]$ into the posterior probability values $p(y=c|x;\theta)$ on the range [0,1] by passing them through a sigmoid function. However, Platt scaling cannot be used because there are no labels for the unlabelled patterns. The question that arises here is: what are the possible solutions for the semi-supervised version? Moreover, does solving the overconfidence problem and fixing the variation problem between predicted and actual class frequencies improve the naïve Bayes classification performance?

5.4.1 Spreading out and fixing the average of the predicted class probabilities

One possible solution would be to rank the predicted class probabilities for the unlabelled patterns, q_{ic}, from the most probable member of the positive class to the least probable member. By doing this, each pattern has a particular rank, r_{ic}, such that, for two patterns x_1 and x_2, if the predicted probability positive class $q_{1c_+} > q_{2c_+}$ then the rank $r_{1c_+} > r_{2c_+}$. Therefore, let r_{ic_+} be the rank values for the predict probabilities positive class for the unlabelled data q_{ic_+},

$$r_{ic_+} = \text{rank}\left(q_{ic_+}\right).$$

The normalised rank values, s_{ic_+}, can be calculated by dividing the rank values by the total number unlabelled data, u ;

$$s_{ic_+} = \frac{r_{ic_+}}{u}.$$

After ranking and normalising the predicted class probabilities for the unlabelled data, q_{ic_+}, calibrated class probabilities might be obtained through a sigmoid function;

$$q_{ic_+} = \frac{1}{1+exp^{-\alpha(s_{ic_+}-\beta)}},$$

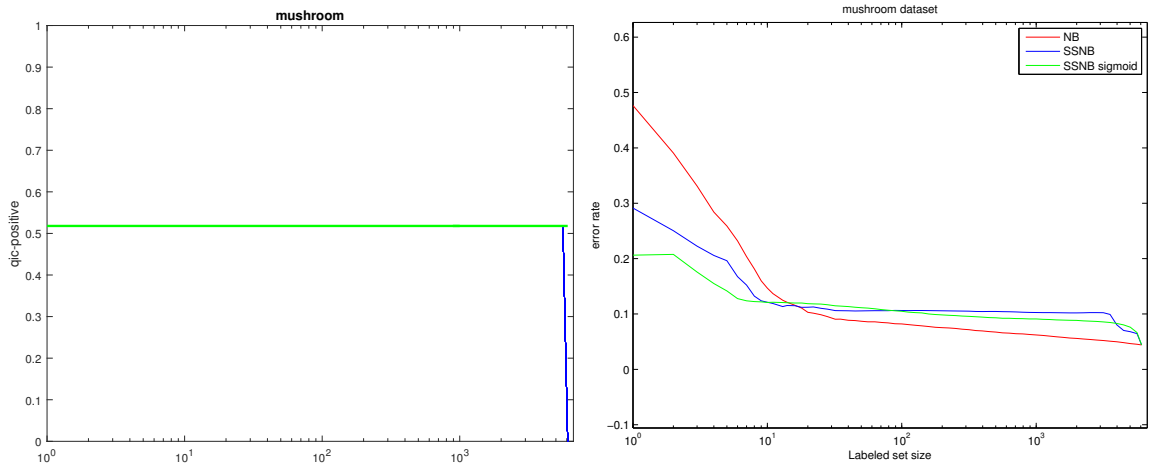
where α is a parameter controlling the slope of the sigmoid function and β is a parameter controlling the position of the sigmoid function. We attempt to spread out the predicted class probability for the unlabelled data, q_{ic_+}, by changing the value of α. In addition, changing the value of β corrects the error in the mean of the predicted class probabilities. To perform this experiment, we want to spread out the class probabilities without changing the level of average value of q_{ic_+} assuming the correct average value is known.

The value of β should be chosen such that the average value of the q_{ic_+} is equal to the

proportion of positive examples in the dataset. Therefore, the average value of the q_{ic_+} does not dependent on unlabelled patterns, instead it is dependent on β. The value of β can be optimised using scaled conjugate gradient methods, implemented by the MATLAB Optimisation Toolbox function `fminunc`, to minimise loss function.

During different trials in each experiment, where the q_{ic_+} variables are updated we need to change β but for a particular trial, it is necessary that the value of β remains the same, assuming the correct average value is known. As can be seen in Figure 5.4a, the value of β remains the same from the beginning to the end of the plot in a particular trial for Mushroom benchmark dataset. Consequently, the mean value of q_{ic_+} would be the same as the proportion of positive examples all the time, regardless of the value of α. Therefore, choosing any value for α, all the lines should be underneath the green line, which is the proportion of patterns in the positive class, θ.

The sigmoid function seems to fit the rank values reasonably well without having difference



(a) (b)

Fig. 5.4 (a) The average value of predicted probabilities for the positive class for unlabelled patterns, q_{ic_+}, using the logistic transformation. (b) The area under the error rate learning curve for 100 replications of the Mushroom benchmark dataset using the logistic transformation.

between the true class proportion and the mean of the predicted class probabilities for unlabelled patterns, q_{ic}. Thus, using the sigmoid function to spread the ranks of the predicted

probabilities class of the unlabelled patterns can reduce the area under the error rate learning curve for the Mushroom benchmark dataset from (0.115 ± 0.0013) to (0.105 ± 0.0008) as shown in Figure 5.4b.

The results obtained suggest that the sigmoid transformation can improve the performance of the semi-supervised naïve Bayes classification. The whole procedure of the semi-supervised naïve Bayes classifier with the sigmoid function for rank values transformation technique (SSNB-sigmoid-fb) is shown in the Algorithm 6.

Algorithm 6 Logistic function and rank transformation for uncalibrated predicted probabilities class for EM based Semi-Supervised naïve Bayes classifier

- **Input** : $X = \{(x_1, y_1), ..., (x_l, y_l), x_{l+1}, ..., x_{l+u}\}$
- **Set** : $t = 0$
- **Initialise** : $\hat{\theta}^{(0)} = argmax_\theta \, P(X_l, Y_l \mid \theta) P(\theta)$
- **Loop while** classifier parameters improve as measured by the change in $l(\theta \mid X_l, Y_l, X_u)$:
  - **(E-Step)** : Use the current classifier, $\hat{\theta}^{(t)}$, to find $q_{ik} = P(y_i = k \mid x_u; \theta)$ as shown in equation 2.14

$$
\begin{aligned}
r_{ic_+} &= rank\left(q_{ic_+}\right) \\
r_{ic_+} &= \frac{r_{ic_+}}{u} \\
\alpha &= 2.^{[}-10:5]; \text{ and } \beta \text{ optimised using the function } fminunc \\
q_{ic_+} &= \frac{1}{1+exp^{-\alpha(r_{ic_+}-\beta)}} \\
q_{ic_-} &= 1 - q_{ic_+}
\end{aligned}
$$

  - **(M-Step)** : Re-estimate the classifier, $\hat{\theta}^{(t+1)}$, using $\hat{\theta}^{(t+1)} = argmax_\theta \, P(X_l, Y_l, X_u \mid \theta^{(t)}) P(\theta^{(t)})$
- **Set** : $t = t + 1$
- **Output** : A classifier, $\hat{\theta}^{(t)}$, that takes unlabelled data and predicts a class label.

In spite of having improved classification performance, the improvement for a single benchmark dataset is not sufficient to decide that the sigmoid function transformation can generally improve classification performance. Therefore, in the next step, we demonstrate an empirical result for the set of benchmark datasets. Table 5.2 gives the details for 12 discrete and 16 continuous binary classification benchmark datasets from the UCI repository. For

each dataset there are 100 random partitions of the data into training and test set for each trial.

#	Dataset	Training Patterns	Testing Patterns	Number of Attributes	Type of Attributes
1	blogger	75	25	5	discrete
2	breast-cancer	215	71	9	discrete
3	breastw	524	175	9	discrete
4	house-votes	326	109	16	discrete
5	kr-vs-kp	2397	799	36	discrete
6	monk1	324	108	6	discrete
7	mushroom	6093	2031	22	discrete
8	promoters	80	26	57	discrete
9	slc	190	63	6	discrete
10	threeOf9	384	128	9	discrete
11	titanic	1651	550	3	discrete
12	xd6	384	128	9	discrete
13	banknote	1029	343	4	continuous
14	Blood-transfusion	561	187	4	continuous
15	bcc	512	171	9	continuous
16	CMSC	405	135	18	continuous
17	haberman	230	76	3	continuous
18	ionosphere	263	88	33	continuous
19	liver-disorder	259	86	6	continuous
20	magic04	14265	4755	10	continuous
21	musk1	357	119	166	continuous
22	arcene	150	50	10000	continuous
23	gisette	5250	1750	5000	continuous
24	madelon	1950	650	500	continuous
25	sonar	156	52	60	continuous
26	spambase	3451	1150	57	continuous
27	Synthetic	938	312	2	continuous
28	vertebral	233	77	6	continuous

Table 5.2 Detials of the datasets used in experiemnts

The purpose of the logistic transformation on the rank values in this experiment, is to fix the average value of the q_{ic} variables to achieve the correct average value of the labels for the unlabelled data that does not depended on unlabelled patterns. Furthermore, the plot of the average value of the q_{ic} variables (see Figure 5.4a) should just be a horizontal line and same

for all runs, because we set it to the correct value for all of the datasets by pretending that we know what the true ratio of labels is. Obviously this is unrealistic and means this approach is biased and gives an optimistic estimate of performance. The question being raised here is: even if we do know what the true value is, would that improve classification performance?

#	**Dataset**	**NB**	**SSNB**	**SSNB-sigmoid-fb**	α
1	blogger	2.173±0.0422	2.521±0.0376	**1.981±0.0430**	0.25
2	breast_cancer	2.656±0.0432	2.965±0.0392	**2.264±0.0297**	1
3	breastw	0.874±0.0225	0.325±0.0293	**0.286±0.0220**	16
4	house_votes	**1.043±0.0234**	1.119±0.0266	1.064±0.0214	0.25
5	kr_vs_kp	**3.136±0.0200**	4.771±0.0249	3.647±0.0179	0.125
6	monk1	3.129±0.0259	3.158±0.0300	**3.045±0.0285**	0.0625
7	mushroom	1.771±0.0226	1.670±0.0313	**1.425±0.0266**	0.125
8	promoters	1.694±0.0360	1.336±0.0361	**1.213±0.0455**	16
9	slc	1.365±0.0275	1.440±0.0386	**1.192±0.0464**	16
10	threeOf9	*2.711±0.0294*	2.967±0.0267	***2.690±0.0241***	1
11	titanic	3.033±0.0451	3.211±0.0411	**2.559±0.0316**	2
12	xd6	2.670±0.0292	2.789±0.0275	**2.458±0.0285**	2
13	banknote	**1.541±0.0225**	2.931±0.0197	2.320±0.0310	32
14	Blood_transfusion	2.070±0.0262	2.564±0.0357	**1.683±0.0185**	0.125
15	bcc	0.500±0.0167	0.343±0.0120	**0.259±0.0090**	32
16	CMSC	0.603±0.0140	0.631±0.0154	**0.536±0.0124**	4
17	haberman	1.778±0.0301	2.045±0.0527	**1.510±0.0254**	1
18	ionosphere	1.415±0.0278	1.742±0.0298	**1.334±0.0208**	0.031
19	liver_disorder	2.719±0.0210	2.989±0.0207	**2.549±0.0210**	0.001
20	magic04	3.506±0.0182	4.832±0.0136	**3.358±0.0102**	0.25
21	musk1	**2.207±0.0225**	2.716±0.0240	2.370±0.0256	0.001
22	arcene	2.160±0.0223	2.051±0.0255	**1.878±0.0286**	0.125
23	gisette	**2.099±0.011**7	3.057±0.0285	2.160±0.0200	0.001
24	madelon	4.116±0.0091	3.935±0.0160	**3.86**2±**0.0224**	32
25	sonar	1.928±0.0239	2.247±0.0210	**1.884±0.0267**	0.001
26	spambase	2.263±0.0195	2.409±0.0288	**2.202±0.0268**	32
27	Synthetic	1.182±0.0217	1.116±0.0269	**1.037±0.0165**	0.125
28	vertebral	**1.421±0.0200**	1.417±0.0231	1.459±0.0227	16

Table 5.3 AULC of the NB, SSNB, and SSNB-sigmoid-fb classifier with the best value of α over 12 discrete and 16 continues benchmark datasets. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier between NB and SSNB-sigmoid-fb classifiers. The results that are statistically equivalent between NB and SSNB-sigmoid-fb classifiers (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifiers is better or equivalent to the AULC for the SSNB-sigmoid-fb classifier.

Table 5.3 shows a comparison of the area under the error rate learning curve of NB, SSNB, and SSNB-sigmoid-fb classifiers for the 28 UCI benchmark datasets. The experiments for each dataset consisted of 100 trials, with random partitioning of the data to form training and test sets in each trial. In this experiment, there is a statistically significant difference between the mean values of the classifiers according to the Friedman test [25]. For test significant pairwise differences between classifiers the Nemenyi post-hoc test is used. As we can see in Figure 5.5 the SSNB-sigmoid-fb achieves the highest average rank. However, in this experiment the biased method is used because in reality we cannot fix the difference between true class and the average value of q_{ic} variables as we do not know the true ratio of positive and negative patterns.

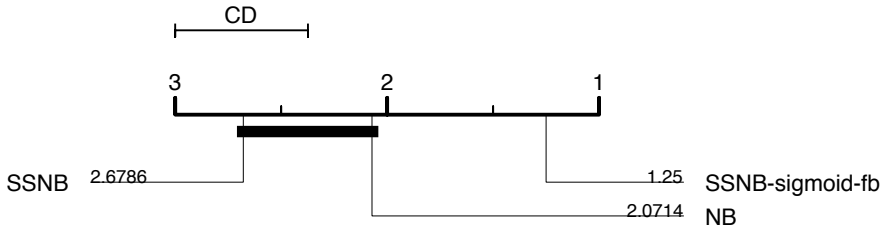


Fig. 5.5 Critical difference diagram for the NB, SSNB, and SSNB-sigmoid-fb over 12 discrete and 16 continues benchmark datasets. It shows that there are statistically significant differences between the means ranks for the SSNB classifier and both NB and SSNB-sigmoid-fb classifiers

This is a useful result, explaining why the unlabelled data do not help to improve naïve Bayes classifier most of the time, which is because the EM based semi-supervised learning has overconfidently predicted the class probabilities. Additionally, there is a variation between the mean of the predicted and actual class labels. Thus, we need to do further experiments to test whether the classification improvement is caused by the spread out of the q_{ic} or if it is caused by fixing the average value of the q_{ic} to be correct. The purpose of this experiment is to test: if we know what the correct average value of the labels for the unlabelled data, does the semi-supervised learning improve the naïve Bayes classifier? If the

unlabelled data does not improve the naïve Bayes classifier even if we know the true average value, obviously it is not going to help when we must estimate it from the limited amount of labelled data. The result in this experiment shows that if we know the correct average value of the labels for the unlabelled data, the performance of the NB classifier can be improved. It is true that we used the biased method but at least we are pointing out what the key issue of SSNB is.

5.4.2 Spreading out without fixing the average of the predicted class probabilities

In the previous section, two different actions are used. The former action is spreading out the rank of the q_{ic} variables and the latter is fixing the average value of the q_{ic} variables. Furthermore, fixing the average value of the q_{ic} variables is incorrect in the real world because the actual class labels known for the unlabelled patterns. Therefore, the previous protocol is a biased. In this section, instead of changing two factors at the same time, a new experiment is designed to discover whether it is the spreading or the fixing the average value of the q_{ic} variables that produces the improvement in performance.

We start by just spreading out the rank of the q_{ic} variables without fixing the average value and expect that the protocol for this experiment is better than the previous one because at least this method is less biased compared to the previous experiment. We spread out the rank of the q_{ic} variables by scaling, q_{ic}, but do not fix the average value of the q_{ic} variables which means we do not fix the bias between predicted and actual class.

In this experiment, we want to make β the average of the q_{ic} variables before they were rescaled and then use the logistic transformation. We set β such that the average of the q_{ic} variables after the transformation is still the average q_{ic} variables before the rescaling. Thus the true average value can be found which is the average value of r_{ic} and should be equal to the average value of the q_{ic} variables before the ranking.

Basically this experiment and the previous experiment are equivalent formula, just the values of α and β differ. Moreover, both experiments tell us whether spreading out the q_{ic} variables or changing their mean value reduces the error rate or both (when we made both changes).

#	Dataset	NB	SSNB	SSNB-sigmoid	α
1	blogger	**2.173±0.0422**	2.521±0.0376	2.224±0.0398	0.001
2	breast_cancer	**2.656±0.0432**	2.965±0.0392	2.963±0.0353	0.500
3	breastw	0.874±0.0225	0.325±0.0293	**0.309±0.0270**	32.000
4	house_votes	**1.043±0.0234**	1.119±0.0266	1.063±0.0246	0.031
5	kr_vs_kp	3.136±0.0200	4.771±0.0249	**3.080±0.0210**	0.001
6	monk1	3.129±0.0259	3.158±0.0300	**3.031±0.0274**	0.250
7	mushroom	1.771±0.0226	1.670±0.0313	**1.423±0.0250**	0.016
8	promoters	1.694±0.0360	1.336±0.0361	**1.238±0.0439**	16.000
9	slc	1.365±0.0275	1.440±0.0386	**1.167±0.0452**	32.000
10	threeOf9	2.711±0.0294	2.967±0.0267	**2.660±0.0271**	0.250
11	titanic	***3.033±0.0451***	3.211±0.0411	*3.034±0.0454*	0.001
12	xd6	**2.670±0.0292**	2.789±0.0275	2.706±0.0332	0.001
13	banknote	**1.541±0.0225**	2.931±0.0197	1.623±0.0247	0.001
14	Blood_transfusion	***2.070±0.0262***	2.564±0.0357	*2.102±0.0304*	0.001
15	bcc	0.500±0.0167	0.343±0.0120	**0.267±0.0090**	16.000
16	CMSC	**0.603±0.0140**	0.631±0.0154	1.365±0.0167	0.001
17	haberman	*1.778±0.0301*	2.045±0.0527	***1.777±0.0302***	0.001
18	ionosphere	1.415±0.0278	1.742±0.0298	**1.333±0.0262**	0.001
19	liver_disorder	**2.719±0.0210**	2.989±0.0207	2.824±0.0223	0.001
20	magic04	3.506±0.0182	4.832±0.0136	**3.337±0.0163**	0.001
21	musk1	**2.207±0.0225**	2.716±0.0240	2.427±0.0250	0.001
22	arcene	2.160±0.0223	2.051±0.0255	**1.886±0.0285**	0.125
23	gisette	**2.099±0.0117**	3.057±0.0285	2.160±0.0199	0.001
24	madelon	4.116±0.0091	3.935±0.0160	**3.883±0.0214**	32.000
25	sonar	*1.928±0.0239*	2.247±0.0210	***1.894±0.0264***	0.001
26	spambase	**2.263±0.0195**	2.409±0.0288	2.539±0.0306	32.000
27	Synthetic	1.182±0.0217	1.116±0.0269	**1.059±0.0167**	0.125
28	vertebral	***1.421±0.0200***	1.417±0.0231	*1.433±0.0244*	32.000

Table 5.4 AULC of the NB, SSNB, and SSNB-sigmoid classifier with the best value of α over 12 discrete and 16 continues benchmark datasets. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier between NB and SSNB-sigmoid classifiers. The results that are statistically equivalent between NB and SSNB-sigmoid classifiers (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifiers is better or equivalent to the AULC for the SSNB-sigmoid classifier.

Table 5.4 shows a comparison of the area under error rate learning curve of the NB, SSNB, and SSNB-sigmoid classifiers over 28 UCI benchmark datasets. The experiments for each dataset consisted of 100 trials, with random partitioning of the data to form training and test sets in each trial. In this case, there is a statistically significant difference between the classifiers according to the Friedman test. As we can see in Figure 5.6 the SSNB-sigmoid and NB classifier are statistical superior to the SSNB, but both SSNB-sigmoid and NB classifier have equivalent performance.

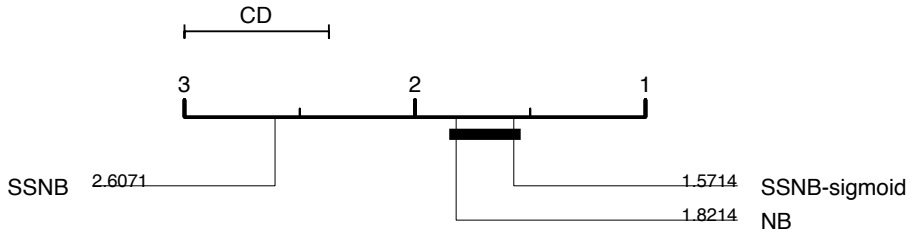


Fig. 5.6 Critical difference diagram for the NB, SSNB, and SSNB-sigmoid over 18 discrete and 16 continues benchmark data sets. It shows that there are statistically significant differences between the means ranks for the SSNB classifier and both NB and SSNB-sigmoid classifiers, however there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar

The SSNB-sigmoid approach is statistically superior to the SSNB classifier. However, the difference between NB and SSNB-sigmoid is very small and in this case SSNB-sigmoid does improve classification performance. Figure 5.6 shows that the SSNB-sigmoid classifier is best compared to the SSNB and NB classifiers but not enough to make a statistically significant difference with the NB classifier.

The two previous experiments show that, the unlabelled data can improve the performance of the NB classifier through the SSNB-sigmoid and SSNB-sigmoid-fb classifiers because they assume we know the true class frequency. When the obvious biased protocol is used, SSNB-sigmoid-fb classifier, it is statistically superior to NB classifier. On the other hand, if the sigmoid transformation is used without fixing the bias, the rank of the SSNB-sigmoid

classifier, which is a less biased protocol, is higher than the NB classifier rank, but the difference is not statistically significant.

The algorithm could be guide if the best value for β knew but the problem is of course in the reality we do not know the true labels for the unlabelled examples. Unfortunately, the SSNB classifier does improve just a little bit by applying some transformation methods, but these methods are biased approaches of getting α and β.

5.4.3 Spreading out the predicted class probabilities using Transfer Learning

In the previous experiment, we applied the biased version of the SSNB-sigmoid classifier that spreads out the q_{ic} without fixing their average value, which improves performance . We next set up a new experiment that is a diagnostic of what happens if we use a default value of α all the time, rather than using a biased method for calculating α or fixing β. This approach is called transfer learning. We estimate α for one dataset from the optimal αs for the other datasets, hence transfer learning means what is learned from one dataset is transferred to another problem. The log scale parameter $\log_2 \alpha$ is used for the experiment Leave one dataset out (LODO). Taking the median of the log scale parameter $\log_2 \alpha$ is probably a more sensible way than having a default value of α.

In this section, the leave one dataset out approach is used over 28 UCI benchmark datasets described in Table 5.2. The SSNB-sigmoid classifier is trained over only 27 benchmark datasets which gives an idea what α should be by taking the median of α. Then, we can use that median value of αs to choose the value of α for this remaining benchmark dataset, which is the leave one dataset out and just using a single value of α. This is called transfer learning, which implies learning something for one task by taking parameters that seem good for another task. Previous experiments show that the sigmoid transformation does help to improve the NB classifier if we use a biased approach, but in this section we want to test

whether without using a biased approach, the NB classifier still improves. The research question for this experiment is: can we predict the value of α via an unbiased approach that is basically established as transfer learning?

#	**Dataset**	**NB**	**SSNB**	**SSNB-sigmoid-LODO**	α
1	blogger	**2.173±0.0422**	2.521±0.0376	2.227±0.0400	0.0156
2	breast_cancer	**2.656±0.0432**	2.965±0.0392	2.992±0.0289	0.0078
3	breastw	0.874±0.0225	0.325±0.0293	**0.482±0.0173**	0.0078
4	house_votes	**1.043±0.0234**	1.119±0.0266	1.069±0.0247	0.0078
5	kr_vs_kp	**3.136±0.0200**	4.771±0.0249	3.185±0.0216	0.0156
6	monk1	3.129±0.0259	3.158±0.0300	**3.048±0.0289**	0.0078
7	mushroom	1.771±0.0226	1.670±0.0313	**1.423±0.0250**	0.0156
8	promoters	*1.694±0.0360*	1.336±0.0361	***1.683±0.0323***	0.0078
9	slc	**1.365±0.0275**	1.440±0.0386	1.536±0.0265	0.0078
10	threeOf9	2.711±0.0294	2.967±0.0267	**2.689±0.0285**	0.0078
11	titanic	**3.033±0.0451**	3.211±0.0411	3.128±0.0386	0.0156
12	xd6	**2.670±0.0292**	2.789±0.0275	2.724±0.0307	0.0156
13	banknote	**1.541±0.0225**	2.931±0.0197	1.685±0.0247	0.0156
14	Blood_transfusion	**2.070±0.0262**	2.564±0.0357	2.142±0.0318	0.0156
15	bcc	0.500±0.0167	0.343±0.0120	**0.283±0.0094**	0.0078
16	CMSC	**0.603±0.0140**	0.631±0.0154	1.382±0.0166	0.0156
17	haberman	***1.778±0.0301***	2.045±0.0527	*1.784±0.0304*	0.0156
18	ionosphere	1.415±0.0278	1.742±0.0298	**1.340±0.0265**	0.0156
19	liver_disorder	**2.719±0.0210**	2.989±0.0207	2.833±0.0224	0.0156
20	magic04	**3.506±0.0182**	4.832±0.0136	3.624±0.0242	0.0156
21	musk1	**2.207±0.0225**	2.716±0.0240	2.449±0.0247	0.0156
22	arcene	2.160±0.0223	2.051±0.0255	**1.894±0.0274**	0.0078
23	gisette	**2.099±0.0117**	3.057±0.0285	2.241±0.0215	0.0156
24	madelon	4.116±0.0091	3.935±0.0160	**4.085±0.0104**	0.0078
25	sonar	*1.928±0.0239*	2.247±0.0210	***1.899±0.0264***	0.0156
26	spambase	**2.263±0.0195**	2.409±0.0288	2.875±0.0493	0.0078
27	Synthetic	1.182±0.0217	1.116±0.0269	**1.106±0.0177**	0.0078
28	vertebral	**1.421±0.0200**	1.417±0.0231	1.853±0.0300	0.0078

Table 5.5 AULC of the NB, SSNB, and SSNB-sigmoid-LODO classifier with the best value of α over 12 discrete and 16 continues benchmark datasets. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier between NB and SSNB-sigmoid-LODO classifiers. The results that are statistically equivalent between NB and SSNB-sigmoid-LODO classifiers (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics. Underlining indicates that the AULC for the SSNB classifiers is better or equivalent to the AULC for the SSNB-sigmoid-LODO classifier.

In this case, there is no statistically significant difference between SSNB-sigmoid-LODO and both NB and SSNB classifiers according to the Friedman test. As we can see in Figure 5.7 the SSNB-sigmoid-LODO outperforms the SSNB classifier and also the SSNB-sigmoid-LODO just underperformed the NB classifier but the difference is not statistically significant.

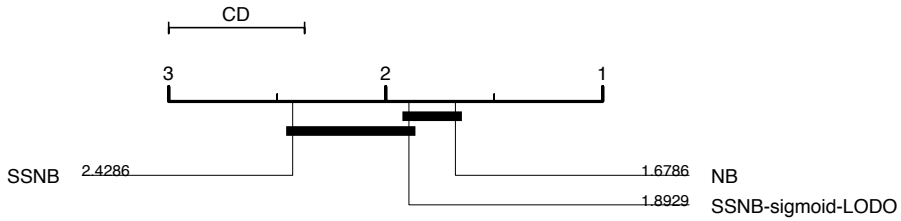


Fig. 5.7 Critical difference diagram for the NB, SSNB, and SSNB-sigmoid-LODO over 12 discrete and 16 continues benchmark datasets. It shows that there are statistically significant differences between the means ranks for the SSNB classifier and both NB and SSNB-sigmoid-LODO classifiers, however there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar

5.4.4 Spreading out the predicted class probabilities using cross-validation method

In the SSNB-sigmoid-LODO experiment, the value of α was chosen by looking at the test data partition, which obviously we can say is still using a bit of a biased method because of choosing a parameter by looking at the test data, which is biased protocol. In addition, this biased method is tried to fixing bias between predictive and actual class, and estimate the test error. A better action is used the cross-validation method in terms of minimizing the test error. If it does not help, then choosing the value of α badly is probably making it much worse which does not really help.

#	Dataset	NB	SSNB	SSNB-sigmoid-cv
1	blogger	**1.765±0.0349**	2.056±0.0318	1.886±0.0314
2	breast_cancer	**2.251±0.0329**	2.501±0.0352	2.530±0.0281
3	breastw	0.585±0.0151	**0.229±0.0131**	0.364±0.0102
4	house_votes	**0.849±0.0163**	0.922±0.0157	0.886±0.0162
5	kr_vs_kp	**2.652±0.0192**	4.273±0.0241	2.744±0.0205
6	monk1	2.650±0.0236	2.664±0.0275	**2.610±0.0232**
7	mushroom	1.337±0.0188	1.398±0.0217	**1.270±0.0155**
8	promoters	1.288±0.0298	**0.983±0.0281**	1.077±0.0291
9	slc	0.998±0.0219	1.008±0.0335	**0.945±0.0258**
10	threeOf9	**2.279±0.0249**	2.503±0.0226	2.296±0.0222
11	titanic	**2.568±0.0358**	2.769±0.0325	2.687±0.0320
12	xd6	**2.211±0.0243**	2.328±0.0243	2.344±0.0241
13	banknote	**1.167±0.0153**	2.464±0.0170	1.351±0.0209
14	Blood_transfusion	**1.786±0.0216**	2.093±0.0314	1.879±0.0244
15	bcc	0.323±0.0113	0.283±0.0099	**0.236±0.0076**
16	CMSC	0.628±0.0165	**0.467±0.0117**	1.340±0.0114
17	haberman	1.487±0.0240	1.567±0.0467	**1.478±0.0263**
18	ionosphere	**1.011±0.0238**	1.319±0.0244	1.096±0.0233
19	liver_disorder	**2.284±0.0175**	2.501±0.0198	2.372±0.0189
20	magic04	3.112±0.0129	4.347±0.0103	**3.045±0.0128**
21	musk1	**1.732±0.0197**	2.273±0.0223	2.027±0.0223
22	arcene	1.661±0.0180	1.582±0.0217	**1.493±0.0228**
23	gisette	**1.564±0.0101**	2.541±0.0277	1.804±0.0173
24	madelon	3.607±0.0087	**3.430±0.016**3	3.450±0.0147
25	sonar	**1.424±0.0238**	1.748±0.0206	1.535±0.0234
26	spambase	**1.889±0.0145**	2.026±0.0204	1.896±0.0211
27	Synthetic	0.861±0.0133	**0.829±0.0144**	0.863±0.0123
28	vertebral	**1.077±0.0164**	1.143±0.0188	1.304±0.0207

Table 5.6 AULC of the NB, SSNB, and SSNB-sigmoid-cv classifier over 12 discrete and 16 continues benchmark datasets. The results for each AULC classifier are presented in the form of the mean and standard error over test data for 100 realisations of each dataset. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent between NB and SSNB-sigmoid-cv classifiers (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

Table 5.6 shows a comparison of the area under the error rate learning curves for the NB, SSNB, and SSNB-sigmoid-cv classifiers over 28 UCI benchmark datasets. The experiments for each dataset consisted of 100 trials, with random partitioning of the data to form training and test sets in each trial. In this case, there is a statistically significant difference between

classifiers according to the Friedman test. As we can see in Figure 5.8 the performance of the NB classifier is statistically superior to that of the SSNB classifier and it has higher rank than the SSNB-sigmoid-cv classifier but the difference is not statistically significant. Furthermore, the SSNB-sigmoid-cv classifier has equivalent classification performance compared to both NB and SSNB classifiers.

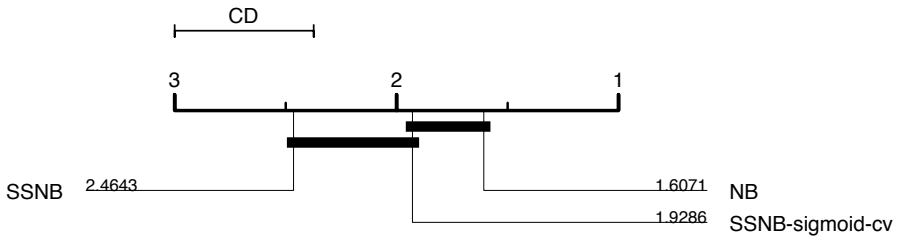


Fig. 5.8 Critical difference diagram for the NB, SSNB, and SSNB-sigmoid-cv over 12 discrete and 16 continues benchmark datasets. It shows that there are statistically significant differences between the means ranks for the SSNB classifier and both NB and SSNB-sigmoid classifiers, however there are no statistically significant differences between the mean ranks for the classifiers which are linked by the bar

5.4.5 Does the logistic transformation improve the SSNB classifier?

Figure 5.9 shows a critical difference diagram for NB, SSNB, SSNB-sigmoid, SSNB-sigmoid-fb, SSNB-sigmoid-LODO, SSNB-sigmoid-cv. Clearly, the logistic transformation improves the SSNB classifier. If the biased protocols are used with the logistic transformation trick, SSNB-sigmoid and SSNB-sigmoid-fb, then the result obtained is better than SSNB and NB, especially if the average value of the q_i variables was fixed. In addition, if the unbiased protocols are used with logistic transformation, SSNB-sigmoid-LODO and SSNB-sigmoid-cv, then the result obtained is equivalent to the NB classification performance but at least superior to the SSNB classifier. This result suggests that using the logistic transformation in the E-step of the EM based semi-supervised learning has not significantly improved the performance of NB classifier but it is a step that at least guides the algorithm better than just

the SSNB classifiers. The logistic transformation is likely to be beneficial, but the problem with estimating the parameters α and θ using the labelled data is unreliable.

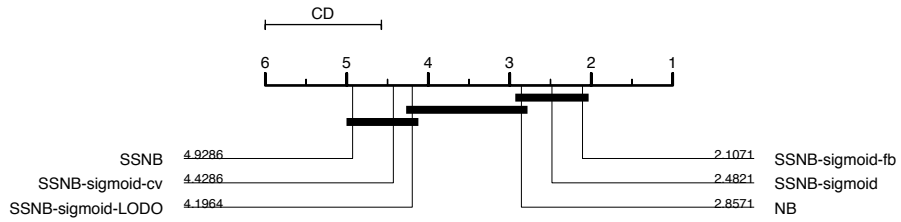


Fig. 5.9 Critical difference diagram for the NB, SSNB, SSNB-sigmoid, SSNB-sigmoid-fb, SSNB-sigmoid-LODO and SSNB-sigmoid-cv over 12 discrete and 16 continues benchmark datasets. It shows that there are statistically significant differences between the means ranks for the SSNB, SSNB-sigmoid-LODO and SSNB-sigmoid-cv classifiers and NB, SSNB-sigmoid and SSNB-sigmoid-fb classifiers, however there are no statistically significant differences between the mean ranks for the NB, SSNB-sigmoid and SSNB-sigmoid-fb classifiers which are linked by the bar

5.5 Conclusions

In this chapter we started off by describing problem in the NB classifier, in which the predicted class probability were overconfident, being extremely close to 0 and 1. Then, we showed the effect of this problem in the NB classifier to the EM algorithm base semi-supervised learning when it used both labelled and unlabelled data to estimate the model parameters. Consequently, the main research question was established: could spreading out the predicted class probabilities for the unlabelled data improve the NB classifier? In order to do that, the logistic transformation method was used with various approaches to choose the logistic function parameters was used in a series of experiments. In the first experiment, the predicted class probabilities for unlabelled data were spread out and we assumed that the correct average value of the labels for unlabelled data was known by using the `SSNB-sigmoid-fb` approach. This approach can improve the baseline classifier because the SSNB-sigmoid-fb has a very biased protocol. Next, the `SSNB-sigmoid` approach was used

without assuming that the correct average value of the labels for unlabelled data was known, which has a slightly biased protocol. The experiment results show that the SSNB-sigmoid approach is better than the NB classifier but there is no statistically significant difference. The transfer learning method was used with the SSNB-sigmoid approach in another experiment known as `SSNB-sigmoid-LODO`. The goal behind using transfer learning is to use a default value for the rate of spread out of the predicted class probabilities for unlabelled data. Finally, the cross-validation method, `SSNB-sigmoid-cv`, was used for choosing the logistic transformation function parameters. The rank of the NB classifier is higher than the `SSNB-sigmoid-LODO` and `SSNB-sigmoid-cv`,but there is no statistically significant difference. The SSNB-sigmoid approaches, (`SSNB-sigmoid-fb` and `SSNB-sigmoid`), which are biased protocols, do help a bit and are approximately equivalent to the NB classifier, while in both cases `SSNB-sigmoid-LODO` and `SSNB-sigmoid-cv`, logistic transformation methods do not help because both are unbiased protocols. In this chapter, we conclude that less improvement in the NB classification performance can be obtained by using less unbiased protocol. The logistic transformation method improve SSNB, but not to the point where it is a significant improvement on NB. The logistic transformation method just moderates a little bit but the experiments results suggest that does not help very much. However, it does generate better performance than the SSNB classifier.

Chapter 6

Investigation of active learning

In previous chapters, various experiments were performed and discussed in order to evaluate whether semi-supervised learning can improve the performance of the naïve Bayes classifier. In this chapter, we present results from active learning experiments for selectively labelling large amounts of unlabelled patterns. The dual goals of using active learning methods are the creation of high-quality labelled data to improve classification performance, and the minimisation of the manual labelling effort required. This chapter starts with a reproduction of some existing experimental results and a description of the experimental design for the real world benchmark datasets. It continues with a discussion of results obtained using the suite of benchmark datasets, including the evaluation measures and statistical tests used. After selecting the least confidently unlabelled pattern by active learning especially in the early stages, we attempt to use the remaining the large amount of unlabelled patterns via semi-supervised leaning to increase the classification performance. Thus, we design some experiments to combine active learning with semi-supervised learning.

6.1 Introduction to active learning

A limitation of supervised learning methods is that they require the training data to be labelled, which is often a time consuming and expensive task. Semi-supervised techniques were used previously to extract predictive information from unlabelled patterns. Nevertheless using unlabelled patterns does not generally improve classification performance in the case of a naïve Bayes classifier. In this chapter, we investigate active learning methods that acquire labelled data incrementally, using the existing model to select particularly helpful additional training patterns for labelling by an oracle. Therefore, active learning methods may reduce the number of examples that must be labelled to achieve a particular level of accuracy. Achieving the same naïve Bayes classification performance with fewer labelled training examples is an important step towards improving the classifier and previous works have shown that this can be done successfully [52].

As mentioned in Chapter 2.3, active learning is a machine learning framework that allows the classifier to automatically select the most informative unlabelled patterns for manual labelling. Thus, the main goal in active learning can be referred to as selective sampling. Different sampling strategies exist, for example, uncertainty sampling is a specific type of active learning which selects informative patterns. These patterns are near of the decision boundaries and so have the lowest confidence score. In this section, we are interested in implementing uncertainty sampling for pool-based active learning, which provides a small set of labelled data, L, and a large set of unlabelled data, U, for training. With the small amount of labelled patterns the initial parameters of model are estimated, this method has been applied by Ramirez-Loaiza et al. [64]. Therefore, before investigating the main hypothesis, we first attempt to reproduce the experiments from Ramirez-Loaiza et al. [64] as a preliminary investigation.

The preliminary experiment attempted to reproduce the ALNB (Active Learning Naïve Bays) classifier results with the `Hiva` benchmark dataset that was originally reported in Ramirez-

Loaiza et al. [64] which is a recent empirical evaluation paper that used ten well-studied benchmark datasets to demonstrate the effectiveness of the ALNB classifier. The `Hiva` benchmark dataset is one of the datasets used, which is a real world binary classification problem with 42,678 examples and 1,617 features, that was originally developed for an active learning challenge by Guyon et al. [34].

The authors of [64] designed their experiments as follows: the performance of the ALNB classifier was averaged over five runs of five-fold cross validation. For each experiment, ten labelled patterns (five from each class) are chosen from the available data and the rest of the training set was treated as the unlabelled pool, U. However, if the unlabelled pool, U, consisted of more than 10,000 patterns, then 10,000 patterns were randomly sub-sampled from the large unlabelled pool. For computational convenience, at each iteration the top ten most informative instances were selected, as determined by the AL strategy. Following this experimental design, we obtained the average recall learning curve plot for `Hiva` benchmark as shown in Figure 6.1.

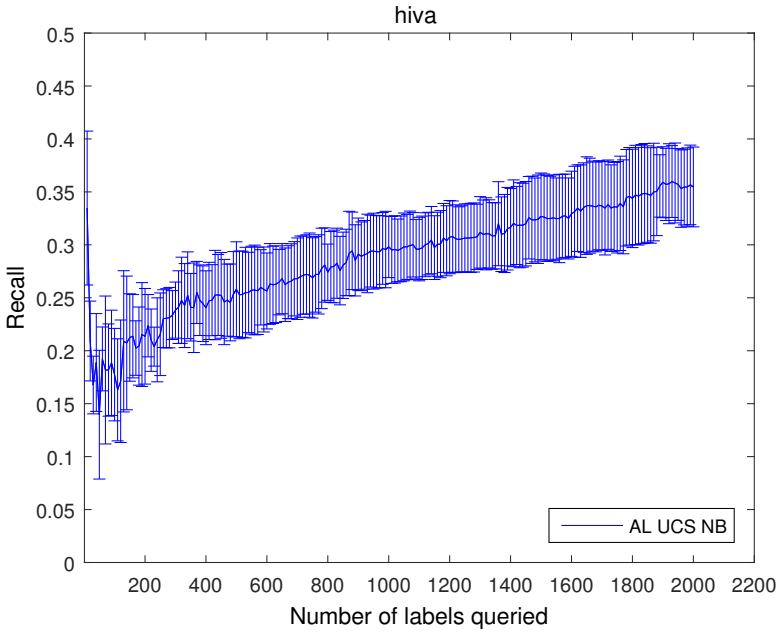


Fig. 6.1 Average test set recall learning curve for the `Hiva` benchmark dataset using the uncertainty sampling strategy of ALNB classifier over 5-run 5-fold cross validation

The learning curve obtained for the ALNB classifier suggested that current experimental results were approximately the same as the learning curve from `Hiva` benchmark dataset given by, [64] suggesting our implementation of ALNB is correct.

After this experiment, we also reproduced the results described in Antonucci et al. [4], which evaluated the uncertainty sampling strategy with pool-based active learning for datasets with various continuous features. This paper is an empirical evaluation for AL algorithms for classification tasks, one of the classifiers investigated being NB, on different data sets from the UCI repository. In order to reproduce some of the results from Antonucci et al. [4], two benchmark datasets, `diabetes` and `iris` , were chosen from UCI repository. The `diabetes` benchmark dataset is a binary classification problem, which consist of 768 examples and 8 features, whereas the `iris` dataset has three classes with 150 examples and 4 features.
The author implemented uncertainty sampling as follows: 50 replications were generated for each dataset and the average of the accuracy learning curve was calculated on the test data. The datasets are partitioned to form the training and test sets in a stratified way. Antonucci et al. [4] randomly draw 10 patterns as the labelled training set and 100 patterns as a test set, then the remaining patterns are treated as unlabelled training patterns. The procedure was started by training on ten labelled patterns, $L = 10$, to estimate the initial parameters for the initial NB classifier. Then, the predicted probabilities of the labels for unlabelled patterns were ranked according to uncertainty sampling. five patterns with the highest uncertainty score were then selected and submitted for labelling by the oracle. Then, newly labelled patterns were added to the labelled training patterns and removed from the large unlabelled pool. Finally, the labelled training examples were used to update the parameters of a standard NB classifier. Then the accuracy of ALNB classifier is evaluated on the test set. This procedure is repeated until the active learning set is empty.
Figure 6.2a shows the mean accuracy of the ALNB classifier for the `diabetes` benchmark using uncertainty sampling with pool based active learning. Figure 6.2b shows the same

algorithm for the `iris` benchmark dataset. The results obtained suggest that the current learning curves for both `diabetes` and `iris` are equivalent to the learning curves of the Antonucci et al. [4]. Based on the initial experiments results, we then evaluated the ALNB classifier over a larger suite of benchmark datasets.

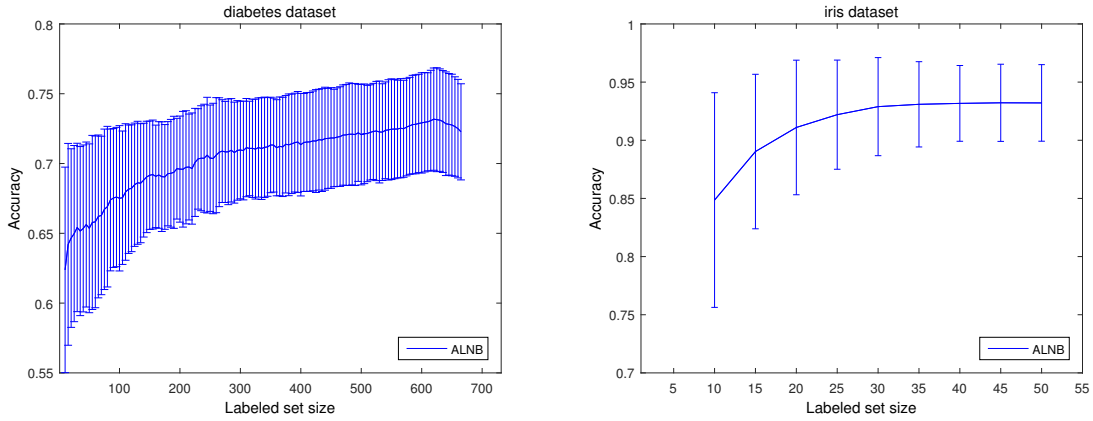


Fig. 6.2 The average accuracy of the ALNB classifier for (a)`diabetes` and (b) `iris`

6.2 Comparing passive and active learning classifier

In this section we provide an experimental comparison of the naïve Bayes (NB) and active learning naïve Bayes (ALNB) classifiers. We performed two sets of experiments for discrete and continuous real world and synthetic benchmark datasets from the UCI machine-learning repository [6], shown in Table 3.1 and 3.2 respectively. The classification performance is presented as the mean AULC over 100 error rate learning curves for the test set partition along with the standard error of the mean. The details of experimental design for active learning is as follows: for the uncertainty sampling approach, the number of oracle queries is displayed on a logarithmic scale. At each step, the best k patterns, i.e. the nearest k patterns from the decision boundary (most k uncertain patterns) of the remaining training patterns are

labelled and the model updated after each query. The AULC for active learning classifier was estimated in each replication after all unlabelled patterns are queried.

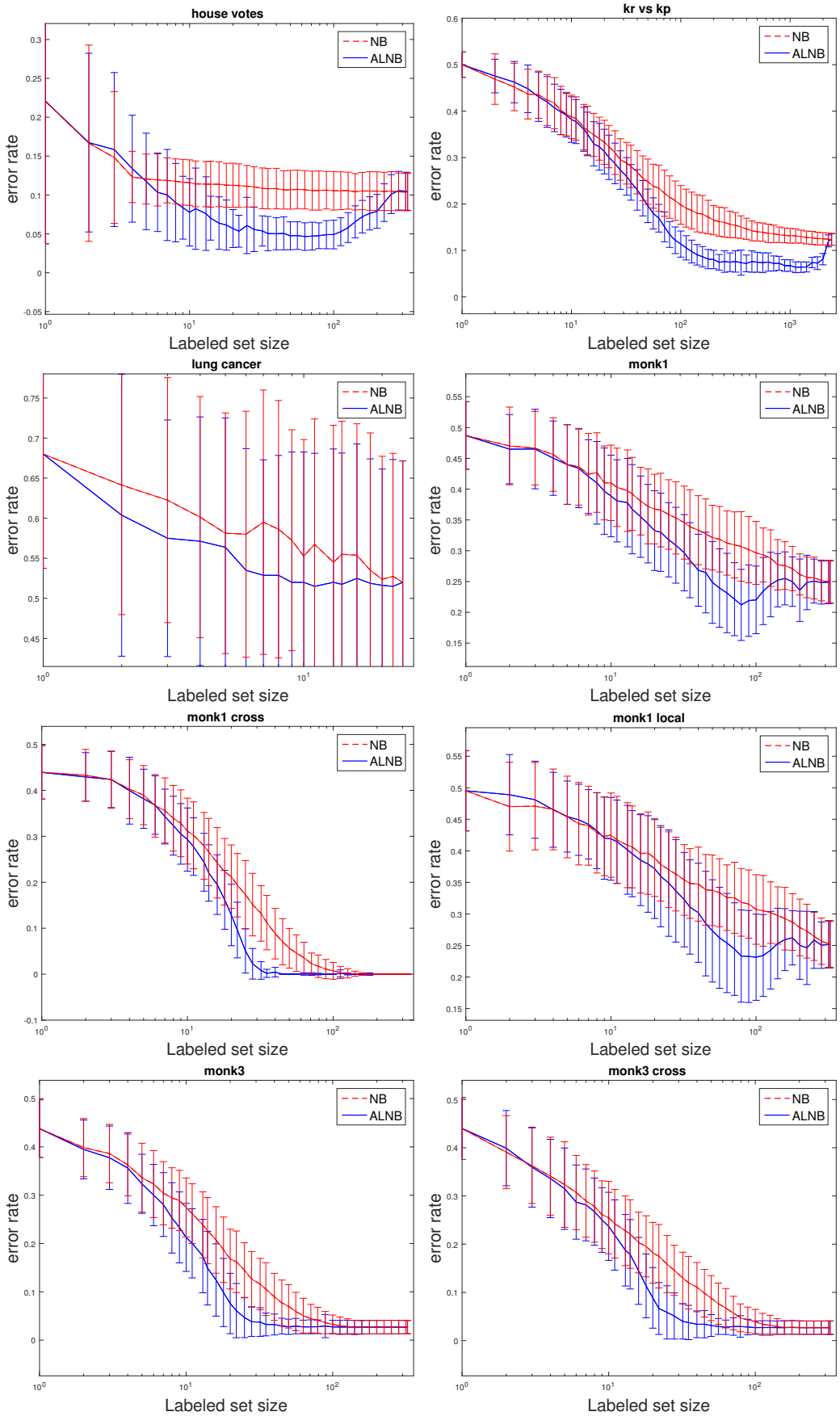
#	Dataset	NB	ALNB
1	audiology	4.891±0.0339	**4.836±0.0320**
2	balance_scale	2.830±0.0312	**2.677±0.0263**
3	blogger	***2.173±0.0422***	*2.211±0.0488*
4	breast_cancer	*2.656±0.0432*	***2.641±0.0445***
5	breastw	***0.874±0.0225***	*0.886±0.0376*
6	car	2.965±0.0308	**2.811±0.0229**
7	DNA	3.181±0.0231	**2.912±0.0196**
8	flare1	4.185±0.0354	**4.071±0.0375**
9	flare2	**4.264±0.0299**	4.392±0.0305
10	hayes_roth	3.140±0.0396	**3.004±0.0315**
11	house_votes	1.043±0.0234	**0.810±0.0222**
12	kr_vs_kp	3.136±0.0200	**2.703±0.0145**
13	led7	**5.200±0.0234**	5.325±0.0231
14	led24	6.193±0.0141	**5.973±0.0161**
15	lung_cancer	2.743±0.0442	**2.595±0.0455**
16	lymphography	2.190±0.0365	**2.063±0.0352**
17	marketing	**9.488±0.0200**	9.596±0.0249
18	monk1_corrupt	3.465±0.0245	**3.385±0.0274**
19	monk1_cross	1.803±0.0237	**1.580±0.0194**
20	monk1_local	3.218±0.0294	**3.052±0.0336**
21	monk1	3.129±0.0259	**2.897±0.0292**
22	monk3_cross	1.681±0.0275	**1.472±0.0264**
23	monk3_local	1.735±0.0223	**1.545±0.0177**
24	monk3	1.695±0.0237	**1.454±0.0224**
25	mushroom	1.771±0.0226	**1.250±0.0275**
26	nursery	3.662±0.0172	**3.449±0.0153**
27	primary_tumor	5.806±0.0301	**5.665±0.0292**
28	promoters	**1.694±0.0360**	1.752±0.0387
29	shuttle_landing_control	1.365±0.0275	**1.235±0.0292**
30	soybean_small	1.362±0.0513	**0.912±0.0230**
31	soybean_large	6.563±0.0191	**6.366±0.0204**
32	splice	3.332±0.0169	**3.164±0.0181**
33	threeOf9	2.711±0.0294	**2.670±0.0308**
34	titanic	**3.033±0.0451**	3.319±0.0688
35	xd6	2.670±0.0292	**2.606±0.0297**
36	zoo	1.696±0.0446	**1.298±0.0345**

Continued on next page

#	Dataset	NB	ALNB

Table 6.1 AULC of the NB, and ALNB classifiers with the uncertainty sampling strategy over 36 discrete benchmark datasets. The boldface font indicates that the AULC for one of the classifiers is better than the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

Table 6.1 shows that the ALNB classifier performs well compared to NB for the 29 out of the 36 benchmark datasets, this suggests that the ALNB classifier is better than the NB classifier in the current experiment. There is also a statistically significant difference between the mean ranks of the ALNB and NB classifiers for AULC according to the Wilcoxon signed rank test and the p-value > 0.01. According to this result we can see that active learning outperforms passive learning. On the following pages, the area under the error rate learning curves are shown for the above experiments to show how active learning reduces the classification error rate. The learning curves indicate the mean of the area under error rate learning curve on the test set over 100 replication. Figure 6.3 shows results for the `house-votes`, `kr-vs-kp`, `lung-cancer` , `monk1`, `monk1-cross`, `monk1-local`, `monk3`, `monk3-cross`, `monk3-local`, `mushroom`, `shuttle-landing-control`, `soybean-small`, `zoo` benchmark datasets, which are the most best results where active learning substantially improves classification performance. For example, the active learning technique for the `mushroom`, dataset can achieve almost minimal level of error rate with only 40 patterns whereas passive learning requires more than 6000 patterns to obtain this error rate level, (see Figure 6.3). Similarly, the passive learning technique for the `kr-vs-kp` benchmark needs more than 2300 patterns to achieve same error rate level in active learning. From the results, it is important to note that an active learning technique can demonstrate good classification performance using a relatively small proportion of the training labelled set which is why it can be said that we only need a small labelling cost if active learning does work.


house votes
kr vs kp
lung cancer
monk1
monk1 cross
monk1 local
monk3
monk3 cross
NB
ALNB
error rate
Labeled set size

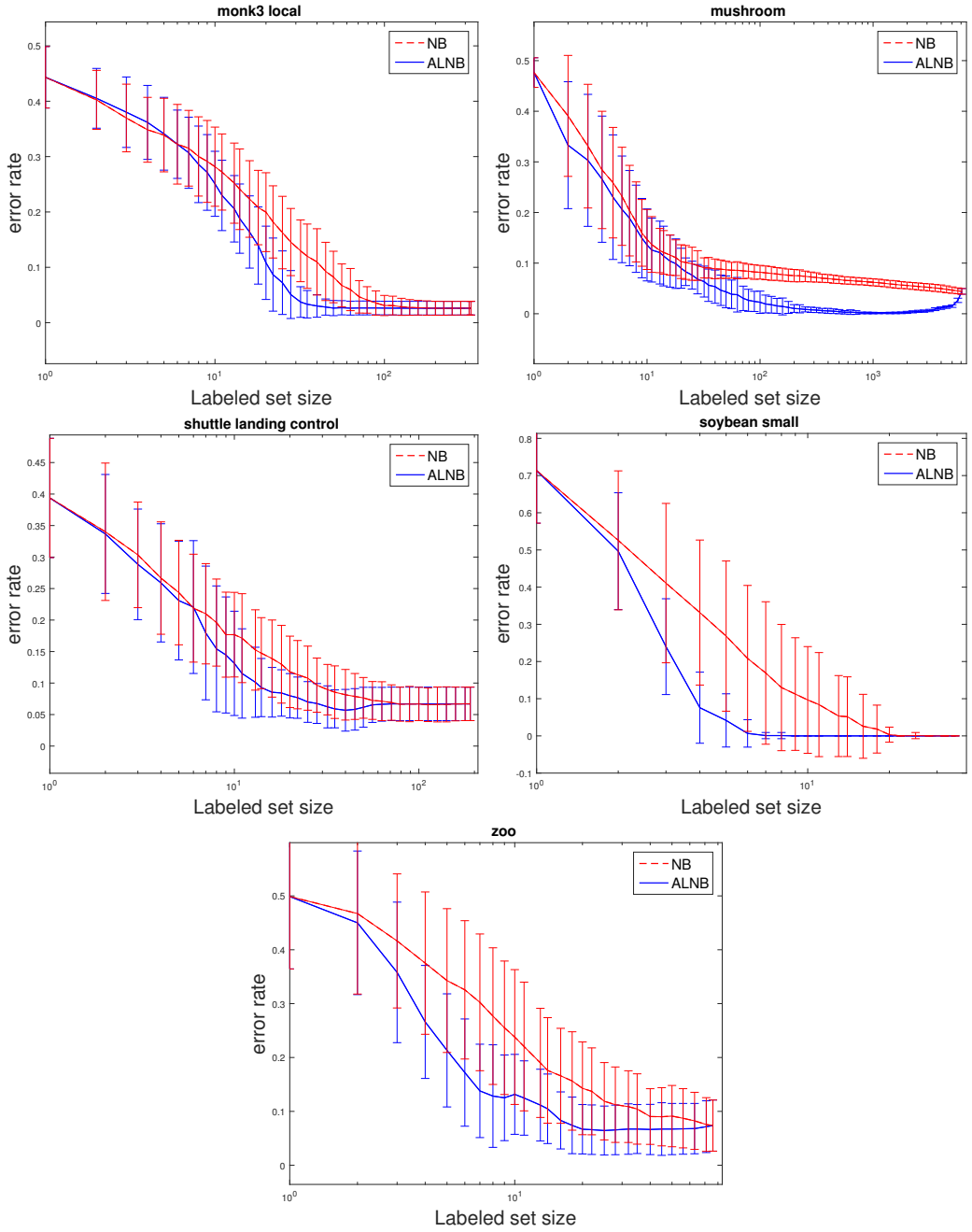
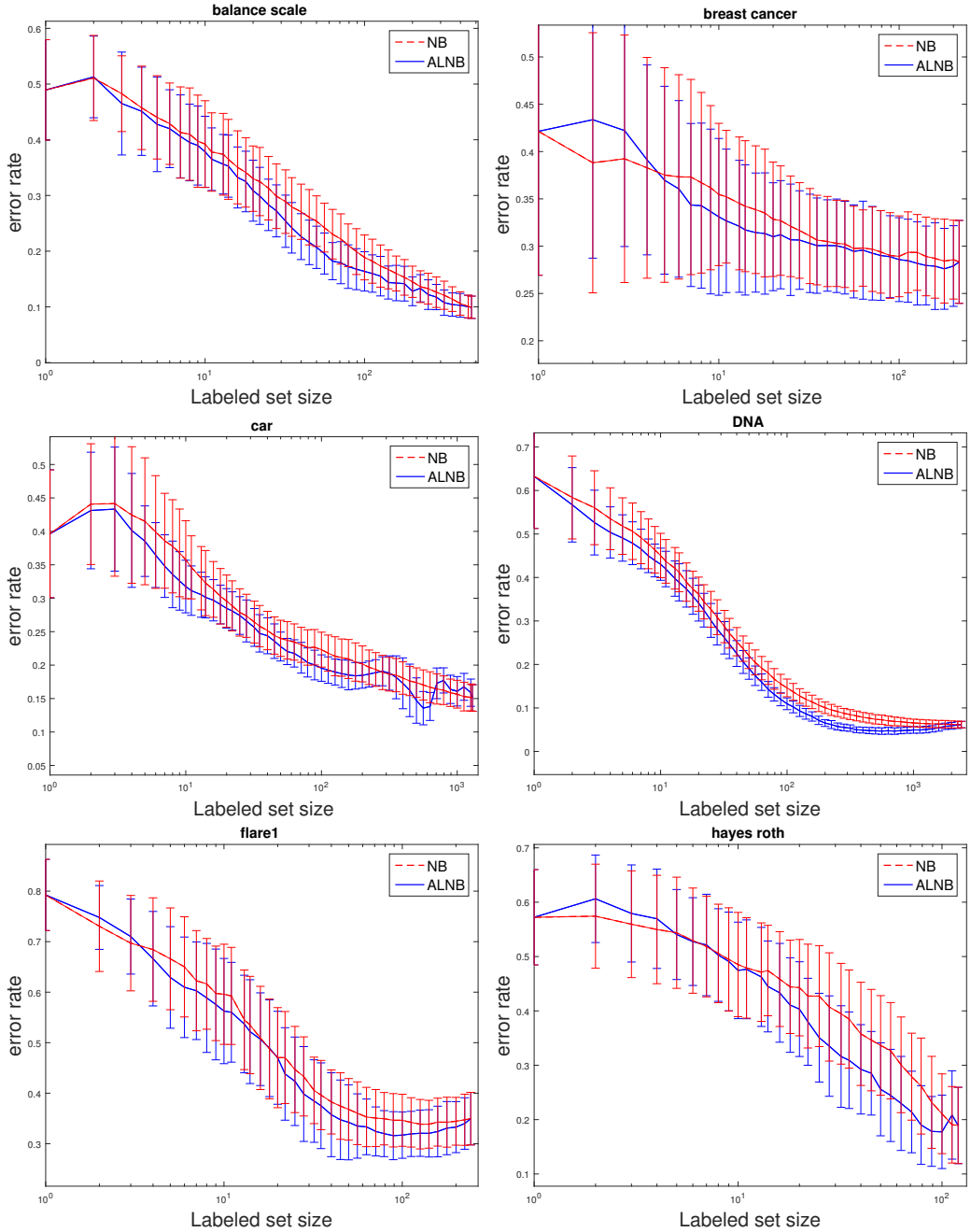


Fig. 6.3 Comparison of the error rate learning curve for the NB and ALNB classifiers. In these cases, the ALNB better than the NB classifier and the difference between these two classifiers are statistically significant for 13 discrete benchmark datasets.

Similar to the previous results, we can present the learning curve results for the `car`, `balance-scale`, `breast-cancer`, `DNA`, `flare1`, `hayes-roth`, `led24`, `lymphography`, `xd6`, `monk3-corrupt`, `primary-tumor`, `soybean-large`, `splice`, `threeOf9` benchmark

datasets (Figure 6.4). The area under the error rate learning curve for active learning demonstrates that reduced classification error rate is achieved slightly faster than for passive learning, but it is not a statically significant improvement according to the Wilcoxon signed rank test.



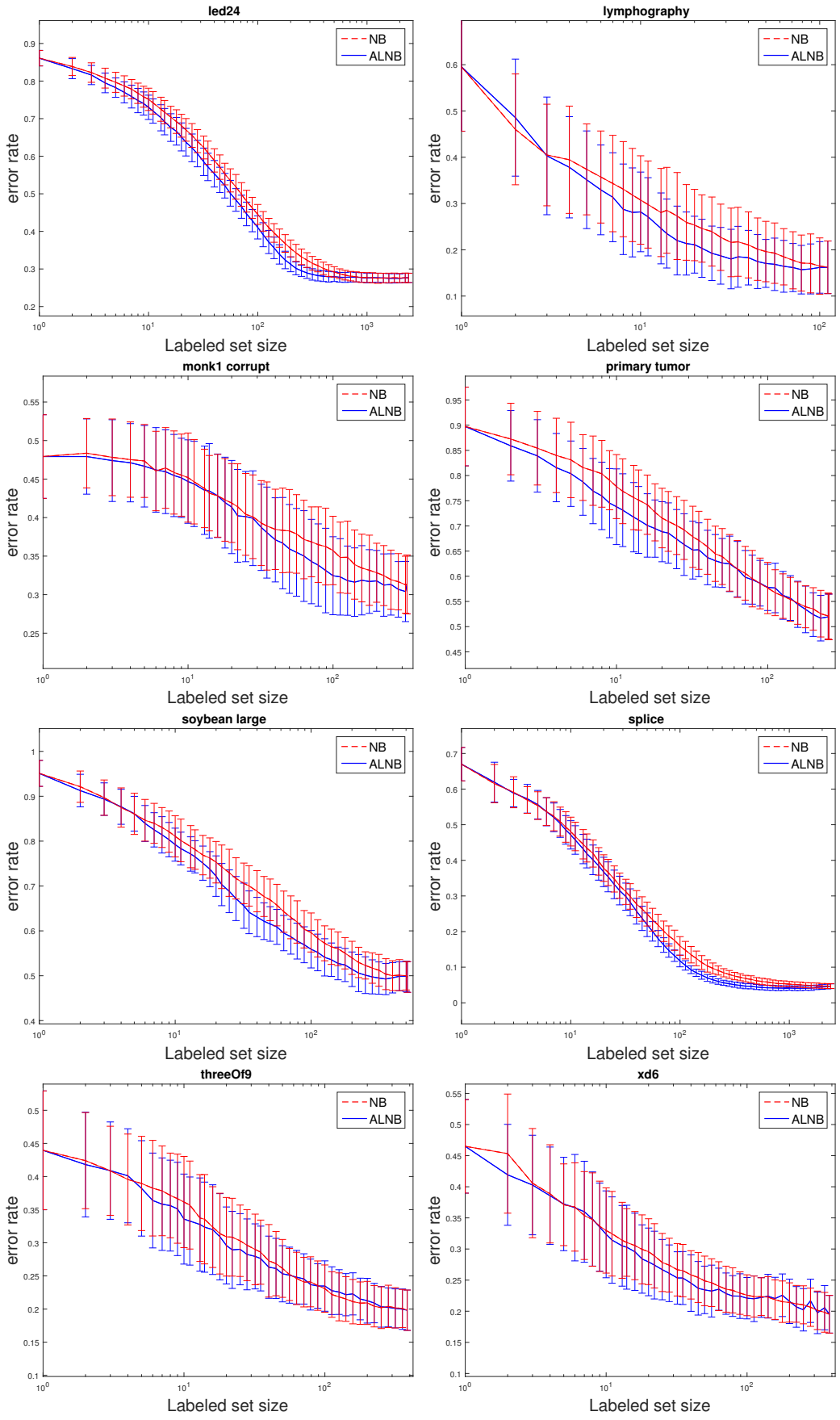


Fig. 6.4 Comparison of the error rate learning curve for NB and ALNB classifiers. In these cases, the ALNB classifier performs slightly better than NB classifier over 14 discrete benchmark datasets.

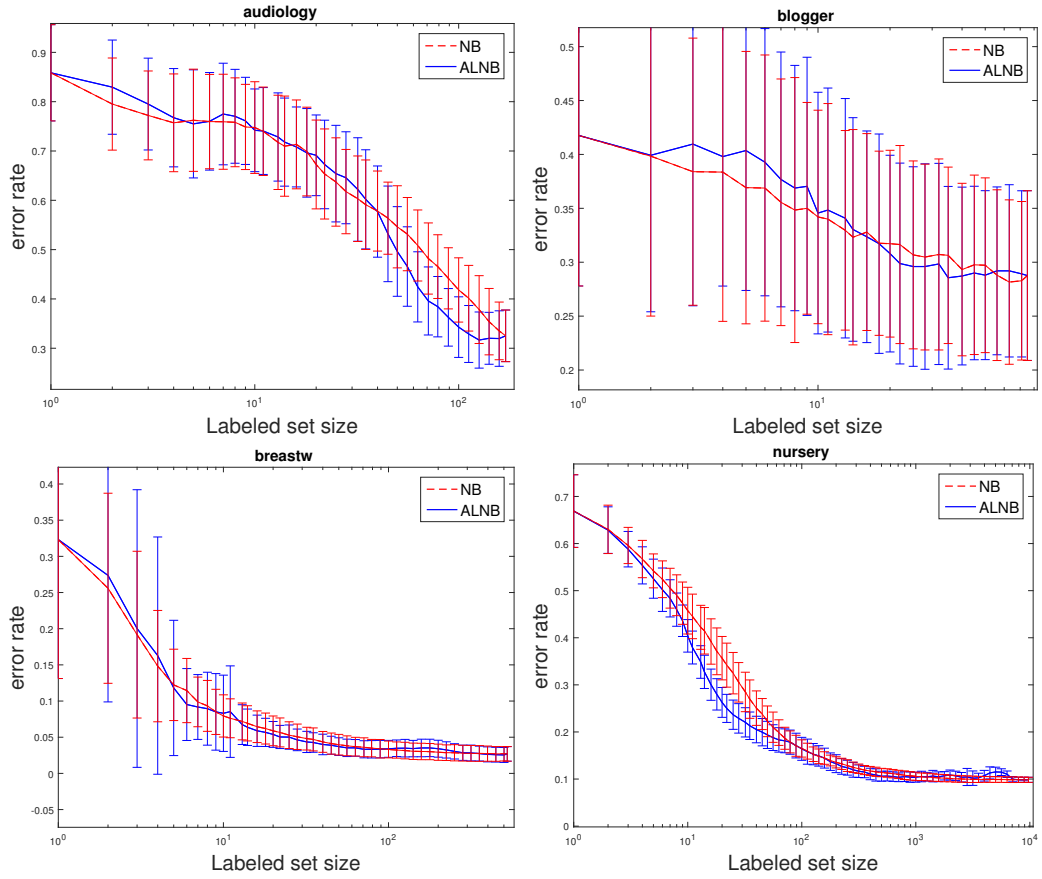


Fig. 6.5 Comparison of the error rate learning curve for NB and ALNB classifiers. In these cases, both classifiers have equivalent performance on four discrete benchmark datasets.

For a few benchmark datasets, for example `audiology`, `blogger`, `breastw`, `led7`, `nursery` active learning shows no improvement in classification performance (Figure 6.5). So, adding additional patterns actively or passively to the training labelled set achieves a similar reduction in error rate. Finally, active learning is worse than passive learning for the remaining benchmark datasets shown in Figure 6.6.

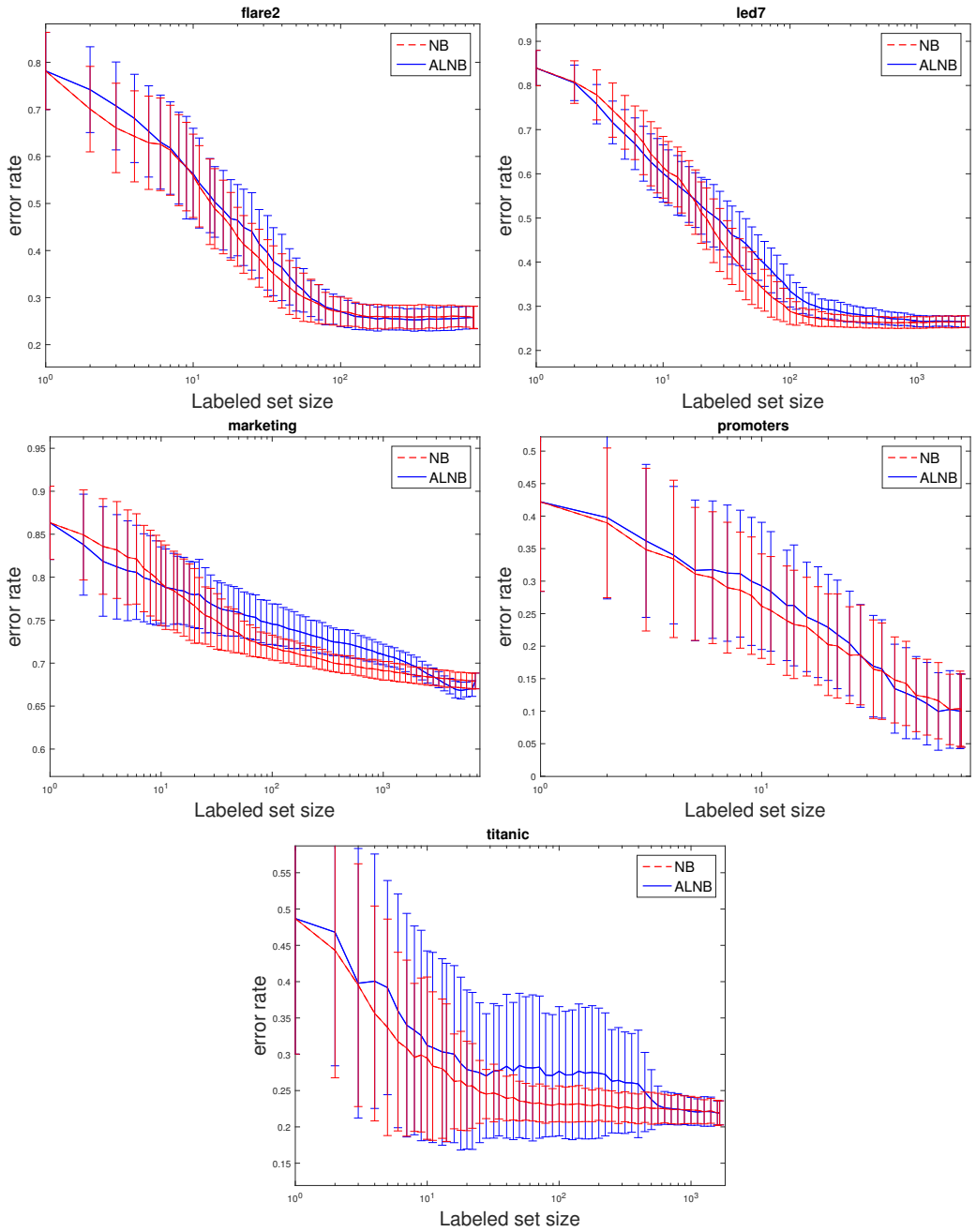


Fig. 6.6 Comparison of the error rate learning curves for NB and ALNB classifiers. The ALNB classifier is inferior to the NB classifier and the differences between these two classifiers are statistically significant for five discrete benchmark datasets.

So far the active learning method is compared to passive learning on benchmark datasets with discrete feature. Clearly, active learning techniques yield superior results on most of the datasets. Thus, it seems that active learning techniques are suitable for problems with

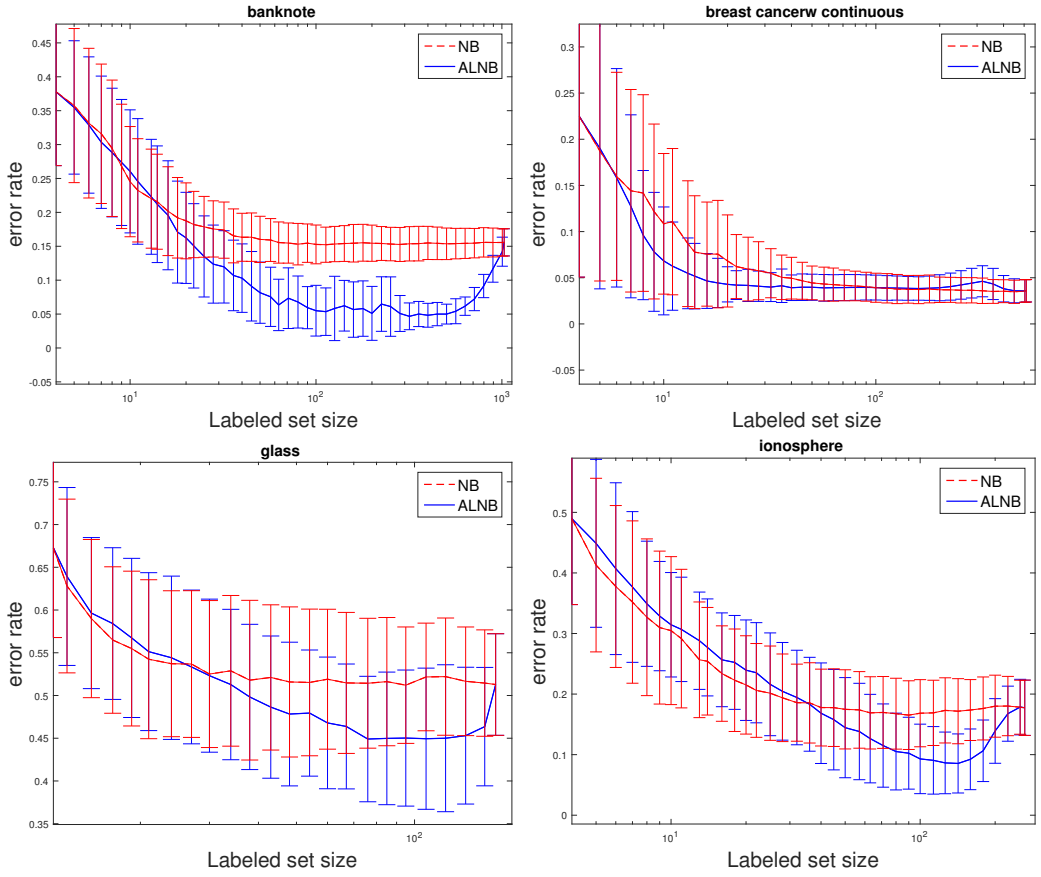
unlabelled data. The second experiment performed in order to evaluate the performance of active learning for datasets with continuous features. This experiment shows the comparison of active and passive learning for the NB classifier on the 28 continuous benchmark datasets and the results presented in Table 6.2.

#	Dataset	NB	ALNB
1	banknote	1.541±0.0225	**1.074±0.0225**
2	Blood_transfusion	***2.070±0.0262***	*2.080±0.0458*
3	breast_cancerw_continuous	0.500±0.0167	**0.432±0.0129**
4	Climate_Model_Simulation_Crashes	***0.603±0.0140***	*0.630±0.0208*
5	glass	2.006±0.0192	**1.902±0.0205**
6	haberman	**1.778±0.0301**	1.870±0.0339
7	ionosphere	1.415±0.0278	**1.340±0.0290**
8	iris	0.418±0.0187	**0.377±0.0180**
9	letter	**4.201±0.0111**	4.567±0.0152
10	liver_disorder	2.719±0.0210	**2.565±0.0304**
11	magic04	3.506±0.0182	**3.194±0.0362**
12	musk1	**2.207±0.0225**	2.263±0.0291
13	new_thyroid	*0.352±0.0150*	***0.347±0.0164***
14	pendigits	***2.104±0.0139***	*2.106±0.0200*
15	sleep	**4.956±0.0145**	5.930±0.0370
16	vehicle	3.517±0.0166	**3.073±0.0230**
17	vowel	2.051±0.0138	**1.945±0.0130**
18	waveform_noise	2.581±0.0145	**2.481±0.0155**
19	waveform	**2.917±0.0121**	2.978±0.0205
20	wine	0.640±0.0187	**0.499±0.0164**
21	arcene	*2.160±0.0223*	***2.156±0.0221***
22	gisette	***2.099±0.0117***	*2.129±0.0145*
23	madelon	***4.116±0.0091***	*4.119±0.0095*
24	sonar	1.928±0.0239	**1.729±0.0211**
25	spambase	*2.263±0.0195*	***2.238±0.0281***
26	Synthetic	**1.182±0.0217**	1.332±0.0320
27	vertebral	1.421±0.0200	**1.357±0.0270**
28	diabetes	**2.137±0.0207**	2.206±0.0234

Table 6.2 AULC of the NB and ALNB classifiers with uncertainty sampling strategies over 28 continuous benchmark datasets. The boldface font indicates that the AULC for one of the classifiers between the NB and ALNB is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

From Table 6.2 it can be seen that the AULC of the ALNB is better than the NB for only half of the datasets. In addition, there is no statistically significant difference between active and passive learning over all datasets according to the Wilcoxon signed rank test at the 95% level of significance (p-value > 0.01). On the following pages, the learning curves results for the above experiment are shown.

Figure 6.7 shows how the average area under the learning curve over 100 random replications for the active learning compares to the passive learning classifier on the test set. As we can see, the area under error rate learning curve for active learning is slightly higher from the beginning of the learning curve but after only few additional training labelled patterns are used, active learning achieves consistently good classification performance on the `banknote`, `breast-cancerw-continuous`, `glass`, `ionosphere`, `iris`, `liver-disorder`, `magic04`, `sonar`, `vehicle`, `vertebral`, `vowel`, and `wine` benchmark datasets.



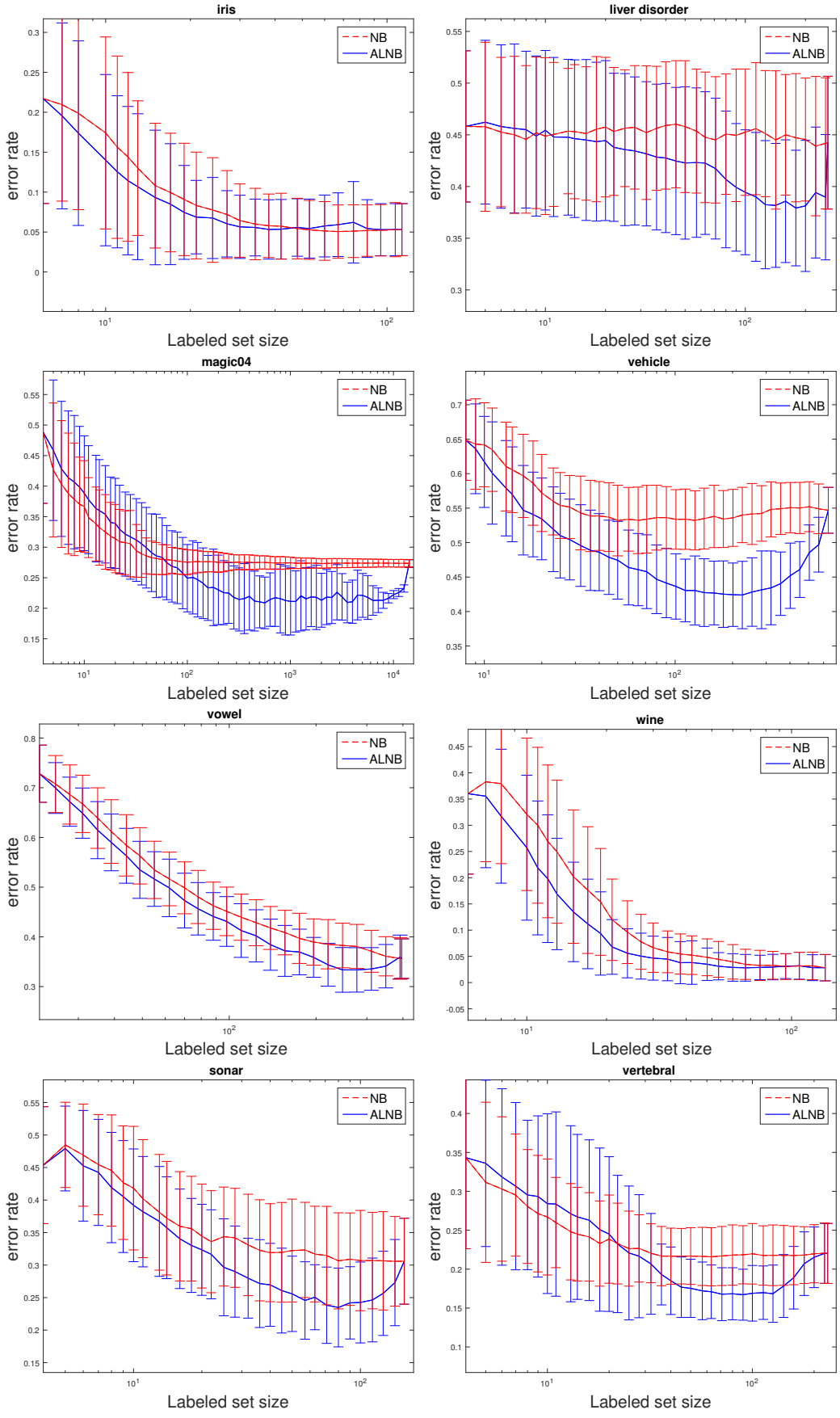


Fig. 6.7 Comparison of the error rate learning curve for NB and ALNB classifiers. In these cases, the ALNB performs better than NB classifier and the differences between these two classified are statistically significant for 12 continuous benchmark datasets.

Figure a 6.8 shows the learning curves for passive and active learning on the `waveform-noise` benchmark dataset. It can be seen that active learning is slightly better than passive learning. Thus, the sample selection of active learning is not very effective for the NB classifier compared to random sampling for this dataset.

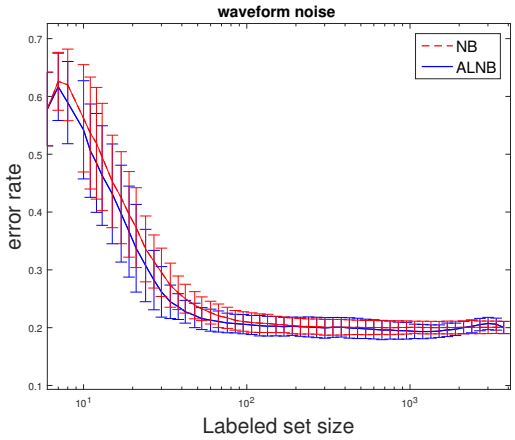


Fig. 6.8 Comparison of the error rate learning curve for the NB and ALNB classifiers on `waveform-noise`. In these cases, the ALNB performs slightly better than NB classifier.

Active learning does not perform better than passive learning for the `arcene`, `gisette`, `madelon`, `pendigits`, `Climate-Model-Simulation-Crashes`, `new-thyroid`, `spambase` benchmark datasets. We show the learning curves for these benchmark datasets in Figure 6.9. The performance of active and passive learning is very similar and stable. Finally, active learning is worse than passive learning for the remaining benchmark datasets, shown in Figure 6.10.

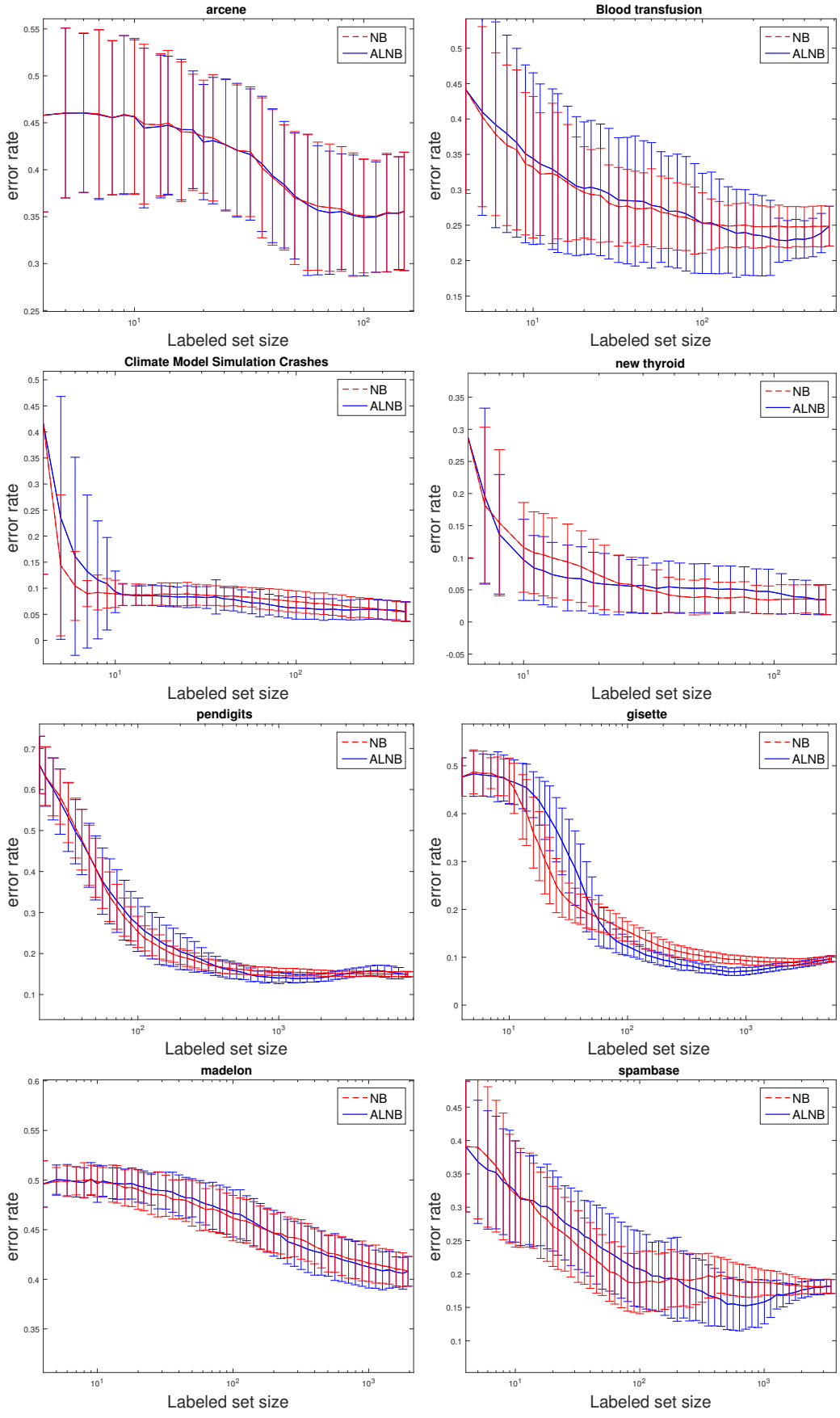


Fig. 6.9 Comparison of the error rate learning curve for NB and ALNB classifiers. In these cases, both classifiers has equivalent performance on eight continuous benchmark datasets.

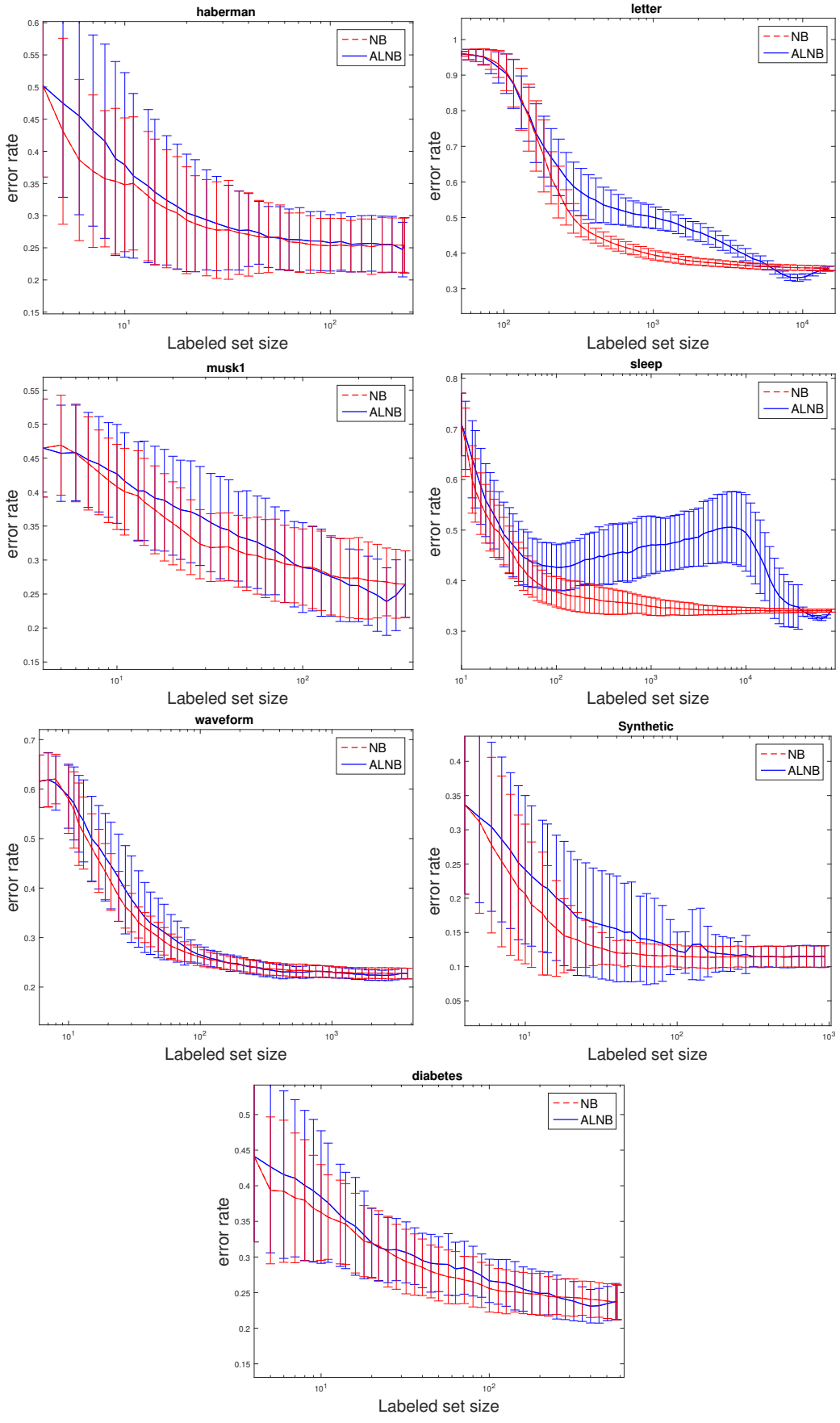


Fig. 6.10 Comparison of the error rate learning curves for NB and ALNB classifiers. The NB better than ALNB classifier and the differences between these two classified are statistically significant for seven continuous benchmark datasets.

In a short summary on the continuous benchmark datasets, these results may indicate that the training patterns are equally informative for both learning techniques therefore adding different amount of the training patterns almost has the same effect and give comparable learning curves for these datasets.

In this experiment the area under the error rate learning curve was used for evaluating the classification performance. For these learning curves, building a good initial model is very helpful for achieving high performance in active learning algorithm, as described in Figure 1.6. The initial labelled training patterns are generally selected at random for building the initial model, so, based on random sampling we may still build a good initial model. However, in reality, occasionally we can build reasonably good initial models due to the small size of initial training labelled patterns. Therefore, the error rate learning curve for active learning classifier is generally inferior to passive learning. However, in some cases the error rate learning curve rises again, as can be seen in Figures 6.3 and 6.7 for 27 benchmark datasets in both previous experiments.

In this section, uncertainty sampling for pool-based active learning was investigated but did not provide much help to improve the classifier. When uncertainly sampling selects the most uncertain unlabelled patterns, it seems it often fails by selecting outliers. The new experiments in the next section are designed, to investigate whether the selected unlabelled patterns at the end of these learning curves are outliers or not.

6.3 Investigation of anomalous learning curves

In theory, active learning iteratively attempts to reduce the classification error rate by labelling the most informative patterns from a large pool of unlabelled examples. Through the error rate learning curve this error rate reduction can be seen when the active learning algorithm continues the labelling process. However, we found that sometimes the error rate learning curve rises again. These results suggest active learning might select uninformative patterns and degrade the classification performance. For example, the learning curves for `banknote`, `glass`, `ionosphere`, `liver-disorder`, `magic04`, `vehicle`, `sonar`, and `vertebral` in the continuous benchmark datasets and `house-votes`, `kr-vs-kp`, `monk1`, and `mushroom`, in the discrete benchmark datasets show this phenomenon. It is important to determine why the active learning classifier shows these kinds of result. It is probably when the active learning classifier has reached its maximum effectiveness and no informative patterns could be found in the remaining unlabelled patterns which might include outliers.

In such a situation, removing patterns at the end of the learning curve might effectively avoid selecting these outliers as the most informative patterns, improving classification performance. However, there is a difficulty in identifying these patterns as outliers at the end of the learning curve because the learning curve come from averaging 100 replications, especially for large benchmark datasets. For example, in the `mushroom`, dataset active learning utilises nearly 1000 unlabelled patterns at the end of learning curve. Therefore, pattern removing techniques cannot be easily applied for uncertainty sampling for active learning. However, at least to determine whether these unlabelled patterns are outliers or not that often cause uncertainty sampling to fail, we used the `liver-disorder` benchmark dataset, shown in Figure 6.11, that utilises seven unlabelled patterns at the end of learning curve where the error rises again sharply.

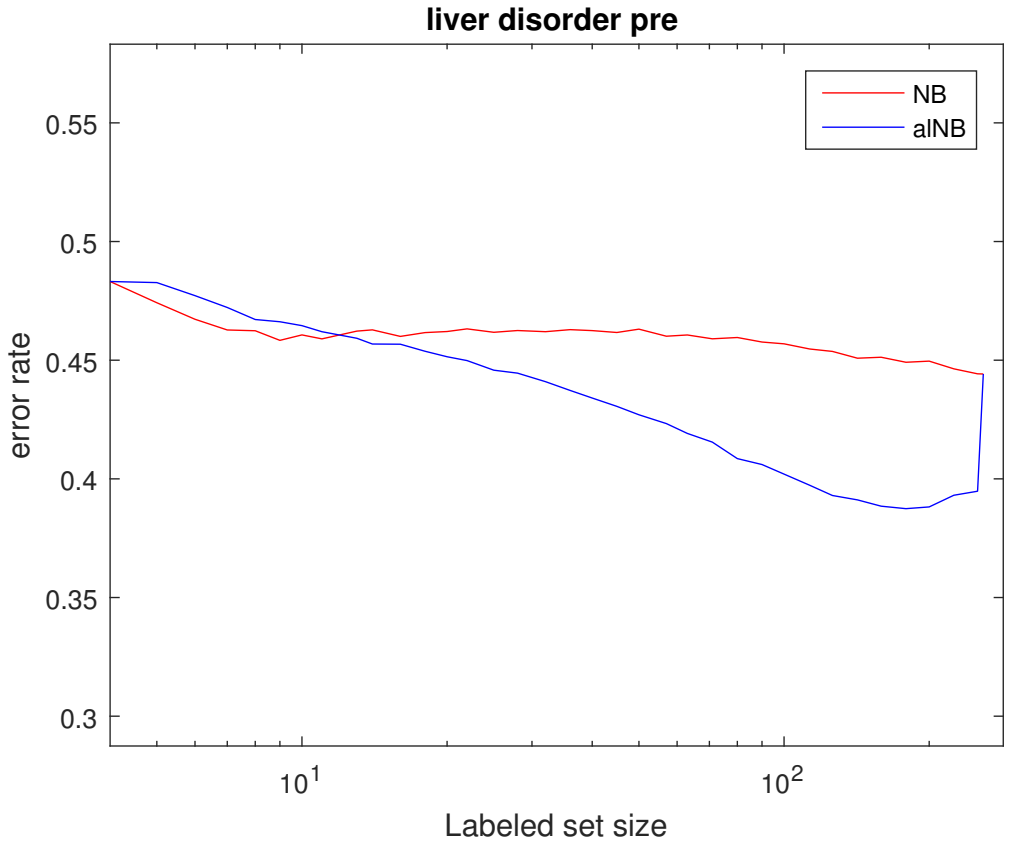


Fig. 6.11 The average of the error rate learning curve for 1000 replications for both active (alNB) and passive (NB) learning for the liver-disorder benchmark dataset

To investigate why the error rate learning curve increases again, we design another simple experiment on the `liver-disorder` dataset chosen because only a few patterns remain at the end of learning curve. In this experiment instead of 100 replications, we increased the number of replications to 1000 in order for each pattern to have the opportunity to appear in a training set a large number of times. Figure 6.11 shows the average result of 1000 learning curves for both active and passive learning on the test set.

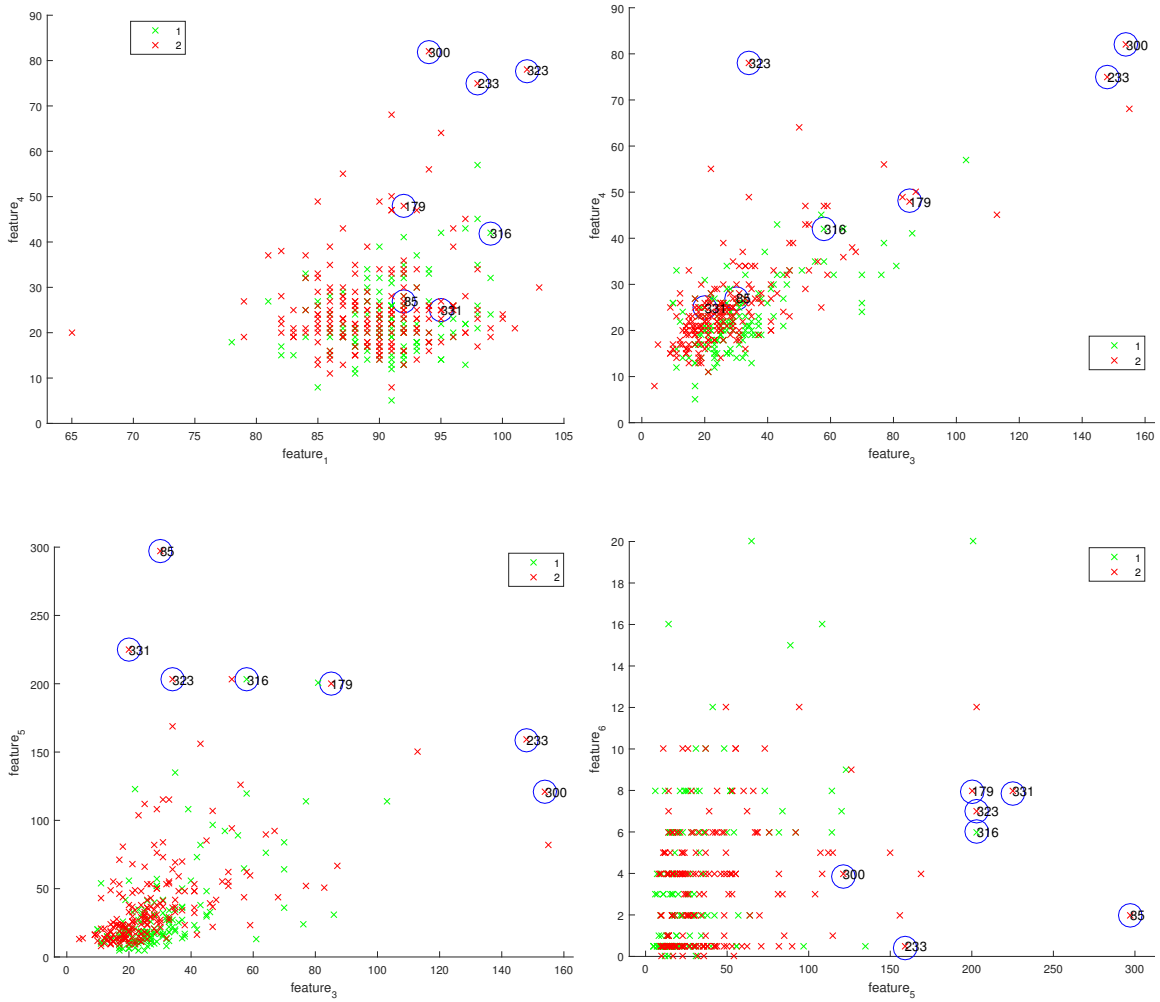


Fig. 6.12 Scatter plot between Liver-disorder benchmark datasets feature patterns. The seven removed patterns are shown inside circles.

Finally, we remove the last seven most uninformative patterns at the end and the passive learning classifier improved significantly. Figure 6.12, shows these seven patterns that were selected as an outliers. Figure 6.13 shows the average of error rate learning curve for both active and passive learning without those seven data points. As we can see the error rate learning curve has jumped at the end and there is huge difference compared to Figure 6.11. This result suggests that these patterns are probably outliers. Finally, we remove the last seven patterns at the end which are uninformative patterns.

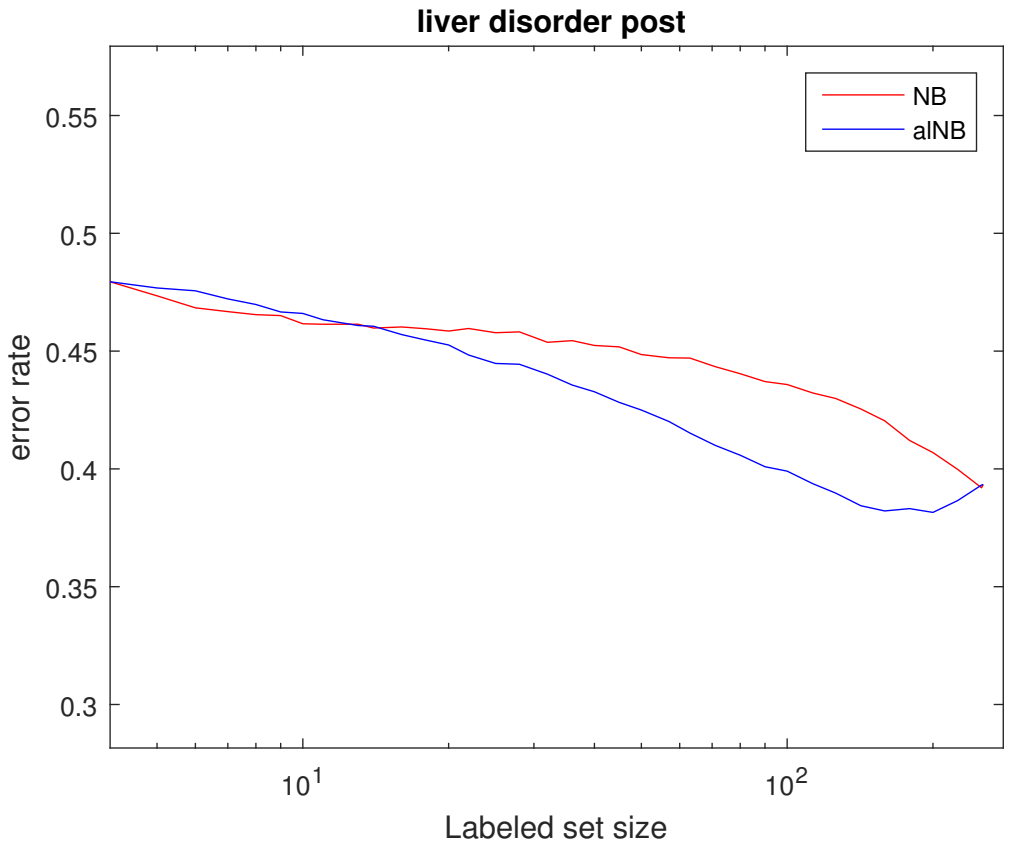


Fig. 6.13 The average of 1000 area under error rate learning curve for both active (alNB) and passive (NB) learning for the liver-disorder benchmark dataset after removal of outliers.

6.4 Combining active and semi-supervised learning

Passive learning techniques attempt to select patterns randomly from a large pool of unlabelled patterns to be labelled, by contrast, active learning techniques attempt to select the most informative patterns to be labelled by using sampling strategies. The previous section 6.2, shows that a lower error rate can be achieved with fewer training labels and so can reduce human labelling effort. Moreover, we show that there is a statistically significant difference between active and passive learning. Those patterns that have the least confidence are selected to be labelled by the oracle in active learning process, but what happen to those patterns that have high confidence scores, which are not selected to be labelled? Especially if

from the beginning of the learning process we left out a considerable number of unlabelled patterns. In this section, we use the semi-supervised learning to further exploit these unlabelled patterns. These unlabelled patterns are attempted to use via semi-supervised learning to improve classification performance. The research question that arises in this section is, does the combination of active learning and semi-supervised learning give better performance than active learning alone?

#	Dataset	NB	ALSSNB
1	audiology	**4.891±0.0339**	5.288±0.0272
2	balance_scale	**2.830±0.0312**	3.010±0.0227
3	blogger	**2.173±0.0422**	2.484±0.0415
4	breast_cancer	**2.656±0.0432**	2.899±0.0366
5	breastw	0.874±0.0225	**0.466±0.0357**
6	car	**2.965±0.0308**	3.295±0.0338
7	DNA	3.181±0.0231	**2.108±0.0642**
8	flare1	4.185±0.0354	**4.117±0.0361**
9	flare2	**4.264±0.0299**	4.357±0.0332
10	hayes_roth	**3.140±0.0396**	3.376±0.0273
11	house_votes	**1.043±0.0234**	1.088±0.0304
12	kr_vs_kp	**3.136±0.0200**	4.734±0.0160
13	led7	5.200±0.0234	**5.113±0.0232**
14	led24	6.193±0.0141	**5.511±0.0226**
15	lung_cancer	2.743±0.0442	**2.507±0.0490**
16	lymphography	***2.190±0.0365***	*2.210±0.0354*
17	marketing	**9.488±0.0200**	9.802±0.0221
18	monk1_corrupt	***3.465±0.0245***	*3.512±0.0294*
19	monk1_cross	**1.803±0.0237**	1.878±0.0220
20	monk1_local	**3.218±0.0294**	3.432±0.0341
21	monk1	3.129±0.0259	**2.996±0.0313**
22	monk3_cross	**1.681±0.0275**	2.101±0.0378
23	monk3_local	**1.735±0.0223**	2.483±0.0266
24	monk3	**1.695±0.0237**	1.786±0.0245
25	mushroom	1.771±0.0226	**1.626±0.0212**
26	nursery	3.662±0.0172	**3.467±0.0182**
27	primary_tumor	**5.806±0.0301**	5.872±0.0252
28	promoters	1.694±0.0360	**1.316±0.0354**
29	shuttle_landing_control	*1.365±0.0275*	***1.307±0.0343***
30	soybean_small	1.362±0.0513	**0.749±0.0236**
31	soybean_large	*6.563±0.0191*	***6.561±0.0211***

Continued on next page

#	Dataset	NB	ALSSNB
32	splice	3.332±0.0169	**2.588±0.0488**
33	threeOf9	**2.711±0.0294**	2.873±0.0233
34	titanic	**3.033±0.0451**	3.411±0.0745
35	xd6	**2.670±0.0292**	2.714±0.0282
36	zoo	1.696±0.0446	**1.252±0.0367**

Table 6.3 AULC of the NB, and ALSSNB classifiers with uncertainty sampling strategy over 36 discrete benchmark datasets. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

Our first experiments found that the use of the unlabelled dataset does not generally reduce the AULC. Table 6.3 shows the results for 36 discrete benchmark datasets. Passive learning was best on 21 out of the 36 benchmark datasets, active learning with semi-supervised naïve Bayes (ALSSNB) being best on 15. The result of the Wilcoxon signed rank test shows that the NB classifier is statistically superior at the 95% level of significance.

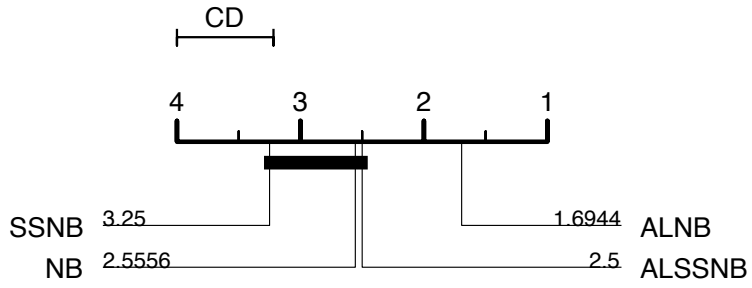
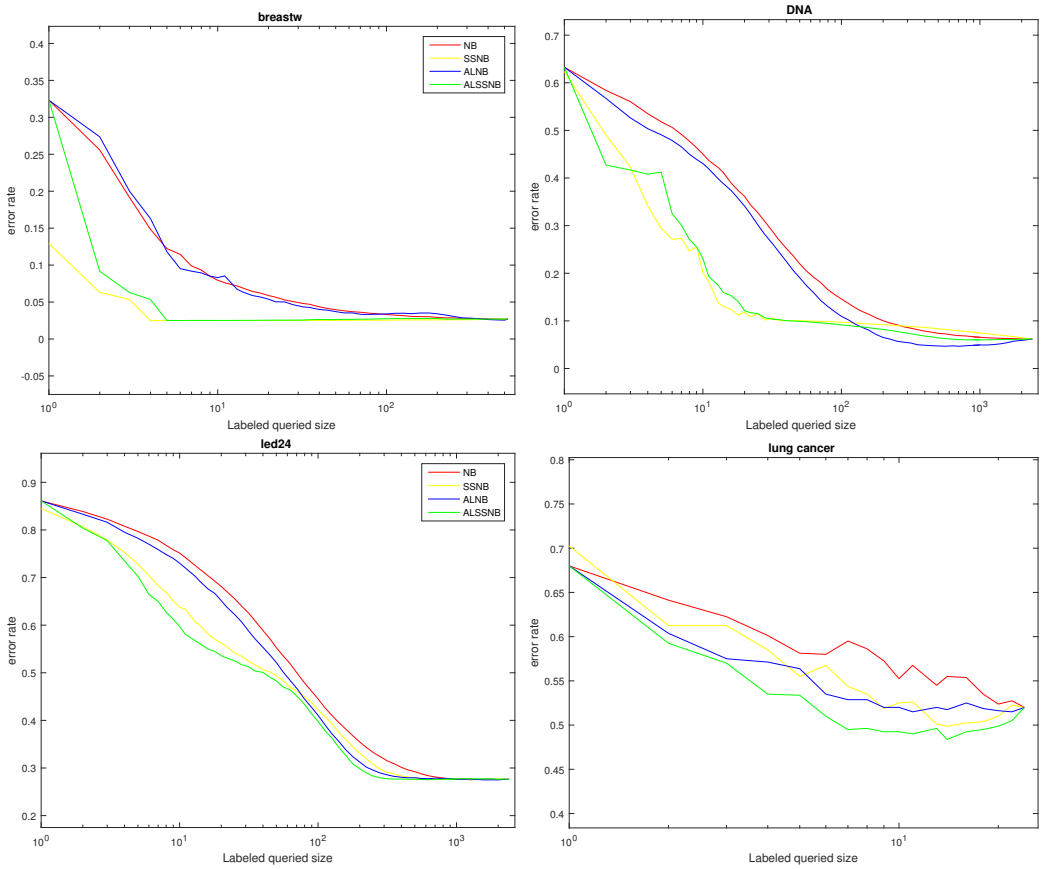


Fig. 6.14 Critical difference diagram for NB, SSNB, ALNB, and ALSSNB over 36 discrete benchmark datasets. It shows that the ALNB classifiers are statistically superior to the other classifiers.

The critical difference diagram can be used to compare the average ranks of the classifiers. Figure 6.14 shows that the average rank of ALSSNB is slightly higher that for NB and SSNB but the difference is not statistically significant. However, the average rank of ALSSNB is inferior to the ALNB, and so the ALNB classifier is statistically superior to the other classifiers. The result suggest that SSNB generally makes the active learning algorithm worse rather than better.

Figure 6.15 shows the efficiency of combing active learning with semi-supervised learning on the ten benchmark datasets. Compared to the baseline classifier, which is the conventional naïve Bayes classifier, and each of active and semi-supervised learning alone, combing active learning with semi-supervised learning can achieve significant improvements on four datasets `lung-cancer`, `monk1`, `soybean-small`, and `splice`. Among the six remaining benchmark datasets, combing active learning with semi-supervised learning gives nearly the same performances as either active learning or semi-supervised learning, while combing active learning with semi-supervised learning outperforms passive learning on the all benchmark datasets in Figure 6.15.



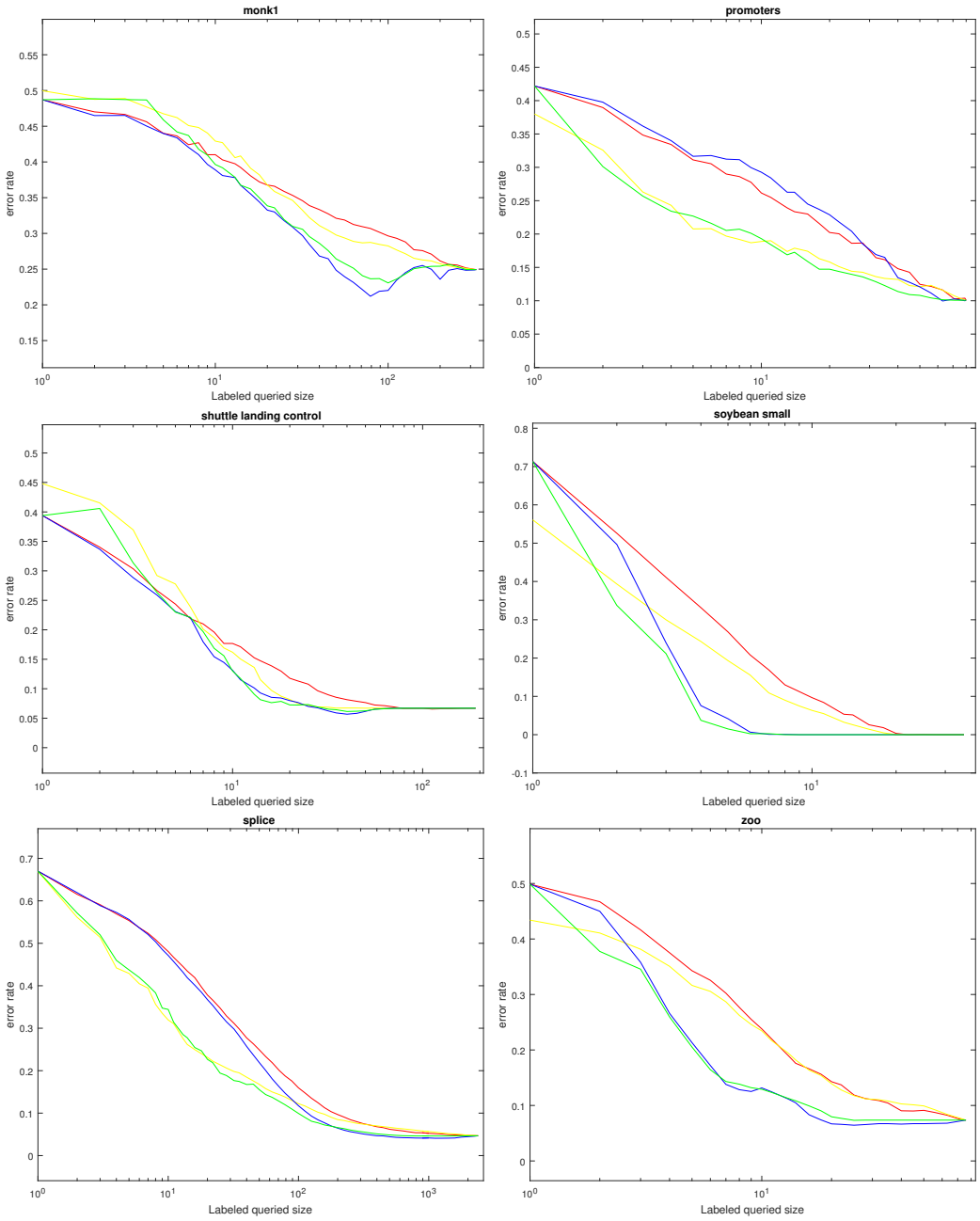
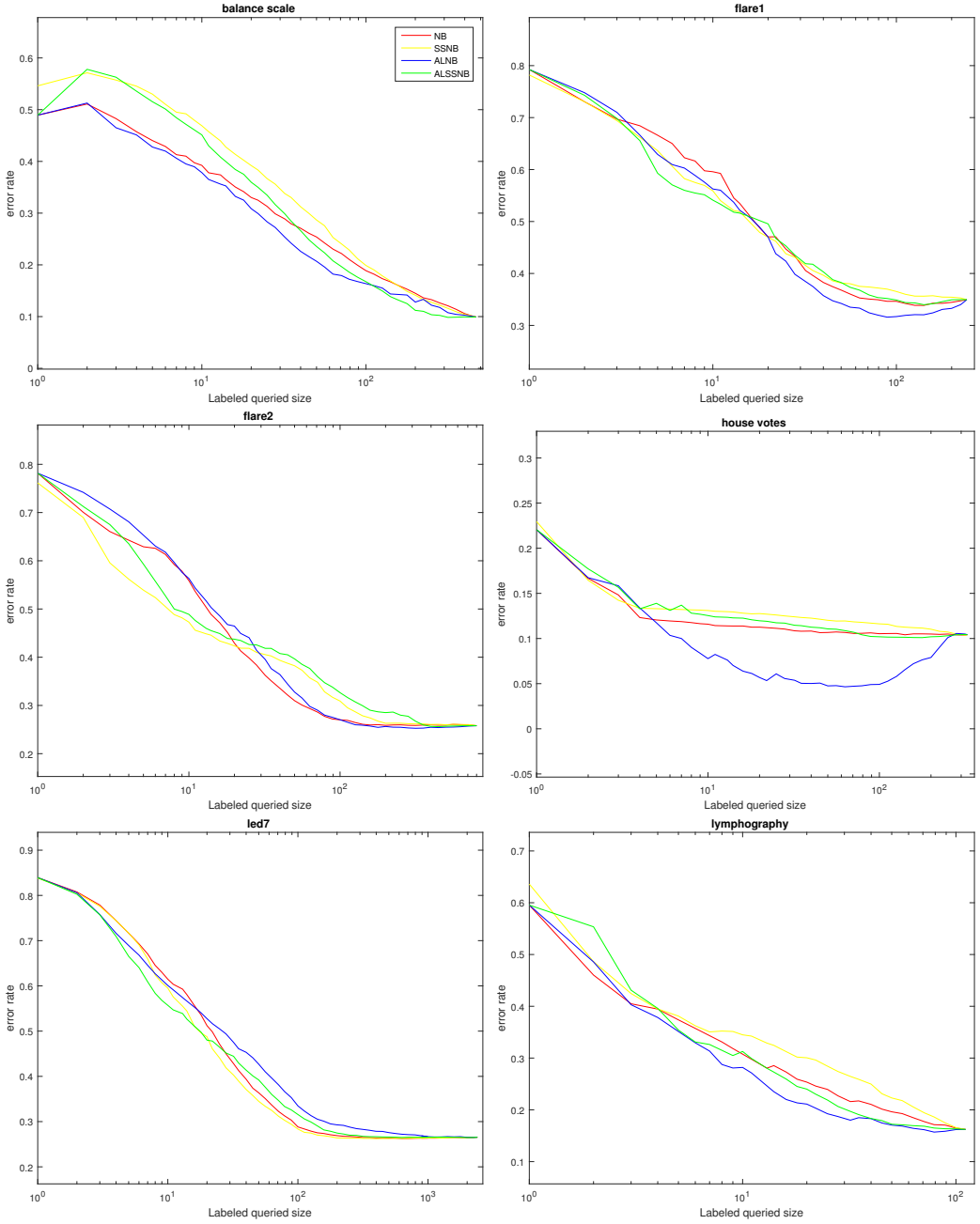


Fig. 6.15 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. In these cases, the ALSSNB is statistically superior to the NB classifier for 10 discrete benchmark datasets.

Figure 6.16 shows that combing active learning with semi-supervised learning yields no significant improvements compared to passive learning. Moreover, active learning gives slightly better performance compared to other classifiers. In addition, active learning achieves

the best performances on the `house-votes`, `monk1-cross`, `monk1-local`, and `mushroom` benchmark datasets, shown in Figure 6.16. These results seem to suggest that using the unlabelled patterns via semi-supervised learning after selecting the least confident classifier patterns through active learning does not improve classification performance.



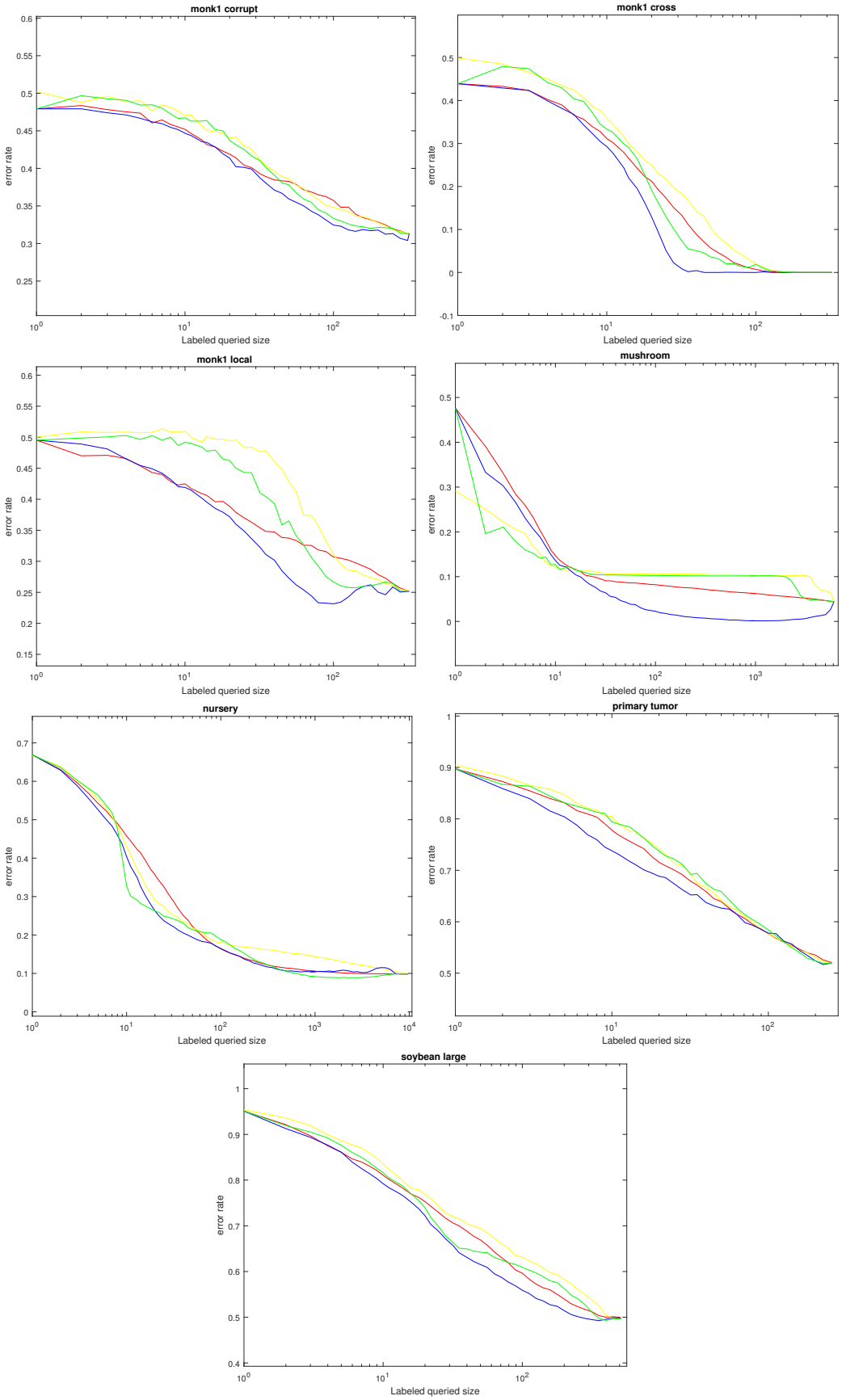
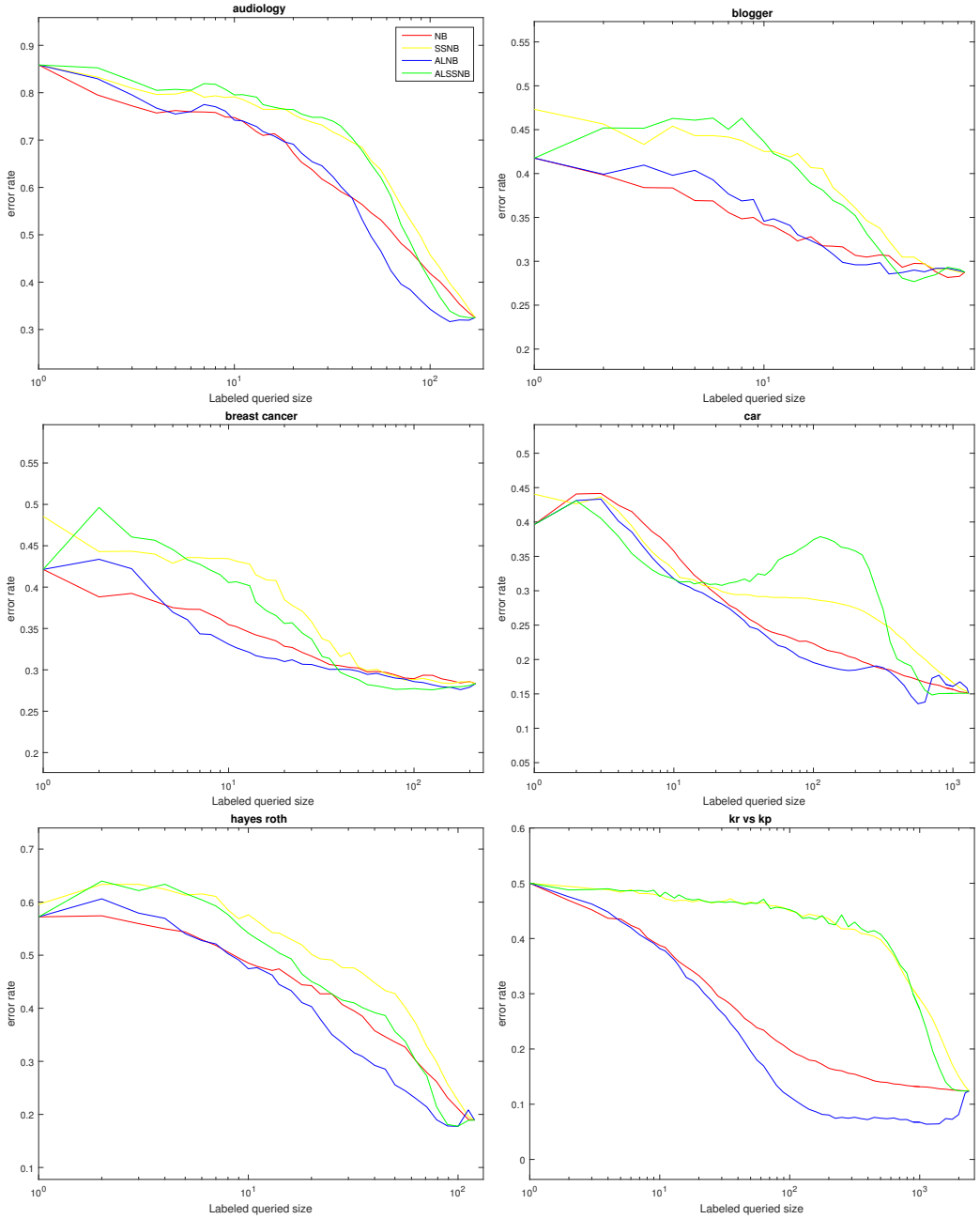


Fig. 6.16 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. The ALSSNB classifier have equivalent performance over 13 discrete benchmark datasets.

Figure 6.17 shows that combining active with semi-supervised learning for one third of the UCI discrete benchmark datasets in Table 6.3 makes the classifier worse. However, in some cases involving unlabelled patterns with the active learning technique can improve the performance of the classifier.



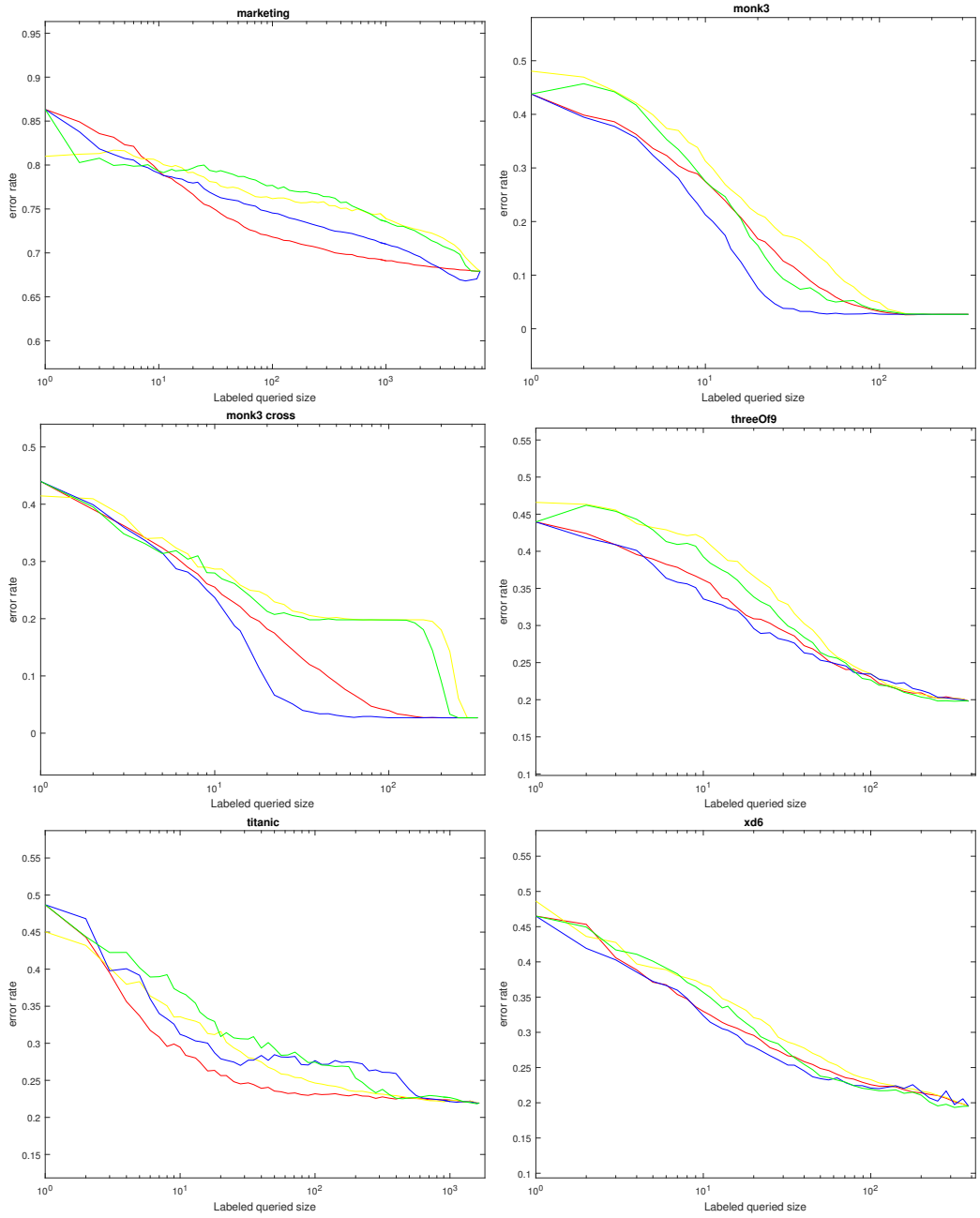


Fig. 6.17 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. In these cases, the ALSSNB is inferior to the other classifiers for 12 discrete benchmark datasets.

This section also shows the mean of the area under error rate learning curves for both passive and combined active and semi-supervised learning in Table 6.4 for 28 continuous benchmark datasets. The corresponding learning curves for these datasets are show in Figures

6.19, 6.20, and 6.21. Passive learning was best on 22 out of 28 benchmark datasets i.e. the ALSSNB best on only 6 benchmark datasets. The results of the Wilcoxon signed rank test at the 95% level of significance show that the ALNB is statistically superior to both ALSSNB and SSNB classifier. In addition, ALNB is slightly higher ranks than NB but the difference is not statistically significant.

#	Dataset	NB	ALSSNB
1	banknote	**1.541±0.0225**	2.920±0.0193
2	Blood_transfusion	**2.070±0.0262**	2.629±0.0557
3	breast_cancerw_continuous	0.500±0.0167	**0.346±0.0108**
4	Climate_Model_Simulation_Crashes	***0.603±0.0140***	*0.638±0.0221*
5	glass	**2.006±0.0192**	2.321±0.0199
6	haberman	**1.778±0.0301**	2.147±0.0710
7	ionosphere	**1.415±0.0278**	1.758±0.0427
8	iris	0.418±0.0187	**0.288±0.0156**
9	letter	**4.201±0.0111**	5.836±0.0084
10	liver_disorder	**2.719±0.0210**	2.912±0.0251
11	magic04	**3.506±0.0182**	5.022±0.0342
12	musk1	**2.207±0.0225**	2.544±0.0332
13	new_thyroid	0.352±0.0150	**0.244±0.0279**
14	pendigits	**2.104±0.0139**	3.094±0.0217
15	sleep	**4.956±0.0145**	7.684±0.0390
16	vehicle	**3.517±0.0166**	3.728±0.0225
17	vowel	**2.051±0.0138**	2.434±0.0108
18	waveform_noise	**2.581±0.0145**	3.006±0.0354
19	waveform	**2.917±0.0121**	3.511±0.0264
20	wine	0.640±0.0187	**0.284±0.0158**
21	arcene	2.160±0.0223	**2.075±0.0235**
22	gisette	**2.099±0.0117**	4.759±0.0089
23	madelon	4.116±0.0091	**3.966±0.0169**
24	sonar	**1.928±0.0239**	2.242±0.0223
25	spambase	**2.263±0.0195**	2.826±0.0289
26	Synthetic	**1.182±0.0217**	1.307±0.0436
27	vertebral	***1.421±0.0200***	*1.469±0.0312*
28	diabetes	**2.137±0.0207**	2.512±0.0365

Table 6.4 AULC of the NB, and ALSSNB classifiers with uncertainty sampling strategies over 28 continuous benchmark datasets. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

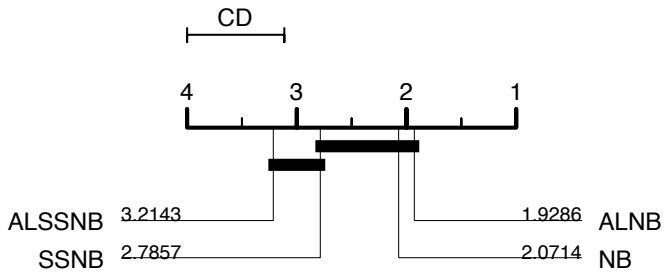


Fig. 6.18 Critical difference diagram for the NB, SSNB, ALNB, and ALSSNB over 28 continuous benchmark datasets. It shows that there are statistically significant differences between the mean ranks for the ALNB and both SSNB and ALSSNB classifiers and that ALNB classifier is superior. Additionally, there is no statistically significant difference between ALNB and NB classifier

The critical difference diagram can be used to compare the average rank of classifiers. Figure 6.18 surprisingly, shows that the average rank of ALSSNB is worse compared to the other classifiers. The average rank of ALNB classifier is statistically superior to the ALSSNB and the SSNB classifiers but ALNB just slightly higher ranks than NB classifier but the difference is not statistically significant. The results suggest that SSNB makes the active learning algorithm worse rather than better.

Figure 6.19 shows the average of 100 replications of the error rate learning curves for passive, active, semi-supervised and combined active and semi-supervised learning on the six continuous benchmark datasets `arcene`, `breast-cancerw-continuous`, `iris`, `new-thyroid`,`wine`, and `madelon`. As shown in this figure, by comparing the combined active and semi-supervised learning with the other three models, the classification error rate of the combined approach is lower. This improvement is most rapid for the first few training patterns labelled, especially in the `iris` and `wine` benchmarks. However, all four classifiers have nearly the same performance on the `Climate-Model-Simulation-Crashes` benchmark dataset, Figure 6.20.

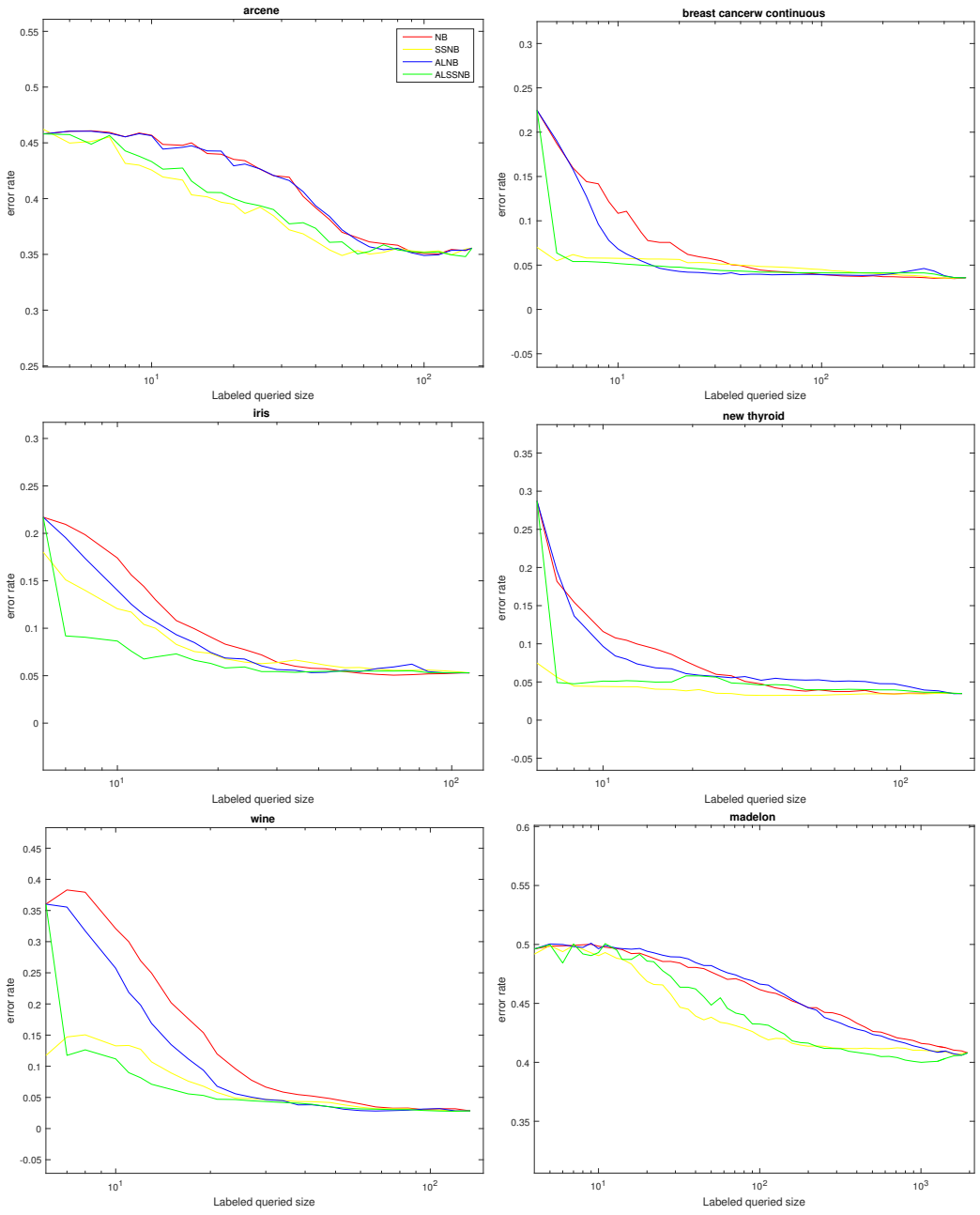


Fig. 6.19 Comparison of the error rate learning curve for NB, SSNB, ALNB, and ALSSNB classifiers. In these cases, the ALSSNB is superior to the NB and ALNB classifiers but have equivalent classification performance as SSNB for six continuous benchmark datasets

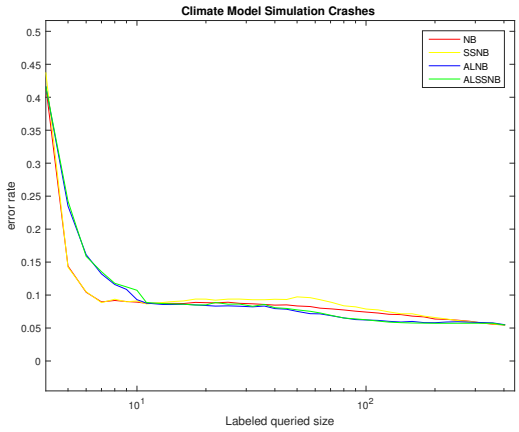
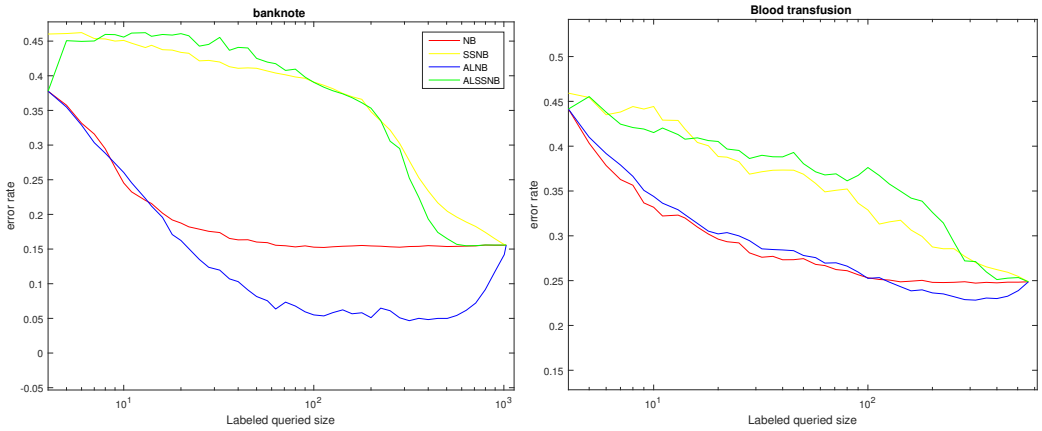
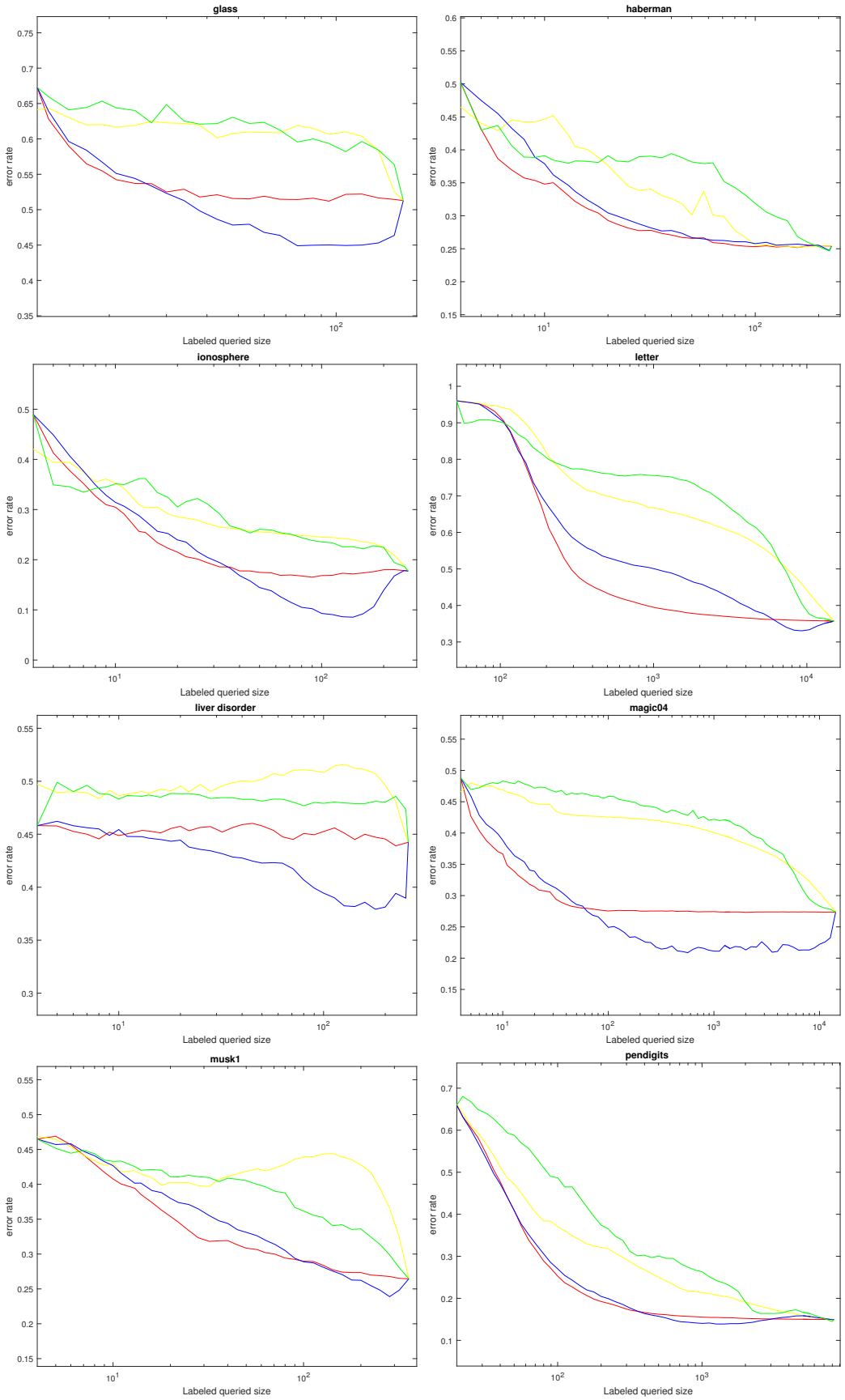
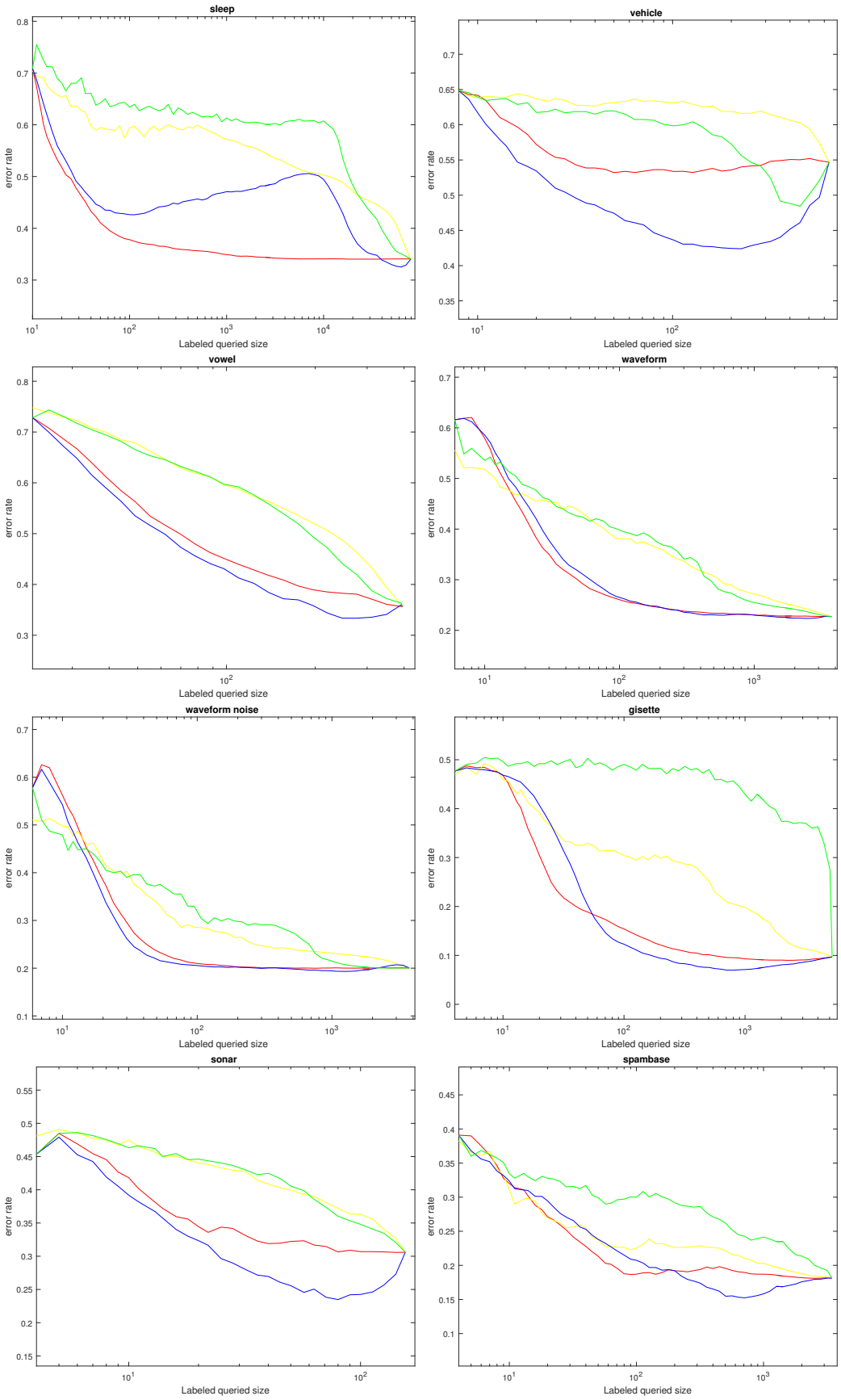


Fig. 6.20 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. All classifiers have equivalent classification performance for the `Climate-Model-Simulation-Crashes` benchmark dataset.

The results shown in Figure 6.21 indicate that using semi-supervised learning after selecting patterns by queries from active learning can degrade classification performance. As can be seen, the classification performance of combining active learning with semi-supervised learning is worse, compared to the NB and ALNB classifiers on 21 benchmark datasets.




glass
haberman
ionosphere
letter
liver disorder
magic04
musk1
pendigits
error rate
Labeled queried size


sleep
vehicle
vowel
waveform
waveform noise
gisette
sonar
spambase
error rate
Labeled queried size

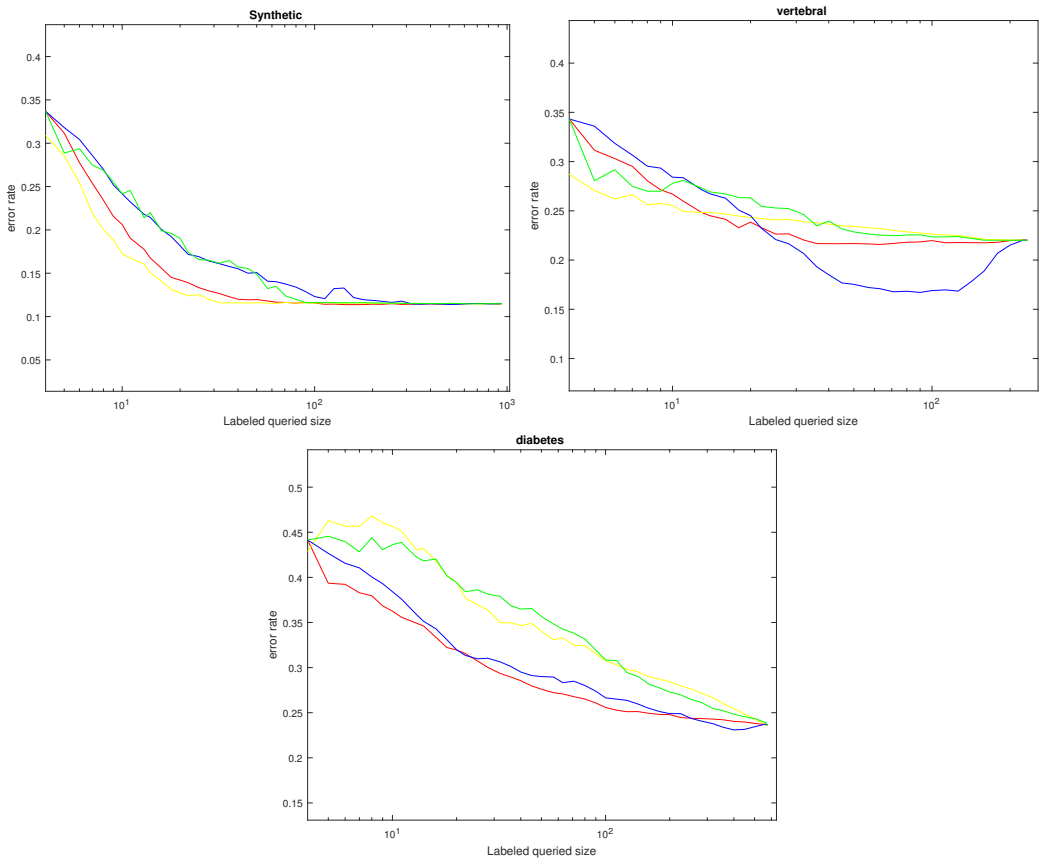


Fig. 6.21 Comparison of the error rate learning curves for NB, SSNB, ALNB, and ALSSNB classifiers. In these cases, the ALSSNB is inferior to the other classifiers for the 21 continuous benchmark datasets

6.5 When does combining active and semi-supervised learning work?

Previous results, show that combining active and semi-supervised learning almost always fails to improve the classifier. In the beginning of this chapter we illustrated that using unlabelled pattens can improve classification performance but this improvement generally does not give a statistically significant difference compared to using only labelled patterns. Therefore, the results of both Section 6.4 and Chapter 3 are good evidence that utilising unlabelled patterns with semi-supervised and active learning generally makes the classifier

worse. In addition, the most obvious explanation for this result is that the assumptions of the classifier are usually false and so it often produces inaccurate probability estimates. To test this hypothesis, we run two further experiments on the discrete and continuous synthetic benchmark datasets that were used in Section 3.1.3. Table 6.5 shows the results for 36 discrete benchmark datasets indicating that combining active with semi-supervised learning techniques can improve the naïve Bayes classifier for synthetic benchmark datasets, provided the assumption of independent is valid.

#	Dataset	NB	ALSSNB
1	audiology	**4.683±0.0361**	5.077±0.0478
2	balance_scale	3.359±0.0316	**3.158±0.0348**
3	blogger	2.350±0.0430	**2.276±0.0502**
4	breast_cancer	*2.553±0.0381*	***2.538±0.0369***
5	breastw	0.705±0.0221	**0.237±0.0334**
6	car	3.286±0.0324	**3.134±0.0332**
7	DNA	2.724±0.0242	**1.779±0.0661**
8	flare1	4.005±0.0292	**3.564±0.0427**
9	flare2	*4.037±0.0278*	***4.009±0.0351***
10	hayes_roth	3.188±0.0417	**3.006±0.0405**
11	house_votes	0.135±0.0099	**0.055±0.0054**
12	kr_vs_kp	3.265±0.0201	**2.302±0.0328**
13	led7	5.199±0.0229	**5.145±0.0262**
14	led24	6.163±0.0155	**5.393±0.0221**
15	lung_cancer	1.988±0.0558	**1.719±0.0553**
16	lymphography	1.832±0.0372	**1.161±0.0375**
17	marketing	8.816±0.0195	**8.298±0.0219**
18	monk1_corrupt	3.096±0.0258	**3.030±0.0336**
19	monk1_cross	2.137±0.0278	**1.229±0.0408**
20	monk1_local	3.205±0.0263	**3.054±0.0364**
21	monk1	3.265±0.0273	**3.148±0.0366**
22	monk3_cross	1.704±0.0267	**0.926±0.0271**
23	monk3_local	1.950±0.0281	**1.184±0.0303**
24	monk3	1.985±0.0288	**1.518±0.0407**
25	mushroom	1.003±0.0309	**0.254±0.0019**
26	nursery	4.022±0.0194	**3.321±0.0517**
27	primary_tumor	*5.561±0.0321*	***5.544±0.0318***
28	promoters	1.215±0.0311	**0.400±0.0232**
29	shuttle_landing_control	1.569±0.0315	**1.296±0.0386**

Continued on next page

#	Dataset	NB	ALSSNB
30	soybean_small	1.235±0.0458	**0.627±0.0352**
31	soybean_large	6.462±0.0220	**6.308±0.0224**
32	splice	3.167±0.0161	**1.499±0.0652**
33	threeOf9	2.953±0.0327	**2.669±0.0404**
34	titanic	2.978±0.0391	**2.831±0.0568**
35	xd6	2.713±0.0289	**2.249±0.0431**
36	zoo	1.897±0.0466	**1.465±0.0415**

Table 6.5 AULC of the NB, and ALSSNB classifiers with uncertainty sampling strategies over 36 discrete synthetic benchmark datasets. The boldface font indicates that the AULC for one of the classifiers is better than for the other classifier. The results that are statistically equivalent (according to the Wilcoxon signed rank test at the 0.95% confidence level) are shown in italics.

As we can see the combination of active and semi-supervised learning was best on 35 out of the 36 discrete synthetic benchmark datasets and, passive learning best on only in one benchmark dataset. The results of the Wilcoxon signed rank test shows that the ALSSNB classifier is statistically superior at the 95% level of significance. Also, we can compare the ALSSNB classifier with both active learning and semi-supervised learning. Figure 6.22 shows the critical difference diagram for the NB, SSNB, ALNB, and NB over the 36 discrete benchmark datasets. The diagram shows that the ALSSNB is statistically superior to both NB and ALNB classifiers, while ALSSNB higher rank than SSNB but it is not statistically significant difference.

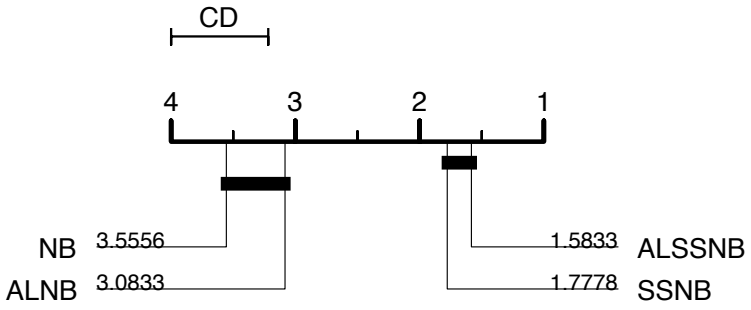


Fig. 6.22 Critical difference diagram for the NB, SSNB, ALNB, and ALSSNB over 36 discrete benchmark datasets. It shows that the ALSSNB is statistically superior compared to the other classifier

The second experiment in this section is run for the same motivation but on 28 continuous synthetic benchmark datasets. Table 6.6 shows that combining active learning with semi-supervised learning was best on 27 out of 28 benchmark datasets, passive learning best on only in one benchmark dataset. The result for the Wilcoxon signed rank test shows that the NB is statistically superior at the 95% level of significance.

#	Dataset	NB	ALSSNB
1	banknote	**2.033±0.0168**	2.373±0.0857
2	Blood_transfusion	1.022±0.0180	**0.837±0.0456**
3	breast_cancerw_continuous	0.220±0.0160	**0.071±0.0096**
4	Climate_Model_Simulation_Crashes	0.280±0.0110	**0.083±0.0093**
5	glass	0.735±0.0178	**0.172±0.0149**
6	haberman	1.417±0.0272	**1.388±0.0504**
7	ionosphere	0.418±0.0207	**0.119±0.0100**
8	iris	0.067±0.0118	**0.010±0.0018**
9	letter	3.086±0.0060	**2.272±0.0106**
10	liver_disorder	1.714±0.0205	**1.507±0.0379**
11	magic04	0.538±0.0219	**0.155±0.0062**
12	musk1	0.760±0.0285	**0.382±0.0299**
13	new_thyroid	0.130±0.0110	**0.043±0.0054**
14	pendigits	1.623±0.0093	**0.698±0.0254**
15	sleep	2.461±0.0107	**1.920±0.0300**
16	vehicle	2.118±0.0141	**1.796±0.0242**
18	waveform_noise	1.205±0.0165	**0.467±0.0183**
19	waveform	1.386±0.0157	**0.671±0.0192**
20	wine	0.199±0.0160	**0.060±0.0144**
21	arcene	0.138±0.0173	**0.044±0.0113**
22	gisette	0.427±0.0190	**0.357±0.0213**
23	madelon	3.306±0.0109	**2.844±0.0178**
24	sonar	0.086±0.0146	**0.005±0.0014**
25	spambase	0.004±0.0023	**0.001±0.0005**
26	Synthetic	0.047±0.0091	**0.014±0.0023**
27	vertebral	0.491±0.0239	**0.150±0.0154**
28	diabetes	1.639±0.0218	**1.337±0.0351**

Table 6.6 AULC of the NB, and ALSSNB classifiers with uncertainty sampling over 28 continuous synthetic benchmark datasets. The boldface font indicates that the AULC for one of the classifiers between the NB and ALNB is better than the other classifier. The result that statistically not different according to the Wilcoxon signed rank test at 0.95% confidence level are shown in italics

Figure 6.23 shows the average rank of NB, SSNB, ALNB, and ALSSNB classifiers. Again, the ALSSNB classifier is statistically superior compared to the NB and ALNB classifiers but slightly worse compare to the SSNB classifier.

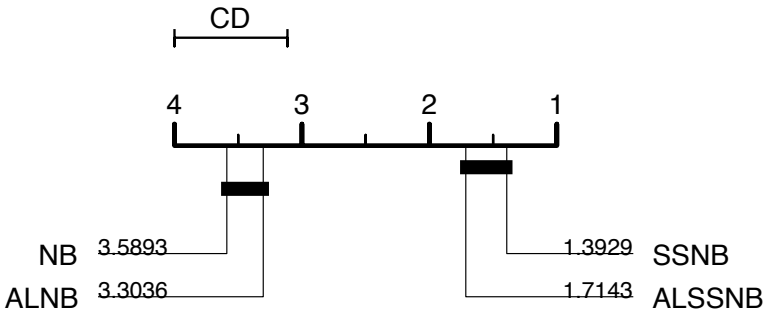


Fig. 6.23 Critical difference diagram for the NB, SSNB, ALNB, and ALSSNB over 28 continuous benchmark datasets. It shows that the ALSSNB classifier is statistically superior compared to the NB and ALNB classifiers. In addition, it shows that SSNB is rather better than ALSSNB but the difference between their mean ranks is not statistically significant

6.6 Summary

In this section, we proposed a new model by combining active learning and semi-supervised learning in order to reduce the human labelling effort and increase the classification performance, especially when very few labelled training patterns are available. We applied this method for different types of benchmark datasets and compared the performance of the four classifiers, NB, ALNB, SSNB, ALSSNB. Both experimental results showed that the SSNB and ALSSNB classifiers outperformed the baseline classifier in the synthetic datasets. Even ALNB was better than NB classifier but not statistically different. Moreover, both the SSNB and ALSSNB classifiers had equivalent performance. In this case, the incorporation of unlabelled patterns can effectively improve performance. The further experimental results on the synthetic datasets showed that the ALSSNB classifier outperforms NB, and ALNB classifier. The main reason for this improvement is the assumption of the model in synthetic

benchmark dataset is valid which means involving semi-supervised learning does improve classification performance and combining semi-supervised learning with active learning does give extra improvement.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis initially investigated the use of semi-supervised and active learning for naïve Bayes classifier individually, and in combination, using a suite of benchmark datasets. Using a substantial series of experiments we have attempted to answer the research question with a small amount of labelled patterns and normally a poor initial model was obtained especially in the early stage. Does the semi-supervised naïve Bayes classifier achieve better performance, when using both labelled and unlabelled data. The results obtained show that the unlabelled data does not substantially improve classification performance in general. Previous research in semi-supervised learning has proposed many different methods but there remain some questions with no clear answer, such as why semi-supervised learning often makes the classifier performance worse. This thesis answered the question for semi-supervised naïve Bayes, through generating a simulated benchmark dataset, where the independence assumption is valid. Experiments with these benchmark datasets demonstrated that the violation of the naïve Bayes model assumption (independence of features) means predictive errors propagate through the self-training methods reducing performance.

We conducted the experiments to determine whether down-weighting the influence of the

unlabelled data can improve the performance of the classifier, but the results obtained show that it does not because tuning the weighting factor, λ, is difficult, especially when the amount of the labelled data is small. A novel algorithm was proposed that used a sigmoid transformation function to re-calibrate the predicted class probabilities of the unlabelled data to overcome the overconfidence of the naïve Bayes classifier. The results obtained are higher rank than those of the naïve Bayes classifier, but there is no statistically significant difference between them. However, in some cases using the sigmoid transformation gives superior results to those of the conventional semi-supervised naïve Bayes classifier. Therefore, we believe that using this novel algorithm might be better than the standard semi-supervised naïve Bayes classifier in practical applications. We also present benchmark dataset results from active learning experiments for naïve Bayes classifier. The aim of using active learning was to create of high quality of labelled patterns in order to improve classification performance, but active learning cannot provide much help to improve the classifier. Later, after selecting the least confident pattern by uncertainty sampling especially in the early stages we attempted to use the remaining unlabelled patterns via semi-supervised learning to improve the classification performance. Thus, we design some experiments to combine active learning with semi-supervised learning, but this does not generally produce better results. Finally, synthetic benchmark datasets was generated to show when the combination of active and semi-supervised learning does improve the classifier. The results obtained show that the combined active learning with semi-supervised learning approach can improve performance if the semi-supervised learning already improve that classifier separately.

7.2 Future Work

- The experimental results obtained suggest that down-weighting the unlabelled data does not improve the performance of the classifier. However, the key challenge in the technique lies in how to choose the best value of λ. Using the train-test partition method is a biased protocol because we look at the test data to find the optimal value for the weighting factor. Then an approach using a validation set was applied, but instead of separating the validation set to evaluate the optimal value it would be better to use this set for training purposes, especially with the a small amount of labelled data. Choosing the value of λ, through k fold cross-validation may not provide a very reliable indicator because there are just a few labelled patterns initially available and leave-one-out-cross-validation may be unreliable because it has a high variance and could give a very different value if the experiment is repeated with a different sample dataset. We also used a new method moving from *k*-fold-cross-validation to leave-one-out-cross-validation, but again it does not improve the classifier. In conclusion, we found that none of these model selection methods for choosing the value of λ can improve the baseline classifier. Furthermore, work is required to develop a more reliable mean of choosing λ by using different classifiers.

- The semi-supervised naïve Bayes classifier is often applied for text classification and the reason that naïve Bayes classifier is useful is that: although the naïve Bayes model assumption is invalid we can estimate the parameter from less data than it is required for the full model which includes the dependence and has no more parameters. However, during the thesis experiments, there are three Molecular Biology datasets `DNA`, `splice`, `promoters` where the unlabelled data does improve the naïve Bayes classifier. This result suggest that semi-supervised naïve Bayes classifier could be useful in analysis of biological data.

References

[1] Agrawala, A. K. (1970). Learning with a probabilistic teacher. *Information Theory, IEEE Transactions on*, 16(4):373–379.

[2] Angluin, D. (1988). Queries and concept learning. *Machine learning*, 2(4):319–342.

[3] Angluin, D. (2001). Queries Revisited. In *Algorithmic Learning Theory, 12th International Conference, ALT 2001, Washington, DC, USA, November 25-28, 2001, Proceedings*, pages 12–31.

[4] Antonucci, A., Corani, G., and Gabaglio, S. (2012). Active learning by the naïve credal classifier. In *Sixth European Workshop on Probabilistic Graphical Models (PGM 2012)*, pages 3–10.

[5] Atlas, L. E., Cohn, D. A., and Ladner, R. E. (1990). Training connectionist networks with queries and selective sampling. In *Advances in Neural Information Processing Systems*, pages 566–573.

[6] Bache, K. and Lichman", M. ("2013"). UCI Machine Learning Repository.

[7] Baluja, S. (1999). Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *Advances in Neural Information Processing Systems*, pages 854–860.

[8] Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(Nov):2399–2434.

[9] Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 19–26.

[10] Blum, A., Lafferty, J., Rwebangira, M. R., and Reddy, R. (2004). Semi-supervised learning using randomized mincuts. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 13. Association for Computing Machinery (ACM).

[11] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100. Association for Computing Machinery (ACM).

[12] Callison-Burch, C., Talbot, D., and Osborne, M. (2004). Statistical Machine Translation with Word- and Sentence-Aligned Parallel Corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 175–182.

[13] Castelli, V. and Cover, T. M. (1996). The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *Information Theory, IEEE Transactions on*, 42(6):2102–2117.

[14] Chakraborty, S. (2011). Bayesian semi-supervised learning with support vector machine. *Statistical Methodology*, 8(1):68–82.

[15] Chapelle, O., Schölkopf, B., Zien, A., et al. (2006). *Semi-Supervised learning*, volume 2. MIT press Cambridge, ISBN: 978-0-262-03358-9.

[16] Chapelle, O. and Zien, A. (2005). Semi-Supervised Classification by Low Density Separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*, pages 57–64.

[17] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

[18] Cohen, I., Cozman, F. G., Sebe, N., Cirelo, M. C., and Huang, T. S. (2004). Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(12):1553–1566.

[19] Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*.

[20] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[21] Cozman, F. G. and Cohen, I. (2002). Unlabeled data can degrade classification performance of generative classifiers. In *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference, May 14-16, 2002, Pensacola Beach, Florida, USA*, pages 327–331.

[22] Dasgupta, S., Littman, M. L., and McAllester, D. A. (2002). Pac generalization bounds for co-training. In *Advances in Neural Information Processing Systems*, pages 375–382.

[23] Demiriz, A., Bennett, K. P., and Embrechts, M. J. (1999). Semi-supervised clustering using genetic algorithms. *Artificial Neural Networks in Engineering (ANNIE-99)*, pages 809–814.

[24] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

[25] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

[26] Efron, B. (1979). Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, pages 1–26.

[27] Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE.

[28] Fralick, S. C. (1967). Learning to recognize patterns without a teacher. *Information Theory, IEEE Transactions on*, 13(1):57–64.

[29] Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

[30] Goldman, S. A. and Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 327–334.

[31] Grandvalet, Y. and Bengio, Y. (2005). Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, pages 529–536.

[32] Grandvalet, Y., Bengio, Y., et al. (2004). Semi-supervised learning by entropy minimization. In *Neural Information Processing Systems (NIPS)*, volume 17, pages 529–536.

[33] Guo, Y., Niu, X., and Zhang, H. (2010). An extensive empirical study on semi-supervised learning. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 186–195. IEEE.

[34] Guyon, I., Cawley, G. C., Dror, G., and Lemaire, V. (2011). Results of the active learning challenge. In *Active Learning and Experimental Design workshop, In conjunction with AISTATS 2010, Sardinia, Italy, May 16, 2010*, pages 19–45.

[35] Hamerly, G., Elkan, C., et al. (2001). Bayesian approaches to failure prediction for disk drives. In *International Conference on Machine Learning (ICML)*, volume 1, pages 202–209.

[36] Hartigan, J. A. (1975). *Clustering Algorithms (Probability & Mathematical Statistics)*. John Wiley & Sons Inc, ISBN: 047135645X.

[37] Inoue, M. and Ueda, N. (2003). Exploitation of unlabeled sequences in hidden markov models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(12):1570–1581.

[38] Jiang, L., Cai, Z., Zhang, H., and Wang, D. (2012). Not so greedy: Randomly selected naïve Bayes. *Expert Systems with Applications*, 39(12):11022–11028.

[39] Jiao, F., Wang, S., Lee, C.-H., Greiner, R., and Schuurmans, D. (2006). Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 209–216. Association for Computational Linguistics.

[40] Joachims, T. (1999a). SVM-Light : Support vector machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund*, 19(4).

[41] Joachims, T. (1999b). Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning (ICML)*, volume 99, pages 200–209.

[42] Klein, D. (2004). Lagrange multipliers without permanent scarring. *University of California at Berkeley, Computer Science Division Tutorial*, pages 1–11.

[43] Lachenbruch, P. A. and Mickey, M. R. (1968). Estimation of error rates in discriminant analysis. *Technometrics*, 10(1):1–11.

[44] Lang, K. and Baum, E. (1992). Query learning can work poorly when a human oracle is used. *In Proceedings of the IEEE International Joint Conference on Neural Networks, IEEE Press*, pages 335—340.

[45] Langley, P., Iba, W., and Thompson, K. (1992). An Analysis of Bayesian Classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA, July 12-16, 1992.*, pages 223–228.

[46] Lee, C.-H., Gutierrez, F., and Dou, D. (2011). Calculating feature weights in naïve bayes with Kullback-Leibler measure. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1146–1151. IEEE.

[47] Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. Springer-Verlag New York, Inc.

[48] Li, M. and Zhou, Z.-H. (2005). Setred: self-training with editing. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 611–621. Springer.

[49] Li, M. and Zhou, Z.-H. (2007). Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(6):1088–1098.

[50] Mann, G. S. and McCallum, A. (2007). Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the 24th International Conference on Machine Learning*, pages 593–600. Association for Computing Machinery (ACM).

[51] McCallum, A., Nigam, K., et al. (1998). Employing em and pool-based active learning for text classification. In *International Conference on Machine Learning (ICML)*, volume 98, pages 350–358.

[52] McCallumzy, A. K. and Nigamy, K. (1998). Employing em and pool-based active learning for text classification. In *Machine Learning: Proceedings of the Fifteenth International Conference, ICML*. Citeseer.

[53] Melacci, S. and Belkin, M. (2011). Laplacian support vector machines trained in the primal. *Journal of Machine Learning Research*, 12(Mar):1149–1184.

[54] Muslea, I., Minton, S., and Knoblock, C. A. (2002). Active semi-supervised learning robust multi-view learning. In *International Conference on Machine Learning (ICML)*, volume 2, pages 435–442.

[55] Ng, A. (2016). Mixtures of gaussians and the em algorithm. *Lecture Notes on Machine Learning, Stanford University, Stanford, CA*.

[56] Ng, S. K., Krishnan, T., and McLachlan, G. J. (2012). The em algorithm. In *Handbook of Computational Statistics*, pages 139–172. Springer.

[57] Nigam, K. and Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 86–93. Association for Computing Machinery (ACM).

[58] Nigam, K., McCallum, A., and Mitchell, T. (2006). Semi-supervised text classification using EM. *Semi-Supervised Learning*, pages 33–56.

[59] Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39(2-3):103–134.

[60] Nigam, K. P. (2001). *Using unlabeled data to improve text classification*. PhD thesis, DEFENSE TECHNICAL INFORMATION CENTER.

[61] Panda, M. and Patra, M. R. (2007). Network intrusion detection using naïve bayes. *International journal of computer science and network security*, 7(12):258–263.

[62] Perlich, C. (2011). Learning curves in machine learning. In *Encyclopedia of Machine Learning*, pages 577–580. Springer.

[63] Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. volume 10, pages 61–74. Massachusetts Institute of Technology (MIT) Press.

[64] Ramirez-Loaiza, M. E., Sharma, M., Kumar, G., and Bilgic, M. (2016). Active learning: an empirical study of common baselines. *Data Mining and Knowledge Discovery*, pages 1–27.

[65] Ramirez-Loaiza, M. E., Sharma, M., Kumar, G., and Bilgic, M. (2017). Active learning: an empirical study of common baselines. *Data Mining and Knowledge Discovery*, 31(2):287–313.

[66] Ratsaby, J. and Venkatesh, S. S. (1995). Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 412–417. Association for Computing Machinery (ACM).

[67] Riloff, E., Wiebe, J., and Wilson, T. (2003). Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning at North American Chapter of the Association for Computational Linguistics: Human Language Technologies 2003-Volume 4*, pages 25–32. Association for Computational Linguistics.

[68] Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *7th IEEE Workshop on Applications of Computer Vision / IEEE Workshop on Motion and Video Computing (WACV/MOTION 2005), 5-7 January 2005, Breckenridge, CO, USA*, pages 29–36.

[69] Roy, N. and McCallum, A. (2001). Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448.

[70] Saeed, A. A., Cawley, G. C., and Bagnall, A. (2015). Benchmarking the semi-supervised naïve Bayes classifier. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE.

[71] Scudder, H. J. (1965). Probability of error of some adaptive pattern-recognition machines. *Information Theory, IEEE Transactions on*, 11(3):363–371.

[72] Settles, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

[73] Settles, B. and Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics.

[74] Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 287–294, New York, NY, USA. Association for Computing Machinery (ACM).

[75] Shahshahani, B. M., Landgrebe, D., et al. (1994). The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *Geoscience and Remote Sensing, IEEE Transactions on*, 32(5):1087–1095.

[76] Sindhwani, V., Belkin, M., and Niyogi, P. (2006). The geometric basis of semi-supervised learning. In *Semi-Supervised Learning*, pages 217–235. The MIT Press, Cambridge, MA.

[77] Sindhwani, V., Chu, W., and Keerthi, S. S. (2007). Semi-Supervised Gaussian Process Classifiers. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1059–1064.

[78] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 111–147.

[79] Tanha, J., van Someren, M., and Afsarmanesh, H. (2017). Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, 8(1):355–370.

[80] Tong, S. and Koller, D. (2002). Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66.

[81] Triguero, I., Sáez, J. A., Luengo, J., García, S., and Herrera, F. (2014). On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification. *Neurocomputing*, 132:30–41.

[82] Tsang, I. W. and Kwok, J. T. (2007). Large-scale sparsified manifold regularization. In *Advances in Neural Information Processing Systems*, pages 1401–1408.

[83] Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.

[84] Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

[85] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

[86] Wu, J. and Cai, Z. (2014). A naïve Bayes probability estimation model based on self-adaptive differential evolution. *Journal of Intelligent Information Systems*, 42(3):671–694.

[87] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics.

[88] Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328.

[89] Zhou, Y. and Goldman, S. (2004). Democratic co-learning. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 594–602. IEEE.

[90] Zhou, Z.-H. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

[91] Zhu, J., Wang, H., Yao, T., and Tsou, B. K. (2008). Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1137–1144. Association for Computational Linguistics.

[92] Zhu, X., Ghahramani, Z., Lafferty, J., et al. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*, volume 3, pages 912–919.

[93] Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130.

[94] Zhu, X., Lafferty, J., and Rosenfeld, R. (2005). *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, language technologies institute, school of computer science.