Design of Multiple Classifier Systems

Fuad M. Alkoot

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing
School of Electronics, Computing and Mathematics
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

© Fuad M. Alkoot 2001
Summary

During the past decade the fusion of multiple experts has become a legitimate necessity due to its capability of achieving improved accuracy over the best single expert. The field has grown diversely and in this thesis we focus on one branch only. Based on their architecture, we can divide multiple classifier systems in two categories: Parallel Homogeneous Classifier systems, (PHC), which involve parallel component classifiers with similar output types for fusion; Complex Multistage Classifiers which are characterised by a complex internal architecture where the decisions may not reach the fusion rule in parallel or if they are in parallel may not be of homogeneous type. The fusion methods used in PHC systems can be grouped into two, Simple methods not requiring any training and Complex methods. The focus of this thesis is on the simple fusion methods. In addition some aspects of PHC system design are also addressed.

In a PHC environment one could search for deficiencies in order to improve the overall classification rate. There is a need to better understand the existing methods to help in selecting the optimum technique for the problem at hand. This understanding may also lead to advances in classifier fusion methods and hence to improvements in their performance. The comparative survey of the simple methods concludes that the strategies perform variably and indicates the need for their detailed analysis and investigation. The theoretical derivation of these strategies is presented followed by an analytical analysis of the sensitivity of Sum and Product to estimation errors. Then the results of the analytical study are validated experimentally for different noise conditions. The experimental study has uncovered the veto effect to be the reason behind the poor performance of Product under high noise conditions. The experimental evaluation is extended further when we investigate Sum and Vote experimentally and theoretically to find when and why one strategy outperforms another.

The experimental findings relating to the veto effect has lead to proposing the heuristic Modified Product and the theoretically based moderated Product. In general the fusion strategy and fusion component experts are related and upgrading the fusion strategy may solve some of the deficiencies of the component experts. However, upgrading the component experts of the PHC system may lead to a further improvement in the overall system performance. This is obvious in the modified bagging methods which we propose. Random feature subset based bagging is another solution to improve the $k - NN$ PHC system.

One can also search for novel methods of designing PHC systems, such as the combiner system based feature selection method to build the ensemble component experts. The proposed method is viewed as a complement to the conventional method of designing PHC systems, which is based on optimising the component experts independently, before fusing them in the system.

Key words: PHC systems, Classifier Combination, Classifier Fusion, Bagging, Moderation, Product, Sum, Vote, Veto, Small Sample classification
Email: f.alkoot@eim.surrey.ac.uk
WWW: http://www.eim.surrey.ac.uk/
Acknowledgements

This thesis is dedicated to “the gateway of the city of knowledge”, Imam Ali ibn Abitaleb (AS), my inspiration to quest for knowledge.

I am in debt to my Supervisor Professor Josef Kittler and would like to thank him not only for the guidance but also for the wonderful personality. I appreciate his patience when I was absorbing knowledge and his tolerance when I was panicking. I benefited tremendously, and from him I learned the art of exploring ideas and evaluating them. Again, thank you Josef for the wonderful three years which led to my life long dream becoming a reality. I would like to thank the examiners Professor Robert Duin and Dr. Terry Windeatt for accepting the task of examining me and for their encouraging remarks, suggestions and comments during the exam.

Many thanks to all researchers and staff of CVSSP. Seminars, discussions and help were very much appreciated. Particularly, I would like to name Dr. William Christmas for answering many of my uncommon questions and Dr. M. Ahmadzadeh who gave the stressful days a taste of fun and helped when Unix was not so friendly, even when he was suffering from the stress associated with PhD research.

A big thanks to PAAET and IDB for the trust they placed in me and hope finishing in three years is a nice present to give back. From PAAET I would like to sincerely thank Dr. H. Almodaf head of PAAET and K. Alabdullah Head of TNI. for arranging the scholarship granted by IDB.

I would like to thank my parents, the source and the centre of my success, but can not do justice in a single page. I can not forget the numerous encouragements since childhood among many factors that lead to my success. Fathers guidance and mothers passion were my source of energy in life. Their faces and advice were always with me. The list of persons I would like to dedicate this work to is very long, however, I would like to mention my brothers, sisters, two uncles and aunts that were also sources of inspiration and encouragement since childhood. The beginning was my brother Fadel’s battery and bulb followed by my fathers gift; the story of Edison, when I was 10 years old.

Also, I would like to thank, with love, my dear, dedicated and patient wife, Poyeh Farhadzadeh, for the unwavering support during my PhD research. Can I do justice in a sentence? No, but I think without her this work would not have succeeded in three years. Hopefully, now that I have reached my childhood dream she can start a normal life with me, or is it just the start?! My consolidation is that although the research is in, however the stress will be out. Equally I would like to dedicate this work to my two lovely daughters Yasmeen and Zahra as well as whomever the future may bring in to our nest. I hope I can be a real father now, and Yasmeen and Zahra can enjoy what they were used to before the commencement of my PhD research. That is being with and being spoiled by thier father, every day.
## Contents

1 Introduction .................................. 1
   1.1 Background ................................ 1
   1.2 Thesis Outline and Contribution ............. 4

2 Fusion Strategies ............................ 9
   2.1 Introduction ................................ 9
   2.2 Fusion Methods for the Parallel Homogeneous Classifier Systems, PHC . 11
      2.2.1 Simple Combiners ......................... 11
      2.2.2 Complex Combiners ....................... 15
   2.3 Complex Multistage Combiner Systems, CMC ........ 21
      2.3.1 Application Specific Systems ............ 22
      2.3.2 Hierarchical Mixtures of Experts .......... 24
      2.3.3 Neural Networks ......................... 25
      2.3.4 Stacked Generalisation ................... 25
   2.4 Conclusion ................................ 26

3 PHC Fusion Systems .......................... 29
   3.1 Introduction ................................ 29
   3.2 Training Set Manipulation Based Ensembles .... 31
      3.2.1 Cross-Validation ......................... 31
      3.2.2 Randomness .............................. 31
      3.2.3 Boosting . .............................. 31
      3.2.4 Bagging ................................ 32
      3.2.5 Modified Bagging, Population Bias Control .. 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 Feature Set Manipulation Based Ensembles</td>
<td>34</td>
</tr>
<tr>
<td>3.3.1 Conventional Methods</td>
<td>34</td>
</tr>
<tr>
<td>3.3.2 Combiner System Based Method</td>
<td>35</td>
</tr>
<tr>
<td>3.3.3 Random Subsets Method, RSM</td>
<td>36</td>
</tr>
<tr>
<td>3.4 Conclusion</td>
<td>38</td>
</tr>
<tr>
<td>4 Investigating The Simple Fusion Strategies</td>
<td>39</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>39</td>
</tr>
<tr>
<td>4.2 Theoretical Derivation of Simple Fusion Strategies</td>
<td>41</td>
</tr>
<tr>
<td>4.3 Theoretical Expectations of Relative Performance in the Presence of Noise</td>
<td>44</td>
</tr>
<tr>
<td>4.4 Comparative Survey on the Performance of Simple Fusion Strategies</td>
<td>47</td>
</tr>
<tr>
<td>4.5 Experimental Study of Fusion Strategies Using Synthetic Experts</td>
<td>49</td>
</tr>
<tr>
<td>4.5.1 Experimental Methodology</td>
<td>50</td>
</tr>
<tr>
<td>4.5.2 Comparison of Strategies Under Uniform Noise</td>
<td>50</td>
</tr>
<tr>
<td>4.5.3 Comparison of Strategies Under Gaussian Noise</td>
<td>53</td>
</tr>
<tr>
<td>4.5.4 Contradicting Experts</td>
<td>56</td>
</tr>
<tr>
<td>4.5.5 Improving The Performance</td>
<td>57</td>
</tr>
<tr>
<td>4.6 Conclusion</td>
<td>58</td>
</tr>
<tr>
<td>5 Alleviating the Veto Effect</td>
<td>61</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>61</td>
</tr>
<tr>
<td>5.2 Setup of Experimental Study Using k-NN Experts</td>
<td>63</td>
</tr>
<tr>
<td>5.2.1 Disclaimer</td>
<td>63</td>
</tr>
<tr>
<td>5.2.2 Experimental Methodology</td>
<td>63</td>
</tr>
<tr>
<td>5.2.3 Calculation of Errors</td>
<td>64</td>
</tr>
<tr>
<td>5.2.4 Comparative Measure</td>
<td>64</td>
</tr>
<tr>
<td>5.2.5 Data Sets</td>
<td>66</td>
</tr>
<tr>
<td>5.3 Modified Product</td>
<td>67</td>
</tr>
<tr>
<td>5.3.1 The Modified Product Procedure</td>
<td>67</td>
</tr>
<tr>
<td>5.3.2 Experiments with Synthetic Experts</td>
<td>67</td>
</tr>
<tr>
<td>5.3.3 Experiments with k-NN Classifiers</td>
<td>70</td>
</tr>
<tr>
<td>5.4 Moderation</td>
<td>75</td>
</tr>
</tbody>
</table>
5.4.1 General Moderation Theory .................................................. 75
5.4.2 k-NN Classifier Output Moderation ............................................. 76
5.4.3 Moderation Experimental Results ............................................... 77
5.5 Conclusion .............................................................................. 83

6 Sum Versus Vote .................................................................. 85
6.1 Introduction ........................................................................ 85
6.2 Theoretical Analysis ................................................................. 87
6.3 Relationship of Sum and Vote ..................................................... 94
6.4 Real Data Experiments ............................................................... 103
6.5 Discussion ............................................................................ 110
6.6 Conclusion ............................................................................ 111

7 A PHC System for k – NN Classifiers ....................................... 113
7.1 Introduction ........................................................................ 113
7.2 Bagging as a Training Set Manipulation Technique to Build PHC Systems 115
7.3 Population Bias Control, Applied to Bagging ................................. 117
7.4 Feature Set Manipulation Technique to Build PHC Systems ................ 125
7.5 Multiple Expert System Design by Combined Feature Selection and Probability Level Fusion 127
7.6 Combining PHC Design Techniques ............................................. 129
  7.6.1 System Design Methods ....................................................... 131
  7.6.2 Experimental Methodology and Data Sets ................................. 132
  7.6.3 Results ........................................................................ 135
  7.6.4 Discussion ...................................................................... 140
7.7 Conclusion ............................................................................ 145

8 Conclusion and Future Research ............................................. 147
  8.0.1 Conclusion ....................................................................... 147
  8.0.2 Future Research ................................................................. 151
A System Based Feature Selection 155

A.1 Experimental Methodology ........................................ 155
   A.1.1 Multiple Expert Systems .................................. 155
   A.1.2 Fusion Strategies ............................................ 156
A.2 Data Sets .................................................................. 158
A.3 Results .................................................................... 159
A.4 Comparing Architectures .......................................... 165
A.5 Comparing Fusion Strategies ..................................... 167
A.6 Conclusion ................................................................ 168
List of Figures

1.1 Separation of fusion systems to two categories, and the division of the fusion strategies to two types ............................................. 3

1.2 Three major strategies for designing parallel homogeneous classifier systems ............................................. 4

4.1 Comparison between the combination rules for a fixed aposteriori probability value (P=.6), and different uniform noise distributions .......... 52

4.2 Comparison between combination rules for different aposteriori probability values and fixed uniform noise interval at b = 0.8 ............... 52

4.3 Comparison between combination rules for fixed aposteriori probability value of 0.6, and varying standard deviation of Gaussian noise .......... 55

4.4 A comparison of the combination rules parameterised by different aposteriori probability values for Gaussian noise with standard deviation σ = 0.3 55

5.1 Sample distribution for the Synthetic data set .......................... 66

5.2 The performance of an 8 expert MProduct combiner as a function of truncation threshold at P(ω|x) = .6 ........................................ 68

5.3 Comparison between the performance of the fusion strategies for t = 5 × 10⁻⁸ ................................................................. 70

5.4 Moderated Product improvement over the single expert .................. 78

5.5 Moderated Product improvement over the unmoderated Product ....... 79

5.6 Comparison of posterior probability estimates when using Moderation to when using Modified Product ........................................... 81

5.7 Moderated Product improvement over the Modified Product .......... 81

5.8 Comparison of classification rates of Product when expert outputs are raw, Moderated or Modified, all compared to the single expert. Results are for bagging method four using regular Seismic learning set ............ 82

5.9 Comparison of classification rates of Product when expert outputs are raw, Moderated or Modified, all compared to the single expert. Results are for bagging method four using regular Wine data learning set .......... 83
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Sum and Vote switching error for normally distributed estimation errors with different values of $\sigma(x)$ using 3 experts.</td>
<td>91</td>
</tr>
<tr>
<td>6.2</td>
<td>Sum and Vote switching error for normally distributed estimation errors with different values of $\sigma(x)$ using 7 experts.</td>
<td>92</td>
</tr>
<tr>
<td>6.3</td>
<td>Sum and Vote switching error for normally distributed estimation errors with different values of $\sigma(x)$ using 11 experts.</td>
<td>92</td>
</tr>
<tr>
<td>6.4</td>
<td>Sum and Vote switching error for normally distributed estimation errors with different values of $\sigma(x)$ using 15 experts.</td>
<td>93</td>
</tr>
<tr>
<td>6.5</td>
<td>Sum and Vote switching error for normally distributed estimation errors with $\sigma(x) = .05, .15, .25$ and .35, using 3, 7, 11 and 15 experts.</td>
<td>93</td>
</tr>
<tr>
<td>6.6</td>
<td>Comparison of experimental Sum and Vote switching errors with theoretical predictions, for $\sigma(x) = 0.15$.</td>
<td>95</td>
</tr>
<tr>
<td>6.7</td>
<td>Comparison of experimental Sum and Vote switching errors with theoretical predictions, for $\sigma(x) = 0.25$.</td>
<td>96</td>
</tr>
<tr>
<td>6.8</td>
<td>Comparison of experimental Sum and Vote switching errors with theoretical predictions, for $\sigma(x) = 0.35$.</td>
<td>97</td>
</tr>
<tr>
<td>6.9</td>
<td>Dirac delta error distribution.</td>
<td>98</td>
</tr>
<tr>
<td>6.10</td>
<td>Theoretical switching error of Sum and Vote in the presence of delta noise at (a) 0.2 and (b) 0.4</td>
<td>99</td>
</tr>
<tr>
<td>6.11</td>
<td>Sum and Vote switching error: A comparison of single class experimental results and theoretical predictions for delta noise located at (1-p)</td>
<td>101</td>
</tr>
<tr>
<td>6.12</td>
<td>Sum and Vote switching error: A comparison of two class experimental results and theoretical predictions for delta noise positioned at (1-p)</td>
<td>102</td>
</tr>
<tr>
<td>7.1</td>
<td>Classification rate of different design methods using real data sets.</td>
<td>133</td>
</tr>
<tr>
<td>7.2</td>
<td>Classification rate of different design methods using normally distributed synthetic data with equal class spread.</td>
<td>135</td>
</tr>
<tr>
<td>7.3</td>
<td>Classification rate of different design methods using normally distributed synthetic data with equal class spread.</td>
<td>136</td>
</tr>
<tr>
<td>7.4</td>
<td>Classification rate of different design methods using normally distributed synthetic data with larger spread of the smaller class.</td>
<td>138</td>
</tr>
<tr>
<td>7.5</td>
<td>Classification rate of different design methods using normally distributed synthetic data with larger spread of the larger class.</td>
<td>139</td>
</tr>
<tr>
<td>7.6</td>
<td>Classification rate of different design methods using uniformly distributed synthetic data with equal class spread.</td>
<td>141</td>
</tr>
<tr>
<td>7.7</td>
<td>Classification rate of different design methods using uniformly distributed synthetic data with equal class spread.</td>
<td>142</td>
</tr>
</tbody>
</table>
7.8 Classification rate of different design methods using uniformly distributed synthetic data with larger spread of the smaller class.  143
7.9 Classification rate of different design methods using uniformly distributed synthetic data with larger spread of the larger class.  144
List of Tables

4.1 Comparison between combining strategies performances using 3 experts, under uniform noise ........................................ 51
4.2 Performance comparison in the three class case under the gaussian noise generator ........................................ 54

5.1 Data sets used and the number of samples available in each data set. . 65
5.2 Simulated Experts: Number of test cases in which fusion strategies give the best or the jointly best performance. The results correspond to MProduct threshold value of $5 \times 10^{-8}$ ........................................ 69
5.3 MProduct classification rate obtained using regular training set and bagging method 4 ........................................ 71
5.4 Degree of improvement of MProduct over Sum using $t = 5 \times 10^{-8}$ . 71
5.5 Degree of improvement of MProduct over Product using $t = 5 \times 10^{-8}$ . 72
5.6 Degree of improvement of MProduct over the single expert using threshold $= 5 \times 10^{-8}$ ........................................ 72
5.7 Number of times a method is significantly better summed over eight bagging methods, four set sizes and four values of nearest neighbours, $k$. 74
5.8 Classification rate of the single expert and bagging using Product of non-moderated aposterioiri class probability estimates .......... 80

6.1 Classification rates of the single experts ........................................ 105
6.2 The classification rate of the best mixture of experts obtained on the XM2VTS evaluation set using Sum and Vote fusion strategies .......... 108
6.3 The test set classification rate of the best mixture of experts selected using the evaluation set. ........................................ 109

7.1 Relative performance measures parameterised by different bagging methods and learning set sizes for the Synthetic data. ............... 118
7.2 Relative performance measures parameterised by different bagging methods and learning set sizes for the Seismic data. ............... 118
7.3 Relative performance measure parameterised by different bagging methods and learning set sizes for the Diabetes data. .......................... 119

7.4 Relative performance measure parameterised by different bagging methods and learning set sizes for the Wine data. .......................... 119

7.5 Relative performance measure parameterised by different bagging methods and learning set sizes for the Ionosphere data. ....................... 120

7.6 Relative performance measure parameterised by different bagging methods and learning set sizes for the BCW data. .......................... 120

7.7 Relative performance measure parameterised by different bagging methods and learning set sizes for the Iris data. .......................... 121

7.8 The number of times a bagging method achieves a rank among the different methods, summed over all data sets. Rank 1 is the highest. .... 122

7.9 The number of times a method achieves a 5% significant improvement or degradation over the single expert out of 56 cases. (all data sets and learning set types) .......................... 123

7.10 The number of times a method achieves a 5% significant improvement or degradation over the regular bagging method 1 out of 56 cases. (all data sets and learning set types) .......................... 123

7.11 Results obtained by Skalak [87], Brieman [18], Quinlan [72] and Opitz [66]. 124

7.12 Comparison of results obtained through MProduct, under bagging method 3 for varying set sizes, to learning set (single expert) results. Learning set is constructed using the regular (random sampling) method. .......................... 125

A.1 Performance comparison of fusion strategies for the serial and parallel system designs on BCW data. .......................... 159

A.2 Performance comparison of fusion strategies for the serial and parallel system designs on BCW data with a large size training set. .......................... 160

A.3 Performance comparison of fusion strategies for the serial system design on Seismic data. .......................... 161

A.4 Performance comparison of fusion strategies for the Optimised Conventional system design using BCW data. The effect of adding features to the termination point of the conventional system, C.S. .......................... 162

A.5 Performance comparison of fusion strategies for the Optimised Conventional system design using BCW data. The effect of removing features from the conventional system, C.S. .......................... 162

A.6 Performance comparison of fusion strategies for the Optimised Conventional system design using BCW data with large training set. The effect of adding features to the termination point of C.S. .......................... 164
A.7 Performance comparison of fusion strategies for the Optimised Conventional system design using BCW data with large training set. The effect of removing features from C.S. ................. 165

A.8 Performance comparison of fusion strategies for the Optimised Conventional system design using Seismic data. The effect of adding features to the termination point of C.S. ......................... 166

A.9 Performance comparison of fusion strategies for the Optimised Conventional system design using Seismic data. The effect of removing features from C.S. .............................. 167
Chapter 1

Introduction

1.1 Background

In the past decade, the use of classifier fusion to improve classification accuracy has become increasingly popular [10, 20, 24, 32, 37, 43, 45, 46, 52, 56, 60, 80, 85, 99, 104, 107, 108, 109]. Classifier combination has been attracting considerable attention because of its potential to ameliorate the performance of pattern recognition systems. The basic idea is to solve each pattern recognition problem by designing a number of classification systems and then combining the designs in some way to achieve reduced recognition error rates. The fusion process may operate on the soft outputs of the individual experts, or it may involve combining the hard decisions of the experts. The literature on classifier combination grows rapidly and by now includes hundreds of articles. There are many strategies that can be used to combine classifiers in order to improve recognition error rates.

The papers advocate different fusion strategies, or demonstrate the benefits of classifier combination on diverse applications. Several novel strategies have been proposed in [74, 75, 82, 10, 18, 25, 56] and existing strategies are investigated in [92, 96, 7, 29, 55], in order to gain better understanding of the properties of the fusion processes involved.

The general findings are the following:

- The aim of combining is to achieve a better estimate of the class a sample be-
Chapter 1. Introduction

longs to. This is done by reducing estimation error variance on a posteriori class probabilities.

- Ensembles are often much more accurate than the component classifiers, especially when component classifiers disagree with each other, on the condition that their error is less than 50% for a two class case.

*Why does combining work?* Dietterich [23] gives three reasons for ensembles outperforming component classifiers.

- Training data may not provide sufficient information for choosing a single best classifier. All resulting hypotheses, or classifiers, are equally accurate with respect to the available training data.

- The learning algorithm may not be able to solve the difficult search problems. We may not be able to find the best algorithm, hence ensembles may be seen as a way of compensating for imperfect search algorithms.

- The hypothesis space may not contain the true function $f$, rather it may include several equally good approximations to $f$.

Besides the lower error rates achieved through combining, we would need to combine if feature types are different forcing the use of different classifiers whose decisions must be combined. For example features could have different scales that must be normalised which could be a job more difficult than using an expert for each group type of features and then combine the results. Another case would be if the feature space is so large that it must be divided into subspaces where individual experts are trained in each feature subspace. Combining based on the feature subspace method for even a small number of features has been shown to be beneficial [42].

*Are there other benefits gained from combining?* Besides the benefits of improved error rates, combining has led to the solution of other problems. For example classifier combination has made it possible to use simpler experts. In an application involving speaker recognition in tactical communications, Ricart et. al. [79] are able to use simpler neural network and k-nearest neighbour classifiers, instead of the commonly
used but more complex HMM. The results of the multiple classifiers are combined to achieve improved results over HMM.

Military applications, for example war scenarios or multi-target tracking or recognition, commonly involve information from multiple sources that are considered un-ignoreable. The multiple feeds of information need to be combined, optimally.

Suen et al [94] point to the problems and difficulties involved with cursive handwriting recognition especially when attempting to match the human performance. Researchers have extracted many feature types none of which can independently achieve the human performance. The integration of the different feature types would lead to an overly complex expert. However, combining many, not necessarily complex, experts based on different features can achieve a high recognition rate.

Van Breukelen and Duin [103] use classifier combination methods to initialise neural network weights. Compared to random initialisation their method of weight initialisation leads to faster convergence with a lower error rate. Kong and Dietterich [53] also use ECOC combining techniques to estimate class conditional probabilities.

Figure 1.1: Separation of fusion systems to two categories, and the division of the fusion strategies to two types. The full name of the CMC systems is presented in Chapter 2.
Figure 1.2: Three major strategies for designing parallel homogeneous classifier systems

Tumer and Ghosh [101, 98] propose using classifier combining techniques to estimate the Bayes error, specifically when the classifiers are combined through averaging.

1.2 Thesis Outline and Contribution

Generally in the classifier fusion field research has been conducted on several levels. These are the decision fusion level, system design level, and sensor or data fusion level. In this thesis we focus on the top two levels, i.e. the decision and the system design levels. For the system design level we divide the fusion systems in two categories, the Parallel Homogeneous Classifier systems, PHC, and the Complex Multistage Combiner systems, CMC. We focus on the PHC systems and survey the fusion strategies used in such systems. These fusion strategies are categorised in two groups, Simple and Complex fusion strategies.

We direct our attention to the simple strategies because, contrary to the complex strategies, they do not require training and can also outperform the complex ones. However, the varying performance of these strategies has highlighted the need to investigate them...
and find when and why they outperform or underperform each other. The theoretical
derivation of the simple strategies by Kittler [50, 48] has legitimised their use. The
derivation shows that Product is directly derived from the Bayesian theory under the
assumption of the independence of component experts. Sum is derived from Product
under more restricting assumptions. However, one could argue that Sum or averaging
does not need to be derived from Product and is an obvious choice when many esti-
mates exist. The rest, i.e. Minimum, Maximum, Median and Vote, are derived from
these two rules. Therefore, one would find it more legitimate to use Product and expect
it to outperform Sum. However, Kittler's error analysis shows that Product is more
sensitive to estimation errors and therefore to underperform below Sum under high
noise conditions. Our experimental study of the rules in Chapter 4 validates the theo-
retical error analysis in addition to shedding light on the reason behind the dramatic
degradation of Product under high noise conditions. Through the empirical study we
found that under high noise conditions, experts estimates reach the extremes of the
probability range leading to zero posterior probability estimates for the correct class.
It is enough to have a single vetoing expert with zero output among any ensemble of
experts to cause a classification error.

Based on this finding we propose in Chapter 5 the heuristic Modified Product (MProd-
uct) [13], and the theoretically derived Moderated Product [11], to remove the veto
effect and significantly improve Product. The improvement puts Product as a strong
contender among the simple fusion strategies which makes it a first choice due to its
direct link to the Bayesian theory.

To further enhance our understanding of the simple fusion strategies in Chapter 6 we
theoretically investigate Sum and Vote to find when and why one outperforms another.
The theoretical analysis is validated experimentally using synthetic and real experts.
The findings are that Sum mostly outperforms Vote. However, for small margins Vote
outperforms Sum if the component experts suffer from a heavy tailed error distribution.

On the system design level an investigation of one method for designing PHC fusion
systems highlights the fact that sometimes fusion may underperform the single expert.
Our experimental investigation shows when and why bagging of k - NN experts un-
derperforms the single expert, when and how it can be modified to outperform the single expert. Although $k - NN$ are unstable under small sample size we found that regular bootstrapping creates biased sets. Accordingly in Chapter 7 we modify the bootstrap method and find that controlling the population in each set would lead to better bagging results. Under large training set size bootstrapping creates stable $k - NN$ component experts. However, when random feature subsets are used for each bootstrap set, we achieve diverse $k - NN$ experts that lead to bagging results which significantly outperform the single expert.

In Chapter 7 we also adopt a novel design methodology by incorporating the feature selection method in the combiner design process. Contrary to the conventional method, where each component expert is designed independently then incorporated in the combiner, the novel method selects the best features for each expert based on the performance of the combined system not the component experts. Although the results are not conclusive some improvement over the single expert was achieved for the gaussian classifier. Bagging with feature subset selection using the system based method does not always outperform the random feature subset selection based on the conventional expert based method. Again the results are not conclusive and further research is needed to find the strengths and weaknesses of the system based design methodology.

Many disciplines have contributed to the advancement of the classifier fusion field. Therefore, different terminologies have been used to refer to relatively the same entity. In this thesis many of these words are used interchangeably. Classifier, expert and ensemble component all refer to the same part of the combiner system, which outputs an opinion based on the input test. Also, the word fusion and combining are used to refer to the process of gathering the opinions output by the different parts or experts of the fusion system. Similarly, fusion system and combiner system refer to the same system, consisting of all the component experts and the fusion processor.

The contribution of this thesis can be summarised as follows:

- The theoretically predicted behaviour of simple fusion strategies has been validated through an experimental evaluation. This also lead to finding when and
1.2. Thesis Outline and Contribution

why Product underperforms. It has also lead to finding the relation between the performance of each of the rules and the estimation error on the expert outputs. [7]

- A novel fusion rule referred to as the Modified Product rule has been proposed. It significantly improves Product. The heuristic method replaces the zero experts outputs with a threshold. [9, 13]

- A theoretical basis for modifying expert outputs has been established by introducing the concept of classifier output moderation. A theoretically founded Moderated Product is derived for the case of $k-NN$ experts. [11, 12]

- The relationship between the popular Sum and Vote rules, which clarifies when and why one rule outperforms another, has been established. The theoretical analysis is validated experimentally. [49]

- A technique called population bias control has been proposed to improve bagging under very small sample size conditions. [14]

- A novel multiple classifier system design philosophy has been introduced. It incorporates the feature selection process in the design procedure of the fusion system, where a feature is selected based on the performance of the fusion system and not the component expert it is assigned to. [8, 10]

- Contrary to the common belief that bagging can not improve a stable expert such as the $k-NN$, we show that $k-NN$ experts can be improved through bagging if another diversification factor, such as the assignment of random feature subsets, is incorporated in the bootstrap sets.
Chapter 2

Fusion Strategies

The fusion systems are categorised in two groups: Parallel Homogeneous Classifier systems, PHC, and Complex Multistage Combiner systems, CMC. We classify the fusion strategies used in the PHC systems as Simple and Complex strategies.

2.1 Introduction

Classifier fusion has become a widely used tool for improving the accuracy of classifiers, and it has grown rapidly. The types of fusion systems and strategies are so numerous that any effort to collectively introduce them in one report would yield an incongruous or a piebald report.

In this thesis we investigate one branch of these fusion systems and the fusion strategies associated with them. Looking from a wider perspective we categorise the combining systems in two groups. The combining systems of one group contain component experts that produce output of the same form and in parallel. These are referred to as Parallel Homogeneous Classifier systems, PHC. Each component classifier output is in one of three information forms, [109]:

- Abstract level: one output which is the label of the winning class.
Chapter 2. Fusion Strategies

- Rank level: all labels of all classes are output in rank from.

- Measurement level: all labels are ranked according to the measurement values of each rank.

The third form contains more information that is useful for decision making. These measurement values may not always be posterior probability values, and must be scaled before they can be used for combining. For example a classifier output could be a distance measurement, in which case it is possible to combine the results of many such classifiers by converting the distance measure into a probability form, as follows:

\[ P_k = \frac{\text{dist}_k}{\sum_{i=1}^{M} \text{dist}_i} \]

The second group comprises Complex Multistage Combiner systems, CMC, that do not require simple or complex fusion strategies. This is because the systems of this category contain component experts that are not connected in parallel [75] and the fusion is an internal complex process. Alternatively the systems contain parallel component experts, but the decisions do not belong to one of the three information forms itemised above, like the stacked generaliser [107].

In this thesis we focus on the PHC systems belonging to the first category. Several strategies for fusing the PHC systems exist. They can be grouped in to two types, Simple and Complex fusion strategies or combiners. Simple combiners do not require any training while complex combiners do. In the following two sections we briefly introduce strategies belonging to these two fusion groups. This is then followed by a brief introduction of the Complex Multistage Combining systems belonging to the second category. Since this thesis is focusing on PHC Systems, we will introduce them in the next chapter separately. In this chapter the presentation of the fusion strategies is conducted in the context of an example where the strategy has been implemented. The aim is to draw attention to the available methods and point out where they have been successfully used, however, not to explain each method in detail.
2.2 Fusion Methods for the Parallel Homogeneous Classifier Systems, PHC

In this section we review and survey methods used to fuse the decisions of PHC systems. The Fusion methods can be grouped in two, Simple Combiners and Complex Combiners.

2.2.1 Simple Combiners

Simple combiners are fusion methods that combine class outputs of different experts without the need for training. The mathematical representation for the fusion strategies discussed in this section can be summarised in the following formula. Given R experts and m classes, if for expert j the posteriori probability that a sample x belongs to class i is \( P_j(\omega_i|x) \), then the combiner probability estimate for class i would be:

- Average or Sum: \( \frac{1}{R} \sum_{j=1}^{R} P_j(\omega_i|x) \)
- Product: \( \frac{\prod_{j=1}^{R} P_j(\omega_i|x)}{\sum_{i=1}^{m} \prod_{j=1}^{R} P_j(\omega_i|x)} \)
- Minimum: \( \frac{\min_{j=1}^{R} P_j(\omega_i|x)}{\sum_{i=1}^{m} \min_{j=1}^{R} P_j(\omega_i|x)} \)
- Maximum: \( \frac{\max_{j=1}^{R} P_j(\omega_i|x)}{\sum_{i=1}^{m} \max_{j=1}^{R} P_j(\omega_i|x)} \)
- Median: \( \frac{\text{med}_{j=1}^{R} P_j(\omega_i|x)}{\sum_{i=1}^{m} \text{med}_{j=1}^{R} P_j(\omega_i|x)} \)
- Majority Vote: \( \frac{\text{vote}_{j=1}^{R} P_j(\omega_i|x)}{\sum_{i=1}^{m} \text{vote}_{j=1}^{R} P_j(\omega_i|x)} \)

Sum and Product

Sum (Average) and Product are the most widely used simple combiners. The \textbf{sum} (average) combiner adds the a posteriori probabilities assigned to each class by all experts and normalises the outcome by dividing by the number of experts. The \textbf{Product} combiner multiplies the expert outputs for each class, then the outcome is normalised by dividing by the sum of the product obtained for all the classes. Two less popular
derivations of Sum are the difference and weight combiners that were used by Yu et.al. [110]. Difference is calculated by finding the difference between the highest score of a class and the score of the class below it. The decision of the expert with the highest difference is accepted. Off-course the experts scores or outputs must be of identical scale. Weights are calculated by first ranking the scores of the classes in a descending order. Then the weight is calculated for each class in the ranked list, as follows: \[ W_1 = \frac{S_2}{S_2} + \frac{S_3}{S_2} \ldots \] Next the score of each class is recomputed by multiplying it by its weight. Finally the new scores of all classes are combined using the sum rule. The weight method requires more steps than Sum and Product and is largely heuristic.

Product and Sum, among other simple fusion strategies, were derived by Kittler et.al. [50, 48] based oil Bayesian theory for experts employing distinct representations. Starting from the Bayes decision rule involving all the available measurements jointly, they derive the Product rule under the assumption that the a posteriori probabilities are output from identically distributed experts. Next the Sum rule is derived from the product rule under the assumption that the a posteriori probabilities will not deviate much from prior probabilities. Then the Minimum rule is derived from the product rule. While Median, Maximum and Vote are derived from Sum.

Vote

Although linear combiners, mentioned in the previous subsection, exhibit very good performance, Vote is also widely used, partly because it can be used to fuse class labels while Sum and Product can not. Vote has been shown to be a very effective method of combining. Srihari [93] applies voting as a classifier combination method in handwritten and machine printed text recognition to achieve near optimum results.

Many versions of voting exist, such as the simple borda count, Threshold Voting and simple Majority Voting. Lam and Suen [55] give a comprehensive analysis of the behaviour of Majority Vote, under the assumption of independence of experts. They found that Voting with an odd number of experts produces a higher recognition rate, while Voting with an even number of experts produces better results, if errors are more costly than rejections. However the work is focused on Vote, without any comparison
2.2. Fusion Methods for the Parallel Homogeneous Classifier Systems, PHC

to other strategies.

Xu et al. [109] propose two voting methods that they claim to outperform Majority Vote. The first method assigns a sample to the class with the highest number of votes as long as another class does not get a vote, in other words all experts assign a sample to the same class or reject the sample. In the second method they propose a winning class to be the one that has the highest vote and its votes are larger than the second largest vote by a threshold, otherwise no decision is made and the test sample is rejected.

Hansen and Salamon [39] train each component of an ensemble of neural networks on a separate part of the training data via cross-validation. It is a procedure they use to optimise the parameters of each component neural network. Next, they use Voting to combine the decisions obtained by the component neural networks. The results show an improvement over any single neural network. However, the work lacks a comparison to an optimal neural network that is trained on the full training set. The optimal neural network would require more neurons or hidden units due to the larger training set. They derive an equation for the probability of error of the majority vote rule as:

$$\sum_{k=\frac{R}{2}}^{R} \binom{R}{k} p^k (1-p)^{R-k}$$

Provided $p < \frac{1}{2}$. R is the number of component neural networks and $p$ is the classifier error rate. It can be seen that the error decreases as R increases.

Order Statistics

Order statistic combiners are Median, Minimum, Maximum, Trimmed Means and Spread Combiners, [99, 102]. As mentioned in the simple combiners subsection, Maximum and Median are derived from Sum, while Minimum is derived from Product [50].

The simple combining methods of Sum or Vote are suitable when individual classifiers perform the same task with comparable success. If outliers exist or the performance of individual classifiers differs then order statistic combiners are expected to outperform other combining methods. Tumer and Ghosh [99] investigate order statistic combiners, Median, Maximum and Minimum. Maximum is equivalent to selecting the class with the highest posterior, hence Maximum has information from the most confident experts. However the drawback is that it would select a combiner that repeatedly outputs high
values, i.e. is stuck at high. **Minimum** deletes experts with little evidence, and its drawback would be that it would select a classifier that repeatedly outputs low values, i.e. is stuck at low. **Median** considers the most typical representation of each class. It is more robust especially for highly noisy data or when outliers exist.

In [102], Tumer and Ghosh propose **Trimmed Means**, which averages as in the simple sum rule, except that certain classifiers are not used if their outputs are at lower or upper ends of the spectrum of decision values. They also propose **Spread combiners**, which instead of throwing away the ends of the spectrum, Maximum and Minimum decision values are averaged. Results in [102] show that the two proposed methods work best only when there is a high variability between classifiers. Classifiers have high variability when the performance of individual classifiers is uneven and class dependent, or when the data size is insufficient or the data contains a high degree of noise. Therefore, contrary to Sum and Product, the last two fusion strategies are best only under certain restricted circumstances.

**Error Correcting Output Codes**

Dietterich and Bakiri [25] propose a method of improving multi-class problems using **Error Correcting Output Codes**, (ECOC). For each class a codeword is assigned such that it is furthest away from the codewords of other classes. When the model is presented with a test sample, the resulting codeword is compared to a table of codewords of all classes in order to find the closest one. The comparison is based on the Hamming distance measure. In addition to maximising the distance between codewords (row separation) they also recommend that the distance between columns is maximised to achieve the best performance. Unless the number of classes is at least five it is difficult to gain from ECOC. They propose four methods of constructing ECOC depending on the number of classes. The combiner decision is based on a distance measure and does not require training. Therefore, ECOC can be viewed as a simple combiner.

They note that when a small number of samples were used, ECOC required complex decision trees and more hidden units in the case of neural networks. More complex trees or neural networks usually require larger training sizes. In contrast their results
showed ECOC not to benefit from increasing the sample size. It actually performed worst on a letter data set, when the data set was large. This leads to a conclusion that ECOC works by reducing the variance of the learning algorithm. For a small sample, the variance could be high, leading to more benefits from ECOC.

In [26], Dietterich and Kong show that combining bagging with ECOC improved the performance of both methods. Also, they show that ECOC does not work well with highly local algorithms such as $k-NN$.

Besides its applicability as a combiner, Kong and Dietterich [53] use ECOC also to estimate class conditional probabilities.

### 2.2.2 Complex Combiners

The fusion methods discussed in this subsection belong to the second category of the fusion strategies used by PHC systems. The main common characteristic of the complex combiners is that they require training prior to their use. Besides the need for training, these fusion methods may require additional processing steps during fusion. Therefore, they share the characteristic of being computationally costly. The survey of the available methods introduced below indicated that most complex combiners, were not compared to the simple combiners. Hence, one can not always conclude that the complex methods would achieve superior performance. Additionally the complex combiners have the following disadvantages compared to the simple combiners.

- Leads to additional combiner system complexity which is undesirable, especially for complex experts or designs.

- Require extra processing steps as in the rank based method [43].

- They are computationally more expensive due to the required training phase.

- They may require splitting the training set in two subsets, one for training the fusion strategy and the second for training the classifiers. This leaves the classifiers with a smaller number of samples to train on.
• May not outperform the simple combiners as in the Logistic Regression fusion [10].

• The performance on the test set depends on how well the training set represents the test set.

In this thesis we consider investigating only the simple combiners due to the disadvantages of the complex combiners, outlined above. Also, some complex combiners eventually require a simple combiner as in the Logistic Regression fusion [10]. However, some complex combiners do not combine the decisions of multiple classifiers as the simple combiners do. They select the decision made by one or a subgroup of the multiple classifiers intelligently as in the handwritten recognition system of Cao et. al. [20].

Weighted strategies

The simple fusion methods are turned into complex methods if different classifiers are assigned unequal weights based on their performance on a validation set. Hashem and Schmeiser [40] propose an optimal liner combination method in which they combine the outputs of component neural networks by a weighted sum method. The weights are found such that they minimise the mean square error. Their tests indicate an improvement over the simple averaging method. However Luis et. al [2] show through a theoretical derivation that when independent experts are combined, equal weights yield the best performance. For weighted Sum expert outputs can be assigned different weights based on their performance on a validation set, i.e. changing Sum into a complex combiner. Some results have shown that the classifier confidence is not uniform over all data space, therefore constant weights in the weighted averaging method may not be optimum. Windeatt and Ghaderi [106] propose a dynamic weighting method such that the weights depend on the classifiers performance over a certain class and data subspace. The weights are found using an oracle which is a neural network trained on a validation set. This is undesirable for applications requiring simple strategies.
2.2. *Fusion Methods for the Parallel Homogeneous Classifier Systems, PHC*

---

**Rank based Fusion**

*Rank based* combining deals with experts outputting ranked class labels, i.e. the second form of decision information.

Ho et al. [43] used ranking for combining the decisions of different types of classifiers. First they reduce the number of competing classes via "Class set reduction" in which a subset of classes is selected such that it has a high probability of containing the true class. Next the class ranks are reordered, via "Class set reordering" which reorders the ranks such that the true class is as close to the top as possible, where the class at the top is considered as the true class. The Borda count and highest rank methods are used to reorder the class ranks. They propose that the difference in classifier performances be reflected by assigning weights to the rank scores produced by each classifier. If two classifiers give a high sum of their weighted ranks then they are correlated or the class is correct. They also propose a method which takes advantage of the strength of classifiers and avoids their weakness. In particular they propose the concept of dynamic classifier selection where the training set is divided into partitions, then the performance of each classifier is tested on each partition. Each test sample is first assigned to a partition then classified by the classifier of that partition. This method attempts to solve the problem of only class labels being available. It draws additional information from these labels to help it reach a final decision on the correct label.

**Dempster-Shafer Fusion**

Rogova [80] combines neural networks using *Dempster-Shafer*, (D-S), theory, which uses statistical information about the relative classification strengths of several classifiers. The neural networks output a measurement level information which can be used for probability assignment. A measure of classifier confidence is calculated from the sets of outputs obtained from the training data. When a test sample is presented, using the D-S theory, the evidence from all classifiers is combined to obtain a measure of confidence for each class. The test sample is assigned to the class with the highest confidence. Rogova finds that better results are not necessarily achieved by combining the best classifiers. Sometimes the combination of most independent classifiers yields better
results. A sample example out of many, where D-S fusion is used, is the handwritten letter recognition system of Ng and Singh [64].

Fuzzy Integral

Gader et al. [37] use Choquet Fuzzy Integral to combine multiple handwritten word classifiers. Their method outperforms weighted borda count, simple borda count, neural networks, and the Sugeno fuzzy integral. Previous handwritten applications of fuzzy integral used prior knowledge about the worth of each classifier for each class to define parameters of the fuzzy measures. This approach could not be used in word recognition because of the huge number of classes. A data dependent approach was used instead to construct the parameters. After a fuzzy measure of expert confidence on each class is constructed, the fuzzy integral measures the overall confidence for each class. The class with the largest integral wins. The literature on the use of fuzzy integral based fusion is very large. One sample example is where Wang et al. [105] successfully used the Sugeno fuzzy integral to combine the decisions of multiple neural networks that recognise bacteria.

Local Accuracy Estimates, Dynamic Classifier Combination

Some have used methods to consider the decision of one or some of the experts. For instance, using a validation set the performance of each classifier in each region of the input space is determined. When a test sample is presented, it is assigned to one of these input space regions and the decision of the classifier that showed the best performance on the test region is considered.

In [45], Huang and Suen propose the Behaviour Knowledge Space, BKS, method which can aggregate the decisions obtained from individual classifiers and derive the best final decision from a statistical point of view. In tests it outperformed Voting, which does not consider the classifiers ability, D-S and Bayesian fusion, which assume independence. BKS does not assume independence and it takes into consideration the classifiers performance on the input data. BKS requires large enough and well representative data. If samples are collected randomly or carelessly the desired performance of
BKS is not guaranteed. The method can be summarised as follows; for $R$ experts and $m$ classes, the BKS is a $R$ dimensional space with each dimension having $m+1$ possible decision values. The intersection of the decisions of individual classifiers contains the number of samples belonging to each class. The focal unit is the unit at the intersection of classifiers decisions for the current input. When the current input is presented the focal unit is found using the component expert outputs. Next, at the focal unit, the class with the highest number of samples is considered as the true class of the input.

Rahman and Fairhurst [73] introduce a Confidence index methodology to build up a knowledge base of a priori $2^{nd}$ order information derived from a data set to be used in a majority rule combiner. They build a database of information about experts confidence by calculating three indices, an overall, a class, and a sample confidence index. When a test sample is presented, it is classified according to the decision of the expert with the highest "sample" confidence index, as long as its index is higher than the next competing expert index by a threshold. If no decision is reached the decision of the expert with the highest "class" index is used. If again no decision is reached the "overall" index is used, and if again no decision is reached the test sample is rejected. Lin et.al. [59] use Logarithmic opinion pool (Log-Op), and Linear opinion pool (Linear-Op), to combine adaptive confidence measures. These measures are distances that have been transformed to probabilities by the authors proposed method of Adaptive Confidence Transform. They propose to use Log-Op to combine independent classifiers, and Linear-Op to combine correlated classifiers. Linear-Op is a variant of weighted Sum, while Log-Op is a variant of weighted Product. They also propose to check the decision of a classifier and if it is above a threshold then the decision is output without combining the decisions of all experts. When this method was tested on a Chinese character recognition problem the results showed that $78\%$ of samples were correctly classified without combining.

In [108], Woods et.al. propose a dynamic classifier selection method using local accuracy estimates. This is done by estimating each classifiers accuracy in local regions of feature space surrounding an unknown test sample. The most locally accurate expert is the winner. The local regions are defined using the K-nearest neighbour algorithm and the training data. The “Local Class Accuracy” is estimated by finding the
percentage of the local training samples correctly assigned to a class by each classifier. The “Overall Local Accuracy” is estimated by finding the percentage of all training samples correctly classified into the region by each classifier. They find that the Local Class Accuracy method outperforms the Overall Local Accuracy method. They also find that for some mixtures of combining experts, lower numbers of component experts give better results than larger number of experts, especially if the worst expert is removed. They recommend that sequential backward chaining may be useful in selecting the best experts for combining.

Neural Network

Outputs of multiple experts could be combined by feeding them to the inputs of a neural network, which outputs a decision indicating the winning class.

Lee and Srihari [56, 57] use an MLP neural network to combine the confidence assigned by component classifiers to classes. This network is named decision combination neural network, DCNN. They further propose to take into consideration the dependencies of the classifiers on the type of input data. They, again, use an MLP neural network to capture classifier output dependencies on image characteristics, called dynamic selection network, DSN. DSN is used to dynamically select, or in other words weight, the classifiers outputs entering into DCNN, by multiplying classifier outputs with a gain factor that is output from DSN for each classifier. The classifier used is a first order Bayesian approximator. Their method is good for combining correlated classifiers.

For handwritten numeral recognition, Lin et.al. [59] use a back-propagation network to combine the confidence value outputs of two neural nets, a self organizing mapping network and a back-propagation network. Using a validation set, they use the outputs of the first stage networks as training values for the combiner network.

All of the methods that use neural networks to fuse the decisions of multiple classifiers have the disadvantage that the exact performance of the combiner stage may not be reproduced once the combiner stage is retrained due to using different initial weights.
2.3 Complex Multistage Combiner Systems, CMC

Regression based Fusion

Regression based methods are based on optimising fusion weights based on the training set. Therefore they can be categorised under the weighted strategies. We have used regression methods [10] to assign weights to each of the component experts based on their performance on a validation set. To find the weights we have used three different methods; nearest mean [36], linear regression and logistic regression [44] methods.

Our results indicate that simple combiners can, on many occasions, outperform the above regression methods. Verlinde [104] uses logistic regression combination in a person identity verification system, but does not show whether it is necessary to use such a complex combiner instead of the simple combiners. Ho et. al. [43] also use logistic regression, but to combine ranked labels. They have not compared the logistic regression results with the vote simple combiner.

2.3 Complex Multistage Combiner Systems, CMC

Some combiner systems do not incorporate decision aggregation as the simple or complex combiners do. However, the complete system works to reach a final decision based on a complex process that occurs in the system as in the serial numeral recognition system of Rahman and Fairhurst [75]. These systems approach the fusion problem from a different angle and we do not consider them, in comparison to the simple combiners, as an alternative choice. We refer to such systems as Complex Multistage Combiner systems, (CMC), that incorporate multistage experts which, according to the classification presented in the introduction, constitute the second category of the fusion system.

CMC is a method of building complete classification systems that contain various experts. The experts could be connected serially, in parallel or using a mixed architecture. Thus, in multistage combination one may not necessarily obtain various decisions or scores that require fusion. In this sense, CMC systems are complementary to the PHC systems which require some type of fusion strategy. In the following section an overview of the different types of fusion systems belonging to the CMC group is presented.
2.3.1 Application Specific Systems

We call CMC systems of this group application specific, (AS), due to their architecture which is dedicated to a specific application. Here expert outputs could be merged in a serial, parallel, hybrid or dynamic manner. Sharkey [84] and Rahman and Fairhurst [75] compare some work done in this field. In the serial method, experts of the lower stages narrow down the choice for the upper stage experts. Decisions of each stage are the input to an upper stage. The importance of having more powerful experts at the later, upper stages is emphasised. The performance of the subsequent stages depends on the performance of the previous stages. In the parallel method the outputs of all experts are channelled to a stage that has to make a decision from the group of decisions it has received. Here, the decisions made by experts are concurrent and not influenced by other experts. The results [75] show that although the parallel architecture offers an improvement over serial, it is computationally expensive.

In the hybrid approach the serial and parallel methods are both used. The first stage tries to make a decision about the correct class, but if this cannot be done with confidence then the approach is to try to narrow down the number of possible classes and serially feed the possible answers to the next stage. In this way the task of the next stage is simplified. This approach requires a high level knowledge regarding the physical structure of patterns to be classified. The types of inputs are known in advance. The decision of this stage is channelled to the final stage to be combined in parallel with the decision of the first stage.

The hybrid method, presented by Rahman and Fairhurst in [74], showed the best performance. The system incorporates the a priori knowledge of numeral classes inferred from the training samples. In other words the architecture reflects the prior knowledge of the numerals that are similar and prone to be confused, for example 4 and 9, 7 and 1 or 8 and 3. The first stage of the system tries to identify an input sample. If the sample is identified, the result is directed to the final stage. If the numeral is not identified, and a confusion between numerals arises, it is directed towards the second stage by channeling it to filters specialising in resolving specific ambiguities. The activated filter decides between candidate hypotheses. If it is not a sample from the
set of hypothesis handled by the filter, then the input is rejected and passed on to a reject recovery classifier. The decision of the reject recovery stage of the dichotomiser is passed on to the final stage. The decisions made by the first stage, reject recovery and dichotomisers are combined at the final stage. The combination of the final stage is more like a selection than combination, since the answer is held by only one of the three experts feeding into the final stage. In general the multistage combination used here seems to be some kind of class set focusing. The decision of a confident expert is accepted otherwise another expert is engaged.

Fairhurst and Rahman [32] used some of the above mentioned techniques to classify characters. Two methods were used. The first method consisted of two classifiers. If a character was rejected by one classifier then it would be sent to a reject recovery classifier for classification. In the second method the character is first directed to one of several groups. The classifier that deals with the specific group of classes then makes a decision. If the character is rejected, then it is sent to a reject recovery for classification. Here they take advantage of similarities and dis-similarities between characters. They find that grouping dissimilar characters results in a better performance than grouping similar characters.

Rahman [77] gives an extensive literature survey of the classifier combining methods specifically applied to character recognition, many of which fall under the CMC category.

Elshishiny et.al. [31] use 3 different classifiers that are progressively more complex to classify pixels of remote sensing images. If the simple first stage can not make a decision, it is sent to the third stage. Or if the first stage associates a pattern with more than one class the second stage is used to classify. If the second stage can not make a decision then the decision making task is passed on to the third stage which is computationally expensive.

Kimura and Shridhar [47], combine two different character recognition algorithms that use two different feature sets. They use several methods to combine, but these can be grouped into two main categories, Parallel and Sequential. In the parallel case the decisions of both algorithms are checked and if they give different results, or both
reject the input pattern then the character is rejected else it is accepted. If an algorithm delivers multiple class membership (i.e. ranked class list) then the second algorithm output is checked for consistency of results in order to make a decision, otherwise the character is rejected. In the serial method, one classifier algorithm is used. If a sample is rejected then the second algorithm is engaged. Other variants of the serial method include the case of class pruning by one algorithm. The resulting set is fed to the second algorithm for processing and selection of a single class. The results showed that the multiple class parallel algorithm performed best.

2.3.2 Hierarchical Mixtures of Experts

Jordan and Jacobs [46] present a tree structured architecture for supervised learning algorithm. They use Expectation Maximisation techniques to find and optimise, i.e. learn, the architecture parameters. They divide the input space into a nested set of regions and fit simple surfaces to the data that fall in these regions. The boundaries between regions are soft and are simple parameterised surfaces that are adjusted by the learning algorithm. The system is called Hierarchical Mixtures of Experts, (HME). The architecture has expert networks at an initial stage and gating networks at all subsequent stages. When presented with an input pattern, a node of the network produces an output as a generalised linear function of the input. The same input is concurrently presented to a gating network. The output of the gating network is the log of the weighted input.

The output vector of each unit is the sum of the node outputs weighted by the associated gating network outputs. In other words, starting from the bottom level, expert outputs are multiplied by the corresponding gating network outputs. Then the result of these multiplications are summed and sent up one level. The same procedure is repeated at the next level, where the sum results of the previous level are multiplied by the gating networks of the new level. The procedure is repeated until we are left with one outcome at the top level. Hence the expert network outputs proceed up the tree, being blended by the gating network outputs. The number of levels and number of network experts depend on the data. At the start of training the parameters of the gating network
are small and the entire system reduces to a single averaged generalised linear model. After training, the parameters in the gating network begin to grow in magnitude and splits are formed. As training proceeds parameters on branches of the splits increase. The training is stopped when the error reaches a goal. Then the effective number of branches needed becomes obvious.

### 2.3.3 Neural Networks

Cao et al. [20] use multistage neural networks, (NNet), to identify numerals. The first stage finds the Euclidean distance between a test vector and each of the possible classes. Its decision about the class membership is based on the distance between the first and second closest classes. If the distance between the closest two classes is below a threshold, then the decision is channelled to the second stage, along with a new set of features (directional histograms). This second stage consists of 45 neural nets, one for each combination of two numerals. The decision of the first stage decides which of the 45 neural networks should be used. Therefore only one of the 45 neurons should output a value, and that value should be larger than a threshold in order to accept its decision, otherwise the input test vector is rejected.

Alkoot and Foda [5, 6] use two stages of neural networks (serial combining) to decode linear hamming codes. The first stage is a one layer perceptron network that identifies the class of the input codeword, while the second stage, which is an outstar neural net, uses the decision of the first stage to identify the exact transmitted message.

### 2.3.4 Stacked Generalisation

Stacked generalisation, (SG), proposed by Wolpert [107], is another method of multistage combination. The first stage is trained on a separate part of the input data set, then its performance is validated using the validation set, a process similar to cross-validation. The results obtained during validation of the first stage are used as the training set for the second stage, the combiner. If the input data set is partitioned, T times, into disjoint training and validation sets, then we have T training sets for
the second stage. The number of second stage training samples equals the number of validation sets $T$ multiplied by the number of validation set samples.

Stacked Generalisation is a method of combining that removes the bias errors of the component classifiers by training the combiner to take them into consideration. After creating the second stage training set, the first stage is trained one more time using all the samples in the input data set. Wolpert does not recommend which classifiers or generalisers to use in the first or the second stage. However, he suggests to use different models that are mutually orthogonal in the first stage while the second stage should be a simple, relatively global and smooth function, not overly concerned with the reproduction of the learning set. It is possible to use more than two stages, in which each stage will take the decisions of the previous stage as its training set.

Friedrich [35] used a 3-level stacked generaliser consisting of evolutionary created neural networks with different architectures and weight settings and a nearest neighbour classifier. The components of the stacked generaliser were obtained by an evolutionary algorithm to optimise the architecture of the neural networks. He compares several combination methods, Entropy, Vote, Sum and Weighted Sum. He finds that entropy outperforms all, while Vote is the worst of all. Weighted Sum performed better than Sum. He uses three different methods for selecting the component classifiers that are used in the system. Although this leads to no difference in performance, there was a difference in the resulting ensemble size. His approach requires a large number of training samples, making it useless for real world problems involving sets of small sample size.

2.4 Conclusion

In this chapter we survey the available fusion methods and systems. We categorised the combining systems in two groups, the Parallel Homogeneous Classifier and the Complex Multistage Classifier systems. Fusion strategies used in the first category are then grouped in to two, simple and complex methods. The available simple and complex fusion methods used in PHC systems were briefly reviewed. We presented Sum, Product, Vote and order statistic combiners as examples of simple fusion strategies.
2.4. Conclusion

Among the complex combiners presented, there were weighted strategies, rank based, Dempster-Shafer based, fuzzy integral, dynamic classifier fusion and local accuracy estimate based methods, neural networks and regression methods. A more detailed comparative survey is left for Chapter 4, where the simple fusion methods are further investigated and experimentally evaluated.

As this thesis focuses on PHC systems only, the CMC systems were introduced briefly in this chapter. The CMC systems introduced were application specific systems, hierarchical mixture of experts, neural networks, stacked generalisation and ECOC methods.
Chapter 3

PHC Fusion Systems

Design strategies of the PHC systems are explored with the emphasis on training set and feature set manipulation approaches. New procedure for bagging and a novel multiple classifier design method based on feature selection are proposed.

3.1 Introduction

In the previous chapter, combination systems were split in two categories. One category contains Parallel Homogeneous Classifier, PHC, systems that output decisions in parallel, which in consequence require simple or complex fusion. The other category contained the rest of the combination systems, termed Complex Multistage Combiner (CMC) Systems, that do not require simple or complex fusion strategies.

As mentioned earlier, in this thesis we are interested in the systems that belong to the first category. There are many methods for constructing multiple experts that benefit from fusion. All available methods acknowledge the necessity for using uncorrelated experts to achieve maximum gains from fusion. Tumer [97] and Sharkey [83] investigate the relation between component expert variance and bias, and fusion performance gains. Tumer [97] notes that some combination methods like boosting reduce bias, however most methods reduce variance. Hence, a good strategy would be to select experts based on their low bias even if they have high variance which can be reduced by combining.
Next, we shall describe methods for obtaining independent experts, then we shall focus on the top two of the following methods.

- **Training Set Manipulation:** It is possible to obtain independent classifiers if each classifier is trained using a different training set. The methods for obtaining different training sets are: cross-validation, randomness, boosting, bootstrapping, varying data sources and preprocessing.

- **Feature Set Manipulation:** Classifiers designed and optimised using different features would generally perform differently over the input space. There are many methods of assigning features to these experts. We propose a novel method \[10\] of assigning features to the component experts, as they are designed and built, such that the system performance is optimised over a validation set.

- **Different Learning Algorithms or Architectures:** Experts can be created using different methods such as Neural Nets, Nearest Neighbours, Decision Trees, Linear Discriminant or statistical Bayes classifiers. Also, when using neural nets different architectures can be achieved by varying the number of hidden units or levels. It is also possible to change the performance of an expert by changing the type of the output function of a neural network. Although retraining neural nets using different initial weights may result in independent experts, the results have shown that this method of obtaining independent experts does not yield a performance superior to the other methods mentioned here. Independent experts can also be acquired by varying training data. This method is considered more efficient than varying initial conditions.

- In \[24\], Dietterich proposes to randomise the internal decisions made by the learning algorithm as an approach to generating ensembles. The way it works is that a split is introduced at each internal node of a tree randomly. He computes the best 20 splits then chooses one randomly.

Following is a detailed explanation of the methods that involve the manipulation of the training set and the manipulation of features.
3.2 Training Set Manipulation Based Ensembles

3.2.1 Cross-Validation

A method of obtaining different training sets per classifier is by randomly dividing the training set into subsets, then building different classifiers using all subsets except a test subset, which is different for each classifier.

3.2.2 Randomness

In order to obtain diverse training sets, Raviv and Intrator [78] bootstrap samples from a training set then inject gaussian noise into the samples. This leads into more independence. The amount of noise must be carefully selected such that its excess does not lead to high errors. There is an optimum level of noise at which classifiers are most independent while their performance is kept at acceptable levels.

3.2.3 Boosting

Freund and Schapire [34] propose a serial method of generating classifiers. After training an expert on a set, the same set is used to design a new classifier, but the samples are assigned weights. The larger the weight the higher is the probability of a sample being selected into the training set. The values of the weights depend on the performance of the previously built experts. They are assigned such that the weights of misclassified samples are increased. The results of the multiple experts are combined by simple or weighted voting.

Schapire [81] claims that boosting is fast, simple and easy to program. It requires no prior knowledge about a weak learner, hence, it can be combined with other methods for finding weak learners. Its actual performance depends on the data and the weak learner. It fails if data is insufficient or if weak hypothesis is overly complex or too weak. It can identify outliers by giving them larger weights.

Breiman [19] gives a different name to Freund and Schapire's boosting, Adaptive Resample and Combine, ARC-fs. He shows that ARCing is more successful in variance
reduction than bagging. It also reduces bias, which bagging does not. He claims that ARCing is successful because of the adaptive resampling property not the specific form of ARC. He proposes another method of modifying the sample weights or probabilities $p(n)$, as follows:

$$p(n) = \frac{1 + m(n)}{\sum_n (1 + m(n))}$$

(3.1)

where $m(n)$ is the number of misclassifications of the $n^{th}$ sample by all classifiers. In contrast, Freund and Schapire's method updates the probabilities of the samples as follows:

$$p^{k+1}(n) = \frac{p^k(n)\beta_k^d(n)}{\sum p^k(n)\beta_k^d(n)}$$

(3.2)

where $\epsilon_k = \sum_n p^k(n)d(n)$, $d(n) = 1$ if $n$ is misclassified, else $d(n) = 0$ and $\beta_k = \frac{(1-\epsilon_k)}{\epsilon_k}$.

After constructing $R$ classifiers their decisions are combined using weighted Vote with classifier $k$ having weight $\log(\beta_k)$.

3.2.4 Bagging

Bagging [18], proposed by Breiman, is a method of generating multiple versions of a predictor or classifier, via bootstrapping and then using these to get an aggregated decision. Methods of combining suggested by Breiman are Voting when classifier outputs are labels, and Averaging when the classifier outputs are numerical measurements. The multiple versions of classifiers are formed by making bootstrap [30] replicas of the training set, and these are then used to train additional experts. He postulates the necessary precondition for bagging to improve accuracy being the classifier instability. By instability we mean that a perturbation of the learning set causes significant changes in the classifier output.

3.2.5 Modified Bagging, Population Bias Control

For large sample sets the k-NN rule has been shown to be stable and therefore not much gain in performance can be expected from bagging $k-NN$ experts, unless some degree of diversification among bootsets is introduced. However, under small sample
size $k - NN$ can be unstable and one may benefit from bagging. In this thesis our experiments involving real data and real experts focus on using and investigating bagging of $k - NN$ experts. When drawing a learning sample set there are two random factors which impact on the classifier performance. The first is reflected in the spatial location of the samples and is governed by the conditional distributions of the respective class populations. The second factor relates to the process of sampling from the prior probability distribution of the classes.

For large data sets the second factor is insignificant. However, when a data set is small, the proportions of training patterns from the different classes may be unrepresentative. The probability of drawing a training set with samples from some class completely missing becomes non negligible. When this occurs, bagging may even become counter-productive. For this reason we set out to investigate whether some form of control over bootstrap sets would result in improved performance. Three modifications of the standard bagging method were considered. We name the standard procedure as method 1 and its modified versions as methods 2-4. The methods which exploit increasing amounts of prior knowledge can be summarised as follows.

We are aware that some, e.g. Skurichina and Duin [92, 89, 90] and Skalak [87] bootstrap each class separately. This is to a degree similar to methods 2 and 3 below. However, most papers do not describe how the sampling process is performed, and we believe that many use method 1. We experiment with the different bootstrapping options, outlined next, to see the effect of the different bootstrapping methods on the performance of bagging.

**Method 1** This is the standard bagging method. Given a learning set, each bootstrap set is created by sampling from the learning set randomly with replacement. The cardinality of each boot set is the same as the size of the training set.

**Method 2** When bootstrap sets are created from the learning set we check the ratio of the number of samples per class in the bootstrap set. This ratio is compared to the ratio of samples per class in the learning set. If the difference between the compared ratios is larger than a certain class population bias tolerance threshold we reject the bootstrap set. We conduct the experiments for a bias tolerance threshold equal to 10%.
Method 3 This method is similar to method 2 except that the bootstrap set ratio is compared to the ratio in the full set. By full set we mean the set containing all samples, learning and test samples. This full set ratio simulates a prior knowledge of the class distribution in the sample space.

Method 4 Here we only require that all classes be represented in the bootstrap set, without enforcing a certain ratio of samples per class. This is done by rejecting any bootstrap set that does not represent all classes.

### 3.3 Feature Set Manipulation Based Ensembles

If different features are used by the component experts of an ensemble, these experts would be diverse and may become uncorrelated. In the following subsections we discuss the available methods to select features and assign them to the corresponding experts. Training set manipulation has attracted more attention than feature set manipulation, however, the interest in the latter has recently grown considerably. Different strategies have been used, ranging from the conventional method of optimising the component experts independently to novel methods that construct an expert with a goal to optimise the combiner. Recently, using random feature subsets has also attracted more attention. The following subsections describe feature manipulation methods for PHIC system design. Methods that fall in to the CMC systems category also exist. For example Chen [21] constructs a system similar to the EM of Jordan and Jacobs [46] where each network, consisting of R neural networks, is designed using a different feature vector.

#### 3.3.1 Conventional Methods

The most commonly used method involves designing each expert independently on different sets of features. The experts are usually different, however if the feature subsets are disjoint then we could combine experts that use similar algorithms. Different feature extraction methods are used to obtain different sets of features. On the other hand one set of features can be used for different experts, where each expert is assigned
the optimum subset of features using feature selection techniques. The main common theme in all the conventional methods is that the experts are designed independently during the training phase, then fused in a combiner.

One recent example is the Input Decimation work of Oza and Tumer [67]. They use feature set manipulation to create diverse experts which focus on a certain class. For an \( l \) class problem, they construct \( l \) MLP neural networks with a single hidden layer. For each neural network they select the best feature subset that yields the highest correlation between each feature and the class for which the current expert is under construction. This method performs best when a large number of irrelevant features exists and when the number of training samples is small.

### 3.3.2 Combiner System Based Method

In Chapter 7 we adopt a completely novel design philosophy [8, 10] when we take the view that the design of individual experts and fusion cannot be solved in isolation. This premise leads to a completely different design methodology whereby each expert is constructed as part of the global design of a final multiple expert system. As the design of component classifiers is optimised using a common performance criterion, that is the error probability of the multiple expert system, it is reasonable to expect that the final design will be at least as good as the fusion of individually designed experts and hopefully much better.

The design process involves jointly adding new experts to the multiple expert architecture and adding new features to each of the experts in the architecture. The feature selection problem itself is of combinatorial complexity and it is clear that the optimisation over different architectures and individual experts in each architecture will be computationally explosive. For this reason, in Section 7.5, we use only the simplistic sequential forward feature selection and the add-1-take-away-r methods to build the individual experts and the fusion system, while in Section 7.6 we use random subset selection.

In Section 7.5 we investigate two distinct design strategies which we refer to as *parallel* and *serial*. In the parallel approach we utilise a pool of features by building a number
of experts simultaneously by distributing the features available to the experts in a "card dealing" manner. In this approach the dimensionality of the feature spaces in which the experts operate increases in a balanced way and also the expert strengths are reasonably balanced. In contrast, in the serial approach any expert is allowed to absorb new features as long as the system performance continues to improve. This inevitably means that the first experts take better and more features than the experts added later, leading to imbalance both in feature space dimensionality and individual expert performance. The two design strategies are compared to the conventional system and the optimised conventional system where the optimisation is based on the system performance.

The novel philosophy is adapted in Section 7.6, where we integrate two PHC design methods to find if there is a merit in mixing two design methods. Out of a pool of random feature subsets we assign a subset to each expert if that leads to improved system performance. This is compared to the conventional method of assigning feature subsets based on the performance of the individual experts.

3.3.3 Random Subsets Method, RSM

A fast feature selection alternative would be to randomly select a prespecified number of features from the full feature set. If a randomly selected subset leads to degraded system performance then another one is selected. To avoid getting stuck in an infinite loop the random feature selection process for each expert is halted after a prespecified number of trials. This is repeated for all experts in an ensemble. This random feature selection process is used in Section 7.6 to diversify bootstrapped experts in experiments involving bagging of $k$-NN experts. The literature on random subset or subspace method is growing, following is a sample of the current research involving RSM.

Kuncheva and Whitaker [54] experiment with combining three experts, each taking a subset of the features. All possible enumerations for subset sizes 4, 4, 2 and 4, 3, 3 were tested. They find that random partition of the features yields more frequent improvements if weak classifiers were combined. The success of random partitioning also depends on the problem complexity, number of features per subset and the number
3.3. Feature Set Manipulation Based Ensembles

of combined classifiers.

Ho [42] uses RSM, to construct C4.5 decision trees, and compares the results to bagging and boosting. Using a table of results she concludes that RSM performs best when a large number of samples and features exist. On the other hand, it degrades when a few features and very small samples exist, or when a large number of classes exists. However, we find the table not in support of these findings. For example, contrary to her conclusions, RSM performs best when the number of classes is large totalling 26. Also, RSM degrades below others when a very small number of classes exist, i.e. two classes. Although it is obvious from the tables that RSM degrades for small number of features, it is not so for small number of samples. For two data sets involving equal number of classes and features we find that RSM degrades on the Nursery data which has a large number of samples, i.e. 12961, and achieves a better performance for the Car data with 1728 samples. Instead of the contradictory table of results, we consider the following reasoning to be more convincing. For training set based methods like bagging, increasing the number of samples does not lead to improved performance, because component classifiers may become more stable. Based on this we can expect bagging with a higher number of samples to underperform and consequently RSM to perform best.

Skurichina and Duin [90] study the effect of pseudo Fisher Linear classifiers and the redundancy in the feature set on the performance of RSM and bagging. Their main findings are that both RSM and bagging are useful in highly redundant feature spaces. However, RSM performs better when discrimination power is distributed over many features. They find bagging not to be affected by feature redundancy and could outperform RSM under small sample size.

Bay [16] explores the use of random feature subsets to combine nearest neighbour, (NN), classifiers. He combines 100 NN experts where each expert is assigned a randomly selected feature subset. The best number of combined experts and the best size of feature subset were found using cross-validation. The resulting combiner outperforms the best NN expert.

Opitz [66] at the initial step uses random feature subset selection to build the first R
experts. Then he uses feature selection based on genetic algorithms to build additional component, neural network, experts. The best R experts are selected using a fitness criterion which is calculated according to the following formula,

\[ \text{Fitness}_i = \text{Accuracy}_i + (\lambda \times \text{Diversity}_i) \]

The calculation of the diversity measure is based on somewhat a similar philosophy as the one underlying our system based feature selection, i.e. considering the performance of the whole ensemble. The diversity is found as the average difference between the performance of the component expert and the ensemble.

### 3.4 Conclusion

In this chapter we reviewed the current methods for designing PIIC systems. The design strategies were grouped into four categories, training set manipulation methods, feature set manipulation methods, methods using different component architectures or algorithms and the randomisation method.

The training set manipulation methods of bagging, boosting, cross validation and randomness were presented followed by a modified bagging method which we propose as a solution for bagging when population bias problems exist in small sample size conditions.

The feature set manipulation method presented include the conventional method of component based feature selection when the PIIC system comprises component experts implementing different algorithms or architectures. A novel method of system based feature selection was then proposed which optimises the system performance on a validation set. This method could be used for PIIC systems containing heterogeneous as well as homogeneous component experts. In addition to the available forward and add-I-take away-r feature selection methods, another method of random subset feature selection method has emerged in the fusion community and was explained briefly.
Chapter 4

Investigating The Simple Fusion Strategies

Simple fusion strategies are theoretically derived and experimentally evaluated.

4.1 Introduction

Although many diverse and sophisticated strategies have been developed, there is still considerable interest in simple fusion methods that do not require any training. Such methods can either perform at the decision level where for a given input signal each classifier (expert) outputs a class label. Alternatively, fusion strategies, such as Sum and Product, operate directly on the soft decision probability outputs of the respective experts. These rules which include also the Maximum, Minimum, Median and Vote rules have been studied extensively in [50, 48], where a bayesian theoretical framework for multiple expert fusion has been presented. There, it has been shown that these combination strategies can be derived from a common starting point under different assumptions. Most interestingly, a theoretical analysis of the sensitivity of these simple strategies to estimation errors is carried out and a plausible model which explains the empirically observed superiority of some of the combination strategies has been put forward. Based on the error sensitivity analysis, the authors have predicted that
the performance of the Product rule and its derivative, i.e. Minimum, would be inferior to Sum, although the Sum combination rule was derived using more restrictive assumptions.

The aim of this chapter is to represent the simple fusion strategies in more detail and, in particular, to validate them experimentally. Tax et al [96] has already contributed to this objective. Specifically, it has been demonstrated that the theoretically predicted relationship between two strategies, namely fusion by averaging and fusion by multiplication holds in the case of relatively high estimation errors. However, for small errors, the Product rule outperforms Sum. The reported study focused on Gaussian classifiers which differed in terms of the actual class distribution parameters used. The perturbation of these parameters from chosen nominal values was performed to emulate the effect of small sample size estimation errors.

However, we adopt a slightly different approach to model validation. We direct our attention to a single point in the class a posteriori probability space at a time and emulate the behaviour of individual experts by subjecting their nominal soft outputs to perturbation errors. This approach allows us to relate the measured effectiveness of various combination strategies to different error distributions acting on these nominal values. As a result we are able to parametrise the behaviour of the fusion strategies in terms of inherent ambiguity of patterns, expert estimation accuracy, and estimation error distribution. We also extend Tax's work to investigate the interaction of multiple experts in pattern classification problems involving several classes. The experimental design adopted facilitates also a study of multiple experts employing either shared feature spaces or distinct feature spaces. Moreover, the approach permits us to explore the effect of estimation error correlations.

The studies showed that for low estimation errors Product is superior to Sum which is consistent with the relative strength of simplifying assumptions behind these strategies. However, as the estimation noise level increases, the Product rule suddenly starts dramatically to deteriorate and eventually its performance becomes considerably inferior to that of the Sum combination. A closer analysis of the behaviour of the Product and Minimum strategies revealed that this degradation is caused by a so called veto effect.
4.2. Theoretical Derivation of Simple Fusion Strategies

When one of the experts outputs an aposteriori class probability close to or equal to zero, it will dominate the output of the combiner, resulting in a low value of the combined multiple expert a posteriori class probability. This will lead to misclassification.

In the following section the simple fusion strategies are derived based on bayesian theory, followed by an error sensitivity analysis of Product and Sum. In Section 4.4 survey of the fusion strategies is conducted to find how they perform relative to each other. In Section 4.5 we evaluate the simple fusion strategies and validate them experimentally. The study investigates the performance of the fusion strategies under Normal (Gaussian) and Uniform noise conditions.

4.2 Theoretical Derivation of Simple Fusion Strategies

In this section we will introduce the theoretical derivation of the simple fusion strategies which was reported by Kittler [48] and Kittler et. al. [50].

Assume in a pattern recognition problem, a test pattern $Z$ is to be assigned to one of $m$ classes ($\omega_1, \ldots, \omega_m$). In a combiner system $R$ component classifiers represent the test pattern by a distinct measurement vector $x_i$, where $i = 1 \ldots R$. In the measurement space each class is modelled by the probability density function $P(x_i|\omega_k)$. The prior probability of occurrence of a class $k$ is denoted by $P(\omega_k)$.

According to bayesian theory an input test pattern $Z$ is assigned to class $\omega_j$ if the a posteriori probability is maximum, i.e.

$$\text{assign } Z \rightarrow \omega_j \text{ if } P(\omega_j|x_1, \ldots, x_R) = \max_{k=1}^m (P(\omega_k|x_1, \ldots, x_R)) \tag{4.1}$$

rewriting the a posteriori using bayes theory we have,

$$P(\omega_k|x_1, \ldots, x_R) = \frac{(P(x_1, \ldots, x_R|\omega_k))P(\omega_k)}{P(x_1, \ldots, x_R)} \tag{4.2}$$

where $P(x_1, \ldots, x_R)$ is the unconditional measurement joint probability density. If we assume that the representations used are conditionally statistically independent, and if
we discard the denominator which is constant for all classes then equation 4.2 becomes.

\[ P(\omega_k|x_1, \ldots, x_R) = P(\omega_k) \prod_{i=1}^{R} P(x_i|\omega_k) \]  

(4.3)

Using equation 4.3 in 4.1 to obtain the decision rule we have

assign \( Z \rightarrow \omega_j \) if

\[ P(\omega_j) \prod_{i=1}^{R} P(x_i|\omega_j) = \max_{k=1}^{m} P(\omega_k) \prod_{i=1}^{R} P(x_i|\omega_k) \]  

(4.4)

or in terms of the a posteriori probabilities yielded by the respective classifiers

assign \( Z \rightarrow \omega_j \) if

\[ P^{-(R-1)}(\omega_j) \prod_{i=1}^{R} P(x_i|\omega_j) = \max_{k=1}^{m} P^{-(R-1)}(\omega_k) \prod_{i=1}^{R} P(x_i|\omega_k) \]  

(4.5)

The decision rule 4.5 which combines the a posteriori probabilities generated by the component classifiers, is the product fusion strategy. Under the assumption of equal priors it simplifies to

assign \( Z \rightarrow \omega_j \) if

\[ \prod_{i=1}^{R} P(\omega_j|x_i) = \max_{k=1}^{m} \prod_{i=1}^{R} P(\omega_k|x_i) \]  

(4.6)

If we further assume that the a posteriori probabilities computed by the component classifiers will not deviate from the prior probabilities, then we can express the a posteriori probabilities as,

\[ P(\omega_k|x_i) = P(\omega_k)(1 + \delta_{ki}) \]  

(4.7)

where \( \delta_{ki} \ll 1 \). Substituting 4.7 in 4.5 we get,

\[ P^{-(R-1)}(\omega_k) \prod_{i=1}^{R} P(\omega_k|x_i) = P(\omega_k) \prod_{i=1}^{R} (1 + \delta_{ki}) \]  

(4.8)

If we expand the product and neglect the second and higher order terms, we can approximate the right-hand side of 4.8 as

\[ P(\omega_k) \prod_{i=1}^{R} (1 + \delta_{ki}) = P(\omega_k) + P(\omega_k) \sum_{i=1}^{R} \delta_{ki} \]  

(4.9)
4.2. Theoretical Derivation of Simple Fusion Strategies

substituting 4.9 and 4.7 into 4.5 we obtain the sum fusion strategy

\[
\text{assign } Z \rightarrow \omega_j \text{ if } (1 - R)P(\omega_j) + \sum_{i=1}^{R} P(\omega_j|x_i) = \frac{m}{\max_{k=1}^{R}[1 - R]P(\omega_k) + \sum_{i=1}^{R} P(\omega_k|x_i)} \tag{4.10}
\]

which under the assumption of equal priors simplifies to

\[
\text{assign } Z \rightarrow \omega_j \text{ if } \frac{1}{R} \sum_{i=1}^{R} P(\omega_j|x_i) = \max_{k=1}^{R} \frac{1}{R} \sum_{i=1}^{R} P(\omega_k|x_i) \tag{4.11}
\]

The assumption leading to Sum is unrealistic and will lead to estimation error, however, as will be shown in the next section Sum is resilient to estimation errors. The assumption is more valid as we get closer to the boundary, while it does not hold as we move away from it. However, when we are far from the boundary no misclassification occurs and all fusion strategies perform equally. We are interested in the cases where the fusion strategies perform differently and therefore we focus our attention on the area close to the boundary.

In equation 4.11, if an outlier classifier exists it will affect the average drastically leading to incorrect combiner decisions. For such cases replacing the average with the median may lead to better fusion results. Hence the median fusion strategy is,

\[
\text{assign } Z \rightarrow \omega_j \text{ if } \text{med}_{i=1}^{R} P(\omega_j|x_i) = \max_{k=1}^{R} \text{med}_{i=1}^{R} P(\omega_k|x_i) \tag{4.12}
\]

Furthermore, the minimum and maximum simple fusion strategies can be derived from the sum and product strategies above. Initially the rules can be related to each other using the following,

\[
\prod_{i=1}^{R} P(\omega_k|x_i) \leq \min_{i=1}^{R} P(\omega_k|x_i) \leq \frac{1}{R} \sum_{i=1}^{R} P(\omega_k|x_i) \leq \max_{i=1}^{R} P(\omega_k|x_i) \tag{4.13}
\]

Which suggests that Product and Sum can be approximated by the upper or lower bounds. In 4.5 bounding the Product from above we replace the \( \prod_{i=1}^{R} \) term by the \( \min_{i=1}^{R} \) to get the minimum fusion strategy,

\[
\text{assign } Z \rightarrow \omega_j \text{ if } P^{-(R-1)}(\omega_j) \min_{i=1}^{R} P(\omega_j|x_i) = \max_{k=1}^{R} P^{-(R-1)}(\omega_k) \min_{i=1}^{R} P(\omega_k|x_i) \tag{4.14}
\]
which under the assumption of equal priors simplifies to

\[ \text{assign } Z \rightarrow \omega_j \text{ if } \min_{i=1}^R P(\omega_j|x_i) = \max_{k=1}^m \max_{i=1}^R P(\omega_k|x_i) \]  
\[ (4.15) \]

Similarly, the sum is approximated by the maximum of the a posteriori probabilities in equation 4.10 to get the maximum fusion strategy,

\[ \text{assign } Z \rightarrow \omega_j \text{ if } (1 - R)P(\omega_j) + \max_{i=1}^R P(\omega_j|x_i) = \max_{k=1}^m \max_{i=1}^R [(1 - R)P(\omega_k) + \max_{i=1}^R P(\omega_k|x_i)] \]  
\[ (4.16) \]

which under the assumption of equal priors simplifies to

\[ \text{assign } Z \rightarrow \omega_j \text{ if } \max_{i=1}^R P(\omega_j|x_i) = \max_{k=1}^m \max_{i=1}^R P(\omega_k|x_i) \]  
\[ (4.17) \]

Finally, the classifier measurement outputs can be transformed to labels, in order to be fused by the vote combiner, as follows,

\[ \Delta_{ki} = \begin{cases} 1 & \text{if } P(\omega_k|x_i) = \max_{j=1}^m P(\omega_j|x_i) \\ 0 & \text{otherwise} \end{cases} \]  
\[ (4.18) \]

Using the hard decisions of equation 4.18 in the sum strategy of equation 4.11 we get the vote fusion strategy,

\[ \text{assign } Z \rightarrow \omega_j \text{ if } \sum_{i=1}^R \Delta_{ji} = \max_{k=1}^m \sum_{i=1}^R \Delta_{ki} \]  
\[ (4.19) \]

4.3 Theoretical Expectations of Relative Performance in the Presence of Noise

In [48] and [50] Kittler et. al. have noticed that Sum outperforms Product, and therefore investigated the sensitivity of both rules to estimation errors. We will repeat the theoretical investigation which shows that Sum is less sensitive to estimation errors.
In the previous section the component classifier measurement of the a posteriori probability was defined as \( P(\omega_j|x_i) \). These classifiers actually can only estimate the a posteriori probability which we denote as \( \hat{P}(\omega_j|x_i) \). The estimate deviates from the true probability by error \( e_{ji} \), hence

\[
\hat{P}(\omega_j|x_i) = P(\omega_j|x_i) + e_{ji} \tag{4.20}
\]

Therefore, the combination strategies actually combine the a posterior probability estimates, \( \hat{P}(\omega_j|x_i) \). Substituting 4.20 into 4.5 we get,

\[
\text{assign } Z \rightarrow \omega_j \text{ if } \frac{\prod_{i=1}^{R} [P(\omega_j|x_i) + e_{ji}]}{\max_{k=1}^{m} \prod_{i=1}^{R} [P(\omega_k|x_i) + e_{ki}]} = \]

Assuming \( P(\omega_k|x_i) \neq 0 \) and that the component classifiers commit small errors, i.e. \( e_{ki} \ll P(\omega_k|x_i) \) then we can rearrange the product term as

\[
\prod_{i=1}^{R} [P(\omega_k|x_i) + e_{ki}] = \prod_{i=1}^{R} [P(\omega_k|x_i)] \prod_{i=1}^{R} \left[1 + \frac{e_{ki}}{P(\omega_k|x_i)}\right] \tag{4.22}
\]

which can then be linearised as

\[
\prod_{i=1}^{R} [P(\omega_k|x_i) + e_{ki}] = \prod_{i=1}^{R} [P(\omega_k|x_i)] \left[1 + \sum_{i=1}^{R} \frac{e_{ki}}{P(\omega_k|x_i)}\right] \tag{4.23}
\]

substituting 4.23 into 4.21 we get

\[
\text{assign } Z \rightarrow \omega_j \text{ if } \frac{\prod_{i=1}^{R} P(\omega_j|x_i)}{\prod_{i=1}^{R} P(\omega_j|x_i)} \left[1 + \sum_{i=1}^{R} \frac{e_{ji}}{P(\omega_j|x_i)}\right] = \]

Comparing 4.5 to 4.24 we notice that every term in the error free combination rule 4.5 is affected by the error factor

\[
\left[1 + \sum_{i=1}^{R} \frac{e_{ki}}{P(\omega_k|x_i)}\right] \tag{4.25}
\]
A similar analysis of the sum rule can be carried out by commencing with the application of equation 4.20 in 4.10 to get

\[
\text{assign } Z \rightarrow \omega_j \text{ if } \\
(1 - R)P(\omega_j) + \left[ \sum_{i=1}^{R} P(\omega_j | x_i) \right] \left[ 1 + \frac{\sum_{i=1}^{R} e_{ji}}{\sum_{i=1}^{R} P(\omega_j | x_i)} \right] = \\
\max_{k=1}^{m} \left\{ (1 - R)P(\omega_k) + \left[ \sum_{i=1}^{R} P(\omega_k | x_i) \right] \left[ 1 + \frac{\sum_{i=1}^{R} e_{ki}}{\sum_{i=1}^{R} P(\omega_k | x_i)} \right] \right\} \tag{4.26}
\]

Comparing 4.10 to 4.26 we notice that every term in 4.10 is affected by error factor

\[
\left[ 1 + \frac{\sum_{i=1}^{R} e_{ki}}{\sum_{i=1}^{R} P(\omega_k | x_i)} \right] \tag{4.27}
\]

Comparing the Product and Sum error factors, 4.25 and 4.27, we find that the error in 4.25 is amplified by the posterior probability estimate then summed over all classifiers. This indicates an amplification of the estimation error when using product. On the contrary the error in 4.27 is dampened if the sum of the posterior probability estimates of all classifiers is greater than 1, which is most probable for the true class. Therefore, Sum seems to dampen the estimation error.

The above analysis indicates that the existence of estimation errors would lead to Product performing worse than Sum. This is confirmed in our synthetical analysis of section 4.5. Before experimentally evaluating the fusion strategies we survey the performance of these strategies in the next section.
4.4 Comparative Survey on the Performance of Simple Fusion Strategies

In this section we survey the simple fusion strategies to see what others have found in relation to their performance.

Experimental results on identity verification data by Kittler et. al. [50] showed Sum to be the best rule followed by Median while Maximum was worse. The results on a problem of handwritten digit recognition using HMM showed Median to be the best followed by Sum while Product was the worst.

Yu et.al. [110] use Sum, Median, Difference and Weight as methods of combining three different experts. The combined experts are Hidden Markov Models (HMM), Time Delay Neural Network (TDNN), and a One Dimensional Fourier Transform (1DFT) classifier that classify spoken words from lip images. These experts have different output representations, therefore they are scaled to a value between 0 and 100, called "score". They find that Weighting is mostly the best, while Median was the best on one data set. They also used borda count and found it gives worse results.

Park and Lee [68] applied weighted Sum, Sum and Vote to HMM classifiers of a large set Korean handwritten characters. Weighted Sum, with weights that were calculated based on the performance of individual classifiers, outperformed Sum and Vote. They do not show why a weighted Sum outperforms the simple sum rule.

Li and Jain [58], apply classifier combination to the text document classification problem. The combination approaches used are: Vote, Dynamic selection and Adaptive Classifier Combination, ACC. ACC assigns a test data to the class that has the highest classification results in a local area around the test sample, over all classifiers. Their tests were applied on two sets of news items down-loaded from the Yahoo newsgroup. While combining had not always improved classification, ACC performed best. None of the combining methods was successful except when combining the naive bayes with the subspace method using ACC. They did not discuss the reason for the unsuccessful combining. It could be that the single classifiers reached the maximum possible classification rate and hence were making correlated errors. In any case they consider their
results to be satisfactory because they are comparable to the human accuracy on the same data. This example serves to uncover the fact that combining may not always work.

To recognise handwritten numerals, Pham and Yan [70] combine three back-propagation neural networks using Choquet fuzzy integral, D-S theory, Borda count, Vote and Maximum. They find that complex fusion strategies outperform the simple ones.

In a text dependent speaker verification system, Farrell [33] uses weighted Sum, Log-opinion pool which is a weighted Product and Vote to combine the posterior probability estimates of eight classifiers, four Neural Tree Networks (NTN) and four Dynamic Time Warping (DTW) classifiers. The average of four DTWs and the average of four NTNs are used as an input to the Product, Sum and Vote rules. Sum had the lowest error rate followed by Product then Vote.

In an identity verification system Kittler et. al. [52] apply Sum, Maximum, Minimum and Median rules to combine soft decision outputs of a single expert on different frames. The soft outputs are posteriori probabilities of one of two classes given an input frame. They note that the performance gains achieved by combining are initially increasing but after 3 or 4 frames no gain is achieved. In the existence of outliers they find that order statistic combiners, such as Median and Maximum seem to outperform Sum. In the absence of outliers Sum performed best. This is consistent with the findings of Tumer and Gosh [99].

Skurichina and Duin [89] Combine LDA experts using bagging and boosting methods. The results indicate that generally Product and Sum are better than weighted vote. The choice of the fusion rule is not critical for boosting, while Product degrades at small sample size.

Tax et.al. [95] compare Sum and Product under different number of classes, different feature sets and rejection rates. They find that Product outperforms Sum when large number of classes exist using gaussian experts. For two classes both perform similarly. For Fisher Linear Discriminant classifiers both perform equally. When using neural network experts a lot of noise exist at the classifier outputs, and Sum outperforms Product.
Procter and Illingworth [71] combine two HMMs that classify handwritten text, using Product, Sum, Borda count, BKS, in addition to Column top k and row top k. The last two, as the name may indicate, take into consideration only the top k results of an expert, either the column or row expert. Their results show that Product performs best followed by Minimum. Maximum, Sum and BKS follow close to each other. Row and Column are worse. We believe the reason they achieved good results using Product and Minimum is that the component classifiers were confident enough such that they did not produce close to zero estimations of the true class.

Finally, Xu et al. [109] have used the D-S theory to combine handwritten recognition experts. They found it to be robust and superior to voting, especially when high reliability is required.

The above survey indicates that no single fusion strategy outperforms the rest on all occasions. Therefore, in the next section and two chapters, we investigate the simple strategies, find when and why they outperform or underperform and find methods of improving their performance where possible.

4.5 Experimental Study of Fusion Strategies Using Synthetic Experts

In this section we investigate the classifier combination models presented in Section 4.2, and validate them experimentally. We emulate the behaviour of individual experts by subjecting their nominal soft outputs to perturbation errors. A relation between the effectiveness of various combination strategies and estimation errors is established.

Different combination strategies have been compared under different conditions. In particular, the Sum, Product, Minimum, Maximum, Median and majority Voting strategies are considered. The comparison of performance is made for different number of classes, varying number of experts, varying noise levels and the results are parameterised by different posterior probabilities. Two noise generators are used, Normal and Uniform, in order to compare the sensitivity of the different combination strategies to different noise distributions. Also to test classifier performance without combination,
the misclassification rate of all experts is measured and averaged over the number of experts used. This result will be referred to as the single expert.

4.5.1 Experimental Methodology

In each experiment, experts estimate the actual posterior probability of classes. Thus, tests are run for a single point in the class aposteriori probability space at a time. This operating point is fixed by selecting a value $P$ of the aposteriori probability for class one from the interval $(1/m$ to $1)$, where $m$ is the number of classes. For each point experts' estimates are considered to be the posterior probability plus noise. Noise values are samples taken from a noise generator that has a zero mean. For each point the tests are repeated for 10 different levels of noise. For gaussian noise we vary the standard deviation $\sigma$ linearly from 0.1 to 1. For uniform noise the support domain of $2b$ is varied by setting $b$ to values in the interval ranging from 0.1 to 1. When the addition of noise values to $P$ results in an expert estimate that falls above one or below zero we clip this experts' output to one or zero, respectively.

The experiments mentioned above are carried out for two class and three class problems using 3 experts and 8 experts. In Figures 4.1 and 4.3 the results of different combiners are connected by a line. This is done for display purposes, and only to simplify the comparison of the rules at each instance of $\sigma$ or $b$.

4.5.2 Comparison of Strategies Under Uniform Noise

For the Uniform noise generator with a support interval $[-b, b]$, we find that there is a boundary across which the relative performance of the rules changes. This boundary is at $b = P + 0.1$. Product and Minimum rules have a superior performance when values of $b$ are below $P + 0.1$. However, for values of $b$ above $P + 0.1$ their performance ranking suddenly switches and they become the worst combination rules (see Figure 4.1). The Sum rule does not seem to be qualitatively affected by the boundary. It shows a linear response with respect to $b$, (i.e. error rate increases as $b$ increases). Its performance is second best to the Product and Minimum rules for $b < P + 0.1$, but when $b > P + 0.1$, it becomes the best, (see Figure 4.2). The Maximum rule
Table 4.1: Comparison between combining strategies performances using 3 experts, under uniform noise

<table>
<thead>
<tr>
<th>Ranking of strategies for 2-class problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case I: $b &lt; P + 0.1$</td>
</tr>
<tr>
<td>1) Product, Min &amp; Max</td>
</tr>
<tr>
<td>2) Sum</td>
</tr>
<tr>
<td>3) Median &amp; Vote</td>
</tr>
<tr>
<td>4) Single expert</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ranking of strategies for 3-class problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case I: $b \leq P$</td>
</tr>
<tr>
<td>1) Prod, Min</td>
</tr>
<tr>
<td>2) Sum</td>
</tr>
<tr>
<td>3) Median &amp; Max</td>
</tr>
<tr>
<td>4) Vote</td>
</tr>
<tr>
<td>5) Single expert</td>
</tr>
</tbody>
</table>

performance is similar to Product and Minimum in the two class problem. The Median and Vote rules performances are identical and always rank next to Sum. The Single expert shows the worst performance at approximately $b \leq P + 0.1$. Sum and Median are always better than the single expert. When 3 classes are involved, the threshold value at which the different cases occur changes a little. No figures are reported for the 3 class case as the overall results are similar to the 2 class case with the exceptions indicated in table 4.5.2. Also for the 3 class case, the Median and Maximum rules performance relative to the other strategies changes. Median always seems to improve, while Maximum deteriorates for $b < P + 0.1$ and improves for $b > P + 0.1$. The single expert exhibits a similar behaviour to the two class case by becoming the worst at approximately $b \leq P + 0.2$, except at $P = 0.4$ (close to the boundary) where Vote showed the worst performance. At $P > 1/m$ Sum, Median and Maximum are always better than the single expert.
Figure 4.1: Comparison between the combination rules for a fixed a-posteriori probability value ($P = 0.6$), and different uniform noise distributions. Figures (a) and (b) are for 3 and 8 experts respectively, using 2 classes.

Figure 4.2: Comparison between combination rules for different a-posteriori probability values and fixed uniform noise interval at $b = 0.8$. Figures (a) and (b) are for 3 and 8 experts, respectively, using 2 classes.
4.5. Experimental Study of Fusion Strategies Using Synthetic Experts

Increasing the number of experts: When we increase the number of experts, the boundary is shifted to $b = P$. Note also that the performance of Sum and Median improves, whereas Product and Minimum become more sensitive to values of $b$. This sensitivity is reflected in a performance improvement when $b < P$, giving close to zero errors and the best performance overall. In contrast, a dramatic deterioration is observed for $b > P$ when Product and Minimum give unacceptably high error rates and become the worst combination strategies. For the two class case, the performance of Maximum is similar to Product. On the other hand, for the three class case, the performance of Maximum improves with an increasing number of experts when $b < P$ and at lower values of $b$ above $P$, but at higher values of $b$ above $P$ it acts similarly to Product and Minimum. The performance of Vote is more complex. For the three class case it improves, but for the two class case it degrades at lower values of $P$, improves at higher values of $P$ and stays the same at medium values of $P$. In general, the performance of all strategies improves with an increasing number of experts. When 8 experts are used, the performance of the single expert adheres more consistently to the regimes identified above: The approximate boundaries become exact, with no overlap. Single expert is worse than any other rule when $b \leq P$, irrespective of the number of classes.

4.5.3 Comparison of Strategies Under Gaussian Noise

For the gaussian noise generator there is also a constant boundary across which the performance of rules changes qualitatively. The boundary was found at standard deviation $\sigma = C$, where $C$ is a constant depending on the number of classes and classifiers used. In the two class problem, for small values of standard deviation $\sigma$, all strategies yield similar results, but above a certain value $C$ of standard deviation, the performance of the Product, Maximum and Minimum strategies becomes worse than that of the others. $C$ is 0.4 in the two class experiments (see Figure (4.3) and (4.4)).

The single expert is worse than any other rule for $\sigma \leq 0.5$, above which Product, Maximum and Minimum become worse than the single expert.

When we increase the number of classes, the location of the boundary line drops to $C =$
Table 4.2: Performance comparison in the three class case under the gaussian noise generator.

<table>
<thead>
<tr>
<th>When $\sigma \leq C$</th>
<th>Best Strategy</th>
<th>When $\sigma &gt; C$</th>
<th>Best Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product &amp; Minimum</td>
<td>Sum &amp; Median</td>
<td>$\sigma &gt; C$</td>
<td>Sum &amp; Median</td>
</tr>
<tr>
<td>Sum &amp; Median</td>
<td>Maximum &amp; Vote</td>
<td>Single expert</td>
<td>Maximum &amp; Vote</td>
</tr>
<tr>
<td>Maximum &amp; Vote</td>
<td>Single expert</td>
<td>Product &amp; Minimum</td>
<td></td>
</tr>
</tbody>
</table>

Again for $\sigma > C$, Sum and Median are the best, while Product and Minimum are the worst. Vote and Maximum interchange positions at different values of the standard deviation. At higher standard deviations, Vote becomes better than Maximum, while at lower standard deviations Maximum becomes better than Vote. This value of standard deviation depends on $P$, but the dividing line was found to be approximately $\sigma = 0.7$. For values of standard deviation below $C$, the similarity of the behaviour of the strategies noted for the two class case disappears. Product and Minimum become the best rules, followed by Sum and Median (see table 2). The Single expert is worse than any other rule for $\sigma < 0.4$. This value increases as $P$ increases, leading to a degradation of its performance rank relative to the other rules. At low $P$ its performance is next to Sum and Median for $\sigma \geq 0.4$. As $P$ increases, it drops below Vote. At even higher values of $P$ it further drops below Maximum, for $\sigma \geq 0.6$

Increasing the number of experts: As the number of experts increases the value of $C$ decreases to 0.1 for the two class case and 0.0 for the three class case. This means the region in which the Product, and Minimum strategies perform best shrinks rapidly. It should also be noted that although for the majority of rules the performance improves only at values of standard deviation less than 0.4, the performance of Sum and Median improves the most, even at values of standard deviation equal to and above 0.4. For both the two class and three class problems, the single expert is worse than any other rule at lower values of $\sigma$. It is worst at $\sigma \leq 0.3$ for $P \leq 0.7$ and at $\sigma \leq 0.4$ for higher values of $P \geq 0.8$.

It is worth mentioning that the value of the constant $C$ at which the boundary exists.
4.5. Experimental Study of Fusion Strategies Using Synthetic Experts

Figure 4.3: Comparison between combination rules for fixed aposteriori probability value of 0.6, and varying standard deviation of Gaussian noise. Figures (a) and (b) are for 3 & 8 experts, respectively, using 2 classes.

Figure 4.4: A comparison of the combination rules parameterised by different aposteriori probability values for Gaussian noise with standard deviation $\sigma = 0.3$. Figure (a) is for 3 experts while Figure (b) is for 8 experts.
for gaussian noise, is explicit for our experiments and may vary for other experiments.

4.5.4 Contradicting Experts

In order to test the performance of the rules under severe conditions, the decision of one of the experts is flipped such that it always contradicts the other experts' decisions. This is done by considering its class 1 output as class 2 and vice versa.

Although the overall performance of all rules degrades due to the experts contradicting one another, the Sum and Median rules show a stable and linear performance at different degrees of noise. They are able to maintain their superiority. The single expert also shows a stable and linear performance. Due to space limitations no figures are reported for experiments involving contradicting experts.

Contradicting experts under Uniform noise

In the combination of low $b < 0.6$ and high $P$, Median is the best. However Sum is the best or equal to Median at almost all other values of $P$ and $b$. Maximum, Minimum and Product show a hyperbolic performance. They are the worst rules at $b > 0.5$ and also when $b < 0.5$ and $P \geq 0.7$. Product becomes better than Minimum and Maximum when $b < 0.4$. The single expert performance falls between the Minimum group and the Sum group. As the number of experts increases to 8, the performance of the rules improves. Sum becomes far better than Median, while Median is better than Vote. Product, Minimum and Maximum are the worst performers, except at $b \leq .6$ and $P = .6$ when they become the best. The single expert is next to the Vote at high values of $P$ and better than Vote at low values of $P$; if $b > .4$. Otherwise Vote is better.

With the number of classes increasing to three, Sum is generally still the best but Median and Vote are better at $b \leq .3$. Minimum, Product and Maximum are the worst rules, although Maximum becomes the best at $P \leq .5$ and $b \leq .8$. The single expert is always worse than Sum and Median, and also worst than Vote if $P \geq .7$. When the number of experts is increased to 8, Maximum never becomes the best. On the other hand Sum is always the best followed by Median and then Vote. At $b \leq .6$ and $P \leq .6$
Maximum becomes better than Vote. The single expert is worst than any other rule except for Product and Minimum. At \( b \geq .8 \) it is also better than Maximum.

In summary, the degrading effect of a contradicting expert at lower values of \( b \) is larger than at higher values of \( b \). Interestingly, the effect at higher \( P \) is larger than at lower \( P \). In general, the relative performance of the rules does not change except below \( b < P \), where Product, Minimum and Maximum which used to be the best rules, suddenly become the worst. The same degradation pattern is observed as the number of experts increases. Our results seem to confirm the findings by Turner and Gosh [102] that Median performs best when outliers, represented by the contradicting experts, exist. However this is true only at low levels of noise. At higher levels of noise Sum is better.

**Contradicting experts under Normal Noise**

The relative performance of the rules is similar to the uniform noise case. Interestingly, when the number of experts increases to 8, the Product group is never better than Sum.

With the number of classes increasing to three, the relative performance of the strategies is similar to the uniform noise problem. Sum and Median are the best rules. The single expert and Vote are very close but Vote is better at \( P \geq 0.6 \). Maximum becomes better than Product and Minimum, but it is worse than the single expert especially at higher values of \( b > 0.3 \). Increasing the number of experts to 8 makes Vote better than the single expert, as was the case in the two class problem.

In summary Product, Minimum and Maximum show a larger percentage of degradation, as compared to other rules, especially at low \( \sigma \) and high \( P \). This suggests that when a confident contradicting expert exists higher misclassification rates occur, even if most of the non-contradicting experts are also confident.

### 4.5.5 Improving The Performance

By investigating the reasons for the degraded performance of the product and minimum rules, we were able to establish that most of the errors committed by these rules were
when at least one expert vetoed the rest of the experts by outputting a zero. We noticed that Product and Minimum were affected most. The results were as follows: at standard deviation $> 0.4$ more than 50% of errors were due to an expert outputting a zero, and the percentage increased with the increase of $\sigma$. At 0.4 the percentage of error due to zero outputting experts was around 50% of total errors committed, and at standard deviation $< 0.4$ less than 50%. The vote rule had only 20% of its errors as zero outputs. Hence we do not expect an improvement in its performance by our method. In order to remove the veto effect one has to eliminate, or replace, the decisions of the vetoing expert.

Tests were performed where experts outputs below a threshold were replaced by the threshold. As the value of the threshold increased the amount of improvements increased but also the amount of degradation also increased. Detailed analysis and experimentation results of the modified product strategy are presented in the next chapter.

4.6 Conclusion

In this chapter we have focused our attention on the commonly used simple fusion strategies Product, Minimum, Sum, Median, Maximum and Vote, which were derived based on a bayesian theoretical framework. Also, a sensitivity analysis of the fusion strategies to estimation errors is presented.

The literature survey has indicated that the simple fusion strategies perform differently under different experiments. This motivated us to conduct a synthetic experiment of the different scenarios that these rules may encounter. We focused our attention on a single point in the class a posteriori probability space at a time and emulated the behaviour of individual experts by subjecting their nominal soft outputs to perturbation errors. Our approach permitted us to explore the effect of estimation error correlations.

For the conditions that we had simulated, we noticed a specific trend in the performance of the simple fusion strategies investigated. We have shown experimentally, that there is a noise level boundary across which the relative performance of classifier combination
rules changes. For experts affected by uniform noise, it was found that for \( b \leq P + 0.1 \) the Minimum and Product performance was the best, while for \( b \geq P + 0.1 \) the Minimum and Product performance degrades causing Sum to become the best rule. Increasing the number of experts does not have any effect on the relative performance, but all the strategies in general exhibit a better performance. Minimum and Product become more sensitive to values of \( b \).

For experts affected by gaussian noise, the boundary is constant regardless of the value of \( P \). It changes (decreases) only when the number of experts or classes increases. It actually reaches zero when the number of classes increases to three and the number of experts grows to 8. For values of the standard deviation below the boundary all rules perform similarly, while above the boundary the Product, Minimum and Maximum deliver the worst performance.

Our experiments were conducted for a single point in space. However, since the relative performance of the fusion rules stayed the same for all such points, the same conclusions hold for the average error.

In the presence of a contradictory expert, the performance of all rules degrades, regardless of the type of noise used. Sum and Median show a stable performance and are better than the single expert. Product, Maximum and Minimum have a hyperbolic performance (i.e. obtain large error rates when \( P \) is small or large) and worse than the single expert most of the time, however Product gives a better performance at lower levels of noise. The Vote performance is similar to Sum when three experts are fused and the performance falls in between Sum and Product when the number of experts increases to 8. In general, all the results prove the combiners to be better than the single expert, especially Sum and Median. However, the single expert may be preferable over the Product, Minimum and Maximum with \( \sigma > 0.5 \) under gaussian noise estimation error. The results also confirm the theoretical prediction that in most scenarios the sum rule outperforms Product and strategies devised from it.

When contradicting experts exist Sum and Median seem to be affected less than the other rules. For low noise Median seems to deal best with outliers however Sum is better when high noise levels exist. The contradicting expert degrades the fusion strategies
more when the experts, including the contradicting one, are confident.

The theoretical derivation of the fusion strategies have shown that Product is directly derived from the bayesian formalism under the assumption of independence, while Sum is derived from Product under further restricting assumptions. Based on this fact Product was expected to outperform Sum. However, as the noise sensitivity analysis indicated Sum is more resilient to noise and can outperform Product under tasks involving high noise. This was shown to hold true in the synthetic experiments of Section 4.5.

The empirical study confirmed the sensitivity of Product to noise. Moreover, the results have shown that Product dramatically drops at a certain boundary which depends on the margin. On the other hand Sum exhibits a linear relation between performance and the amount of noise. The theory does not indicate the existence of the nonlinearity and the boundary at which Product changes dramatically. Maximum, Median and Vote were all derived from Sum with more restrictions. Hence were expected to exhibit similar or weaker performance. Minimum was derived from Product and was expected to perform equally. In the synthetical experiments Sum and Product did outperform their derivatives for the noise types and experiments of this chapter. However, the order statistic combiners may outperform Sum and Product when outliers exist. The contradicting expert experiments showed Median to slightly outperform Sum however it falls below Sum as the amount of noise increases.

The experimental study showed that Product drops sharply at the boundary due to some expert outputs reaching the ends of the range of the probability estimate values. This will lead to vetoing experts which strongly lead to classification errors. The noise sensitivity analysis and the theoretical derivation of Product did not account for estimation errors due to the veto effect and hence the theory did not predict the degradation of Product under veto conditions. Therefore, the theory was not able to forecast the sharp drop of Product at the boundary.
Chapter 5

Alleviating the Veto Effect

A heuristic and a theoretical methods for improving Product are introduced and experimentally validated.

5.1 Introduction

In this chapter we shall focus on the decision probability level fusion in general and on the product rule in particular. The product rule, which combines the multiple expert outputs by multiplication plays a prominent role because of its theoretically sound basis in probability calculus [50, 48] as pointed in Chapter 4. It is the proper fusion strategy when combining the outputs of experts utilising distinct (statistically independent) signal representations. It is also the optimal operator for combining the outputs of experts responding to an identical stimulus, under the assumption that the experts have been designed using statistically independent training sets. In spite of its theoretical underpinning, in our experimental studies in the previous chapter Product was shown to be outperformed by the less rigorously founded sum rule. This was also reported by Tax and Duin [96]. The inferior performance was attributed to the veto effect. If estimation errors drive one of the class aposteriori probability estimates to zero, the output of the product fusion will also be zero, even if other experts provide a lot of support for the class. This severity of the product fusion strategy has motivated our research in Section 4.5. This research has led to this chapters development of a
heuristic modification of the classifier outputs before the product fusion is carried out. The advocated MProduct which stands for Modified Product is presented in Section 5.3. The idea is to set a threshold on the class posteriori probability and if an actual estimate falls below the threshold, it will be replaced by a constant. We show in a number of experiments that this modification significantly improves the performance of Product and raises it above the sum rule and the related strategies. Section 5.3 is devoted to a detailed experimental evaluation of the proposed strategy. First we report the results of a comparative experimental study involving the synthetic experts presented in Section 4.5. Then, we experiment with a committee of $k-NN$ classifiers obtained by bagging. The experiments are performed on real data available in the public domain.

In Section 5.4 we argue that estimation errors are often caused by small sample problems. We show that by taking small sample effects into account we can develop a formula for correcting the outputs of individual experts, provided the sampling distribution is known and can be incorporated as a Bayes prior. We introduce the concept of classifier output moderation. Then, we focus on the $k-NN$ decision rule and derive the formula for correcting the outputs of $k-NN$ experts. Incidentally, Product is affected by small sample problems even when the size of the training set is large, as each decision is made by drawing a small number of samples from the training set. We then validate our correction formula experimentally on synthetic and real data sets. We demonstrate that Product using moderated outputs of multiple $k-NN$ classifiers strongly outperforms the product fusion of raw classifier outputs. Finally, we compare the proposed scheme with the heuristic MProduct and show that they are quite similar in performance. The former has the advantage that the modification formula is very simple and adaptive to the number of nearest neighbours used by the decision rule. The latter has the advantage that the correction procedure is effective, even when the assumptions behind the $k-NN$ rule break down, i.e. when the training set is very small.

Experiments of this chapter involve a repeat of the synthetic experts experiments of the previous chapter, as well as real data experiments. In the next section we will only introduce the experimental methodology for the real data experiments. The synthetic
experts experiments can be referred to in Section 4.5.

5.2 Setup of Experimental Study Using $k$-NN Experts

5.2.1 Disclaimer

To validate the results obtained using synthetic experts we combine $k$-NN experts designed by bootstrapping, which are then aggregated using the fusion method under study. In a bagging scenario all component experts are estimating the same true posterior probability, as in the synthetic experiments. However, these experts are no longer completely independent. They start to become independent as we decrease the training set size. Also, in the synthetic experiments we focused our attention on a single point in the probability space. In the real experiments we are no longer able to give results for a single point because we are averaging the results for many test samples.

5.2.2 Experimental Methodology

For experiments involving each data set a single training set is randomly taken from the original sample space, i.e. the full data set. The $k$-NN classifier built using this original learning set is referred to as the single expert. The remaining samples are used as a test set. Using the learning set, 25 boot sets are generated, by bootstrapping. The decision of the 25 boot sets are aggregated to classify the test set. These results are referred to as the bagged expert results. We compare these results to those obtained from the single expert, and to those obtained from other bagging methods. The above is repeated for four training set sizes. The sizes used were 10, 20, 40, and 80 samples.

We investigate the performance of the four methods of creating bootstrap sets for two types of learning sets. In the first case the learning set is created by randomly taking samples from the full data set. This results in a set that may contain samples from all classes with a population bias towards a certain class. The second type of learning set is referred to as a modified learning set. It is constructed using Method 3 which was mentioned as a technique to create unbiased bootstrap sets in Section 3.2.5. This
results in a set that is representative of all the classes, with class population ratios similar to those of the full set. The modified learning set simulates an unbiased sample space.

All experiments are repeated 100 times and we average the error rates by dividing by the number of repetitions.

### 5.2.3 Calculation of Errors

To find the misclassification error rate, a test sample is presented to the $k - NN$ classifier, where $k$ is the closest integer of $\sqrt{n}$ and $n$ is the number of samples in the learning set. The class posterior probabilities $P(\omega_i|x)$ for each test sample are estimated as:

$$P(\omega_i|x) = \frac{\kappa_i}{k} \tag{5.1}$$

where $\kappa_i$ is the number of neighbours from the $i$-th class among $k$. The test sample is assigned a class label that corresponds to the largest posterior probability. If the original label of the sample is found to be different from the assigned label the error counter is incremented. This is repeated for all samples in the test set. After presenting all test samples the error counter is divided by the number of test samples used, in order to get the misclassification rate.

To find the error rate of the combiners, the a posteriori probability of each test sample is found using each of the 25 boot sets, as was done in the training set case. Next, the resulting 25 posterior probabilities per class are combined using the combination strategy under investigation. Hence, using the 25 pairs of a posteriori probabilities, each of the combination rules gives its estimate of the final posteriori probability value. The resulting posterior probability determines the label of the test sample and any error is recorded.

### 5.2.4 Comparative Measure

In order to determine whether a certain method outperforms a baseline method we perform a significance check. This is done by calculating the degree of improvement
Table 5.1: Data sets used and the number of samples available in each data set.

<table>
<thead>
<tr>
<th>Data Name</th>
<th>No. of samples</th>
<th>No. of features</th>
<th>No. of classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic</td>
<td>1232</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Seismic</td>
<td>300</td>
<td>25</td>
<td>3</td>
</tr>
<tr>
<td>Diabet</td>
<td>768</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Breast cancer (BCW)</td>
<td>699</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>34</td>
<td>2</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>Iris</td>
<td>150</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>


over the single expert trained using a particular learning set. Accordingly for each method of creating bootstrap sets we have a fusion or bagging classification rate, \( f \), that is compared to the learning set performance, \( l \). The degree of improvement is measured as

\[
s = \frac{f - l}{100 - l} \times 100
\]  

(5.2)

If the improvement exceeds 5 percent we consider it as significant. This value is calculated for all four bagging methods and each of the two learning set types under varying set sizes.

We adopt two criteria to assess each Bagging method. As the first criterion we measure the degree of improvement, \( s \), of each method at each set size. Then at each set size we rank the methods according to these improvement measures and the best method is assigned rank 1, while the worst is rank 4. Next we sum the number of times a method holds each rank, over all data sets and sizes. The second criterion counts the number of times a method achieves an improvement or degradation, i.e. above 5% or below -5%.

The actual fusion classification rate \( f \) can be obtained from the tabulated results using the single expert performance rates as a reference.
Figure 5.1: Sample distribution for the Synthetic data set

5.2.5 Data Sets

Both synthetic and real data sets were used in our experiments. Synthetic data was chosen to carry out controlled experiments for which the achievable recognition rate is known. The computer generated data is two dimensional involving two classes. The two class densities have an overlap area which was designed to achieve an instability of the class boundary. The feature in the first dimension was drawn from a uniform distribution. The feature of the second dimension has a density that is uniform in a non overlapping area, and a ramp in an overlapping area as shown in Figure 5.1. The densities of the two classes have a mirror symmetry. The ramp was approximated using a uniform random number generator, by cutting the ramp into very small bins. The theoretical Bayes error of this data set is 6.67%. Using the generated samples the empirical Bayes error was found to be 6.82%. Most of the real data used were the standard sets obtained from the UCI repository [17]. The exception is the seismic data set made available by Shell. Table 5.1 summarises the essential information on these data sets.
5.3 Modified Product

We propose a novel method of combining classifier outputs that alleviates the veto effect. The veto effect in the Product fusion is caused by small classifier measurement output values dominating the product, i.e., giving a close to zero result. This commonly occurs when noise levels are high. The basic idea is to modify the output of an expert if it falls below a specified threshold. For all experts, we examine the output for one class at a time. Hence for an expert, its estimate of the probability of one class may change while its estimates for other classes may stay the same. No normalisation is made at this point.

5.3.1 The Modified Product Procedure

MProduct fusion is formally defined as follows:

For each class \( \omega_i \), the expert outputs that fall below the threshold value are modified by setting their output to the threshold value.

Thus in MProduct the \( j^{th} \) expert output \( P_j(\omega_i|x) \) which estimates the a posteriori probability for class \( \omega_i \) given pattern vector \( x \) is modified before entering the product fusion rule as follows:

\[
\tilde{P}_j(\omega_i|x) = \begin{cases} 
  t & \text{if } P_j(\omega_i|x) \leq t \\
  P_j(\omega_i|x) & \text{if } P_j(\omega_i|x) > t 
\end{cases} \tag{5.3}
\]

In the following subsections we present the results of several experiments which demonstrate the merits of the proposed modification. We also investigate the sensitivity of the performance of MProduct to the prespecified truncation threshold.

5.3.2 Experiments with Synthetic Experts

The aim of this experiment was to investigate and experimentally evaluate the proposed MProduct fusion strategy in a controlled situation when the true a posteriori probabilities for the respective classes are known. We emulate the behaviour of the individual
experts by subjecting the known a posteriori class probabilities to perturbation errors drawn according to different probability distributions.

The experiment follows closely the experimental study of the sensitivity of simple fusion strategies to expert estimation errors which has been reported in Section 4.5. The study involved all the classifier combination schemes discussed in [50, 48] and confirmed the superiority of the Sum rule over Product. Here we use the results obtained with these two rules as a reference to assess the proposed MProduct.

MProduct is evaluated for different truncation threshold values. The experimental results obtained for 8 experts under uniform distribution noise for varying threshold values when $P(\omega|x) = .6$ are shown in Figure 5.2. The threshold values tested are: $5 \times 10^{-128}$, $5 \times 10^{-64}$, $5 \times 10^{-32}$, $5 \times 10^{-14}$, $5 \times 10^{-8}$, $5 \times 10^{-4}$, .005, .05, and .1.

We note that MProduct's performance improves as the threshold is decreased. However, it remains constant between $5 \times 10^{-8}$ and $5 \times 10^{-64}$, before it starts to degrade at $5 \times 10^{-128}$. This is true for both cases tested (3 and 8 experts). Interestingly for
5.3. Modified Product

Table 5.2: Simulated Experts: Number of test cases in which fusion strategies give the best or the jointly best performance. The results correspond to MProduct threshold value of $5 \times 10^{-8}$.

<table>
<thead>
<tr>
<th>Noise</th>
<th>Experts</th>
<th>Sum</th>
<th>Mprod</th>
<th>Prod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>3</td>
<td>29</td>
<td>32</td>
<td>9</td>
</tr>
<tr>
<td>Gaussian</td>
<td>8</td>
<td>31</td>
<td>31</td>
<td>8</td>
</tr>
<tr>
<td>Uniform</td>
<td>3</td>
<td>22</td>
<td>49</td>
<td>40</td>
</tr>
<tr>
<td>Uniform</td>
<td>8</td>
<td>25</td>
<td>49</td>
<td>40</td>
</tr>
<tr>
<td>Total</td>
<td>107</td>
<td>161</td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>

gaussian noise the variance of the results obtained under different threshold values is relatively low. Overall, the largest threshold with the best performance was found to be approximately $5 \times 10^{-8}$. The reason for selecting the largest threshold from the range of $5 \times 10^{-8}$ to $5 \times 10^{-64}$ is that at some instances very large thresholds, for example 0.1, yield the best performance. Comparing MProduct to Sum and Product at this threshold we find in table 5.2 that MProduct is the best in a significant number of cases.

For all posterior probability values we notice MProduct outperforms Sum. There is a trend which is apparent from Figure 5.3. We notice that at low levels of noise all rules perform equally because all experts are classifying perfectly without errors. As the noise is increased there is an interval, especially for uniform noise, for which Product and MProduct perform equally and better than Sum. Then at even higher noise levels Product deteriorates severely and MProduct becomes the best fusion rule. When a threshold larger than $5 \times 10^{-8}$ is used, occasionally an interval at an intermediate noise level emerges where Product outperforms all rules. This interval is very small and therefore does not appear in all the repetitions of our experiments for the varying parameters (noise type, posterior probability value, number of experts). This level is right below the boundary $b$, above which Product’s performance deteriorates. In Section 4.5 this boundary was found to be at $b = P(\omega|x) + .1$, for uniform noise.

In general we found that MProduct, relative to Product, is the best rule twice as often.
Figure 5.3: Comparison between the performance of the fusion strategies for $t = 5 \times 10^{-8}$. On the x-axis, the symbols Sm, Mp, Pd and xp represent Sum, MProduct, Product and single expert, respectively. The y-axis represent the standard deviation, $\delta$, for normal noise and half the support domain, $b$, for uniform noise. The top row is for normal noise while the bottom row is for uniform noise.

In most cases, when MProduct outperforms Product, the improvement is substantial. When Product is superior, the accuracy gains are marginal. Thus MProduct represents a significant improvement over Product. Considering the number of repetitions, i.e. 750 for the synthetic experiment and 100 for the bagging experiments, we believe the results are statistically significant. We also note that MProduct is robust to changes in noise type.

5.3.3 Experiments with $k-NN$ Classifiers

We validate the results obtained in the previous subsection using simulated experts on the synthetic data by experiments involving real experts and real data outlined in Section 5.2.
### 5.3. Modified Product

Table 5.3: MProduct classification rate obtained using regular training set and bagging method 4

<table>
<thead>
<tr>
<th>Set size</th>
<th>Data set</th>
<th>Number of nearest neighbors</th>
<th>Data set</th>
<th>Number of nearest neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Seis.</td>
<td>96.38</td>
<td>95.22</td>
<td>90.46</td>
</tr>
<tr>
<td>2</td>
<td>98.23</td>
<td>98.08</td>
<td>97.57</td>
<td>96.78</td>
</tr>
<tr>
<td>3</td>
<td>99.48</td>
<td>99.23</td>
<td>99.05</td>
<td>98.85</td>
</tr>
<tr>
<td>4</td>
<td>99.75</td>
<td>99.70</td>
<td>99.63</td>
<td>99.55</td>
</tr>
<tr>
<td>1</td>
<td>Wine</td>
<td>83.67</td>
<td>82.64</td>
<td>78.22</td>
</tr>
<tr>
<td>2</td>
<td>90.13</td>
<td>89.75</td>
<td>89.16</td>
<td>87.90</td>
</tr>
<tr>
<td>3</td>
<td>92.76</td>
<td>92.91</td>
<td>92.86</td>
<td>93.09</td>
</tr>
<tr>
<td>4</td>
<td>94.18</td>
<td>94.21</td>
<td>94.10</td>
<td>94.04</td>
</tr>
<tr>
<td>1</td>
<td>Iono.</td>
<td>69.57</td>
<td>67.86</td>
<td>66.21</td>
</tr>
<tr>
<td>2</td>
<td>73.51</td>
<td>72.22</td>
<td>71.13</td>
<td>69.90</td>
</tr>
<tr>
<td>3</td>
<td>78.74</td>
<td>77.06</td>
<td>75.95</td>
<td>74.54</td>
</tr>
<tr>
<td>4</td>
<td>81.94</td>
<td>81.32</td>
<td>80.61</td>
<td>79.53</td>
</tr>
</tbody>
</table>

Table 5.4: Degree of improvement of MProduct over Sum using $t = 5 \times 10^{-8}$

<table>
<thead>
<tr>
<th>Set size</th>
<th>Data set</th>
<th>Number of nearest neighbors</th>
<th>Data set</th>
<th>Number of nearest neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Seis.</td>
<td>0</td>
<td>34.55</td>
<td>41.34</td>
</tr>
<tr>
<td>2</td>
<td>0.56</td>
<td>5.81</td>
<td>5.36</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>1.95</td>
<td>3.13</td>
<td>0.99</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>2.90</td>
<td>2.38</td>
<td>6.67</td>
<td>0.00</td>
</tr>
<tr>
<td>1</td>
<td>Wine</td>
<td>0.40</td>
<td>11.48</td>
<td>15.89</td>
</tr>
<tr>
<td>2</td>
<td>0.13</td>
<td>3.00</td>
<td>5.52</td>
<td>13.25</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>-1.98</td>
<td>-0.41</td>
<td>0.42</td>
</tr>
<tr>
<td>4</td>
<td>0.00</td>
<td>0.70</td>
<td>1.20</td>
<td>0.68</td>
</tr>
<tr>
<td>1</td>
<td>Iono.</td>
<td>0.00</td>
<td>0.57</td>
<td>0.04</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>0.13</td>
<td>0.74</td>
<td>0.77</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>-0.27</td>
<td>0.57</td>
<td>0.89</td>
</tr>
<tr>
<td>4</td>
<td>0.00</td>
<td>0.43</td>
<td>1.57</td>
<td>0.16</td>
</tr>
</tbody>
</table>
Table 5.5: Degree of improvement of MProduct over Product using $t = 5 \times 10^{-8}$

<table>
<thead>
<tr>
<th>Set size</th>
<th>Data set</th>
<th>Number of nearest neighbours</th>
<th>Data set</th>
<th>Number of nearest neighbours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Seis.</td>
<td>62.22 36.81 8.08 0.23</td>
<td>DCW</td>
<td>44.86 33.49 17.55 6.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76.38 70.21 53.83 29.55</td>
<td></td>
<td>50.61 43.75 29.09 15.81</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>84.51 72.05 56.79 41.63</td>
<td></td>
<td>47.00 42.24 26.78 14.29</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>77.22 67.63 56.38 40.96</td>
<td></td>
<td>47.05 40.98 26.28 11.69</td>
</tr>
<tr>
<td>1</td>
<td>Wine</td>
<td>50.28 40.02 20.18 9.31</td>
<td>Iris</td>
<td>65.75 53.20 23.56 5.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64.37 56.41 42.80 27.00</td>
<td></td>
<td>72.47 64.71 52.33 33.14</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>62.26 54.38 40.60 27.58</td>
<td></td>
<td>67.17 60.84 48.67 35.03</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>61.64 53.29 39.67 24.94</td>
<td></td>
<td>65.12 57.77 46.71 31.01</td>
</tr>
<tr>
<td>1</td>
<td>Iono.</td>
<td>11.38 5.70 0.35 -1.17</td>
<td>Synth.</td>
<td>35.65 26.43 12.09 1.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22.34 16.60 11.43 5.97</td>
<td></td>
<td>28.51 21.19 11.32 5.74</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>33.84 26.08 18.30 9.54</td>
<td></td>
<td>23.58 19.17 8.96 3.60</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>37.92 31.46 22.04 13.16</td>
<td></td>
<td>15.58 12.08 6.45 3.26</td>
</tr>
</tbody>
</table>

Table 5.6: Degree of improvement of MProduct over the single expert using threshold $= 5 \times 10^{-8}$

<table>
<thead>
<tr>
<th>Set size</th>
<th>Data set</th>
<th>Number of nearest neighbours</th>
<th>Data set</th>
<th>Number of nearest neighbours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Seis.</td>
<td>61.80 65.96 56.86 41.93</td>
<td>BCW</td>
<td>40.86 20.83 41.01 16.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26.45 3.93 32.17 12.01</td>
<td></td>
<td>34.28 -3.75 17.25 0.70</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>39.56 -3.08 33.87 9.37</td>
<td></td>
<td>33.15 -0.55 14.51 0.71</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>34.15 0.00 29.92 -4.26</td>
<td></td>
<td>33.78 1.70 14.87 0.45</td>
</tr>
<tr>
<td>1</td>
<td>Wine</td>
<td>36.16 29.24 32.97 30.52</td>
<td>Iris</td>
<td>31.06 38.27 33.28 25.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34.58 0.06 27.04 10.44</td>
<td></td>
<td>23.01 2.20 13.61 8.02</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>29.45 1.61 17.90 3.74</td>
<td></td>
<td>18.27 2.47 10.38 2.05</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>28.21 4.87 22.00 5.81</td>
<td></td>
<td>20.72 4.48 15.38 6.64</td>
</tr>
<tr>
<td>1</td>
<td>Iono.</td>
<td>6.55 0.99 1.95 2.00</td>
<td>Synth.</td>
<td>20.09 2.96 17.36 11.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11.50 -1.96 8.04 0.39</td>
<td></td>
<td>9.29 -0.26 6.18 -0.93</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>19.23 -2.77 13.89 1.41</td>
<td></td>
<td>9.80 0.05 4.26 0.62</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>20.21 -1.12 15.28 -2.19</td>
<td></td>
<td>6.15 1.14 4.45 -0.62</td>
</tr>
</tbody>
</table>
5.3. Modified Product

The synthetic experiments of the previous section indicate that the lowest threshold of $5 \times 10^{-8}$ leads to the best performance. Hence, in this section experiments are conducted using this threshold, and are compared to the results obtained using a threshold of $5 \times 10^{-4}$ and 0.1 as well as to the results of the single expert, as shown in Table 5.7.

Results shown in Tables 5.4 to 5.6 are for comparing MProduct against Product, Sum and the single expert, respectively, using a threshold of $5 \times 10^{-8}$ under varying set sizes for four values of $k$ mentioned in Section 5.2. These tables show detailed comparison results only for the modified bagging method 4. In these tables, in order to see the effect of a fusion method, we adopt a simple relative performance measure $s$ defined in equation 5.2 of Subsection 5.2.4.

If the improvement or degradation exceeds 5% we consider it as significant. These tables confirm that a major improvement in performance can be gained using MProduct which is able to outperform Product at all set sizes. MProduct outperforms the single expert mostly at the smallest set sizes. As $k$ - $NN$ is more stable at larger set sizes, bagging cannot improve the single expert regardless of the fusion method. Consequently for the largest data set MProduct, Product and Sum yield similar results which are equal to the single expert. For bagging method 4 when the training set size decreases MProduct mostly outperforms Sum, as shown in Table 5.4.

Table 5.7 summarise the difference in performance between the different fusion methods for 8 different bagging methods at 4 training set sizes using 4 different nearest neighbours, $k = \{2, 3, 4, 5\}$. Therefore, we have a total of 128 classification rates for each fusion method. In Table 5.7 each of the horizontal divisions shows results relating to the comparison of MProduct with a baseline using three different threshold values, $\{1, 5 \times 10^{-4}$ and $5 \times 10^{-8}\}$. In the first horizontal section of Table 5.7 MProduct is compared to Product, in the second to Sum and in the third to the single expert. In each comparison the number of times a fusion method outperforms another is summed over 128 instances, excluding ties. When neither method outperforms the other by 5% we have a tie.

When the truncation threshold value is increased to 0.1, we notice the performance of MProduct drops. It equals Sum in all instances. Nevertheless MProduct is still better
than Product by a large margin. However, both $5 \times 10^{-4}$ and $5 \times 10^{-8}$ yield similar counts of MProduct outperforming other rules. A detailed look at its classification rate at these thresholds indicates that both thresholds yield similar results. Although $5 \times 10^{-8}$ yields significantly better results for some of the instances, the larger threshold of $5 \times 10^{-4}$ yields a slightly better performance at other instances. For the smaller threshold values MProduct outperforms Sum mostly when bagging method 4 is used. Otherwise, both rules yield similar results.

Table 5.7: Number of times a method is significantly better summed over eight bagging methods, four set sizes and four values of nearest neighbours, k, totaling 128 cases. In the first horizontal section MProduct is compared to Product, in the second to Sum and in the third to the single expert.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Rule</th>
<th>Seis</th>
<th>Wine</th>
<th>Iono</th>
<th>BCW</th>
<th>Iris</th>
<th>Synth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MProd.</td>
<td>117</td>
<td>122</td>
<td>104</td>
<td>120</td>
<td>120</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>Prod.</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$5 \times 10^{-4}$</td>
<td>MProd.</td>
<td>118</td>
<td>124</td>
<td>105</td>
<td>122</td>
<td>121</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>Prod.</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$5 \times 10^{-8}$</td>
<td>MProd.</td>
<td>120</td>
<td>125</td>
<td>105</td>
<td>123</td>
<td>122</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>Prod.</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>.1</td>
<td>MProd.</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Sum</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$5 \times 10^{-4}$</td>
<td>MProd.</td>
<td>10</td>
<td>8</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Sum</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$5 \times 10^{-8}$</td>
<td>MProd.</td>
<td>16</td>
<td>9</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Sum</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>.1</td>
<td>MProd.</td>
<td>73</td>
<td>86</td>
<td>56</td>
<td>69</td>
<td>71</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>Single</td>
<td>17</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>$5 \times 10^{-4}$</td>
<td>MProd.</td>
<td>75</td>
<td>93</td>
<td>56</td>
<td>69</td>
<td>72</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Single</td>
<td>12</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>$5 \times 10^{-8}$</td>
<td>MProd.</td>
<td>77</td>
<td>92</td>
<td>56</td>
<td>68</td>
<td>71</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Single</td>
<td>11</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>12</td>
<td>1</td>
</tr>
</tbody>
</table>

Note that when estimation errors are corrupted by uniform noise, altering the value
of the truncation threshold has a minimal effect on the performance of MProduct at high noise levels. It also has no effect when noise is low as all rules perform equally well. Otherwise, as the threshold value is decreased the MProduct error rate decreases, as evident from Figure 5.2. The effect of changing the threshold value under gaussian noise is not as obvious. However a general trend, similar to the uniform noise case, has been observed. When the gaussian noise level increases a few instances emerge, at which larger threshold values lead to better results. This suggests that it is possible to improve the performance of MProduct if the value of the threshold is tuned during training. This is exemplified by the Synthetic data set for which better results can be obtained with the truncation threshold set to .1.

5.4 Moderation

In the previous section we have proposed an effective modification to improve the product fusion strategy, which alleviates the veto effect. However, the proposed MProduct fusion strategy is heuristic and its only parameter is selected heuristically or based on the performance on a validation set. The modification method lacks a theoretical background and derivation. In this section we show that the veto problem can be minimised by marginalising the k-NN estimates using the bayesian prior. A formula for the resulting moderated k-NN estimate is theoretically derived. The merits of moderation are examined on real data sets. Tests with different bagging procedures indicate that the proposed moderation method improves the performance of the multiple classifier system significantly.

5.4.1 General Moderation Theory

Given $R$ experts and $m$ classes, the product rule assigns an input pattern vector $x$ to class $\omega_j$ if

$$\prod_{i=1}^{R} P_i(\omega_j|x) = \max_{k=1}^{m} \prod_{i=1}^{R} P_i(\omega_k|x)$$

(5.4)
where \( P_i(\omega_k|x) \) is an estimate of the \( k^{th} \) class a posteriori probability \( P(\omega_k|x) \) delivered by expert \( i \). Note that the estimate will be influenced by a training set, \( X_i \), used for the design of expert \( i \). Once the form of the classifier is chosen, the training set is then deployed to estimate the underlying model parameters denoted by vector \( \gamma_i \). A particular value of the parameter vector obtained through training will then define the \( i^{th} \) expert output. This can be made explicit by denoting the output by \( P(\omega_k|x, X_i, \gamma_i) \). However, \( \gamma_i \) is only an estimate of the true model parameters. The estimate will be just a single realisation of the random variable drawn from the sampling distribution \( P(\gamma_i) \). If the sampling distribution is known a priori, then the raw estimate

\[
P_i(\omega_k|x) = P(\omega_k|x, X_i, \gamma_i)
\]

(5.5)
can be moderated by taking the prior into consideration. In other words, a new estimate is obtained by integrating parameter dependent estimates over the model parameter space as

\[
\hat{P}_i(\omega_k|x) = \int P_i(\omega_k|x, X_i, \gamma_i)P(\gamma_i)d\gamma_i
\]

(5.6)
This is known as marginalisation in Bayesian estimation.

5.4.2 k-NN Classifier Output Moderation

In Subsection 5.4.1 we argued for a moderation of raw expert outputs. The moderation is warranted for pragmatic reasons, namely to minimise the veto effect of overconfident erroneous classifiers.

It is perhaps true to say that for training sets of reasonable size there should not be any appreciable difference between moderated and raw expert outputs. However, for some types of classifiers, moderation is pertinent even for sample sets of respectable size. An important case is the \( k-NN \) classifier. Even if the training set is relatively large, say hundreds of samples or more, the need for moderation is determined by the value of \( k \), which may be as low as \( k = 1 \). Considering just the simplest case, a two class problem, it is perfectly possible to draw all \( k \)-Nearest Neighbours from the same class which means that one of the classes will have the expert output set to zero. In the subsequent (product) fusion this will then dominate the fused output and may impose a veto on the class even if other experts are supportive of that particular hypothesis.
We shall now consider this situation in more detail. Suppose that we draw $k$-Nearest Neighbours and find that $\kappa$ of these belong to class $\omega$. Then, the unbiased estimate $P_1(\omega|x)$ of the posterior probability $P(\omega|x)$ is given by

$$P_1(\omega|x) = \frac{\kappa}{k} \tag{5.7}$$

It should be noted that the actual observation $\kappa$ out of $k$ could arise for any value of $P(\omega|x)$ with the probability

$$q(\kappa) = \binom{k}{\kappa} P^\kappa(\omega|x)[1 - P(\omega|x)]^{k-\kappa} \tag{5.8}$$

Assuming that a priori the probability $P(\omega|x)$ taking any value between zero and one is equally likely, we can find an a posteriori estimate of the a posteriori class probability $P(\omega|x)$ as

$$P_2(\omega|x) = \frac{\int_0^1 \binom{k}{\kappa} P^\kappa(\omega|x)[1 - P(\omega|x)]^{k-\kappa} dP(\omega|x)}{\int_0^1 P^\kappa(\omega|x)[1 - P(\omega|x)]^{k-\kappa} dP(\omega|x)} \tag{5.9}$$

where the denominator is a normalising factor ensuring that the total probability mass equals to one. By expanding the term $[1 - P(\omega|x)]^{k-\kappa}$ and integrating, it can be easily verified that the right hand side of (5.9) becomes

$$P_2(\omega|x) = \frac{\kappa + 1}{k + 2} \tag{5.10}$$

which is the beta distribution. Thus the moderated equivalent of $\frac{\kappa}{k}$ is $\frac{\kappa + 1}{k + 2}$. Clearly our estimates of a posteriori class probabilities will never reach zero which could cause a veto effect. For instance, for the Nearest Neighbour classifier with $k = 1$ the smallest expert output will be $\frac{1}{3}$. As $k$ increases the smallest estimate will approach zero as $\frac{1}{k+2}$ and will assume zero only when $k = \infty$.

For $m$ class problems equation (5.10) can be extended to become

$$P_2(\omega|x) = \frac{\kappa + 1}{k + m} \tag{5.11}$$

### 5.4.3 Moderation Experimental Results

Table 5.8 displays the baseline classification rates of the single expert and bagging using the product rule with raw a posteriori class probability estimates. These results will be analysed and discussed in Chapter 7 For the purpose of this section we only note the main points, namely that
Figure 5.4: Moderated Product improvement over the single expert

- Bagging does not improve the $k - NN$ rule performance for sufficiently large training sets (in excess of 80 samples)

- For smaller sample sets, bootstrapping and aggregation of moderated estimates of a posteriori class probabilities via product can be useful

- For very small training and bootstrap sets created by means of regular sampling, bagging can lead to degradation in performance

The benefits of moderating the $k - NN$ classifier outputs can be gleaned from Figures 5.4 and 5.5 which plot the relative performance measure defined in equation 5.2. In Figure 5.4 we show the improvement gained over the single expert whereas Figure 5.5 relates the product aggregation of moderated $k - NN$ outputs to the product of raw outputs. We note that for training sets of size less than 80 samples the performance improves significantly. The gains are inversely proportional to the training set size. Bagging with moderation can largely compensate for the lack of training data.
The method for moderating the outputs of the $k-NN$ classifier advocated in this section is based on the principles of sampling with a Bayesian prior. The Modified Product (MProduct) proposed in the previous section, has the same motivation, i.e. eliminating the veto effect of the product fusion rule that can be caused by raw aposteriori class probability estimates. However, MProduct is heuristic and it is of interest to compare its performance with the moderated output scheme based on theoretical foundations. In MProduct the $j^{th}$ expert output $P_j(\omega_i|x)$, which estimates the a posteriori probability for class $\omega_i$ given pattern vector $x$, is modified before entering the product fusion rule, as shown in section 5.3.

The respective transfer functions between the raw inputs and outputs delivered by moderation and MProduct are shown in Figure 5.6. The posterior probability estimates of MProduct cover almost the full range 0 to 1, regardless of the value of $k$. In contrast, the range of moderated posterior probability estimates reduces as $k$ decreases. However, these differences in transfer functions do not seem to translate to any significant differences in performance as can be seen from the results of Figure 5.7. In all the
### Table 5.8: Classification rate of the single expert and bagging using Product of non-moderated aposteriori class probability estimates

<table>
<thead>
<tr>
<th>Learn set type</th>
<th>Bagging method</th>
<th>Data set</th>
<th>No. of learning samples</th>
<th>Data set</th>
<th>No. of learning samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>40</td>
<td>80</td>
</tr>
<tr>
<td>Regular</td>
<td>1</td>
<td>Seis.</td>
<td>89.87</td>
<td>98.16</td>
<td>99.17</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>95.29</td>
<td>98.12</td>
<td>99.13</td>
<td>91.69</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>96.56</td>
<td>98.62</td>
<td>99.16</td>
<td>92.70</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>94.73</td>
<td>98.13</td>
<td>99.17</td>
<td>91.43</td>
</tr>
<tr>
<td></td>
<td>Single xp</td>
<td>96.41</td>
<td>98.08</td>
<td>99.16</td>
<td>92.66</td>
</tr>
<tr>
<td>Modified</td>
<td>1</td>
<td>94.83</td>
<td>98.03</td>
<td>99.08</td>
<td>90.89</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>96.41</td>
<td>98.16</td>
<td>99.09</td>
<td>92.01</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>96.34</td>
<td>98.37</td>
<td>99.15</td>
<td>92.43</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>95.50</td>
<td>97.96</td>
<td>99.10</td>
<td>91.50</td>
</tr>
<tr>
<td></td>
<td>Single xp</td>
<td>96.90</td>
<td>97.91</td>
<td>99.10</td>
<td>93.32</td>
</tr>
<tr>
<td>Regular</td>
<td>1</td>
<td>Wine</td>
<td>76.05</td>
<td>91.39</td>
<td>93.54</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>83.20</td>
<td>92.09</td>
<td>93.65</td>
<td>87.00</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>86.07</td>
<td>92.37</td>
<td>93.84</td>
<td>88.76</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>81.04</td>
<td>91.80</td>
<td>93.78</td>
<td>84.33</td>
</tr>
<tr>
<td></td>
<td>Single xp</td>
<td>85.14</td>
<td>91.75</td>
<td>93.56</td>
<td>91.35</td>
</tr>
<tr>
<td>Modified</td>
<td>1</td>
<td>79.54</td>
<td>91.22</td>
<td>94.28</td>
<td>83.45</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>85.84</td>
<td>91.74</td>
<td>93.95</td>
<td>88.75</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>86.43</td>
<td>92.01</td>
<td>94.26</td>
<td>89.22</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>82.72</td>
<td>91.09</td>
<td>94.16</td>
<td>84.52</td>
</tr>
<tr>
<td></td>
<td>Single xp</td>
<td>88.27</td>
<td>91.34</td>
<td>94.02</td>
<td>92.15</td>
</tr>
<tr>
<td>Regular</td>
<td>1</td>
<td>Iono.</td>
<td>68.28</td>
<td>71.97</td>
<td>76.00</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>68.24</td>
<td>72.63</td>
<td>75.86</td>
<td>89.23</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>68.31</td>
<td>72.99</td>
<td>75.62</td>
<td>89.85</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>67.40</td>
<td>72.31</td>
<td>75.85</td>
<td>88.72</td>
</tr>
<tr>
<td></td>
<td>Single xp</td>
<td>68.60</td>
<td>70.25</td>
<td>75.93</td>
<td>89.34</td>
</tr>
<tr>
<td>Modified</td>
<td>1</td>
<td>67.21</td>
<td>71.76</td>
<td>75.62</td>
<td>89.35</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>68.10</td>
<td>71.90</td>
<td>75.56</td>
<td>90.03</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>67.68</td>
<td>72.36</td>
<td>76.15</td>
<td>90.06</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>66.97</td>
<td>71.70</td>
<td>75.48</td>
<td>89.32</td>
</tr>
<tr>
<td></td>
<td>Single xp</td>
<td>67.55</td>
<td>70.22</td>
<td>75.61</td>
<td>89.97</td>
</tr>
</tbody>
</table>
5.4. Moderation

Figure 5.6: Comparison of posterior probability estimates when using Moderation to when using Modified Product

Figure 5.7: Moderated Product improvement over the Modified Product
above experiments the value of $k$ for a particular training set size was automatically chosen using the usual square root rule, i.e. $k = \sqrt{n}$ where $n$ is the size of the training set. This leads to using an odd value for $k$ at the largest set size and even $k$ for the smaller two sizes. We wanted to check the consistency of the above results as $k$ varies. We repeated the same experiments for the number of Nearest Neighbours varying from 1 to 10. Typical results are shown in Figures 5.8 and 5.9, where we have added results for an additional smaller set size of 10 samples. Figure 5.8 gives the classification error rates for the Seismic data for a single expert, and a product fusion combining raw, moderated and heuristically modified $k$-NN classifier outputs respectively. Note that the performance of Product of $k$-NN raw outputs improves with increasing $k$. Clearly the probability of the veto effect occurring will be highest for the smallest $k$ and will go down as $k$ increases. However, the improvement in performance of this fusion method is undermined by the general downward trend of the $k$-NN classifier as a function of $k$ for small training sets. Thus the performance curve of Product of raw outputs will peak at some point and then monotonically decay with increasing $k$. 

Figure 5.8: Comparison of classification rates of Product when expert outputs are raw, Moderated or Modified, all compared to the single expert. Results are for bagging method four using regular Seismic learning set
Figure 5.9: Comparison of classification rates of Product when expert outputs are raw, Moderated or Modified, all compared to the single expert. Results are for bagging method four using regular Wine data learning set.

The product of moderated outputs and MProduct peak at the lowest value of $k$ i.e. at $k = 1$ and then monotonically fall off as $k$ increases. On average, MProduct is marginally better than moderation, but at the peak ($k = 1$) the curves meet. Interestingly, the single expert does extremely well for odd numbers of $k$ and larger sets, but less well for smaller training sets. Most importantly, it does not do well for $k=2$ where the probability of indecision is high. Thus one of the benefits of bagging is that it produces consistent performance for odd and even values of $k$. The general conclusions emerging from these results are that the bootstrapping of $k$–NN classifiers with aggregation by Product is beneficial for small training sets if the classifier outputs are moderated. The best performance is obtained for $k = 1$. This general behaviour of bagging $k$–NN classifiers is confirmed for the Wine data as shown in Figure 5.9.

5.5 Conclusion

Based on the outcome of the previous chapter we have found that the Product fusion strategy, which is derived under the bayesian framework, underperforms Sum which is
also derived under the bayesian framework but with restricting assumptions. Results indicated that the reason for the degraded performance of Product is the veto effect. The veto effect caused by contradicting experts outputting zero probability estimates leads to some fusion strategies outputting zero estimates for the true class and hence performing sub optimally.

The performance of Product can be significantly improved by truncating the expert outputs falling below a certain threshold before combining. The resulting Modified Product (MProduct), presented in this chapter, identifies the degrading classifier outputs and replaces each output by a value that does not cause a veto effect. The new fusion rule exhibits the most significant improvements over the performance of Product when the estimation noise is high. It is also able to match the performance of Product when the noise level is low. Thus MProduct consistently exhibits a superior performance to Product. On average it also outperforms Sum. For this reason it is advocated as a strong contender among simple fusion strategies which do not require any fusion stage training. Although MProduct performs very well at a wide range of thresholds in most cases its best performance is achieved when truncation threshold is set to $5 \times 10^{-4}$ or lower.

Furthermore, we establish a theoretical basis for modifying expert outputs by introducing the concept of classifier output moderation. We focused on the $k$-NN decision rule and derived the formula for correcting the outputs of $k$-NN experts, by taking the model prior into consideration. The derived moderation equation was examined on real data sets. Tests on different bagging methods indicated that the proposed moderation method improves the performance of Product significantly when problems relating to training set sample distribution exist.
Chapter 6

Sum Versus Vote

Sum and Vote are compared theoretically and experimentally under varying noise conditions.

6.1 Introduction

Among the many combination rules suggested in the literature [4, 8, 15, 18, 24, 35, 29, 39, 40, 43, 50, 48, 55, 72, 76, 83, 94, 109] Sum and Vote are used the most frequently. The previous chapters indicated that the sum rule operates directly on the soft outputs of individual experts for each class hypothesis, normally delivered in terms of a posteriori class probabilities. Accordingly, the expert outputs are combined by averaging. The fused decision is obtained by applying the maximum value selector to the class dependent averages. Vote, on the other hand, operates on class labels assigned to each pattern by the respective experts by hardening their soft decision outputs using the maximum value selector. The Vote rule output is a function of the votes received for each class in terms of these single expert class labels.

Classification systems implementing the Bayes decision rule incur classification errors over and above the Bayes rate due to errors in their estimates of the a posteriori class probabilities. The larger the variance of the error distribution the larger the additional classification error. A multiple classifier system which deploys the Sum rule reduces this
variance and as a result diminishes the additional classification error. The properties of
the rule have been widely investigated [39, 40, 50, 48, 7, 10, 96, 95, 100]. When fusing
by Sum the experts outputs can be treated equally or they could be assigned different
weights based on their performance on a validation set. When independent experts are
combined, equal weights appear to yield the best performance [2].

As for Vote, many versions of the combination rule exist, such as unanimous vote,
threshold voting, weighted voting and simple majority voting [55, 109]. In addition
to these basic rules, the authors in [109] propose two voting methods claimed to out-
perform the majority voting. The first method assigns a pattern to a class by the
unanimity vote, otherwise the sample is rejected. In the second method the authors
propose a winning class to be the one that has the highest vote and its votes are larger
than the second largest vote by a threshold. Lam and Suen [55] give a comprehen-
sive analysis of the behaviour of the majority vote (Vote), under the assumption of
conditional independence of the experts. They show that Vote with an odd number
of experts produces a higher recognition rate, while voting with an even number of
experts produces better results, if errors are more costly than rejections.

In our theoretical deliberations we focus on the basic Sum and Vote rules. Clearly both
the weighted average (see e.g. [51]) and modified voting [109] can outperform the basic
rules. However, the advanced strategies require training which is a negative aspect of
these approaches. In any case, we believe that the conclusions drawn from the analysis
of the simple cases will extend also to the more complex procedures.

Many researchers [7, 50, 29, 69, 88] have found that Sum outperforms Vote, while a few
[29, 69] have demonstrated that Vote can equal or outperform Sum. In this chapter
we will investigate the relationship between these two rules in more detail. We shall
argue that the relative merits of Sum and Vote depend on the distribution of estimation
errors.

We show analytically that, for normally distributed estimation errors, Sum always
outperforms Vote, whereas for heavy tail distributions Vote may outperform Sum. We
then confirm our theoretical predictions by experiments on both synthetic and real
data. In the synthetic experts case, we show for Gaussian error distributions that,
6.2. Theoretical Analysis

as expected, Sum outperforms Vote. The differences in performance are particularly significant at high estimation noise levels. However, for heavy tail distributions the superiority of Sum may be eroded for any number of experts if the margin between the two a posteriori class probabilities is small or for a small number of cooperating experts even when the margin is large. In the latter case, once the number of experts exceeds a certain threshold, Sum tends to be superior to Vote. Experiments on real data support the general findings but also show the effect of the usual assumptions of conditional independence, identical error distributions and common target outputs of the experts not being fully satisfied.

In the next section we introduce the necessary formalism and develop the basic theory of classifier combination by averaging and majority voting. The relationship of the two strategies is discussed in Section 6.3. We set up a synthetic experiment in which we compare Sum to majority Vote in a two class problem under the assumption that the error distributions are Gaussian, in order to confirm the validity of the theoretical predictions made. The experimental comparison is performed for different combinations of the key influential parameters, namely error distribution variance, number of experts, and the local difference between class a posteriori probabilities. In Section 6.4 we carry out a real data experiment involving the XM2VTS database [65]. We adopt the Lausane experimental protocol [61] so that the results are comparable with the baseline established by other workers on the same data. In Section 6.5 we discuss the results and draw the chapter to conclusion in Section 6.6.

6.2 Theoretical Analysis

Consider a two class pattern recognition problem where pattern Z is to be assigned to one of the two possible classes \( \{ \omega_1, \omega_2 \} \). Let us assume that we have \( R \) classifiers each representing the given pattern by an identical measurement vector \( x \). In the measurement space each class \( \omega_k \) is modelled by the probability density function \( p(x|\omega_k) \) and the a priori probability of occurrence denoted by \( P(\omega_k) \). We shall consider the models to be mutually exclusive which means that only one model can be associated with each pattern.
Now according to the Bayesian decision theory, given measurements $x$, the pattern, $Z$, should be assigned to class $\omega_j$, i.e. its label $\theta$ should assume value $\theta = \omega_j$, provided the aposteriori probability of that interpretation is maximum, i.e.

$$assign \quad \theta \rightarrow \omega_j \quad if$$

$$P(\theta = \omega_j | x) = \max_k P(\theta = \omega_k | x) \quad (6.1)$$

In practice, the $j$-th expert will provide only an estimate $P_j(\omega_l | x)$ of the true aposteriori class probability $P(\omega_l | x)$ given pattern $x$, rather than the true probability. The idea of classifier combination is to obtain a better estimate of the aposteriori class probabilities by combining all the individual expert estimates and thus reducing the classification error. A typical estimator is the averaging estimator

$$\hat{P}(\omega_l | x) = \frac{1}{R} \sum_{j=1}^{R} P_j(\omega_l | x) \quad (6.2)$$

where $\hat{P}(\omega_l | x)$ is the combined estimate based on $R$ observations.

Let us denote the error on the $j$-th estimate of the $i$-th class aposteriori probability at point $x$ as $e_j(\omega_l | x)$ and let the probability distribution of the errors be $p_{ij}[e_j(\omega_l | x)]$. Then the probability distribution of the unscaled error $e_i(x)$

$$e_i(x) = \sum_{j=1}^{R} e_j(\omega_l | x) \quad (6.3)$$

on the combined estimate will be given by the convolution of the component error densities, i.e.

$$p(e_i(x)) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p_{i1}(\lambda_1)p_{i2}(\lambda_2 - \lambda_1)\cdots p_{iR}(e_i(x) - \lambda_{R-1})d\lambda_1d\lambda_2\cdots d\lambda_{R-1} \quad (6.4)$$

The distribution of the scaled error $\epsilon_i(x) = \frac{1}{R} e_i(x)$ is then given by

$$p(\epsilon_i(x)) = p(\frac{1}{R} e_i(x)) \quad (6.5)$$

In order to investigate the effect of classifier combination, let us examine the two class aposteriori probabilities at a single point $x$. Suppose the aposteriori probability of class $\omega_s$ is maximum, i.e. $P(\omega_s | x) = \max_{l=1}^{2} P(\omega_l | x)$ giving the local Bayes error
6.2. Theoretical Analysis

\( e_B(x) = 1 - \max_{i=1}^2 P(\omega_i|x) \). However, our classifiers only estimate these a posteriori class probabilities and the associated estimation errors may result in suboptimal decisions, and consequently in an additional classification error. In order to quantify this additional error we have to establish what the probability is for the recognition system to make a suboptimal decision. This situation will occur when the a posteriori class probability estimates for the other class becomes maximum. Let us derive the probability of the event occurring for a single expert \( j \) for class \( \omega_i \), \( i \neq s \), i.e. when

\[
P_j(\omega_i|x) - P_j(\omega_s|x) > 0  \tag{6.6}
\]

Note that the left hand side of (6.6) can be expressed as

\[
P(\omega_i|x) - P(\omega_s|x) + e_j(\omega_i|x) - e_j(\omega_s|x) > 0  \tag{6.7}
\]

Equation (6.7) defines a constraint for the two estimation errors \( e_j(\omega_k|x), \ k = 1, 2 \) as

\[
e_j(\omega_i|x) - e_j(\omega_s|x) > P(\omega_s|x) - P(\omega_i|x)  \tag{6.8}
\]

In a two class case the errors on the left hand side satisfy

\[
e_j(\omega_s|x) = -e_j(\omega_i|x)  \tag{6.9}
\]

and thus an additional labelling error will occur if

\[
2e_j(\omega_i|x) > P(\omega_s|x) - P(\omega_i|x)  \tag{6.10}
\]

The probability \( e_A(x) \) of this event occurring will be given by the integral of the error distribution under the tail defined by the margin \( \Delta P_{si}(x) = P(\omega_s|x) - P(\omega_i|x) \), i.e.

\[
e_A(x) = \int_{\Delta P_{si}(x)}^\infty p_{ij}[2e_j(\omega_i|x)]de_j(\omega_i|x)  \tag{6.11}
\]

In contrast, after classifier fusion by averaging, the labelling error with respect to the Bayes decision rule will be given by

\[
e_S(x) = \int_{\Delta P_{si}(x)}^\infty p[2\epsilon_i(x)]d\epsilon_i(x)  \tag{6.12}
\]

Now how do these labelling errors translate to classification error probabilities? We know that for the Bayes minimum error decision rule the error probability at point \( x \)
will be \( e_B(x) \). If our pseudo Bayesian decision rule, i.e. the rule that assigns patterns according to the maximum estimated a posteriori class probability, deviates from the Bayesian rule with probability \( e_A(x) \) the local error of the decision rule will be given by

\[
\alpha(x) = e_B(x)[1 - e_A(x)] + e_A(x)[1 - e_B(x)]
\]  

(6.13)

which simplifies to

\[
\alpha(x) = e_B(x) + e_A(x)[1 - 2e_B(x)] = e_B(x) + e_A(x)|\Delta P_{12}(x)|
\]  

(6.14)

as \(|(1-e_B(x))-e_B(x)|\) is the absolute value of the margin between the two a posteriori class probabilities. For the multiple classifier system which averages the expert outputs, the classification error probability is

\[
\beta(x) = e_B(x) + e_S(x)|\Delta P_{12}(x)|
\]  

(6.15)

Thus for a multiple classifier system to achieve a better performance the labelling error after fusion, \( e_S(x) \), should be smaller than the labelling error, \( e_A(x) \), of a single expert.

Let us now consider fusion by voting. In this strategy all single expert decisions are hardened and therefore each expert will make suboptimal decisions with probability \( e_A(x) \). When combined by voting for the most representative class, the probability distribution of \( k \) decisions, among a pool of \( R \), being suboptimal is given by the binomial distribution. A switch of labels will occur whenever the majority of individual expert decisions is suboptimal. This will happen with probability

\[
e_V(x) = \sum_{k=\frac{R}{2}+1}^{R} \binom{R}{k} e_A(x)[1 - e_A(x)]^{R-k}
\]  

(6.16)

Provided \( e_A(x) < 0.5 \) this probability will decrease with increasing \( R \). Note that for a large value of \( R \) the binomial distribution in equation (6.16) can be approximated by a Gaussian distribution with mean \( e_A(x) \) and variance \( \frac{e_A(x)[1 - e_A(x)]}{R} \), i.e.

\[
e_V(x) = \frac{\sqrt{R}}{\sqrt{2\pi e_A(x)[1 - e_A(x)]}} \int_{0.5}^{\infty} exp\left(-\frac{R}{2} \frac{[k - e_A(x)]^2}{e_A(x)[1 - e_A(x)]}\right) dk
\]  

(6.17)

For large \( R \) the integral will go to zero because of the decreasing variance.

After fusion by Vote, the error probability of the multiple classifier will then be

\[
\gamma(x) = e_B(x) + e_V(x)|\Delta P_{12}(x)|
\]  

(6.18)
6.2. Theoretical Analysis

Figure 6.1: Sum and Vote switching error for normally distributed estimation errors with different values of $\sigma(x)$ using 3 experts.

Before discussing the relationship between Sum and Vote in the next section, let us pause and consider the formulae (6.13), (6.15) and (6.18). The additional classification error, over and above the Bayesian error, is given by the second term in the expressions. Note that the term depends on the probability $e_X(x)$ of the decision rule being suboptimal and the margin $\Delta P_{12}(x)$. The former is also a function of the margin, the number of experts $N$ and the estimation error distribution. Now, at the boundary $\Delta P_{12}(x) = 0$ and the multiple classifier system will be Bayes optimal, although at this point $e_X(x)$ is maximum. As we move away from the boundary $\Delta P_{12}(x)$ increases but at the same time $e_X(x)$ decreases. The product of the two nonnegative functions will be zero for $\Delta P_{12}(x) = 0$ and as $\Delta P_{12}(x)$ increases, it will reach a maximum, followed by a rapid decay to zero.

The above behaviour is illustrated in figure 6.1 for Sum and Vote combination strategies for normally distributed estimation errors with different values of $\sigma(x)$ and $N = 3$. We note that the additional error injected by the sum rule is always lower than that due to Vote. As the standard deviation of the estimation error distribution increases, the probability of the decision rule being suboptimal increases for all margins. At the same time the peak of the two functions shifts towards the higher margin values. As
Figure 6.2: Sum and Vote switching error for normally distributed estimation errors with different values of $\sigma(x)$ using 7 experts.

Figure 6.3: Sum and Vote switching error for normally distributed estimation errors with different values of $\sigma(x)$ using 11 experts.
6.2. Theoretical Analysis

Figure 6.4: Sum and Vote switching error for normally distributed estimation errors with different values of $\sigma(x)$ using 15 experts.

Figure 6.5: Sum and Vote switching error for normally distributed estimation errors with $\sigma(x) = .05, .15, .25$ and .35, using 3, 7, 11 and 15 experts.
the number of experts increases the above relationship between \( \sigma(x) \) and \( \Delta P_{12}(x) \) is preserved. However, the additional experts push the family of curves towards the origin of the graph. This can be gleaned from figures 6.2 - 6.4. The results are summarised in figure 6.5.

### 6.3 Relationship of Sum and Vote

In this section we shall investigate the relationship between the Sum and Vote fusion strategies. For the sake of simplicity we shall commence by assuming that the errors \( e_j(\omega_i|x) \) between the true class aposteriori probabilities \( P(\omega_i|x) \) and their estimates are unbiased, i.e.

\[
E\{e_j(\omega_i|x)\} = E\{\hat{P}_j(\omega_i|x) - P(\omega_i|x)\} = 0 \quad \forall i, j, x \quad (6.19)
\]

Then the combined estimate \( \hat{P}(\omega_i|x) \) will also be an unbiased estimate of \( P(\omega_i|x) \). Suppose the standard deviations \( \sigma_j(\omega_i|x) \) of errors \( e_j(\omega_i|x) \) are equal, i.e.

\[
\sigma_j(\omega_i|x) = \sigma(\omega_i|x) \quad \forall i, j \quad (6.20)
\]

Then, provided the errors \( e_j(\omega_i|x) \) are independent, the variance of the error distribution for the combined estimate \( \sigma^2(x) \) will be

\[
\sigma^2(x) = \frac{\sigma^2(x)}{N} \quad (6.21)
\]

Let us assume that the error distributions \( p_{ij}[e_j(\omega_i|x)] \) are gaussian. This in practice will approximate the true distribution of estimation errors very coarsely as both ends of the \([0, 1]\) interval from which the aposteriori class probabilities can assume values will clip the errors. Nevertheless, the analysis, under even such a simplistic assumption, will give an indication of the benefits of classifier combination.

Since the error distributions are gaussian, the distribution of the difference of the two errors with equal magnitude but opposite sign will also be gaussian with four times as large variance. The probability of the constraint (6.8) being satisfied is given by the area under the gaussian tail with a cut off point at \( P(\omega_i|x) - P(\omega_i|x) \). More specifically, this probability, \( e_A(x) \), is given by

\[
e_A(x) = 1 - erf\left(\frac{\Delta P_{st}(x)}{4\sigma}\right) \quad (6.22)
\]
6.3. Relationship of Sum and Vote

where \( \text{erf}\left(\frac{\Delta P_{\text{st}}(x)}{4\sigma}\right) \) is the error function defined as

\[
\text{erf}\left(\frac{\Delta P_{\text{st}}(x)}{4\sigma}\right) = \frac{1}{2\sqrt{2\pi}\sigma} \int_{0}^{\Delta P_{\text{st}}(x)} \exp^{-\frac{y^2}{2\sigma^2}} dy
\]  

(6.23)

In order to compare the performance gains of the Sum and Vote fusion under the gaussian assumption we have designed a simulation experiment involving \( N \) experts, each estimating the same aposteriori probability \( P(\omega_i|x) \) \( i = 1, 2 \). Estimation errors are simulated by perturbing the target probability \( P(\omega_i|x) \) with statistically independent errors drawn from a gaussian distribution with a zero mean and standard deviation \( \sigma(x) \). We have chosen the aposteriori probability of class \( \omega_1 \) to be always greater than 0.5. The decision margin \( \Delta P_{12}(x) \) is given by \( 2P(\omega_1|x) - 1 \). The Bayesian decision rule assigns all the test patterns to class \( \omega_1 \). For each test sample the expert outputs are combined using the Sum rule and the resulting value compared against the decision

Figure 6.6: Comparison of experimental Sum and Vote switching errors with theoretical predictions, for \( \sigma(x) = 0.15 \).
Figure 6.7: Comparison of experimental Sum and Vote switching errors with theoretical predictions, for $\sigma(x) = 0.25$.

threshold of 0.5. If the estimated aposteriori probability for a test sample from class $\omega_1$ is less than 0.5 or if the value is greater than 0.5 for a sample from class $\omega_2$ an error counter is incremented.

This particular method of estimating the probability of the decision rule being suboptimal, which we shall refer to as two class set testing is dependent on the random process of sampling the aposteriori class probability distributions. In order to eliminate the inherent stochasticity of the sampling process and its impact on the estimated error we also ran the same experiment by testing with samples from a single class. The corresponding one class set testing method involved samples from class $\omega_1$ only and the switching error was estimated by counting the number of misclassified patterns.

Similarly, the decision errors of the majority vote are estimated by converting the expert
6.3. Relationship of Sum and Vote

outputs into class labels using the pseudo Bayesian decision rule and then counting the support for each class among the $N$ labels. The label of the winning class is then checked against the identity of the test pattern and any errors recorded. The results are averaged over 500 experiments for each combination of $P(\omega_1|x)$ and $\sigma(x)$, the parameters of the simulation experiment.

The empirical results showing the additional error incurred are plotted as a function of the number of experts $N$ in Figures 6.6-6.8. The results were obtained using the two class set testing approach. The theoretical values predicted by formulas (6.12) and (6.16) are also plotted for comparison. The experimental results mirror closely the theoretically predicted behaviour, i.e. Sum being superior to Vote. All the results shown in Figures 6.6-6.8 indicate that Sum outperforms majority Vote at all error levels and all margins $\Delta P_{12}(x)$ except for the boundary where no improvement
is possible. For a large number of experts Vote approaches the performance of Sum. However, for high values of \( \sigma(x) \) the initial discrepancy in performance between Sum and Vote is large and the convergence of the two strategies as the number of experts increases is slow. The slight positive bias of the empirical errors as compared with their theoretical predictions is believed to be due to sampling effects (sampling of the aposteriori class probability distributions and of the distribution of gaussian estimation errors). As \( \sigma(x) \) increases the additional classification error also increases. In contrast, increasing the margin has the opposite effect.

While under the Gaussian assumption the Sum rule always outperforms Vote it is pertinent to ask whether this relationship holds for other distributions. Intuitively, if the error distribution has heavy tails it is easy to see that fusion by Sum will not result in improvement until the probability mass in the tail of \( p_{ij}[e_j(\omega_i|x)] \) moves within the margin \( \Delta P_{12}(x) \). In order to gain better understanding of the situation let us consider a specific example with the error distribution \( p_{ij}[e_j(\omega_i|x)] \) being defined as a mixture of three Dirac delta functions with the weights and positions shown in figure 6.9. Using the convolution integral in equation (6.4) and substituting into (6.12) we can derive the probability, \( e_S(x) \) of the decision rule being suboptimal for a given margin \( \Delta P_{12}(x) \).

Figure 6.10 shows this probability as a function of the number of expert outputs fused. The function has been computed for a range of margins from \( \Delta P_{12}(x) = 0.04 \) to \( \Delta P_{12}(x) = 0.2 \). The figure shows clearly an oscillating behaviour of \( e_S(x) \). It is
Figure 6.10: Theoretical switching error of Sum and Vote in the presence of delta noise at (a) 0.2 and (b) 0.4
Chapter 6. Sum Versus Vote

interesting to note that for small margins, initially (i.e. for a small number of experts) the error probability of the sum combiner has a tendency to grow above the probability of the decision rule being suboptimal for a single expert. First the performance improves when $N = 2$ but as further experts are added the error builds up as the probability mass shifts from the origin to the periphery by the process of convolution. It is also interesting to note that for $N = 2$ Vote degrades in performance. However, this is only an artifact of a vote tie not being randomised in the theoretical formula. Once the first line of the probability distribution of the sum of estimation errors falls below the threshold defined by the margin between the two class aposteriori probabilities the performance dramatically improves. However, by adding further experts the error build up will start all over again, though it will culminate at a lower value than at the previous peak. We can see that for instance for $\Delta P_{12}(x) = 0.04$ the benefits from fusion by the sum rule will be very poor and there may be a wide range of $N$ for which fusion would result in performance deterioration.

Once the margin reaches 0.16 Sum will generally outperform Vote but there may be specific numbers of experts for which Vote is better than Sum. The same kind of behaviour is demonstrated in figures 6.11 and 6.12 where the position of the Dirac delta components of the error distribution offset from the origin is at $\pm[1 - P(\omega_1|x)]$. Figure 6.12 shows the additional effect of sampling the aposteriori class probability distribution inherent in the two class set testing approach.

In contrast the corresponding probability, $e_V(x)$, given for the majority vote by formula (6.16), diminishes monotonically (also in an oscillating fashion) with the increasing number of experts. Thus there are situations where Vote outperforms Sum. Most importantly, this is likely to happen close to the decision boundary where the margins are small.

By the central limit theorem, as the number of experts increases, the probability distribution of the sum of expert outputs will become more and more gaussian. At the same time the variance of the labelling error distribution will decay with a factor $\frac{1}{\sqrt{N}}$. Thus at some point the result of fusing $N$ expert outputs subject to error distribution in Figure 6.9 will be indistinguishable from the effect of fusing estimates corrupted by
6.3. Relationship of Sum and Vote

Figure 6.11: Sum and Vote switching error: A comparison of single class experimental results and theoretical predictions for delta noise located at (1-p). (a) up to 100 experts, (b) up to 20 experts
Figure 6.12: Sum and Vote switching error: A comparison of two class experimental results and theoretical predictions for delta noise positioned at (1-p). (a) up to 100 experts, (b) up to 20 experts
normally distributed noise with the same initial variance. For our distribution in Figure 6.9 the standard deviation equals $\sigma(x) = 0.357$. From the experiments presented in Section 6.2 we already established that for this regime Sum should be better than Vote. As the effective $\sigma(x)$ is quite high it should take relatively long time for the two fusion strategies to converge which is borne out by the plots in Figure 6.10.

In summary, for error distributions with heavy tails we can expect Vote to outperform Sum for small margins. At some point Sum will overtake Vote and build up a significant margin between the two which will eventually diminish as Vote converges to Sum from above.

### 6.4 Real Data Experiments

In Sections 6.2 and 6.3 we investigated the relationship of Sum and Vote analytically and by simulation studies. In this section we shall compare these two combination rules on real data. Real pattern classification problems differ from idealised situations in many different ways. First of all the point wise analysis performed in Sections 6.2 and 6.3 is impossible as we do not have enough data at each and every single point of the expert output space. Second, in realistic scenarios the ground truth, at best, is known only coarsely, in terms of class labels, rather than true a posteriori probabilities. Third, expert outputs are likely to be correlated. Moreover, it is unlikely that each expert would be estimating the same a posteriori probability functions. Nevertheless, Sum and Vote are useful practical fusion rules and it is interesting to know how they compare when we depart significantly from the underlying assumptions.

As a vehicle for our experimental study we consider the problem of personal identity verification using face and voice biometrics extracted from the multimedia data in the XM2VTS database. The XM2VTS database is a multimodal database consisting of face images, video sequences and speech recordings taken of 295 subjects at one month intervals. The database contains 4 sessions. During each session two head rotation and speaking shots were taken. The Lausanne protocol [61] splits randomly all 295 subjects into 200 clients, 25 evaluation imposters and 70 test imposters. Two different evaluation configurations were defined. They differ in the distribution of client training
and client evaluation data. Configuration I takes the first shot from the first 3 sessions as training data, while the second shots are taken for evaluation. Configuration II takes sessions 1 and 2 as training data and session 3 as evaluation data. Our experiments are based on configuration I. Hence, the evaluation set contains 600 client shots (200 clients x three shots), and 40000 imposter cases (25 imposters x 8 shots x 200 clients). The test set contains 400 client shots (200 clients x 2 shots) and 112000 imposter cases (70 imposters x 8 shots x 200 clients).

Scores from eight different experts are used as our single expert outputs that we need to combine. FACE2 and FACE4 are two of the experts designed at University of Surrey which confirm or reject the claimed identity using face biometrics. Both use normalised correlation metric and global threshold. FACE2 uses full automatic registration while FACE4 uses semi-automatic registration, where the eyes of people were located manually and then their position perturbed by adding gaussian noise of 1 pixel standard deviation to the manually detected coordinates. SPEECH2 and SPEECH3 experts designed at IDIAP, Switzerland, base the identity on the speaker's voice characteristics. Experts number five to seven are from the Aristotle University of Thessaloniki and are based on elastic graph matching as a means of face representation. The methods differ in the internal threshold settings which respectively favour low rejection rates, low false acceptance rates, and equal error rates. SydneyCI is the eighth expert, from the University of Sydney. It is a face verification expert based on fractal image coding. The expert scores on the XM2VTS database are available from [65]. The database and the eight experts used in our experiments are described in [63].

Using the combiner performance on the evaluation set we select three different threshold values, as suggested by the Lausanne protocol. The three thresholds are calculated as follows:

\[
T_{FAE=0} = \arg \min_T (FRE|FAE = 0) \\
T_{ERR} = T_{FAE=FRE} = (T|FAE = FRE) \\
T_{FRE=0} = \arg \min_T (FAE|FRE = 0)
\]

where FAE and FRE are the false acceptance and false rejection error rates using the evaluation set.
### Table 6.1: Classification rates of the single experts

<table>
<thead>
<tr>
<th>Set type</th>
<th>sample type</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eval.</td>
<td>Average rate</td>
<td>51.83</td>
<td>86.67</td>
<td>94.34</td>
<td>59.45</td>
<td><strong>91.92</strong></td>
<td>96.50</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>73.33</td>
<td>94.33</td>
<td>100</td>
<td>83.83</td>
<td>96.50</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>3.66</td>
<td>100</td>
<td>94.35</td>
<td>18.90</td>
<td>100</td>
<td>96.50</td>
</tr>
<tr>
<td>Test</td>
<td>Average rate</td>
<td>51.65</td>
<td>86.25</td>
<td>92.82</td>
<td>50.42</td>
<td><strong>92.75</strong></td>
<td>96.90</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>99.75</td>
<td>72.50</td>
<td>93.25</td>
<td>100</td>
<td>85.50</td>
<td>97.25</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>3.56</td>
<td>100</td>
<td>92.38</td>
<td>18.83</td>
<td>99.99</td>
<td>96.54</td>
</tr>
<tr>
<td>threshold</td>
<td></td>
<td>0.0034</td>
<td>0.4179</td>
<td>0.1434</td>
<td>0.0168</td>
<td>0.3887</td>
<td>0.1597</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set type</th>
<th>sample type</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eval.</td>
<td>Average rate</td>
<td>99.35</td>
<td>55.75</td>
<td>99.38</td>
<td><strong>99.99</strong></td>
<td>50.00</td>
<td><strong>99.91</strong></td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>11.50</td>
<td>99.17</td>
<td>100</td>
<td>0</td>
<td>99.83</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>98.69</td>
<td>100</td>
<td>99.60</td>
<td>99.97</td>
<td>100</td>
<td>99.99</td>
</tr>
<tr>
<td>Test</td>
<td>Average rate</td>
<td>96.69</td>
<td>52.37</td>
<td>95.80</td>
<td><strong>99.26</strong></td>
<td>50.00</td>
<td><strong>99.26</strong></td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>95.25</td>
<td>4.75</td>
<td>93.00</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>98.13</td>
<td>100</td>
<td>98.50</td>
<td>98.52</td>
<td>100</td>
<td>98.52</td>
</tr>
<tr>
<td>threshold</td>
<td></td>
<td>0.4992</td>
<td>0.5158</td>
<td>0.5000</td>
<td>0.4999</td>
<td>1.0000</td>
<td>0.5000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set type</th>
<th>sample type</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eval.</td>
<td>Average rate</td>
<td>50.39</td>
<td>82.50</td>
<td>92.01</td>
<td>50.45</td>
<td>81.25</td>
<td>91.84</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>65.00</td>
<td>92.00</td>
<td>100</td>
<td>62.50</td>
<td>91.83</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>0.78</td>
<td>100</td>
<td>92.01</td>
<td>0.91</td>
<td>100</td>
<td>91.84</td>
</tr>
<tr>
<td>Test</td>
<td>Average rate</td>
<td>50.48</td>
<td>74.87</td>
<td>92.94</td>
<td>50.48</td>
<td>73.12</td>
<td>92.49</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>49.75</td>
<td>94.00</td>
<td>100</td>
<td>46.25</td>
<td>92.75</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>0.96</td>
<td>99.99</td>
<td>91.89</td>
<td>0.96</td>
<td>100</td>
<td>92.23</td>
</tr>
<tr>
<td>threshold</td>
<td></td>
<td>0.2016</td>
<td>0.7374</td>
<td>0.5006</td>
<td>0.1542</td>
<td>0.6376</td>
<td>0.4113</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set type</th>
<th>sample type</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eval.</td>
<td>Average rate</td>
<td>50.59</td>
<td>80.58</td>
<td>91.83</td>
<td>52.81</td>
<td>64.67</td>
<td>87.08</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>61.17</td>
<td>91.83</td>
<td>100</td>
<td>29.33</td>
<td>87.00</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>0.18</td>
<td>100</td>
<td>91.83</td>
<td>5.63</td>
<td>100</td>
<td>87.15</td>
</tr>
<tr>
<td>Test</td>
<td>Average rate</td>
<td>50.67</td>
<td>73.12</td>
<td>92.69</td>
<td>53.01</td>
<td>59.37</td>
<td>87.06</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>46.25</td>
<td>93.50</td>
<td>100</td>
<td>18.75</td>
<td>87.75</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>0.34</td>
<td>100</td>
<td>91.88</td>
<td>6.02</td>
<td>99.99</td>
<td>86.37</td>
</tr>
<tr>
<td>threshold</td>
<td></td>
<td>0.2899</td>
<td>0.8641</td>
<td>0.6331</td>
<td>0.4452</td>
<td>0.7839</td>
<td>0.6687</td>
</tr>
</tbody>
</table>
Next, using each of these threshold values we measure two error rates, client and imposter error rates. Also, for each threshold we calculate the total error rate as the average of the client and imposter rates. All the rates are measured on both the evaluation set and the test set.

The strength of the individual experts can be gleaned from the single expert results in Table 6.1. We note that the SPEECH3 expert is the best expert for $T_{EER}$ and $T_{FRE}=0$, while FACE4 is the best for $T_{FAE}=0$. The test set results confirm the performance on the evaluation set.

It is also interesting to note that for $T_{EER}$ the test set error rates using expert SPEECH3 is about 7 times better than the error rates achieved by the next best two experts, SPEECH2 and FACE4. Similar observations can be made for the $T_{FRE}=0$ thresholding strategy, although here the test set error rate of the best image expert is about two orders of magnitude worse than SPEECH3. Interestingly, for $T_{FAE}=0$ the test set performance of FACE4 is superior to any of the two speech experts. The difference is remarkably an order of magnitude.

When fusing the available experts we have the choice of combining any subset of them. The number of experts combined, $R$, could range from 2 up to 8 experts. For each value of $R$ and each threshold type, we find the set of experts that performs best on the evaluation set. Therefore, for the threshold type giving $FRE=0$ we find the mixture leading to the best imposter rate. Similarly for $T_{FAE}=0$ we find the combination of experts leading to the best client rate. Finally, for $T_{EER}$ we find the best combination of experts to optimise the equal error rate.

The two speech experts perform very well on $T_{FRE}=0$ and $T_{EER}$. However they have a very high error rate on $T_{FAE}=0$. Hence, for $T_{FRE}=0$ and $T_{EER}$ we expect the need to combine a lower number of experts than for $T_{FAE}=0$ to attain comparable performance.

We notice that although FACE2 has a lower performance than FACE4, it is preferred by both combiners, Sum and Vote. It is consistently selected along with the two speech experts. Actually Sum selected FACE2 and SPEECH3 for $T_{FRE}=0$ and $T_{EER}$ for all values of $R$. This leads us to conclude that we should not be looking for the best single expert to combine but the most complementary ones.
The fusion results obtained with the best combinations of experts for the Sum and Vote are shown in Table 6.3. Overall, we can see that as predicted theoretically, initially, for small number of experts \((R = 2)\) Sum is the best. As the number of experts increases Vote is better than Sum but for \(R > 5\) Sum outperforms Vote. However, in contrast to the results of the simulation experiments, in this parameter range we fail to observe a monotonic improvement of both fusion strategies. This is due to the fact that our experts have unequal strengths and therefore one of the assumptions made is not satisfied. Although we reduce the variance of the probability distribution of the fused decision being suboptimal by including more experts, the Bayes error rate decreases and so does the overall performance of the multiple classifier system.

Not surprisingly, the SPEECH3 expert, the best single expert, is always selected as one of the experts to be fused. For the Sum rule, the performance on the evaluation set indicates that the combination of 2 – 4 experts for \(T_{FE}=0\) and \(T_{EE}\) yield the results which are flat as indicated in Table 6.2. For \(T_{FAE}=0\), the best results with Sum in this range of \(R\) slightly oscillate. Thus the Sum rule appears to exhibit the same oscillatory behaviour that we noted in the simulation experiment with a non gaussian probability distribution \(p_{ij}[e_j(\omega_i|x)]\). This would suggest that that error distribution has relatively heavy tails.

From the test set results shown in Table 6.3 we find that the best performance for \(T_{FAE}=0\) using Sum is delivered when \(R=2\) as was suggested by the evaluation set. Similarly, for \(T_{FE}=0\) and \(T_{EE}\) the best performance is when \(R=4\), which is in agreement with the best performance on the evaluation set.

For the vote rule the evaluation set yields the highest performance rate when \(R=2\) or \(R=3\). Recall from the synthetic data experiments that we should normally avoid using an even number of experts for the vote. Thus we should select as the best combination experts FACE2, SPEECH2 and SPEECH3, indicated in bold in Table 6.2. Based on the test set we find that Vote fails to yield the best performance at \(R=2\) or 3. Its performance peaks when \(R=4\) and 5, which would not be considered given the information provided by the evaluation set.

Note in Table 6.2 that the performance of the sum rule does not improve as the num-
Table 6.2: The classification rate of the best mixture of experts obtained on the evaluation set. The mixture of experts used is indicated by their ID number below each rate. The best number of experts for each threshold type and combiner is indicated in bold. Below the expert ID's is the threshold used. To obtain expert names from their ID, refer to table 6.1.

<table>
<thead>
<tr>
<th>No. of Experts</th>
<th>FR=0 (impos rate)</th>
<th>FA=0 (clint rate)</th>
<th>EER (total rate)</th>
<th>FR=0 (impos rate)</th>
<th>FA=0 (clint rate)</th>
<th>EER (total rate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td></td>
<td></td>
<td></td>
<td>Vote</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>14</td>
<td>14</td>
<td>34</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>0.5022</td>
<td>0.7419</td>
<td>0.5023</td>
<td>0.4992</td>
<td>0.4995</td>
<td>0.4993</td>
</tr>
<tr>
<td></td>
<td>134</td>
<td>148</td>
<td>134</td>
<td>134</td>
<td>134</td>
<td>134</td>
</tr>
<tr>
<td></td>
<td>0.4885</td>
<td>0.5303</td>
<td>0.4886</td>
<td>0.4992</td>
<td>0.4995</td>
<td>0.4993</td>
</tr>
<tr>
<td></td>
<td>1348</td>
<td>1348</td>
<td>1348</td>
<td>1348</td>
<td>1348</td>
<td>1348</td>
</tr>
<tr>
<td></td>
<td>0.5173</td>
<td>0.5509</td>
<td>0.5174</td>
<td>0.4967</td>
<td>0.4995</td>
<td>0.4968</td>
</tr>
<tr>
<td></td>
<td>12348</td>
<td>12348</td>
<td>12348</td>
<td>12348</td>
<td>12348</td>
<td>12348</td>
</tr>
<tr>
<td></td>
<td>0.4230</td>
<td>0.4766</td>
<td>0.4231</td>
<td>0.4967</td>
<td>0.4995</td>
<td>0.4968</td>
</tr>
<tr>
<td></td>
<td>123468</td>
<td>123468</td>
<td>123468</td>
<td>123478</td>
<td>123478</td>
<td>123478</td>
</tr>
<tr>
<td></td>
<td>0.4009</td>
<td>0.5029</td>
<td>0.4010</td>
<td>0.2899</td>
<td>0.4995</td>
<td>0.4002</td>
</tr>
<tr>
<td>7</td>
<td>99.3625</td>
<td>97.6667</td>
<td>99.8329</td>
<td>41.6325</td>
<td>98.5000</td>
<td>98.8329</td>
</tr>
<tr>
<td></td>
<td>1234568</td>
<td>1234568</td>
<td>1234568</td>
<td>1234678</td>
<td>1234678</td>
<td>1234678</td>
</tr>
<tr>
<td></td>
<td>0.3656</td>
<td>0.4856</td>
<td>0.3956</td>
<td>0.2899</td>
<td>0.5000</td>
<td>0.4782</td>
</tr>
</tbody>
</table>
### 6.4. Real Data Experiments

Table 6.3: The test set classification rate of the best mixture of experts selected using the evaluation set.

<table>
<thead>
<tr>
<th>No. of experts</th>
<th>Sample type</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
<th>FR=0</th>
<th>FA=0</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Average rate</td>
<td>99.30</td>
<td>99.62</td>
<td>99.31</td>
<td>97.59</td>
<td>97.22</td>
<td>97.34</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>99.75</td>
<td>100</td>
<td>99.75</td>
<td>95.25</td>
<td>94.50</td>
<td>94.75</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>98.86</td>
<td>99.24</td>
<td>98.86</td>
<td>99.92</td>
<td>99.93</td>
<td>99.93</td>
</tr>
<tr>
<td>3</td>
<td>Average rate</td>
<td>99.38</td>
<td>99.45</td>
<td>99.38</td>
<td>99.34</td>
<td>99.22</td>
<td>99.21</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>98.75</td>
<td>98.50</td>
<td>98.50</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>98.76</td>
<td>98.90</td>
<td>98.76</td>
<td>99.92</td>
<td>99.93</td>
<td>99.93</td>
</tr>
<tr>
<td>4</td>
<td>Average rate</td>
<td><strong>99.44</strong></td>
<td>98.93</td>
<td><strong>99.44</strong></td>
<td>99.55</td>
<td>99.22</td>
<td>99.55</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>98.25</td>
<td>100</td>
<td>99.25</td>
<td>98.50</td>
<td>99.25</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>98.88</td>
<td>99.60</td>
<td>98.88</td>
<td>99.84</td>
<td>99.93</td>
<td>99.85</td>
</tr>
<tr>
<td>5</td>
<td>Average rate</td>
<td>99.43</td>
<td>99.32</td>
<td>99.44</td>
<td>99.55</td>
<td>99.22</td>
<td>99.55</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>98.75</td>
<td>100</td>
<td>99.25</td>
<td>98.50</td>
<td>99.25</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>98.87</td>
<td>99.90</td>
<td>98.87</td>
<td>99.94</td>
<td>99.93</td>
<td>99.85</td>
</tr>
<tr>
<td>6</td>
<td>Average rate</td>
<td>99.42</td>
<td>98.00</td>
<td>99.42</td>
<td>99.10</td>
<td>99.10</td>
<td>99.29</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>96.00</td>
<td>100</td>
<td>100</td>
<td>98.25</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>98.83</td>
<td>100</td>
<td>98.83</td>
<td>98.20</td>
<td>99.95</td>
<td>98.58</td>
</tr>
<tr>
<td>7</td>
<td>Average rate</td>
<td>99.16</td>
<td>98.74</td>
<td>99.40</td>
<td>71.16</td>
<td>99.47</td>
<td>99.17</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>97.50</td>
<td>100</td>
<td>100</td>
<td>99.00</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>98.33</td>
<td>99.97</td>
<td>98.81</td>
<td>42.32</td>
<td>99.95</td>
<td>98.34</td>
</tr>
<tr>
<td>8</td>
<td>Average rate</td>
<td>94.06</td>
<td>98.60</td>
<td>99.18</td>
<td>63.22</td>
<td>97.86</td>
<td>97.18</td>
</tr>
<tr>
<td></td>
<td>Client rate</td>
<td>100</td>
<td>97.25</td>
<td>100</td>
<td>100</td>
<td>95.75</td>
<td>97.75</td>
</tr>
<tr>
<td></td>
<td>Imposter rate</td>
<td>88.11</td>
<td>99.95</td>
<td>98.35</td>
<td>6.44</td>
<td>99.97</td>
<td>96.60</td>
</tr>
</tbody>
</table>
ber of experts increases. This would suggest that Vote might be preferable to Sum. However, as Vote also fails to improve as the number of experts increases, we know from the theoretical expectations that we are most likely combining experts of unequal strength. In these circumstances a more conservative option is to adopt the sum rule, as the relationship of its results on the evaluation and test sets appears to be less volatile. This strategy, based on the results of this chapter, gives quite good performance on the test set, almost matching the absolute best performance achievable aposteriori.

6.5 Discussion

We showed in Sections 6.2 and 6.3 that for Gaussian distributions of estimation errors the Sum fusion strategy will always outperform the majority vote. However, for heavy tail distributions the majority vote may give better results than Sum. This may happen when the margin between the two aposteriori class probabilities is small. However, even if the margin is reasonable, Vote may be superior to Sum when the number of experts is small. This behaviour may explain the observations made by a number of researchers when experimenting with these two combination rules whereby none of the rules was uniformly better than the other.

The above ideal behaviour of the two rules applies only under the assumption that the expert errors are independent and identically distributed. Moreover, the target aposteriori class probability estimated by each expert must be the same. In practice, none of the assumptions are likely to hold exactly. The effect of the first two not being satisfied is likely to be reflected in slower rates of performance improvement than those predicted by the theory, as the number of experts increases.

The implications of the third assumption being invalid are much more serious. Here, as we are adding more experts to reduce variance, we may be including experts which inject additional ambiguity in the output of the multiple classifier system. Thus any gains in reduced variance will have to be weighed against potential losses due to increased ambiguity. This raises a serious methodological question of how to decide when the fusion of more experts starts being counterproductive. One possibility is to obtain several designs as in Section 6.4 and choose the best solution using the performance
rates on an evaluation set. However, this has the drawback that we lose the nice property of the Sum and Vote strategies, namely that no training is required. Another possibility would be to try to estimate the Bayes error and the estimation error variance for each expert. However, it would be practically impossible to estimate these quantities point wise. Thus at best, one would be able to decide whether another expert's opinion is worth considering only on average. This should still result in improved performance and such a scheme is currently being developed. Alternatively, other schemes such as the expert output clustering method of Roli and Giancarlo [38] could be considered as a method for selecting which experts to fuse.

6.6 Conclusion

The relationship of the Sum and Vote classifier combination rules was investigated. The main advantage of these rules is their simplicity and their applicability without the need for training the classifier fusion stage. We showed analytically that, for normally distributed estimation errors, Sum always outperforms Vote, whereas for heavy tail distributions Vote may outperform Sum. We then confirmed our theoretical predictions by experiments on both synthetic and real data. In the synthetic experts case, we showed for Gaussian error distributions that, as expected, Sum outperforms Vote. The differences in performance are particularly significant at high estimation noise levels. However, for heavy tail distributions the superiority of Sum may be eroded for any number of experts if the margin between the two a posteriori class probabilities is small or for a small number of cooperating experts even when the margin is large. In the latter case, once the number of experts exceeds a certain threshold, Sum tends to be superior to Vote. Experiments on real data supported the general findings but also showed the effect of the usual assumptions of conditional independence, identical error distributions and common target outputs of the experts not being fully satisfied.
Chapter 7

A PHC System for \(k - NN\) Classifiers

Two PHC system design methods are investigated. An improved modified Bagging technique and a new feature based PHC design method are proposed. The two PHC design strategies are combined to improve bagging at large training set size.

7.1 Introduction

In Chapter 3 we presented several methods for creating Parallel Homogeneous Classifier systems, PHC. Bagging is one particular example of such methods. To evaluate the performance of certain fusion strategies, in Chapter 5 we experimented with PHC systems designed using bagging procedures. In this chapter we focus on the methods outlined in Chapter 3 as a tool for building PHC systems. We investigate training set manipulation and feature set manipulation techniques for PHC system construction. We analyse the results obtained with the bagging and modified bagging methods proposed in Chapter 3 and experimented in Chapter 5, from the point of training set manipulation technique. As far as feature set manipulation is concerned we experiment with conventional and the proposed system based PHC design method introduced in Chapter 3 and detailed here.
Our investigation of bagging $k$–NN experts confirmed the findings of others that some bagging methods may not yield a successful fusion system. We propose modified methods that improve the performance of these systems over the single expert. The modified bagging methods improve bagging when very small training set exists. For training sets equal to, or larger than 80 samples, bagging does not suffer from sampling problems, therefore modified bagging does not improve bagging. Under large set size bagging degrades because $k$–NN experts become stable and bagging does not yields diverse bootstrap sets. A solution for the large set size is also proposed.

For the feature set manipulation technique category we develop a novel feature selection strategy by adopting a completely novel design philosophy. We take the view that the design of individual experts and fusion cannot be solved in isolation. This premise leads to a completely different design methodology whereby each expert is constructed as part of the global design of a final multiple expert system. The design process involves jointly adding new experts to the multiple expert architecture and adding new features to each of the experts in the architecture. We propose two distinct design strategies which we refer to as parallel and serial. In the parallel approach we utilise a pool of features by building a number of experts simultaneously by distributing the features available to the experts in a "card dealing" manner. In this approach the dimensionality of the feature spaces in which the experts operate increases in a balanced way and also the expert strengths are reasonably balanced. In contrast, in the serial approach any expert is allowed to absorb new features as long as the system performance continues to improve. This inevitably means that the first experts take better and more features than the experts added later, leading to imbalance both in feature space dimensionality and individual expert performance. We show that the fusion strategy yielding the best results appears to be different for the two applications investigated. The results are compared to the conventional and the optimised conventional system.

In the next section we discuss bagging as a method based on training set manipulation, followed by an analysis of the modified bagging results of Chapter 5 in Section 7.3. In Section 7.4 we discuss feature set manipulation techniques for building PHC systems. A novel feature based multiexpert system design method is presented in Section 7.5. In Section 7.6 we integrate two PHC design techniques to improve the $k$–NN expert
under large training set size, and compare the feature subset selection using the novel method to the feature selection using the conventional method. The chapter is drawn to conclusion in Section 7.7.

7.2 Bagging as a Training Set Manipulation Technique to Build PHC Systems

Bagging has been successfully applied to practical cases to improve the performance of unstable classifiers. A sample of such papers includes [28, 22, 62]. Many authors have investigated its merits and compared bagging to boosting or other methods [41, 66, 62, 24, 91, 15, 72, 19, 3, 11]. In [18] Breiman argued that bagging would not improve a nearest neighbour (NN) classifier, because it is stable. He confirmed this experimentally by showing that, when bagging NN classifiers, good results were achieved on 3 data sets out of 6. But, when bagging decision trees, better error rate on all 6 data sets was observed. Most research to date has been directed towards applying bagging to unstable classifiers, such as decision trees and neural networks. However, Skalak [87] did apply a technique similar to bagging to a NN classifier successfully. This was done by forcing the classifier to be unstable. In Skalak's work the classifier is carefully designed by selecting the best representative sample per class which achieves the minimum error rate. His design goal is to construct classifiers that independently classify well, but differently and quickly. He experimentally proves that the component classifier accuracy and diversity does not always lead to combiner accuracy. The simplest and most accurate classifiers are selected using a stratified random sampling. In contrast to bagging which has a large number of classifiers, he works with a small number of classifiers, 2 or 3, that are coarse in regard to the hypothesis space.

Breiman's results [18] show that bagging more than 25 replicas does not further improve the performance. He also notes that fewer replicas are required when the classifier outputs are numerical results rather than labels, but more are required as the number of classes increases. Regarding the bootstrap training set size, he used the size equal to the cardinality of the original training set and his tests showed no improvement when the boot training set was double the size of the original training set. Skurichina and Duin
investigated instability and its effect on bagging for linear discriminant classifiers. They found that the optimum number of bootstrap replicas depends on the classifier and data set used. They also propose a tool for measuring classifier instability. They show that techniques for stabilisation exist but some improve performance rather than stabilise. Bagging is such a technique, while regularization is a stabilising technique. They measure instability by calculating the changes in classification of a test set caused by bootstrap replicates of the original training set. It is possible to measure instability using the training set as a test set. Then we get a training set instability measure. Their results show that generalisation error and instability are related. More unstable classifiers perform worse. They notice the dependency of the classifier stability on the composition of the training set. The usefulness of bagging can be predicted by considering the training set instability measure. Classifiers are most unstable if the size of the training set is comparable with the data dimensionality. An exception arises when data set is very large or very small. The first case a classifier is very strong and stable, while the second it is stable and weak, i.e. it always mis-classifies. The bagged classifier can be more stable or less stable than the original classifier, depending on the number of bootstrap replicates used to create the bagged classifier. Their tests were also done on Nice Bagging, which aggregates components trained using the best bootstrap sets. The best bootstrap sets are the ones that lead to an error rate lower than or equal to the error rate when the original training set is used. Nice Bagging is more stable than bagging. The results show that, in general, nice bagging does not give better results than bagging. Depending on the classifier and data set used an optimum number of bootstrap replicates must be used. It has been shown that Bootstrap replicates can help avoid outliers, giving another reason for the success of bagging. Domingos \cite{Domingos1998} states that bagging works because it shifts the prior to a more appropriate region of model space. An over-fit learner is unstable hence bagging is successful with it.

In Chapter 5 MProduct and Moderation were experimentally validated using bagged \textit{k} - \textit{NN} experts. However, the tables also contain the results for three modified bagging methods presented in Section 3.2.5. In Section 7.3 we focus on these results and analyse the performance achieved by the different bagging methods. We confirm that for larger data sets bagging \textit{k} - \textit{NN} classifiers using the standard method of constructing boot
sets does not improve the performance. However, for smaller data sets, bagging can be beneficial, but for very small data sets any gains from bagging vanish once again. Most interestingly, we found that while bagging can counteract spatial instability of points in small sample sets, it is detrimentally affected by swings in population bias. This observation led us to modify the bagging procedure itself. We show that using the modified method of constructing the boot sets much more significant gains can be achieved from bagging for small sample sets.

7.3 Population Bias Control, Applied to Bagging

In this section we analyse the modified bagging results for the experiments of Chapter 5. The modified bagging procedures were presented in Chapter 3. The results generally indicate that the modification to the standard bagging method designed to avoid population bias leads to substantial performance gains, especially under very small sample size conditions. The choice of the modification method used depends on whether prior knowledge exists or not. If no prior knowledge exists then modified bagging method 4, i.e. insuring that all classes exist in the bootstrap set, yields the best results.

The results from MProduct and Sum were relatively close and the difference in their performance was not significant enough to justify drawing separate conclusions for these two fusion rules. Since MProduct exhibited superior performance most of the time, in this section we document only the results obtained with this fusion method to minimise the information overload.

The relative performance measures of the different bagging methods defined in Subsection 5.2.4, for all data sets, are reported in Tables 7.1 to 7.7, in two sets of four columns. The left set contains the results of different bagging methods as compared to the performance on the learning set. The right set of columns present the results as a comparison to regular bagging.

Looking at the left set of columns we notice some common trends to all data sets.

- All rules significantly outperform the single expert at set sizes 2 and 3.
Table 7.1: Relative performance measures parameterised by different bagging methods and learning set sizes for the Synthetic data.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Learn set type</th>
<th>Bagging Method</th>
<th>No. of learning samples</th>
<th>Comparison to single expert</th>
<th>Comparison to regular bagging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>10 20 40 80</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Synth</td>
<td>Regular</td>
<td>-2.62 7.29 3.50 0.09</td>
<td>-10.21 7.61 4.39 0.02</td>
<td>-7.30 0.35 0.92 -0.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13.38 12.97 6.09 1.20</td>
<td>15.60 6.13 3.61 1.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.02 6.17 4.36 0.14</td>
<td>5.50 -1.20 0.89 0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>87.67 89.34 91.02 92.28</td>
<td>87.35 90.11 91.34 92.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modified</td>
<td></td>
<td>0.63 10.11 3.11 -0.41</td>
<td>-7.88 28.09 25.00 -2.16</td>
<td>-1.95 -0.98 1.06 -1.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.97 10.40 4.92 -0.21</td>
<td>5.37 0.43 1.88 0.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.28 11.37 5.47 1.55</td>
<td>2.67 1.40 2.44 1.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.40 9.87 4.47 -0.09</td>
<td>2.79 -0.27 1.41 0.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>89.62 89.97 91.59 92.30</td>
<td>89.68 90.99 91.85 92.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>85.95 96.41 98.08 90.16</td>
<td>85.13 97.45 98.54 99.15</td>
</tr>
</tbody>
</table>

Baseline Classification rate 87.35 90.11 91.34 92.29

Table 7.2: Relative performance measures parameterised by different bagging methods and learning set sizes for the Seismic data.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Learn set type</th>
<th>Bagging Method</th>
<th>No. of learning samples</th>
<th>Comparison to single expert</th>
<th>Comparison to regular bagging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>10 20 40 80</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Seis.</td>
<td>Regular</td>
<td>-5.82 28.78 24.20 -1.08</td>
<td>-7.88 28.09 25.00 -2.16</td>
<td>-1.95 -0.98 1.06 -1.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>74.28 45.22 34.80 2.70</td>
<td>75.70 23.08 13.98 3.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64.91 32.47 30.00 2.16</td>
<td>66.84 5.17 7.65 3.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>85.95 96.41 98.08 90.16</td>
<td>85.13 97.45 98.54 99.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modified</td>
<td></td>
<td>5.53 31.68 27.21 3.05</td>
<td>-2.12 28.23 25.92 0.00</td>
<td>-8.10 -5.06 -1.77 -3.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.15 33.48 31.07 7.11</td>
<td>-1.46 5.56 5.30 4.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.84 23.95 24.26 -0.51</td>
<td>-3.90 -2.53 -4.04 -3.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>96.26 96.90 97.91 99.10</td>
<td>96.47 97.88 98.48 99.13</td>
</tr>
</tbody>
</table>

Baseline Classification rate 96.26 96.90 97.91 99.10
Table 7.3: Relative performance measure parameterised by different bagging methods and learning set sizes for the Diabetes data.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Learn set type</th>
<th>Bagging Method</th>
<th>No. of learning samples</th>
<th>Comparison to single expert</th>
<th>Comparison to regular bagging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>Diab.</td>
<td>Regular</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>-0.13</td>
<td>-1.31</td>
<td>1.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>-0.87</td>
<td>-0.24</td>
<td>1.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>3.52</td>
<td>1.54</td>
<td>2.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0.73</td>
<td>-0.47</td>
<td>1.48</td>
</tr>
<tr>
<td>Baseline Classification rate</td>
<td></td>
<td></td>
<td>64.77</td>
<td>67.15</td>
<td>68.19</td>
</tr>
<tr>
<td></td>
<td>Modified</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>-0.22</td>
<td>2.18</td>
<td>3.44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>-1.61</td>
<td>2.63</td>
<td>2.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>1.25</td>
<td>2.71</td>
<td>3.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0.14</td>
<td>2.26</td>
<td>3.33</td>
</tr>
<tr>
<td>Baseline Classification rate</td>
<td></td>
<td></td>
<td>66.34</td>
<td>67.18</td>
<td>67.81</td>
</tr>
</tbody>
</table>

Table 7.4: Relative performance measure parameterised by different bagging methods and learning set sizes for the Wine data.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Learn set type</th>
<th>Bagging Method</th>
<th>No. of learning samples</th>
<th>Comparison to single expert</th>
<th>Comparison to regular bagging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>Wine</td>
<td>Regular</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>-3.83</td>
<td>19.46</td>
<td>16.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>-7.11</td>
<td>23.98</td>
<td>15.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>38.78</td>
<td>39.48</td>
<td>14.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>27.06</td>
<td>26.66</td>
<td>16.42</td>
</tr>
<tr>
<td>Baseline Classification rate</td>
<td></td>
<td></td>
<td>75.47</td>
<td>85.14</td>
<td>91.75</td>
</tr>
<tr>
<td></td>
<td>Modified</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>6.56</td>
<td>22.56</td>
<td>16.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>-15.90</td>
<td>26.82</td>
<td>15.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>7.93</td>
<td>27.52</td>
<td>18.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>6.94</td>
<td>24.34</td>
<td>16.07</td>
</tr>
<tr>
<td>Baseline Classification rate</td>
<td></td>
<td></td>
<td>86.11</td>
<td>88.27</td>
<td>91.34</td>
</tr>
</tbody>
</table>
Table 7.5: Relative performance measure parameterised by different bagging methods and learning set sizes for the Ionosphere data.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Learn set type</th>
<th>Bagging Method</th>
<th>Comparison to single expert</th>
<th>Comparison to regular bagging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>No. of learning samples</td>
<td>No. of learning samples</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Iono.</td>
<td>Regular</td>
<td>1</td>
<td>-1.23</td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0.96</td>
<td>8.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>2.40</td>
<td>8.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>1.03</td>
<td>7.99</td>
</tr>
<tr>
<td>Baseline Classification rate</td>
<td>67.53</td>
<td>68.60</td>
<td>70.25</td>
<td>75.93</td>
</tr>
</tbody>
</table>

Table 7.6: Relative performance measure parameterised by different bagging methods and learning set sizes for the BCW data.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Learn set type</th>
<th>Bagging Method</th>
<th>Comparison to single expert</th>
<th>Comparison to regular bagging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>No. of learning samples</td>
<td>No. of learning samples</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>BCW</td>
<td>Regular</td>
<td>1</td>
<td>-6.71</td>
<td>14.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>-29.90</td>
<td>13.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>26.70</td>
<td>22.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>18.48</td>
<td>21.55</td>
</tr>
<tr>
<td>Baseline Classification rate</td>
<td>89.78</td>
<td>92.66</td>
<td>94.47</td>
<td>95.44</td>
</tr>
</tbody>
</table>

|          | Modified       | 1              | -4.12 | 14.06 | 7.35 | -1.46 | 5.78 | 0.08 | 0.89 | 0.32 |
|          |                | 2              | 1.90 | 14.13 | 8.17 | -1.14 | 5.45 | 1.85 | 0.61 | 2.04 |
|          |                | 3              | 1.55 | 15.65 | 7.91 | 0.61 | 4.50 | 1.85 | 0.61 | 2.04 |
|          |                | 4              | -4.71 | 14.28 | 7.89 | -1.00 | -0.56 | 0.26 | 0.58 | 0.46 |
| Baseline Classification rate | 93.28 | 93.32 | 94.65 | 95.48 | 93.00 | 94.26 | 95.04 | 95.41 |
Table 7.7: Relative performance measure parameterised by different bagging methods and learning set sizes for the Iris data.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Learn set type</th>
<th>Bagging Method</th>
<th>No. of learning samples</th>
<th>Comparison to single expert</th>
<th>Comparison to regular bagging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>Iris</td>
<td>Regular</td>
<td>1</td>
<td>-9.20</td>
<td>3.65</td>
<td>5.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>-24.99</td>
<td>4.09</td>
<td>8.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>42.50</td>
<td>24.38</td>
<td>11.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>34.61</td>
<td>13.17</td>
<td>5.67</td>
</tr>
<tr>
<td>Baseline Classification rate</td>
<td>Regular</td>
<td>80.91</td>
<td>91.35</td>
<td>94.07</td>
<td>96.27</td>
</tr>
<tr>
<td></td>
<td>Modified</td>
<td>1</td>
<td>-6.73</td>
<td>11.07</td>
<td>6.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>-10.80</td>
<td>11.85</td>
<td>10.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>4.66</td>
<td>15.96</td>
<td>13.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0.00</td>
<td>14.59</td>
<td>10.15</td>
</tr>
<tr>
<td>Baseline Classification rate</td>
<td>Modified</td>
<td>90.34</td>
<td>92.15</td>
<td>94.54</td>
<td>96.13</td>
</tr>
</tbody>
</table>

- Method 3 outperforms all methods, followed by Method 4.
- For regular learning sets, methods 3 and 4 significantly improve on the single expert at set size 1.
- Modifying the learning set impacts beneficially on the single expert performance. The effect disappears as the set size increases.

Bagging an unmodified learning set leads to a more significant improvement than bagging a modified learning set. Interestingly, when modified learning set is used the performance of all four methods becomes relatively equal.

We notice that regular bagging degrades significantly below the single expert for the very small set size. Hence a bootstrap bias control is a necessity if we wish to improve the single expert in such circumstances.

Referring to the right set of four columns in Tables 7.1 to 7.7, which compare the modified to the regular bagging methods we notice the following:

- At the largest set size there is no difference between the different bagging methods.
Table 7.8: The number of times a bagging method achieves a rank among the different methods, summed over all data sets. Rank 1 is the highest.

<table>
<thead>
<tr>
<th>method</th>
<th>regular learning set</th>
<th>modified learning set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prior knowledge</td>
<td>No Prior knowledge</td>
</tr>
<tr>
<td>Rank 1</td>
<td>3 1 22 2</td>
<td>4 6 19</td>
</tr>
<tr>
<td>Rank 2</td>
<td>2 6 3 20</td>
<td>15 7 5</td>
</tr>
<tr>
<td>Rank 3</td>
<td>17 7 1 3</td>
<td>9 15 4</td>
</tr>
<tr>
<td>Rank 4</td>
<td>6 14 2 3</td>
<td>- - -</td>
</tr>
</tbody>
</table>

- For two data sets bagging method 2 improves the performance but on the whole it is not very useful.

- Bagging method 3 is significantly better at set sizes 1, 2 and occasionally at size 3. This also holds true when modified learning set is used at sizes 1 and 2.

- Bagging method 4 is significantly better when regular learning set is used at set sizes 1 and 2. Moreover, it is never significantly worse.

Reducing the learning set size increases the error rate of all classifiers, whether single experts or combiners. At very small set sizes some bootstrap sets contain samples from a single class only. Such heavily biased training sets result in higher classification errors which degrade the combiner error rate. Our proposed methods work to alleviate this deficiency and result in significant improvements.

Method 3 is superior and consistently outperforms all the other methods as shown in Table 7.8. But if prior knowledge regarding the class distribution is not available we can not use it. If we exclude method 3 we note that method 4 achieves superior performance if the learning set is not modified. Otherwise, if a modified learning set is used then method 2 slightly outperforms method 4, as shown in Table 7.8. This slight difference warrants a more detailed look at the performance of the methods in Tables 7.1 to 7.7. A review of these tables reveals that method 4 is superior to method
Table 7.9: The number of times a method achieves a 5% significant improvement or degradation over the single expert out of 56 cases. (all data sets and learning set types)

<table>
<thead>
<tr>
<th>Method</th>
<th>regular learning set</th>
<th>modified learning set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+5% improve</td>
<td>-5% degrade</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.10: The number of times a method achieves a 5% significant improvement or degradation over the regular bagging method 1 out of 56 cases. (all data sets and learning set types)

<table>
<thead>
<tr>
<th>Method</th>
<th>regular learning set</th>
<th>modified learning set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+5% improve</td>
<td>-5% degrade</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

2. Although method 2 accumulated a larger number of higher rank positions than method 4, most of these higher ranks were due to a small improvement over method 4. However, when method 4 ranked above method 2 it outperformed method 2 by a large margin. Hence method 4 is preferable to method 2 overall. The comparison of the results obtained with the different bagging methods to the regular bagging method confirms the above finding. Thus in general methods 3 and 4 significantly improve the performance over that of the single expert at all set sizes except at size 4.

Another interesting observation can be made in Table 7.9. Methods 3 and 4 not only significantly improve the single expert more often than methods 1 and 2, they also never perform worse than the single expert. The same holds when they are compared to the regular bagging method in Table 7.10.
Table 7.11: Results obtained by Skalak [87], Brieman [18], Quinlan [72] and Opitz [66].

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Diab.</td>
</tr>
<tr>
<td>Prototype Selection (Skalak)</td>
<td>73.9</td>
</tr>
<tr>
<td>Single NN</td>
<td>69.7</td>
</tr>
<tr>
<td>Bagged classification tree (Brieman)</td>
<td>76.1</td>
</tr>
<tr>
<td>Single classification tree</td>
<td>74.7</td>
</tr>
<tr>
<td>Bagging C4.5 (Quinlan)</td>
<td>76.37</td>
</tr>
<tr>
<td>Single C4.5</td>
<td>74.61</td>
</tr>
<tr>
<td>Bagging neural networks (Opitz)</td>
<td>76.8</td>
</tr>
<tr>
<td>Single neural network</td>
<td>76.4</td>
</tr>
</tbody>
</table>

Table 7.11 shows the amount of improvement that others achieved via bagging. These results can be compared to our results in Table 7.12. Skalak [87] compares the performance of a standard nearest neighbour (NN) classifier to the nearest neighbour classifier constructed by sampling through prototype selection (P.S.N.N). Brieman [18] bags classification trees under a moderate size training set condition. Quinlan [72] bags C4.5 classifiers while Opitz [66] bags neural networks. All use 10-fold cross validation with 90% of the original samples constituting the training set. The reason we compare to the results of Skalak is because his method can be seen as a variant of bagging, although the method by which the classifier samples are selected is different from bagging.

In our experiments, for each set size, the number of nearest neighbours, \( k \), is found automatically as the square root of the number of training sets rounded to the closest integer. This leads to using odd values for \( k \) at the smallest and largest set sizes and even \( k \) for the middle set sizes. In Chapter 5 an investigation of the bagging and learning set performance at different values of \( k \) for each set size indicated that for even \( k \) the single expert performance was worse than when the next odd \( k \) was selected, as shown in Figures 5.8 and 5.9. In other words the bagging advantage over the learning set is reduced when \( k \) is forced to be odd. Thus one of the benefits of bagging is that
Table 7.12: Comparison of results obtained through MProduct, under bagging method 3 for varying set sizes, to learning set (single expert) results. Learning set is constructed using the regular (random sampling) method.

<table>
<thead>
<tr>
<th>Data name</th>
<th>MProduct size 4</th>
<th>No fusion size 4</th>
<th>MProduct size 3</th>
<th>No fusion size 3</th>
<th>MProduct size 2</th>
<th>No fusion size 2</th>
<th>MProduct size 1</th>
<th>No fusion size 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diab.</td>
<td>70.59</td>
<td>70.33</td>
<td>69.06</td>
<td>68.19</td>
<td>67.66</td>
<td>67.15</td>
<td>65.92</td>
<td>64.77</td>
</tr>
<tr>
<td>BCW</td>
<td>95.42</td>
<td>95.44</td>
<td>95.03</td>
<td>94.47</td>
<td>94.30</td>
<td>92.66</td>
<td>92.65</td>
<td>89.78</td>
</tr>
<tr>
<td>Ion.</td>
<td>75.67</td>
<td>75.93</td>
<td>73.58</td>
<td>70.25</td>
<td>71.48</td>
<td>68.60</td>
<td>68.39</td>
<td>67.53</td>
</tr>
<tr>
<td>Wine</td>
<td>93.95</td>
<td>93.56</td>
<td>93.16</td>
<td>91.75</td>
<td>91.01</td>
<td>85.14</td>
<td>84.83</td>
<td>75.47</td>
</tr>
<tr>
<td>Iris</td>
<td>96.07</td>
<td>96.27</td>
<td>94.70</td>
<td>94.07</td>
<td>93.49</td>
<td>91.35</td>
<td>88.76</td>
<td>80.91</td>
</tr>
</tbody>
</table>

it produces consistent performance for odd and even values of k.

7.4 Feature Set Manipulation Technique to Build PHC Systems

As mentioned in Chapter 3 one method for designing PHC systems is to use different feature sets for the component classifiers. In spite of the difference in the underlying fusion principles, almost all the techniques proposed in the literature share the same premise, namely that the experts involved in fusion have already been designed using conventional methodology. This implies that the fusion is considered as an after thought, addressing the question how the experts existing already can be combined in the most effective way. The design of individual experts which involves the choice of classifier (e.g. Gaussian, k-nearest neighbour, radial basis function, multi-layer perceptron) and the corresponding feature space is carried out using the standard criterion of classification performance the experts would achieve individually. It is at the stage of fusion, when a classifier combination scheme is being devised, that the merits of
the individual experts are assessed and the expert opinions combined accordingly. In some cases such a design approach may adhere to a deliberate policy of breaking up a complex design problem into simpler tasks. However, in most cases it will simply be a reflection of the incremental development of the design methodology.

It is generally believed that the fusion of multiple experts is beneficial if they voice independent opinions as indicated in Chapter 3. While it may be largely true that combining independent experts leads to performance improvement, it is very easy to demonstrate that even statistically dependent experts can be combined profitably. This is why the problem of choosing a suitable fusion strategy is so difficult. As we do not know the characteristics of the respective experts, it is far from clear which strategy should be used as the fusion schemes which are able to capitalise on expert independence will not necessarily fare well when the experts outputs are dependent.

In the next section we experiment with a completely novel design philosophy. We take the view that the design of individual experts and fusion cannot be solved in isolation. This premise leads to a completely different design methodology whereby each expert is constructed as part of the global design of a final multiple expert system. As the design of component classifiers is optimised using a common performance criterion, that is the error probability of the multiple expert system, it is reasonable to expect that the final design will be at least as good as the fusion of individually designed experts and hopefully much better.

The design process involves jointly adding new experts to the multiple expert architecture and adding new features to each of the experts in the architecture. The feature selection problem itself is of combinatorial complexity and it is clear that the optimisation over different architectures and individual experts in each architecture will be computationally explosive. For this reason in this initial study we use only the plus-l-take away-r and the simplistic sequential forward feature selection methods to build the individual experts and the fusion system. Results for the forward selection method are similar and are reported in [10].

We experiment with two classification problems: breast cancer detection [17], and Seismic data interpretation. In both cases we show that the proposed integrated design
7.5 Multiple Expert System Design by Combined Feature Selection and Probability Level Fusion

In this section we propose a novel design method to build a combiner system based on the feature set manipulation technique. In our multiple expert system, in order to achieve diversity, we propose a distinct set of features for each expert. Hence experts are designed by assigning features to them. At the end of the design process each expert will hold a distinct set of features. We propose two methods of building the experts of the system, parallel and serial then compare their performance to the optimised conventional system. The proposed methods differ from the conventional method in the optimisation criteria based on which a feature is accepted or rejected. The parallel, serial and optimised conventional methods are named according to the sequence by which features are added to the component experts of the system. In the following subsections these design methods are explained.

Parallel system

In the parallel method, at any stage of the design an expert is allowed to take the feature that will deliver the best system performance. In order to encourage a balance in the performance of individual experts each expert is allowed to take only one feature at a time. The number of experts is a system parameter that is specified at the beginning of the design process. since the minimum number of features that we experiment with is nine, we heuristically consider the use of four experts in the parallel design method to be sufficient. When a feature is introduced to an expert the system classification rate is estimated using a validation set. The feature that optimises the system performance is assigned to the expert. When the first feature is introduced the system consists of only one expert, hence no fusion occurs. The second feature is introduced to a new expert and the feature that results in the maximum system performance is selected for this
second expert. When all experts are assigned a feature we go back to the first expert and assign it a second feature. This process continues until all features are selected or the termination condition is reached. A possible termination condition is when the addition of any of the remaining features fails to improve the system performance. The parallel method allows best features to be distributed among many experts. When all experts use similar criterion functions we expect the experts to have approximately equal strength.

Serial system

In the serial method the first expert is allowed to take all the features that it needs to achieve maximum performance. Next, if any feature remains, a second expert is built from the remaining features. It is now, when a feature is added to the second expert, that we observe the effect of fusion. The process of building experts continues until all features are selected or a termination condition is reached.

This method allows the first expert to take the best features. Therefore the subsequent experts will be weaker. This will lead into a more diverse set of experts, as far as their relative strength is concerned, compared to the experts in the parallel system. Logistic regression or any kind of weighted fusion is expected to perform better than the simple Sum strategy.

In the serial system the number of experts is variable and depends on the number of features taken by the experts. It could vary from a single expert up to a maximum equal to the number of features.

Optimised conventional system

In this method we use a different approach to build the system. We start by optimising experts built independently as is conventionally done. Next we integrate these experts to define a baseline multiple classifier. This system is referred to in the tables below as C.S., which stands for “conventional system”. Now we can optimise the system by adding a feature to any expert in the system that would lead to improving the system
performance. Therefore, when using plus l, take away r feature selection method, we add l features to the system. They are not necessarily all added to the same expert in the system. Also, when removing r features they are removed from the system and not from the expert to which we just added features. We experiment with two alternative strategies that allow us to decide whether we should add features to the C.S. system or not. One option is to add a feature regardless of the C.S. system performance on the validation set, i.e. we force a feature. Then before adding further features we check the validation set rate. We stop adding further features if no improvements are achieved. This strategy is referred to as O.C.S.-f. An alternative strategy is to check the C.S. performance on the validation set. If it is below 100% then we continue adding features that would lead to improvement of system performance on the validation set, otherwise no features are added. This strategy is referred to as O.C.S.

Results

The experimental methodology and the results of the design methods are detailed in Appendix A. The results were not conclusive and further research is required to establish when and why the system based method outperforms the conventional one. Tests on different data sets with different characteristics would be beneficial. The data parameters that can be investigated are feature types, training sample size, number of classes and class distributions and finally the coherence level between the training and test sets.

In the next section the two feature and training set manipulation techniques are integrated in one system. The system based feature selection strategy is revisited in another context, and is compared to the conventional one on real and synthetic data satisfying some of the data characteristics outlined above.

7.6 Combining PHC Design Techniques

Results of Section 7.3 indicated that there is no merit in bagging \( k - NN \) experts under large sample size. The reason is that at such a sample size bagging does not
produce diverse $k - NN$ experts. We have indicated the necessity of fusing diverse component experts as a precondition for a multiple classifier system to gain from fusion. In this section we show that it is possible to improve $k - NN$ experts using bagging by diversifying component $k - NN$ experts using feature selection.

The main purpose of this chapter is to improve bagged $k - NN$ experts under large sample size. The proposed solution also gives insight into whether the combination of two PHC design techniques yields better results than using either technique alone.

The proposed method to improve bagged $k - NN$ experts involves using a different feature subset for each bootstrap set. For this initial study we use a quick method of randomly selecting half of the features to create a feature subset from the full set. The commonly used forward and plus-1, take away-r feature selection methods are computationally more expensive than the random feature subset selection method. This is especially true for the large number of bootstrap sets which we experiment with. Assigning feature subsets to each bootstrap set will yield a system incorporating the two techniques for building PHC systems, the training and the feature set manipulation techniques. The combination of the two methods is applied to diversify the component $k - NN$ experts used in a bagging scenario. We shall demonstrate that this proves to be a quick solution that yields substantial gains.

Also, following the introduction of the novel design method of the previous section, we have the option to use two optimisation methods using feature selection; system based as proposed in the previous section and expert based as is conventionally used. We experiment with both methods and find when a certain method outperforms another. We are especially interested in knowing when our proposed system based method outperforms the conventional one. However, the novel system based feature selection method of Section 7.4, and the system based of this section, are different in the way features are grouped but are similar in the optimisation method which we consider to be novel. That is both methods select the feature(s) that optimise the system not the expert for which the feature(s) is(are) assigned.

The results indicate that the combination of the two PHC design methods is a useful strategy for creating diverse bootstrap designs which yield improved bagging results.
Combining PHC Design Techniques

Using random feature subsets, bagging is an effective approach for improving \( k - NN \) experts. This indicates that for \( k - NN \) experts, feature set manipulation yields more diverse experts than training set manipulation (bagging). However, a multiple expert system that uses the full training set (FTS), where each expert is assigned a random feature subset seems to contend well with bagging.

In Section 7.6.1 we present the system which uses the combination of both training and feature set manipulation techniques and also present the system which uses the full training set, i.e. it uses only the feature set manipulation technique. The design methodology and data sets are presented in subsection 7.6.2, while the results are presented in subsection 7.6.3. We discuss the systems and results in subsection 7.6.4.

7.6.1 System Design Methods

In this section we introduce the systems used in our experiments. With the exception of the simple regular bagging and the single expert, all systems use a feature selection method based on the random feature subset selection technique. We experiment with three PHC systems that incorporate bagging as a design technique; regular bagging, bagging with best expert based random feature selection, (bagxpt), and bagging with best system based random feature selection, (bagsys). We also experiment with PHC systems using the full training set (FTS), instead of the bootstrap sets. Similarly we randomly assign feature subsets to each of the experts. Again, the best subset is selected based on the performance of the expert and the performance of the system, FTSxpt and FTSsys, respectively. The reason for experimenting with the FTS method is to see if there is a merit in combining both the training set and feature set manipulation techniques.

In an expert based random feature selection method, for each expert we randomly select a feature subset from the full feature set. If, based on a validation set, the expert using the selected feature subset yields a higher classification rate than the previously selected subset, we save it as the best subset for the current expert. The number of feature subsets, presented to each expert is heuristically chosen to be equal to the total number of features in the data set. Feature selection based on the performance of
the component expert is also known as the conventional method. The system based random feature subset selection is similar to the expert based one, except that instead of selecting the feature subset that yields the best expert performance, we select the feature subset that yields the best combiner system performance. That is for each expert we create a feature subset and using a validation set we test the performance of the combination of the previously created experts in addition to the current expert under construction. The combination method used employs moderation which was presented in Chapter 5. The system based feature selection method explained above is based on the same principal presented in the previous section, where the feature that improves the system not the individual expert is selected. The difference is in the feature presentation method. Here subsets are selected in group and randomly to speed up the process, while in the previous section features were selected based on the forward and the plus-l, take away-r methods. There were two reasons why we refrained from using any conventional feature selection algorithm instead of the random feature subset to assign features to the bootstrap sets. One is that we are dealing with a large number of experts and features where even the simplest search algorithm would be computationally intensive. The second reason is that the sophisticated search algorithm would, most likely, result in relatively similar feature subsets due to all bootstrap sets being similar under the $k$-NN method. Random feature subset assignment would lead to more diversification among the bootstrap sets.

7.6.2 Experimental Methodology and Data Sets

The system design procedures of this section are tested using the same experimental methodology and setup used in Chapter 5.

Data Sets

The same data sets are used as in Chapter 5 except that the training set size is different. We focus on a large training set size at which bagging is least beneficial, i.e. 80 percent of the full data set. 10 percent of the full set is reserved for validation to select the best feature subset. The remaining 10 percent of the full data set is used as a test set. We
use the three data sets which contain a large number of features totalling 9 or more. Hence we experiment with the BCW, Ionosphere and Wine data sets. The results on the rest of the data were not encouraging. This is consistent with the conclusion drawn by Ho[42] who argued that for random subset selection to be successful the data set must contain a large number of features and samples.

Our results in Figure 7.1 indicate that the bagsys method outperforms all the other methods for the BCW data. In order to find when bagsys performs best we created a synthetic data similar to the BCW data. We set up an experiment using synthetic data sets having different characteristics. The main characteristics that we suspect to have an influence on the performance of any of the systems, and bagging in particular, are

- The size of the training set. We use 500 samples for all the synthetic data sets
which is of comparable size to the BCW data set.

- The ratio of the number of class samples. We focus on the ratio of \( \frac{2}{1} \) as in the BCW data.

- The number of features. We use a total of 10 features in all the synthetic data sets.

- The number of overlapping features. We experimented with 2, 4, 6, 8 and 10 overlapping features. The rest constitute non-overlapping features. We generate overlapping features using a random number generator with equal mean and standard deviation for both classes.

- The spread of features of each class, i.e. the standard deviation for the gaussian data and \( \frac{1}{2}b \) for the uniform data. We experiment with equal spread for both classes, larger class having a standard deviation twice as large as the smaller class, and the smaller class having twice the standard deviation of the larger class.

The last two characteristics yield 13 different data sets. One containing all overlapping features, and four for each of the three types of spread of features. For the data with unequal spread of classes we repeat the data with 8 non-overlapping features but increase the degree of overlap. That is for both types of the unequal class spreads two additional data with increasing overlap are created. The non-overlapping features are created with the mean of one class equal to the mean of the other class plus an overlap factor. The larger this factor the smaller the overlap region. For the data with equal class spread the overlap factor is equal to \( 0.01 \times \delta_2 \) where \( \delta_2 \) is the standard deviation of class 2 for gaussian distributed data, and is \( \frac{1}{2} \) the support domain for the uniformly distributed data. For the data with unequal class spread the overlap factor is \( 0.01 \times (\delta_1 + \delta_2) \). The degree of overlap is increased by reducing the overlap factor to \( 0.01 \times \delta_2 \) and again to \( 0.005 \times \delta_2 \).
7.6. Combining PHC Design Techniques

7.6.3 Results

Real Data

Figure 7.1 indicates that, on real data, we gain from using random feature subsets to create diverse bootstrap sets. The expert based feature selection method yields better performance than the system based feature selection method except for the BCW data. However, the FTS combining system using a feature selection method and bagging using a similar feature selection method perform relatively equally. All outperform regular bagging using the full feature set. Bagging with the system based random feature selection clearly outperforms FTSsys on BCW data. The full feature set single expert outperforms the single expert using the best feature subset found using the forward feature selection or random feature subset selection methods. Therefore, the full feature single expert is the baseline against which we will compare the fusion results.

Generally the results indicate that combining both feature set and training set manipulation techniques, as in bagsys and bagxpt systems, yields better results than using bagging alone. However, they may not necessarily yield better results than feature selection alone as in the FTSsys and FTSxpt systems. Nevertheless, our results generally indicate that it is possible to gain from bagging $k - NN$ experts if we incorporate techniques that diversify the bootstrap sets. The results show that using a different
feature subset for each bootstrap set leads to increased classification rate, which could be attributed to an increased diversification among the bootstrap sets.

Synthetic Data

The synthetic data results shown in Figure 7.2 indicate that if all features are overlapping then all methods are better than simple bagging. This can be attributed to the peaking phenomenon, which occurs when the features are redundant and have little

![Figure 7.3](image)

Figure 7.3: Classification rate of different design methods using normally distributed synthetic data with equal class spread. Figures (a) to (d) are for data with 2, 4, 6 and 8 non-overlapping features, respectively.
information in them. Therefore reducing them improves the classification rate. All methods outperform bagging and the single expert, and all perform equally except at $k = 1$ where bagging with either feature selection method outperforms FTS using a similar feature selection method. At $k = 1$ the system based feature subset selection outperforms expert based subset selection for both bagging and FTS. However, the difference is not significant. The findings are also true for the uniformly distributed data sets of Figure 7.6.

When non-overlapping features exist we have three different cases based on the spread of the non-overlapping features. The first case is when both classes have equal standard deviation, or support domain, shown in Figures 7.3 and 7.7. For this case, using gaussian data, the single expert performs best followed by the full feature bagging. However, the other methods outperform simple bagging and the single expert at approximately $k < 3$. Comparing the feature selection based methods we find that mostly the expert based feature selection is better except when 8 non-overlapping features exist. Generally the feature based bagging and FTS are close, with bagxpt yielding slightly better performance than FTSSxpt. For the uniform data we find that regardless of the number of non-overlapping features all methods perform equally and outperform simple bagging and the single expert.

The second case involves a smaller standard deviation, or support domain, for the larger class, shown in Figures 7.4 and 7.8. Here we note that, for gaussian data, system based feature selection outperforms the expert based one, except that the opposite is true when only 2 non-overlapping features exist. FTS outperforms bagging when system based feature selection is used and yield mixed results when expert based feature selection is used. With the exception of the case when 2 non-overlapping features exist all methods outperform simple bagging and the single expert. For the data with 8 non-overlapping features we increased the overlap and tested the classification rates. However the relative performances remain unchanged. The same is true for uniform data sets except that bagxpt never outperforms FTSSxpt.

The third case involves a larger standard deviation, or support domain, for the larger class, shown in Figures 7.5 and 7.9. For gaussian data when only two non-overlapping
Figure 7.4: Classification rate of different design methods using normally distributed synthetic data with larger spread of the smaller class. Data created using normally distributed random number generator. Figures (a) to (d) are for data with 2, 4, 6 and 8 non-overlapping features, respectively. Figures (e) and (f) are for data of figure (d) with an increased degree of overlap.
7.6. Combining PHC Design Techniques

Figure 7.5: Classification rate of different design methods using normally distributed synthetic data with larger spread of the larger class. Figures (a) to (d) are for data with 2, 4, 6 and 8 non-overlapping features, respectively. Figures (e) and (f) are for data of figure (d) with an increased degree of overlap.
features exist both bagging and FTS methods using the expert based feature selection outperform all, and are equal to each other at all k except at $k = 1$ where bagging outperforms FTS. For the data with 4 non-overlapping features, at $k < 4$, simple bagging outperforms all, while FTS degrades below all. However, at larger k, the feature selection based methods outperform simple bagging with FTSsys outperforming all. At higher numbers of non-overlapping features, feature based bagging outperforms FTS. However, once the degree of overlap between the 8 non-overlapping features is increased FTS outperforms bagging. As the degree of overlap is further increased bagging outperforms FTS. When more than 2 non-overlapping features exist the system based feature selection outperforms the expert based one. For the uniformly distributed data sets FTS is always better than bagging if the system based feature selection is used. However, when using the expert based feature selection method, bagging and FTS perform equally with bagging marginally outperforming at smaller number of non-overlapping features and smaller values of k. As was the case for the gaussian data, simple bagging and the single expert outperform the system based methods, when 2 non-overlapping features exist. However they do not outperform the expert based methods.

7.6.4 Discussion

When to Bagg

The choice of the design technique that yields the best performance depends on the type of expert combined. For certain expert types a design technique may not produce diverse experts and consequently may not produce a successful combining system. An example is the training set manipulation design method applied to an expert that is resilient to the changes in the training set, i.e. bagging $k$ – NN experts. However, adding another factor of diversity, like a feature set manipulation technique, could yield diverse bootstrap sets, even under stable $k$ – NN experts.

Our question in this chapter was whether the combination of training set manipulation, i.e. bagging, and feature set manipulation would outperform a system using either one alone. In other words when would bagsys or bagxpt outperform simple bagging and
7.6. Combining PHC Design Techniques

Figure 7.6: Classification rate of different design methods using uniformly distributed synthetic data with equal class spread.

feature selection based systems like the FTSsys and FTSxpt systems presented above. The results have shown that bagsys and bagxpt do not outperform simple bagging when features of both classes have relatively equal standard deviation with a gaussian density. Otherwise, when features have uniform distribution or when one class has a larger spread than the other then bagsys and bagxpt outperform bagging. Bagsys and bagxpt outperform FTS systems when the larger class has a larger spread on the condition that more than half of the features are non-overlapping. When features have uniform distribution then FTS is always equal to or better than bagging with a similar feature selection method.

Hence, we can not confirm that combining both PHC design techniques would always yield the best performance. However, for the $k$ – $NN$ case, combining them does yield superior performance than using the training set manipulation technique alone.

Comparing the conventional expert based feature selection method to the system based one proposed by us we notice that bagsys outperforms bagxpt when more than 2 non-overlapping features exist only for the unequally spread cases. This indicates that when more information exists in the features, the system based method outperforms the expert based one. One can conclude that bagging with system based feature selection is expected to perform best when more information exists in the features. This is a natural extension to the fact that bagging improves when training set manipulation leads to diverse experts as in the case of unstable classifiers. The same holds true
Figure 7.7: Classification rate of different design methods using uniformly distributed synthetic data with equal class spread. Figures (a) to (d) are for data with 2, 4, 6 and 8 non-overlapping features, respectively.
7.6. Combining PHC Design Techniques

Figure 7.8: Classification rate of different design methods using uniformly distributed synthetic data with larger spread of the smaller class. Figures (a) to (d) are for data with 2, 4, 6 and 8 non-overlapping features, respectively. Figures (e) and (f) are for data of figure (d) with an increased degree of overlap.
Figure 7.9: Classification rate of different design methods using uniformly distributed synthetic data with larger spread of the larger class. Figures (a) to (d) are for data with 2, 4, 6 and 8 non-overlapping features, respectively. Figures (e) and (f) are for data of figure (d) with an increased degree of overlap.
for the feature set manipulation. Bagging with feature set manipulation outperforms regular bagging if the feature sets are diverse and yield diverse feature subsets which consequently yield diverse bootsets.

When to Use System Based Feature Selection

It is obvious from the synthetic data results that the system based feature selection method outperforms the expert based one when more than two of the ten features are non-overlapping. This indicates that we can use this method if we have many features with discriminatory information.

7.7 Conclusion

In this chapter we investigated two techniques for building PHC systems, the training set and the feature set manipulation techniques. We investigated bagging as an example of a training set manipulation technique. We found that although the problems of Product due to small sample size were solved using MProduct or Moderation, bagging also suffers from the small sample size. It was noticed that for cases involving a very small sample size the standard bagging method may degrade the performance as compared to the single expert. At very small sample sizes the single expert outperformed the standard bagging method because the bootstrap sets often contained samples from one class only, i.e. population bias. In such situations the resulting very high error rates dominated the bootstrap expert outputs and the underlying benefits of bagging were cancelled. In order to benefit from bagging, and alleviate the shortcoming exhibited by bagging we modified the bagging method so that all classes be represented in the bootstrap set. The different modification methods were all beneficial, although method 3, which ensures class ratio similar to the full data set, was the best, followed by method 4, which ensures all classes are present, regardless of the ratio.

Additionally, we noticed that the smaller sizes may have contained learning sets that were biased towards a certain class and the reconstruction of the population distribution could improve the single expert performance. Using such an unbiased learning set all
methods improved the single expert relatively equally, and the choice of the bagging method was not very critical, although method 3 could outperform others.

For the feature set manipulation technique we proposed a novel design philosophy for expert fusion by taking the view that the design of individual experts and fusion cannot be solved in isolation. Each expert was constructed as part of the global design of a final multiple expert system. We investigated three distinct design strategies which we referred to as parallel, serial and conventional. Results, shown in Appendix A were not conclusive and further research is required to find when and why the system based method outperforms the conventional one. The last section partially contributed to this investigation.

We have found that combining \( k - NN \) experts using modified bagging under large training set sizes does not improve regular bagging or the single expert. Therefore, we have proposed that bagging should be combined with feature selection to achieve diversity among the bootstrap sets, and consequently improve bagging. The feature selection method is based on random subset selection which is computationally more feasible than the forward feature selection method due to the large number of component experts and features. We showed that using feature selection to diversify bootstrap sets yields a significant improvement over simple bagging. However, other PHC design methods may yield a larger improvement over the single expert than feature based bagging. For example the use of only the feature set manipulation technique under the full training set (FTS), where each expert uses a different feature subset, occasionally outperformed bagging with a similar feature selection method. Experiments involving artificial data sets indicated that bagging with feature selection outperforms other techniques studied when more than half of the features have discriminatory information. That is because only if features have discriminatory powers we achieve diversity.

Also, the results have shown that the system based feature selection method outperforms the expert based one if more than 20 percent of the features are non-overlapping.
Chapter 8

Conclusion and Future Research

8.0.1 Conclusion

This dissertation focuses on simple fusion strategies commonly used to combine the decisions of Parallel Homogeneous Classifiers, PHC. The fusion systems were divided in two categories based on their architecture: PHC and CMC systems. The PHC systems contain component experts that produce output of the same form and in parallel. However, the CMC systems contain experts that are not connected in parallel and the fusion is an internal complex process. Alternatively, the systems contain parallel component experts, but the decisions do not belong to one of three information forms itemised in Chapter 2. Furthermore, the fusion strategies used in PHC systems were also divided in two groups, simple and complex based on whether the rule requires fusion stage training or not. The simple strategies investigated were Product, Sum, Vote, Median, Maximum and Minimum. We experimentally validated the theoretically derived prediction of the behaviour of these fusion strategies. The empirical study confirmed the sensitivity of Product to noise. Moreover, the results indicated the existence of a noise level boundary across which Product properties changed dramatically. For experts affected by uniform noise, where $b = \frac{1}{2}$ support domain, it was found that for $P(\omega|x) \geq b$ the Minimum and Product performance was the best, while for $P(\omega|x) < b$ the Minimum and Product performance degraded causing Sum to become the best rule. Increasing the number of experts did not have any effect on the relative performance,
but all the strategies in general exhibited a better performance. For experts affected by gaussian noise, the boundary was constant regardless of the value of $P(\omega|x)$. It changed (decreased) only when the number of experts or classes increased. It actually reached zero when the number of classes increased to three and the number of experts grew to 8. For values of the standard deviation below the boundary all rules performed similarly, while above the boundary the Product, Minimum and Maximum delivered the worst performance.

The experimental study showed that Product dropped sharply at the boundary due to some expert outputs reaching the end of the range of the probability estimate values. This leads to having experts with zero outputs, which veto the outputs of all other experts if product fusion rule is used. Based on this finding we proposed Modified Product (MProduct) which replaces all zero outputs with a threshold. The method is heuristic but resulted in a large improvement over Product. Although Product performed well at low noise conditions, MProduct was able to improve Product also at low noise levels. This is a significant achievement since Product is preferred over Sum due to its direct derivation from the Bayesian formalism, but was so far avoided due to its sensitivity to the veto effect. Although MProduct performed very well at a wide range of thresholds in most cases its best performance was achieved when truncation threshold was set to $5 \times 10^{-4}$ or lower.

We also established a theoretical basis for modifying expert outputs by introducing the concept of classifier output moderation. We focused on the $k-\text{NN}$ decision rule and derived the formula for correcting the outputs of $k-\text{NN}$ experts, by taking the model prior into consideration. The derived moderation equation was examined on real data sets. Tests indicated that the proposed moderation method improved the performance of Product significantly.

Even with the introduction of MProduct, Sum and Vote are and will be two of the most widely used methods. However, more investigation is needed to find when and why one of these rules outperforms another. In Chapter 6 the relationship of the sum and vote classifier combination rules was investigated. We showed analytically that, for normally distributed estimation errors, Sum always outperforms Vote, whereas for
heavy tail distributions Vote may outperform Sum. We then confirmed our theoretical predictions by experiments on both synthetic and real data. In the synthetic experts case, we showed for Gaussian error distributions that, as expected, Sum outperforms Vote. The differences in performance are particularly significant at high estimation noise levels. However, for heavy tail distributions the superiority of Sum may be eroded for any number of experts if the margin between the two a posteriori class probabilities is small or for a small number of cooperating experts even when the margin is large. In the latter case, once the number of experts exceeds a certain threshold, Sum tends to be superior to Vote. Experiments on real data supported the general findings but also showed the effect of the usual assumptions of conditional independence, identical error distributions and common target outputs of the experts not being fully satisfied.

On the PHC system design level we investigated two widely used techniques for building successful fusion systems. The methods belong to the training set and the feature set manipulation technique categories. For the training set manipulation case, we investigated bagging applied to $k-NN$ experts. The results confirmed the previous finding that bagging does not improve $k-NN$ experts, at all training set sizes. At very small set size one may generate diverse bootstrap sets, however, the resulting ensemble would contain sets with some classes misrepresented or completely missing. As the training set increases in size problems associated with population bias in the bootstrap sets disappear, however, all bootstrap sets become equal and the diversity achieved under the smaller training set size also disappears. We propose two solutions. At very small sample size we propose population bias control using prior knowledge through which we reject bootstrap sets that do not contain samples from all classes, or the ratio of samples per class is not close to the ratio in the full set or the training set. Results indicated that the bagging method which used prior knowledge from the full set, i.e. had the most prior knowledge, yielded the best performance followed by the method which only checks for the existence of all classes. This indicates that if complete prior knowledge does not exist, then just ensuring that all classes are represented yields better results than using biased prior knowledge. The strength of this bootstrapping method may be in its ability to create more diverse bootstrap sets than the ones using class ratio knowledge.
A second technique to improve bagging $k-NN$ experts is based on the idea of injecting diversity among the bootstrap sets. This was tested for large training sets but is expected to improve bagging even at smaller set sizes. $k-NN$ experts are very robust to changes in training sets, and hence training set manipulation will not yield diverse experts. However, if bootstrap sets used different feature subsets they may become diverse. Although this philosophy was found to be successful when using feature set manipulation techniques, not any measure of diversity would yield an improvement. For example our experiments using two different metrics, the euclidean and the optimum metric proposed by Short and Fukunaga [86], for each half of the bootstrap sets, respectively, did not result in any improvement. In any case, the use of random feature subsets for each bootstrap set yielded a significant improvement. We found the combination of random feature subset and bagging yielded a large improvement over simple bagging. However, we also found that the assignment of random feature subsets to the full training sets mostly yielded better results. By this we confirmed that mixing two PHC design methods may not necessarily yield a superior fusion system.

We also proposed a different design philosophy for building any fusion system. We advocated the view that the design of individual experts and fusion can not be solved in isolation. Based on this philosophy each expert is constructed as part of the global fusion system and is optimised to yield the best combiner performance rather than the best expert performance, the expert being the one under construction. This design methodology yielded good results for gaussian experts when tested on two data sets. However, the outcome was not conclusive and did not significantly outperform the conventional method of designing $k-NN$ experts. The conventional method is one that optimises the component experts independently and then merges them in the combiner system. The system based design method, i.e. the proposed design philosophy, was further investigated and compared to the conventional method in experiments involving random feature subset selection and bagging. The findings were that the system based method may outperform the conventional one when many non-overlapping or discriminatory features exist.
8.0.2 Future Research

On the decision fusion level one can alter the existing methods to improve them, however one should be careful not to impose training otherwise the simple strategies will lose the advantage they have over many good complex strategies. For example, it is possible to improve Vote, when there is a tie at the expert output between two classes, by bringing both labels to the combiner stage and allowing the combiner to poll all votes including ties. Hence a tie between two classes would mean that we are adding an expert to the total number of experts being combined. For example, in a two class case, if we have three experts, with one expert outputting a tie, then considering the results from the tied expert as two outputs, (i.e. two labels), would change the number of decisions at the combiner level to four instead of three.

There is also a need to better understand the fusion strategies to find when and why they outperform one another. For example what is the relationship between the outliers and the performance of Median and MProduct. When would Median outperform MProduct and Sum. The experimental evaluation of Chapter 4 has shown Median to perform better than Sum when contradicting experts exist only to degrade below Sum when expert estimation errors increased.

In general the relation between fusion systems and strategies is complex and has not been investigated thoroughly. We found that MProduct improved Product most when using bagging method 4. The prior on which the experts are built has an effect on the performance of the fusion strategies. The independence of the component experts also has an effect on their performance. We expect Product to perform best when independent experts are used, while weighted Sum or weighted MProduct may perform best if the expert independence assumption does not hold. Would weighted MProduct outperform Sum and weighted Sum? Occasionally Product outperforms MProduct. Is MProduct a closer version to Sum or to Product. We know that Product is more suitable for independent experts, because it follows directly from the independence assumption while Sum does not. On the other hand equally-weighted Sum is sufficient if experts are independent. Hence, when independence among experts does not exist one should not use equally-weighted Sum, and if independence exits then Product, (or
MProduct) is more legitimate, not equally-weighted Sum. Based on this logic then equally-weighted Sum should never be used, except if independence does not exist and weighting is not possible due to the lack of a validation set, for example. Even there MProduct may outperform Sum. An investigation of these rules for varying degrees of independence could yield answers to the questions raised.

One could experiment to see which of the fusion rules performs best under different diversity levels. Our synthetic experiments using contradicting experts partially contributed to this knowledge, however the investigation was not focused on the issue of diversity among experts.

As the number of classes increases the probability of classes having zero posteriors increases. This becomes more obvious under $k - NN$ experts when $k$ is smaller than the number of classes. This leads to the assumption that Product may degrade under a large number of classes due to the veto effect. There the benefit of MProduct and Moderation becomes more obvious. Our experiments involved a maximum of three classes. Experiments involving a larger number of classes is required to validate this prediction.

On the system design level we experimented with bagging $k - NN$ experts. Bagging may not improve a $k - NN$ expert indefinitely. However, to obtain an optimum classification rate, a $k - NN$ would need to be fused if we have different sets of training samples, different features, or types of features. Therefore, we may improve over a single $k - NN$ expert if numeric and nominal features are used in two different experts. Most state that there is no merit in bagging stable experts. Many stable experts can be unstable at certain training set sizes. Also, some unstable experts may become stable at a certain training set size. The type and complexity of data used is a crucial factor. Bagging can be useful with any classifier as long as the classifier is unstable at the available training set size for the data under investigation. The option to use bagging should be open for any classifier until a stability test is performed for the available training set. A clearer statement regarding the use of bagging on stable experts would be the following: *bagging is useful only at the training set size at which the component classifier is unstable.*
The system based feature selection to construct combiner systems raised many unanswered questions. Would such a design method lead to increased overfitting. We found that this approach outperforms the conventional method when many informative, non-redundant or non-overlapping features exist. For experiments with random feature subsets combined with bagging on real data we can remove the features containing the smallest discrepancy and see if the system based method outperforms the conventional one, or either method involving all the features. The results with synthetic data suggest that the system based method outperforms the conventional one. However, due to the loss of information resulting from removing some weak features, the final design may be worse overall. We would certainly loose a degree of diversity among bootstrap sets.

Certainly the use of different features among an ensemble of experts leads to increased diversity. However, is a random selection the best feature selection method for large ensembles of experts? Should not a careful selection of features for each component result in stronger experts? To what degree does random subset selection lead to increased diversity and good experts, simultaneously? When do we stop obtaining accuracy for the sake of diversity? Is there a boundary? If we are selecting our features, using random subsets or more sophisticated methods, would the system based method achieve better results over the conventional method, for other types of experts or data that we have not explored? Should not the researcher have both options open when designing a system? In some real world applications, if one trusts his data, would not some degree of overfitting be useful? And therefore, the system based feature selection method to be more beneficial.

The training set classification rate and consequently the amount of correlation reduction among experts, (if such correlation reduction leads to better classification rate), is expected to be higher when the system based design method is used instead of the conventional one. The correlation reduction capability of the system based method should be investigated and compared to the correlation achieved among the components constructed independently as in the conventional design method.

In our version of random feature subset selection method we selected the best subset out of a heuristic number of subsets. Does the selection lead to improved performance
over the commonly used method of using the first generated subset? Should not we test to see if the subset leads to a classification rate higher than 50 percent? This does not seem to be done, or is not clearly stated in the literature.

Finally, would class set manipulation yield improved combiner system? There are at least three parameters involved with any problem. The training set, the feature set and the number of classes. Research in multiple classifier fusion has focused on manipulating the training set and the feature set. Expert diversification by controlling the set of classes an expert is trained on has drawn less attention. ECOC [25] and Input Decimation [67] are two methods that exploit the class set, however, further research in this area could result in interesting achievements. For example one can construct many component classifiers, such that each group of them is optimised for one class. The experts are created by randomly sampling, but allowing more samples from the class we want to optimise. Instead of random sampling one can also select the samples that lead to the best expert performance, for a certain class, on a validation set.
Appendix A

System Based Feature Selection

In this appendix the proposed system based design philosophy of Section 7.5 is studied experimentally. Although the results are inconclusive they are presented because we believe the novel system based strategy, when developed further, may outperform the conventional approach or the single expert. Further investigation using different data sets with different characteristics, feature types, sample sizes and training and test set coherence levels will be required in the future to find when it outperforms the single and the conventional systems.

A.1 Experimental Methodology

A.1.1 Multiple Expert Systems

We experiment with three types of classification experts generating the class aposteriori probabilities in different ways, namely the gaussian, K-nearest neighbour \((k - NN)\) and nearest neighbour \((NN)\) experts. When using \(k - NN\) experts we set \(k = \sqrt{N}\), where \(N\) is the number of training samples. Besides the homogeneous system resulting from using either one of the experts mentioned above in a single system, we also experimental with a heterogeneous system that uses all three types of experts in parallel. It is referred to as \textit{mult3}. In the \textit{mult3} system, the first expert used is the gaussian followed by \(k - NN\) then \(NN\) experts. Any additional experts needed are gaussian.
A.1.2 Fusion Strategies

One of the design specifications of the proposed methodology for multiple expert system construction is the fusion rule. In our approach we focus on decision probability level fusion. We assume that our experts deliver a soft opinion for each possible class expressed in terms of the a-posteriori class probabilities. More specifically, expert $i$ using feature vector $x_j$, obtained from the original pattern vector $x$ by feature allocation, outputs its estimate $P(\omega_i|x_j)$ of the a-posteriori probability for class $\omega_i$, $i = 1, \ldots, m$.

For the sake of notational simplicity we shall denote the expert output $P(\omega_i|x_j)$ as $s_{ji}$. Wherever necessary, the output corresponding to a specific pattern, say the $k-th$ pattern $x^k$, will be denoted by $s_{ji}^k$. Note that the experts scores for each class take values from the interval $[0,1]$.

We experimented with several different rules, simple combiners which do not require any training and trainable linear combiners. The parameters of trainable fusion strategies were estimated using an independent validation set. Each fusion rule derives for class $\omega_i$ a specific fused output $S_i$. These outputs are normalised as required to ensure that the normalised quantities sum up to one.

The fusion rules used which require no training are the Sum and MProduct strategies presented in Chapters 4 and 5, respectively.

The linear fusion rules established by training on the validation set are of the form

$$S_i = \beta_{0i} + \beta_{i}^T s_i$$

$$= \sum_{j=0}^{R} \beta_{ji} s_{ji}$$

(A.1)

where $\beta_{ji}$, $j = 1, R$, the components of the parameter vector $\beta_i$, are the mixing parameters of the scores $s_{ji}$ and $\beta_{0i}$ represents the off-set with $s_{0i} = 1$. The following linear rules have been studied:

**Nearest Mean Fusion** [36] The linear discriminant function corresponding to this decision rule is given by

$$\bar{\beta}_i = \frac{X_i^T Y_i}{\|X_i^T Y_i\|}$$

(A.2)
where $X$ is the data matrix of scores for the elements in the training set

$$X_i = \begin{bmatrix}
s_{1i}^1 & \cdots & s_{1i}^R \\
s_{2i}^1 & \cdots & s_{2i}^R \\
\vdots & \cdots & \vdots \\
\vdots & \cdots & \vdots \\
s_{Ni}^1 & \cdots & s_{Ni}^R
\end{bmatrix}$$  \hspace{1cm} (A.3)

and the vector of target responses $Y_i$ has elements $1/N_i$ for patterns from class $\omega_i$ and $\frac{1}{N - N_i}$ for all the other patterns. $\beta_{0i}$ is set so that a point on the boundary maps to $\frac{1}{2}$. This fusion strategy will be referred to as NrMean in the tables.

**Linear Regression** By augmenting the vector of scores by an $(R+1)^{st}$ component $s_{0i}$ set to one and the parameter vector by the corresponding element $\beta_{0i}$ we can find the least squares fit to the binary target output. The solution is given by

$$\beta_i = (X_i^T X_i)^{-1} X_i^T Y_i$$  \hspace{1cm} (A.4)

where

$$X = \begin{bmatrix}
1 & s_{1i}^1 & \cdots & s_{1i}^R \\
1 & s_{2i}^1 & \cdots & s_{2i}^R \\
\vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \cdots & \vdots \\
1 & s_{Ni}^1 & \cdots & s_{Ni}^R
\end{bmatrix}$$  \hspace{1cm} (A.5)

and the vector of target responses $Y_i$ has its elements set to one for patterns from class $\omega_i$ else zero. This regression will be referred to as LnrReg in the tables.

**Logistic Regression** Under the assumption that the expert outputs are conditionally distributed according to a multivariate normal, they can be combined using logistic
regression with weighting parameters

$$\beta_i = (\mu_i - \mu_0)^T \Sigma^{-1}$$  \hspace{1cm} (A.6)

and an off-set

$$\beta_{oi} = \log\left(\frac{N_i}{N - N_i}\right) - 0.5(\mu_i - \mu_0)^T \Sigma^{-1}(\mu_i + \mu_0)$$  \hspace{1cm} (A.7)

where $\mu_i$ is the mean vector of expert scores for patterns in class $\omega_i$ and $\mu_0$ is the vector of mean scores for all the other classes. $\Sigma$ is the weighted average covariance matrix of the respective score populations. This strategy will be referred to as LogReg in the tables.

### A.2 Data Sets

Both of the data sets used here were also used in Chapter 5. However, for the experiments of this section we used a different training set size and also used a validation set in addition to the test set. Here we reintroduce the data sets again.

**Breast Cancer Wisconsin [17] (BCW):** It consists of 699 nine feature samples. The training set consists of 50 randomly selected samples. The number of samples per class are taken such that their proportion is equal to their original proportion. Using the rest of the samples we construct the 50 sample validation set in a similar manner. The remaining samples constitute the test set. We repeat the experiments for a large training set where 175 samples constitute the training set, 175 constitute the validation set and the rest constitute the test set.

**Seismic data:** This data set was obtained from Shell for research purposes at CVSSP, University of Surrey. It is also used in chapter 5 and contains three classes. The data is divided into two sets L0 and T0. L0 contains 25229 samples while T0 contains 17560 samples. Samples of both files have 25 features. We randomly sample 500 samples from each class to build each of the training, validation and test sets. Samples from set L0 are used for the training and validation sets, while the test set is constructed from T0.
Table A.1: Performance comparison of fusion strategies for the serial and parallel system designs on BCW data

<table>
<thead>
<tr>
<th>Termination Condition</th>
<th>Considering all features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gauss</td>
<td>K - NN</td>
</tr>
<tr>
<td><strong>Serial</strong></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>93.76</td>
</tr>
<tr>
<td>Mprod</td>
<td>93.72</td>
</tr>
<tr>
<td>NrMean</td>
<td>90.99</td>
</tr>
<tr>
<td>LnrReg</td>
<td>93.56</td>
</tr>
<tr>
<td>LogReg</td>
<td>93.56</td>
</tr>
<tr>
<td><strong>Parallel</strong></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>93.32</td>
</tr>
<tr>
<td>Mprod</td>
<td>93.16</td>
</tr>
<tr>
<td>NrMean</td>
<td>90.95</td>
</tr>
<tr>
<td>LnrReg</td>
<td>92.92</td>
</tr>
<tr>
<td>LogReg</td>
<td>92.92</td>
</tr>
<tr>
<td><strong>Single expert</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>92.55</td>
</tr>
</tbody>
</table>

A.3 Results

In this thesis we present and discuss the results obtained using the plus-l-take away-r feature selection method. The results for the simple forward selection optimisation strategy are reported in [10]. Overall both feature selection methods yielded relatively similar results. The classification rates are indicated in tables A.1 - A.9. Whenever the performance is higher than the highest single expert it is highlighted in bold.

The data sets used for our experiments had different properties. The dimensionality of the BCW data is relatively low. There was a very little scope for duplication of discriminatory information. Any attempt to build classifiers that would deploy diverse features was extremely curtailed by the small number of features available. This meant
that the classifiers built so that their measurements space do not overlap were very
simple, and of low dimensionality. The BCW data was reasonably homogeneous and any
training set of reasonable size drawn from the total data set available would normally
be representative of the rest of the data.

In contrast the dimensionality of the seismic data was 25. The features which repre-
sented the texture of seismic images using different sets of filters were correlated. This
provided a greater opportunity for overfitting. The data complexity was further am-
plified by its natural diversity. The training and test sets were drawn from physically
different sites. In consequence the training data was not fully representative of the total
population. These aspects provided interesting diverse scenarios for testing the fusion
system design methods.

Considering the various classifiers, in general one can observe that the Gaussian clas-
Table A.3: Performance comparison of fusion strategies for the serial system design on Seismic data

<table>
<thead>
<tr>
<th>Termination Condition</th>
<th>Considering all features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gauss</td>
</tr>
<tr>
<td><strong>Serial</strong></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>77.61</td>
</tr>
<tr>
<td>Mprod</td>
<td>78.08</td>
</tr>
<tr>
<td>NrMean</td>
<td>73.36</td>
</tr>
<tr>
<td>LurReg</td>
<td><strong>84.19</strong></td>
</tr>
<tr>
<td>LogReg</td>
<td>78.12</td>
</tr>
<tr>
<td><strong>Parallel</strong></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>80.03</td>
</tr>
<tr>
<td>Mprod</td>
<td>74.57</td>
</tr>
<tr>
<td>NrMean</td>
<td><strong>85.56</strong></td>
</tr>
<tr>
<td>LurReg</td>
<td>80.65</td>
</tr>
<tr>
<td>LogReg</td>
<td>78.87</td>
</tr>
<tr>
<td><strong>Singl xprt</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>66.37</td>
</tr>
</tbody>
</table>

A classifier, which is most restrictive in terms of the assumptions imposed on the classifier design, delivers the worst performance as a single expert. However, it remarkably catches up and often outperforms the other classifiers when used in multiple expert design. Surprisingly the mixed expert system referred to in Tables A.1-A.3 as mult3 did not exhibit superior performance over the homogeneous multiple classifier designs.

Comparing the fusion rules, the Sum and MProduct are incredibly robust and in most cases offer the best performance. This appears to be true regardless of the design approach. However, when the classifier outputs are correlated, weighted averaging as afforded by linear or logistic regression on the whole delivered better results.

The two proposed design approaches, serial and parallel, appeared to achieve compa-
### Table A.4: Performance comparison of fusion strategies for the Optimised Conventional system design using BCW data. The effect of adding features to the termination point of the conventional system, C.S.

<table>
<thead>
<tr>
<th>Feature</th>
<th>C.S.</th>
<th>O.C.S.-f</th>
<th>O.C.S.</th>
<th>C.S.-all-feats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>94.56</td>
<td>94.06</td>
<td>94.22</td>
<td>96.49</td>
</tr>
<tr>
<td>Mprod</td>
<td>94.56</td>
<td>94.06</td>
<td>94.22</td>
<td>96.49</td>
</tr>
<tr>
<td>NrMean</td>
<td>89.95</td>
<td>90.75</td>
<td>90.28</td>
<td>94.36</td>
</tr>
<tr>
<td>LnrReg</td>
<td>93.59</td>
<td>92.45</td>
<td>92.45</td>
<td>95.83</td>
</tr>
<tr>
<td>LogReg</td>
<td>77.40</td>
<td>93.42</td>
<td>93.42</td>
<td>95.83</td>
</tr>
</tbody>
</table>

### Table A.5: Performance comparison of fusion strategies for the Optimised Conventional system design using BCW data. The effect of removing features from the conventional system, C.S.

<table>
<thead>
<tr>
<th>Condition</th>
<th>C.S.</th>
<th>O.C.S.-f</th>
<th>O.C.S.</th>
</tr>
</thead>
<tbody>
<tr>
<td>At termination condition</td>
<td>Sum</td>
<td>94.56</td>
<td>87.55</td>
</tr>
<tr>
<td></td>
<td>Mprod</td>
<td>94.56</td>
<td>88.21</td>
</tr>
<tr>
<td></td>
<td>NrMean</td>
<td>89.95</td>
<td>87.01</td>
</tr>
<tr>
<td></td>
<td>LnrReg</td>
<td>93.59</td>
<td>92.62</td>
</tr>
<tr>
<td></td>
<td>LogReg</td>
<td>77.40</td>
<td>93.29</td>
</tr>
<tr>
<td>Using all features</td>
<td>Sum</td>
<td>96.49</td>
<td>90.28</td>
</tr>
<tr>
<td></td>
<td>Mprod</td>
<td>96.49</td>
<td>95.96</td>
</tr>
<tr>
<td></td>
<td>NrMean</td>
<td>94.36</td>
<td>88.91</td>
</tr>
<tr>
<td></td>
<td>LnrReg</td>
<td>95.83</td>
<td>88.95</td>
</tr>
<tr>
<td></td>
<td>LogReg</td>
<td>95.83</td>
<td>88.95</td>
</tr>
</tbody>
</table>
A.3. Results

rable performance. A more influential factor was the stopping criterion used for the classifier construction. When the classifier building process continues until all the features are used up, problems with redundant representations such as the seismic data classification, could lead to overfitting of individual experts. The poor generalisation of component classifiers may be difficult to recover from, even using fusion. More over, the overfitting may also result in expert output saturation which may lead to data dependence and consequently, to numerical problems for fusion rules which involve matrix inversion. To avert such a problem the pseudo inverse method is used instead of the regular inversion method. Note that this measure failed to solve the poor generalisation for the logistic regression case of table A.9.

Tables A.4-A.9 present the results of various experiments involving the conventional design approach in its various variants. For the BCW data the fusion of best individual designs was most successful with the simplest fusion rules (Sum, MProduct). The performance was not improved by further system optimisation. The designs exploiting all the features rather than stopping at the termination conditions were marginally better. The overall performance of these designs was very similar to that achieved by the advocated serial and parallel construction methods. However, these observations do not extend to the seismic data where the conventional design methods and its variations did not do as well, as shown in Tables A.8-A.9. The use of linear regression saved the situation somewhat. However, this has been established only aposteriori, from the performance on the test set. Thus it appears that it would be a much less risky strategy to design the multiple expert systems based on the system optimization method proposed by us, rather than adopting the conventional approach. On the BCW data, some experiments were conducted with a small training set. The results of these experiments are given in Tables A.1, A.4 and A.5. On the whole the conclusions drawn earlier extend also to this situation, with perhaps even stronger emphasis on the simplicity of fusion.

Finally, it is pertinent to compare the proposed integrated design with the previous method and with the baseline methods achieved on the BCW data elsewhere. It would not be possible to claim that the use of a more sophisticated optimisation strategy resulted in a better performance. This suggests that the objective function is quite flat.
Table A.6: Performance comparison of fusion strategies for the Optimised Conventional system design using BCW data with large training set. The effect of adding features to the termination point of C.S.

<table>
<thead>
<tr>
<th></th>
<th>C.S.</th>
<th>O.C.S.-f</th>
<th>O.C.S.</th>
<th>C.S.-all-feats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>96.33</td>
<td>96.29</td>
<td>96.29</td>
<td>96.49</td>
</tr>
<tr>
<td>Mprod</td>
<td>96.36</td>
<td>96.33</td>
<td>96.33</td>
<td>96.49</td>
</tr>
<tr>
<td>NrMean</td>
<td>93.62</td>
<td>93.76</td>
<td>93.76</td>
<td>94.36</td>
</tr>
<tr>
<td>LnrReg</td>
<td>94.46</td>
<td>95.53</td>
<td>95.63</td>
<td>95.83</td>
</tr>
<tr>
<td>LogReg</td>
<td>94.46</td>
<td>95.53</td>
<td>95.63</td>
<td>95.83</td>
</tr>
</tbody>
</table>

and a number of solutions achieve reasonable performance.

Comparing with baseline methods, the best performance on BCW that we achieve is when all features are considered using gaussian experts and fusing with Sum and MProduct. Both serial and parallel systems obtain rates higher than 96 compared to a single experts performance of 94.79 as shown in Table A.2. Skalak [87] on BCW achieves 96.2 using NN, 96.4 using his single prototype method and 96.8 using his two prototype method. He uses 90% of samples as the training set. Since our NN rate is lower than his due to working with a smaller training set size, we could expect our method to outperform his method. Further experiments on a large training set is required to confirm this. When MProduct fuses a parallel gaussian system we achieve the highest rate of 96.68 compared to the single expert performance of 94.79.

For the seismic data the results obtained suggest that the more sophisticated optimisation with backtracking may result in overfitting, as in the case of the $k-NN$ classifier in table A.3, with adverse effects on performance. Thus overall, from the results of the two data sets one would recommend the use of the sequential forward strategy which is computationally simpler. It seems that when using the Seismic data the systems over train and hence can not generalise well during testing. When the test and training sets are both from one file, for example L0, we get a very high classification rate of 99.97 using the Sum rule in a parallel gaussian system.
A.4 Comparing Architectures

Table A.7: Performance comparison of fusion strategies for the Optimised Conventional system design using BCW data with large training set. The effect of removing features from C.S.

<table>
<thead>
<tr>
<th></th>
<th>C.S.</th>
<th>O.C.S.-f</th>
<th>O.C.S.</th>
</tr>
</thead>
<tbody>
<tr>
<td>At stop</td>
<td>Sum</td>
<td>96.33</td>
<td>95.43</td>
</tr>
<tr>
<td>termination</td>
<td>Mprod</td>
<td>96.36</td>
<td>95.43</td>
</tr>
<tr>
<td>condition</td>
<td>NrMean</td>
<td>93.62</td>
<td>91.19</td>
</tr>
<tr>
<td></td>
<td>LnrReg</td>
<td>94.46</td>
<td>94.06</td>
</tr>
<tr>
<td></td>
<td>LogReg</td>
<td>94.46</td>
<td>93.86</td>
</tr>
<tr>
<td>Using all</td>
<td>Sum</td>
<td>96.49</td>
<td>96.39</td>
</tr>
<tr>
<td></td>
<td>Mprod</td>
<td>96.49</td>
<td>96.39</td>
</tr>
<tr>
<td></td>
<td>NrMean</td>
<td>94.36</td>
<td>88.08</td>
</tr>
<tr>
<td></td>
<td>LnrReg</td>
<td>95.83</td>
<td>95.69</td>
</tr>
<tr>
<td></td>
<td>LogReg</td>
<td>95.83</td>
<td>95.69</td>
</tr>
</tbody>
</table>

A.4 Comparing Architectures

In general the serial architecture is better with $k - NN$ and NN classifiers while the parallel is better with the gaussian method, although both design methods improve gaussian experts. The parallel architecture sometimes improves $k - NN$.

The serial architecture does not degrade much if all features are used. In the serial case assigning new features to new experts does not degrade old experts and keeps the system resilient to the peaking phenomenon. The parallel design degrades when new features are added.

When fusing $k - NN$ or NN by the serial design, regression methods do better than the single expert. In general we do not benefit from fusing $k - NN$ or NN experts. Few exceptions exist like when the serial system is used on the Seismic data. The reason could be due to the ability of the serial system to build un-correlated experts even at the presence of redundant features. It is more successful than the parallel system at building uncorrelated experts.
Table A.8: Performance comparison of fusion strategies for the Optimised Conventional system design using Seismic data. The effect of adding features to the termination point of C.S.

<table>
<thead>
<tr>
<th>Method</th>
<th>C.S.</th>
<th>O.C.S.-f</th>
<th>O.C.S.</th>
<th>C.S.-all-feats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>77.68</td>
<td>75.29</td>
<td>77.68</td>
<td>73.00</td>
</tr>
<tr>
<td>Mprod</td>
<td>77.87</td>
<td>75.41</td>
<td>77.87</td>
<td>73.00</td>
</tr>
<tr>
<td>NrMean</td>
<td>77.68</td>
<td>75.29</td>
<td>77.68</td>
<td>73.00</td>
</tr>
<tr>
<td>LnrReg</td>
<td>82.41</td>
<td>82.41</td>
<td>82.41</td>
<td>71.69</td>
</tr>
<tr>
<td>LogReg</td>
<td>67.81</td>
<td>68.67</td>
<td>68.67</td>
<td>52.03</td>
</tr>
</tbody>
</table>

In general applying the parallel method to gaussian experts is successful. However the benefits of this approach are less so when NN experts are involved. In a parallel system when fusing using Sum, MProduct or Logistic regression, Gaussian based systems exhibit the best performance. Gaussian experts are weaker than $k-NN$, hence even in the presence of redundant features each expert is different from the rest and hence fusion is successful on both data types and both systems.

We notice that the heterogeneous system, if fused using regression methods, yields very good results. For BCW data it achieves an optimum performance in a parallel system. The combination of a heterogeneous system and regression fusion yield results that are better than those produced by any heterogeneous system using any other fusion method or a homogeneous system using regression fusion methods. Using the termination condition method the heterogeneous system yields good results although none of the fusion methods yield optimum performance, except for regression strategies. When all features are used in a heterogeneous system Sum and MProduct yield a very good system performance that is better than any type of single expert.

When using all BCW features, both parallel and serial designs respond with the highest performance if Sum or MProduct are used to combine the experts.
Table A.9: Performance comparison of fusion strategies for the Optimised Conventional system design using Seismic data. The effect of removing features from C.S.

<table>
<thead>
<tr>
<th></th>
<th>C.S.</th>
<th>O.C.S.-f</th>
<th>O.C.S.</th>
</tr>
</thead>
<tbody>
<tr>
<td>At termination</td>
<td>Sum</td>
<td>77.68</td>
<td>78.37</td>
</tr>
<tr>
<td>condition</td>
<td>Mprod</td>
<td>77.87</td>
<td>78.36</td>
</tr>
<tr>
<td></td>
<td>NrMean</td>
<td>77.68</td>
<td>79.91</td>
</tr>
<tr>
<td></td>
<td>LnrReg</td>
<td>82.41</td>
<td>82.41</td>
</tr>
<tr>
<td></td>
<td>LogReg</td>
<td>67.81</td>
<td>68.39</td>
</tr>
<tr>
<td>Using all</td>
<td>Sum</td>
<td>73.00</td>
<td>73.19</td>
</tr>
<tr>
<td></td>
<td>Mprod</td>
<td>73.00</td>
<td>73.52</td>
</tr>
<tr>
<td></td>
<td>NrMean</td>
<td>73.00</td>
<td>73.19</td>
</tr>
<tr>
<td></td>
<td>LnrReg</td>
<td>71.69</td>
<td>69.80</td>
</tr>
<tr>
<td></td>
<td>LogReg</td>
<td>52.03</td>
<td>33.33</td>
</tr>
</tbody>
</table>

A.5 Comparing Fusion Strategies

Among all the rules we notice that Sum, MProduct and Linear Regression were most successful. Linear Regression was best on the Seismic data when the serial system was used. Sum and MProduct were best on the BCW data when either system was used. But their superiority was more obvious in the parallel system. Linear regression was close to and mostly better than Logistic regression.

When compared to Sum, MProduct was better when the serial system was used, while Sum yielded better performance when the parallel system was used. This could be related to the independence between experts, among other factors.

When using gaussian experts on Seismic data we noticed that Sum and MProduct performed better under the parallel than under the serial method for the designs produced by the termination condition. In the parallel case the performance is close to the regression methods. The reason NN does not show a good performance when using regression methods is that the covariance matrix in equations A.4 and A.6 becomes singular.
A.6 Conclusion

For all design architecture methods we showed that the proposed integrated design approach have the potential to deliver comparable or improved performance.

We evaluated the performance of different fusion strategies ranging from linear untrainable strategies like Sum and Modified Product to linear trainable strategies such as linear and logistic regression. In general MProduct yielded a better performance than Sum. We showed that for correlated experts, fusion by logistic regression gives superior performance. As the degree of correlation of experts outputs could be easily measured, this could be used as a criterion for the selection of an appropriate fusion rule for the multiple expert system.

The serial and parallel methods were not always superior to the conventional or the single $k-NN$ and $NN$ experts. Further investigation under different data sets and parameters may shed some light on the strength of our proposed method.
Bibliography


and F. Roli, editors, First International Workshop on Multiple Classifier systems,

Hall, 1993.

algorithm for fast classification of patterns. Pattern Recognition Letters, 10:211–

[32] M C Fairhurst and A F R Rahman. Generalised approach to the recognition of
structurally similar handwritten characters using multiple expert classifiers. IEE

[33] K.R. Farrell. Text dependent speaker verification using data fusion. In ICASSP-
95., 1995 International Conference on Acoustics, Speech, and Signal Processing,

[34] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In 13th

and nearest neighbour classifiers. In R. Roy, T. Furuhashi, and P.K.Chawdhry,


[37] P.D. Gader, M.A. Mohammad, and J.M. Keller. Fusion of handwritten word

[38] G. Giacinto and F. Roli. Dynamic classifier selection. In Josef Kittler and Fabio
Roli, editors, First International Workshop on Multiple Classifier Systems, pages


