

This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (<https://dspace.lboro.ac.uk/>) under the following Creative Commons Licence conditions.



For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>

University Library

Author/Filing Title SMITH, Duncan.

Class Mark T

**Please note that fines are charged on ALL
overdue items.**

--	--	--

0403820170



An evolutionary approach to optimising neural network predictors for
passive sonar target tracking

by

Duncan Smith

Doctoral Thesis

Submitted in partial fulfilment of the requirements

for the award of

Doctor Of Philosophy of Loughborough University

October 2009

© by Duncan Smith 2009

13/5/10


Date 13/5/10

Class T

Acc No. 0403820170

CERTIFICATE OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this thesis, that the original work is my own except as specified in acknowledgments or in footnotes, and that neither the thesis nor the original work contained therein has been submitted to this or any other institution for a degree

.....  (Signed)

..... 26th October 2009 (Date)

Abstract

Object tracking is important in autonomous robotics, military applications, financial time-series forecasting, and mobile systems. In order to correctly track through clutter, algorithms which predict the next value in a time series are essential.

The competence of standard machine learning techniques to create bearing prediction estimates was examined. The results show that the classification based algorithms produce more accurate estimates than the state-of-the-art statistical models. Artificial Neural Networks (ANNs) and K-Nearest Neighbour were used, demonstrating that this technique is not specific to a single classifier.

Furthering this work, ensembles of predictors were tested. The outputs of ensembles of ANNs were fused to give the system's output. Initially NCL was used to train the ensemble.

A faster more accurate process was then created. Learning was performed entirely through the genetic design process rather than using the usual ANN learning algorithms to train each ensemble member individually.

A multi-objective genetic algorithm further optimised the ensembles by maximising ensemble diversity and minimising ensemble error. This proved effective giving a 32% improvement in RMS accuracy on a very large semi-synthetic data created as part of this thesis.

The major novelties are the use of classifier ensembles as a time-series predictor in target tracking, using negative correlation learning in target tracking, using genetic algorithms to evolve whole classifier ensembles, with each genetic algorithm individual representing an ensemble rather than an ensemble member, evolving ensemble connection weights.

Keywords:

Target tracking, Neural Network Ensembles, Time series prediction, genetic algorithms

To Vicky

Acknowledgements

Firstly I would like to thank my supervisor Professor Sameer Singh for his help and guidance along the way, and for reminding me of the overall direction when I got lost in the detail.

A big thank you to Xin Yao and Chris Hinde who both provided invaluable feedback and made my viva an extremely informative event which was not just useful for improving the quality of this thesis immeasurably but also as an enjoyable experience in its own right.

I must also thank everyone at QinetiQ, firstly Chris Brook for agreeing to sponsor this work. Also thanks to Roger Manning and Charles Whitworth who made many helpful suggestions during the early months

Prior to attending Loughborough I was an undergraduate student at the University of Leeds I would like to thank my personal tutor Kevin McEvoy for his faith in my ability, without whose patience I would not have completed my bachelors degree.

Further back still I am grateful to the staff of Aldridge School for helping me realise what I could achieve with hard work, especially Steven Randall and David Key who were both an inspiration to me throughout my time there.

Of course I would also like to thank my parents for their enthusiasm for learning that has given me the drive to reach this point. Also for their constant belief in me and my abilities, even when it was contrary to all available evidence! I would also like to thank them for their attention to detail when they read (and reread!) this document.

Most importantly I would like to thank my wife Vicky for her constant support over the last few years, her understanding when I have spent all of my free time studying, and for not getting (outwardly!) frustrated with the ever changing finish date

Duncan Smith

2009

Table of contents

Abstract ..	1
Acknowledgements ..	3
1 Introduction.....	19
1.1 Introduction to target tracking.....	19
1.2 Sonar data processing ..	20
1.2.1 Hydrophones ..	20
1.2.2 Amplifiers ..	20
1.2.3 Beamformer ..	21
1.2.4 Waterfall plot ..	22
1.2.5 Peak detector ..	23
1.2.6 Tracker ..	23
1.2.7 Target Motion Analysis (TMA) ..	23
1.3 Motivation ..	23
1.4 Previous work and limitations ..	24
1.5 Proposed work ..	24
1.5.1 Contribution ..	24
1.5.2 Objectives ..	25
1.6 Structure ..	25
2 State-of-the-art in target tracking.....	27
2.1 Summary ..	27
2.2 Data fusion ..	27
2.3 JDL Level 1 – “Object refinement” ..	29
2.3.1 Data registration ..	29
2.3.2 Data association ..	29
2.3.2.1 Nearest neighbour ..	30
2.3.2.2 Artificial Neural Networks (ANNs) ..	30
2.3.3 Position/attribute estimation.....	30
2.3.3.1 Kalman Filter (KF) ..	31
2.3.3.1.1 Predict.....	31
2.3.3.1.2 Update ..	32
2.3.3.2 Particle filter (PF) ..	33
2.3.3.3 AI approaches..	36
2.3.3.3.1 Artificial Neural Networks (ANNs) ..	36
2.3.4 Classification ..	37
2.3.4.1 Artificial Neural Networks (ANNs) ..	37
2.4 JDL Level 2 – “situation assessment”..	38
2.5 JDL Level 3 – “threat assessment” ..	38
2.6 JDL Level 4 – “process assessment” ..	39
2.6.1 Distributed sensing..	41
2.7 Evaluating state-of-the-art approaches to target tracking ..	43
2.7.1 Summary ..	43
2.7.2 Extended Kalman Filter (EKF) ..	44
2.7.2.1 Kalman filter tuning ..	44
2.7.2.2 Initialisation.....	44
2.7.2.3 Running parameters.....	44
2.7.3 Particle filter. ..	45
2.7.4 Single output ANN.....	45

2.7.5 Experimental design	45
2.7.5.1 Accuracy measurement.....	47
2.7.6 Data.....	48
2.7.6.1 Fully synthetic.....	48
2.7.6.1.1 Set one... ..	48
2.7.6.1.1.1 Summary.....	48
2.7.6.1.1.2 Definition	48
2.7.6.1.2 Set two.....	52
2.7.6.1.2.1 Summary	52
2.7.6.1.2.2 Definition.....	52
2.7.6.1.3 Set three... ..	56
2.7.6.1.3.1 Summary	56
2.7.6.1.3.2 Definition	56
2.7.6.1.4 Set four.....	60
2.7.6.1.4.1 Summary... ..	60
2.7.6.1.4.2 Definition	60
2.7.6.2 Semi-synthetic	64
2.7.6.2.1 Conversion to time-series ..	67
2.7.6.2.2 Results	69
2.7.6.2.3 Statistical comparison of algorithms ..	70
2.8 Conclusions	72
3 Improving upon existing predictors	73
3.1 Summary.....	73
3.2 Introduction.. ..	73
3.3 Problem statement.	74
3.4 Proposal and methodology	74
3.4.1 Binning function.. ..	75
3.4.1.1 Converting bin number to network training output ..	77
3.4.2 Learning algorithms	77
3.4.2.1 K-nearest neighbour (KNN)	78
3.4.2.2 Artificial neural networks (ANN)	78
3.5 Experimental design... ..	79
3.5.1 Experimental design	79
3.6 Results and comparison	79
3.6.1 Synthetic data.....	81
3.6.2 Semi-synthetic data	85
3.6.2.1 Binning parameters	85
3.6.2.2 Input parameters	89
3.7 Conclusions	96
4 Ensembles of predictors	97
4.1 Summary.	97
4.2 Introduction	97
4.3 Ensembles in time-series forecasting.	98
4.4 Negative Correlation Learning	99
4.5 Experimental design	100
4.6 Results	101
4.7 Conclusion.. ..	108
5 Genetic algorithms to create ensembles	109
5.1 Summary	109
5.2 Introduction	109

5.3	Evolving ANNs in the literature	110
5.3.1	Weights.....	110
5.3.2	Architecture.....	110
5.3.2.1	Direct encoding.....	110
5.3.2.2	Indirect encoding.....	111
5.3.3	Architecture and weights	112
5.3.4	Evolving ensembles	113
5.4	Proposal.....	113
5.4.1	Original aim.....	113
5.4.1.1	Attempted hardware solution to running speed problem	116
5.4.2	Improved process	117
5.4.3	Artificial neural network used.....	119
5.4.3.1	Ensemble output fusion	119
5.4.4	Genetic algorithm	119
5.4.4.1	Selection	120
5.4.4.2	Crossing	122
5.4.4.3	Genome.....	122
5.4.4.4	Worked example of ensemble construction	124
5.5	Experiments.....	127
5.5.1	Objective.....	127
5.5.2	Genetic algorithm	127
5.5.3	Neural network ensembles	128
5.6	Results	128
5.6.1	Pure synthetic data	128
5.6.1.1	Experiment 1	128
5.6.1.2	Experiment 2.....	132
5.6.1.3	Experiment 3	135
5.6.1.4	Experiment 4	138
5.6.1.5	Conclusions	140
5.6.2	Semi-synthetic data set	142
5.7	Conclusions	152
6	Multi-objective GA	154
6.1	Summary.....	154
6.2	Introduction.....	154
6.3	Previous work on diversity in GAs	155
6.3.1	Genotype or phenotype diversity.....	156
6.3.2	Restriction of selection procedure.....	156
6.3.3	Restriction of mating	156
6.3.4	Altering mutation rate and population size	157
6.3.5	Fitness sharing	157
6.3.6	Diversity as a MOEA objective.....	159
6.4	Multi-objective Genetic algorithm	160
6.4.1	Evaluating fitnesses.....	161
6.5	Results.....	163
6.6	Conclusions	171
7	Discovering accuracy/diversity balance which gives best overall accuracy.....	173
7.1	Summary.....	173
7.2	Introduction.....	173
7.3	Changes to selection procedure	174
7.4	Results	175

7.5 Conclusions	185
7.5.1 Comparison to NCL.....	186
8 Conclusions and further work.....	188
8.1 Further work.....	190
9 References.....	193
9.1 World Wide Web references.....	221
Appendix A – Description of the genetic code	223
Appendix B – Example data after sliding window.....	224
Appendix C – Example data with binning function applied.	226
Appendix D - Optimisation results.....	228
EKF	228
Particle Filter	232
Single Output Neural Network	245
Pure synthetic 1	245
Pure synthetic 2	253
Pure synthetic 3.	261
Pure synthetic 4	269
Semi-synthetic	277
Appendix E - Standard definitions	285
Artificial Neural Networks	285
Training algorithms.	286
Genetic algorithms.	287
Multi-objective genetic algorithms	288
Pareto optimality	288
Measuring diversity	290
Measuring accuracy	291
Geometric Mean of Relative Absolute Error (GMRAE)	292
Median RAE (MdRAE)	292
Median Absolute Percentage Error (MdAPE)	292

Index of Tables

Table 1	The tenth of the data set used in each run of the ANN.....	46
Table 2	The confidence that algorithm a is more accurate than algorithm b given the number of folds that algorithm a was more accurate than b out of a total number of ten folds.	71
Table 3	The outputs (per fold) of the three algorithms ANN, EKF & PF on the semi-synthetic data set	71
Table 4	The number of folds for which algorithm a is more accurate than algorithm b for the ANN, EKF & PF...	71
Table 5	The confidence that algorithm a is more accurate than algorithm b for the ANN, EKF & PF...	71
Table 6	The best RMS bearing error for each algorithm on each data set.....	72
Table 7	The number of entries in the KNN database and the number of examples per entry against number of inputs when using uniform binning with a maximum bearing of 10, a bin size of 0.1 and training with the semi-synthetic data set.....	91
Table 8	Results per fold for the ANN, EKF, PF & GANN	95
Table 9	The number of folds for which algorithm A was more accurate than algorithm B for the ANN, EKF, PF & GANN	95
Table 10	The percentage confidence that algorithm A was more accurate than algorithm B for the ANN, EKF, PF & GANN	95
Table 11	The best RMS bearing error for each algorithm on each data set.....	95
Table 12	The best RMS bearing error for NCL trained ensembles on each data set....	106
Table 13	The best value for NCL trained ensembles on each metric for the semi-synthetic data set..	106
Table 14	Results per fold for the ANN, EKF, PF, GANN & NCL	107
Table 15	The number of folds for which algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN & NCL	108
Table 16	The percentage confidence that algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN & NCL..	108
Table 17	The tenth of the data set used in each run of the ensemble	120
Table 18	The ensemble chromosome.....	123
Table 19	The ANN chromosome.	123
Table 20	The hidden node chromosome.....	124
Table 21	Example of binary to floating point conversion (minimum value 0, maximum 1, 5 bits).....	124
Table 22	Example binary chromosome to ensemble parameters	126
Table 23	Results per fold for the ANN, EKF, PF, GANN, NCL & GA.....	144
Table 24	The number of folds for which algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN, NCL & GA.	144
Table 25	The percentage confidence that algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN, NCL & GA....	145
Table 26	The best RMS bearing error for GA generated ensembles on each data set..	152
Table 27	The best value for GA generated ensembles on each metric on the semi-synthetic data set...	152
Table 28	The best RMS bearing error for MOGA generated ensembles on each data set	171
Table 29	The best value for MOGA generated ensembles on each metric on the semi-	

synthetic data compared to the SOGA and the best NCL trained ensemble	171
Table 30 Results per fold of the best of the new algorithms, the Gaussian binning Artificial Neural Network against the per-fold results for the baseline algorithms tested in section 2.7..	177
Table 31 The number of folds for which algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN, NCL, GA and λ MOGA.....	177
Table 32 The percentage confidence that algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN, NCL, GA and λ MOGA	178
Table 33 The best RMS bearing error for MOGA generated ensembles on each data set	185
Table 34 The best value for MOGA generated ensembles on each metric.	185
Table 35 RMS bearing error for the PF with different values for parameter plant noise between 0 and 5000 as tested on pure synthetic data set 1.....	234
Table 36 RMS bearing error for the PF with different values for parameter plant noise between 0 and 5000 as tested on pure synthetic data set 2	235
Table 37 RMS bearing error for the PF with 200 particles with plant noise values between 0 and 5000 as tested on pure synthetic data set 3.....	237
Table 38 RMS bearing error for the PF with 200 particles with plant noise values between 0 and 5000 as tested on pure synthetic data set 4.....	238
Table 39 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 1	240
Table 40 Pure synthetic data 2	241
Table 41 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 3	242
Table 42 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 4.....	243
Table 43 Run time in seconds on pure synthetic data over number of particles in particle filter between 50 and 1000 particles.....	244

Index of Figures

Figure 1 Data flow in a typical passive sonar system.....	20
Figure 2 Omnidirectional response of a single hydrophone.	22
Figure 3 Unidirectional response of a beamformed 2d array of hydrophones	22
Figure 4 Unidirectional (though ambiguous) response of a beamformed 1d line array of hydrophones	22
Figure 5 Three possible data sources for a network enabled submarine	28
Figure 6 Flow chart of experimental plan.....	47
Figure 7 Small data set 1.....	49
Figure 8 Small data set 1 restricted between -180° and $+180^\circ$	49
Figure 9 Output of the target tracking algorithms on simple data set 1	50
Figure 10 Bearing residuals of the target tracking algorithms on simple data set 1....	51
Figure 11 Bearing residuals of the target tracking algorithms on simple data set 1... .	52
Figure 12 Small data set 2.. . . .	53
Figure 13 Small data set 2 restricted between -180° and $+180^\circ$	53
Figure 14 Output of the target tracking algorithms on simple data set 2...	54
Figure 15 Bearing residuals of the target tracking algorithms on simple data set 2.....	55
Figure 16 Cumulative bearing residuals of the target tracking algorithms on simple data set 2	56
Figure 17 Small data set 3	57
Figure 18 Small data set 3 restricted between -180° and $+180^\circ$	57
Figure 19 Output of the target tracking algorithms on simple data set 3.....	58
Figure 20 Bearing residuals of the target tracking algorithms on simple data set 3.....	59
Figure 21 Cumulative bearing residuals of the target tracking algorithms on simple data set 3	60
Figure 22 Small data set 4	61
Figure 23 Small data set 4 restricted between -180° and $+180^\circ$	61
Figure 24 Output of the target tracking algorithms on simple data set 4.	62
Figure 25 Bearing residuals of the target tracking algorithms on simple data set 4.....	63
Figure 26 Cumulative bearing residuals of the target tracking algorithms on simple data set 4.....	64
Figure 27 A simple example of processing beam data for a sonar with eight beams for a single instant.....	65
Figure 28 Plan view of simulated target over time.	67
Figure 29 Time/bearing plot of sonar measurements.	67
Figure 30 Plot of recorded noise, shown with time over bearing.	67
Figure 31 Plot of simulated data with recorded noise, overlaid with threshold track, shown with time over bearing.	67
Figure 32 Sliding window data conversion, black lines represent input data, while the orange denotes the value to be predicted, converting a continuous time series into discrete samples	68
Figure 33 RMS bearing residuals for single output ANN, EKF and PF when run on the semi-synthetic data set with number of inputs varying from 1 to 45.	69
Figure 34 An illustration of uniform binning of prediction space	75
Figure 35 An illustration of Gaussian binning of prediction space.....	75
Figure 36 Cumulative bearing residual for Gaussian Binning ANN, Uniform Binning ANN, Gaussian Binning KNN and Uniform Binning KNN when run on synthetic data	

Figure 54 Using development rules to create an ANN112

Figure 55 Flow chart of method originally proposed115

Figure 56 Flow chart of new methodology... .. .118

Figure 57 Evaluating ensemble accuracy 121

Figure 58 Ensemble consisting of two ANNs produced by parameters in Table 22 ...126

Figure 59 Size of chromosome given different maximum number of ANNs 127

Figure 60 Output bearing values for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 1129

Figure 61 Output bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 1130

Figure 62 Cumulative bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 1.....131

Figure 63 Output bearing values for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 2132

Figure 64 Bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 2.133

Figure 65 Cumulative bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 2134

Figure 66 Output bearing values for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 3 135

Figure 67 Bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 3 136

Figure 68 Cumulative bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 3..... .. .137

Figure 69 Output bearing values for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 4138

Figure 70 Bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 4.139

Figure 71 Cumulative bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 4140

Figure 72 RMS bearing residuals of the ensembles in the GA population compared to the EKF, PF, Gaussian Binned ANN and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line. 143

Figure 73 Entropy of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line..... .. . 145

Figure 74 Kohavi-Wolpert diversity of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the

range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line. 146

Figure 75 Generalised diversity of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line..... 147

Figure 76 MdRAE of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line..... 148

Figure 77 MdAPE of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line. 149

Figure 78 GMRAE of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line.. 150

Figure 79 Hamming distance of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line. 151

Figure 80 RMS bearing residuals of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only), EKF, PF, Gaussian Binned ANN and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line.... .. 163

Figure 81 Entropy of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line..... 164

Figure 82 Kohavi-Wolpert diversity of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line... .. 165

Figure 83 Generalised diversity of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line..... 166

Figure 84 MdRAE of the ensemble with the highest entropy averaged across all folds 167

Figure 85 MdAPE of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line..... 168

Figure 86 GMRAE of the ensembles in the MOGA population compared to the SOGA

(accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line... 169

Figure 87 Hamming distance of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line. ... 170

Figure 88 Final generation RMS error over percentage of individuals selected for their accuracy were the line represents the mean level of accuracy in the MOGA population in the final generation, while the error bars show the range between the most and least accurate individuals..... 175

Figure 89 RMS bearing residuals of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only), EKF, PF, Gaussian Binned ANN and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line 176

Figure 90 Entropy of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line.. . . . 178

Figure 91 Kohavi-Wolpert diversity of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line..... 179

Figure 92 Generalised diversity of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line 180

Figure 93 MdRAE of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line..... 181

Figure 94 MdAPE of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line.. . . . 182

Figure 95GMRAE of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line..... 183

Figure 96 Hamming distance of the ensembles in the MOGA population for the MOGA

with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line.....	184
Figure 97 RMS bearing error for the EKF with different values for parameters; standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on pure synthetic data set 1.....	228
Figure 98 RMS bearing error for the EKF with different values for parameters; standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on pure synthetic data set 2.	229
Figure 99 RMS bearing error for the EKF with different values for parameters; standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on pure synthetic data set 3.....	230
Figure 100 RMS bearing error for the EKF with different values for parameters; standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on pure synthetic data set 4.	231
Figure 101 RMS bearing error for the EKF with different values for parameters, standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on semi-synthetic data set	232
Figure 102 RMS bearing error for the PF with different values for parameter plant noise between 0 and 5000 as tested on pure synthetic data set 1.....	233
Figure 103 RMS bearing error for the PF with different values for parameter plant noise between 0 and 5000 as tested on pure synthetic data set 2.....	234
Figure 104 RMS bearing error for the PF with 200 particles with plant noise values between 0 and 5000 as tested on pure synthetic data set 3.....	236
Figure 105 RMS bearing error for the PF with 200 particles with plant noise values between 0 and 5000 as tested on pure synthetic data set 4.....	237
Figure 106 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 1....	239
Figure 107 Pure synthetic data set 2	240
Figure 108 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 3	241
Figure 109 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 4.....	242
Figure 110 Run time in seconds on pure synthetic data over number of particles in particle filter between 50 and 1000 particles.....	243
Figure 111 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 5 input nodes trained with backpropagation on synthetic data set 1.....	245
Figure 112 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 10 input nodes trained with backpropagation on synthetic data set 1....	246
Figure 113 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 15 input nodes trained with backpropagation on synthetic data set 1....	247
Figure 114 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 20 input nodes trained with backpropagation on synthetic data set 1 ..	248
Figure 115 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural	

learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 30 input nodes trained with backpropagation on synthetic data set 3... 266

Figure 133 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 35 input nodes trained with backpropagation on synthetic data set 3... 267

Figure 134 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 40 input nodes trained with backpropagation on synthetic data set 3...268

Figure 135 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 5 input nodes trained with backpropagation on synthetic data set 4.....269

Figure 136 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 10 input nodes trained with backpropagation on synthetic data set 4....270

Figure 137 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 15 input nodes trained with backpropagation on synthetic data set 4... 271

Figure 138 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 20 input nodes trained with backpropagation on synthetic data set 4... 272

Figure 139 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 25 input nodes trained with backpropagation on synthetic data set 4....273

Figure 140 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 30 input nodes trained with backpropagation on synthetic data set 4... 274

Figure 141 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 35 input nodes trained with backpropagation on synthetic data set 4. 275

Figure 142 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 40 input nodes trained with backpropagation on synthetic data set 4....276

Figure 143 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 5 input nodes trained with backpropagation on semi-synthetic data set
277

Figure 144 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 10 input nodes trained with backpropagation on semi-synthetic data set
278

Figure 145 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 20 input nodes trained with backpropagation on semi-synthetic data set
 279

Figure 146 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 25 input nodes trained with backpropagation on semi-synthetic data set
280

Figure 147 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural

network with 25 input nodes trained with backpropagation on semi-synthetic data set
 281
 Figure 148 RMS bearing residual over number of hidden nodes between 5 and 45 and
 learning parameter between 0.01 and 0.9 for a single output feedforward neural
 network with 30 input nodes trained with backpropagation on semi-synthetic data set
282
 Figure 149 RMS bearing residual over number of hidden nodes between 5 and 45 and
 learning parameter between 0.01 and 0.9 for a single output feedforward neural
 network with 35 input nodes trained with backpropagation on semi-synthetic data set
283
 Figure 150 RMS bearing residual over number of hidden nodes between 5 and 45 and
 learning parameter between 0.01 and 0.9 for a single output feedforward neural
 network with 40 input nodes trained with backpropagation on semi-synthetic data set
 284
 Figure 151 Artificial neuron or node. 285
 Figure 152 Artificial neural network. 286
 Figure 153 An example Pareto Front. 289
 Figure 154 Calculating dominance ranking .. 290

1 Introduction

This thesis describes a novel technique for tracking sonar targets on a passive sonar system using classification algorithms. It also describes both how these classifier-predictors can be improved by forming ensembles of predictors trained with Negative Correlation Learning (NCL), how these ensembles may be created using a genetic algorithm (GA), and finally how the accuracy of the classifier-predictors can be further improved by using diversity as an objective in a Multiple Objective GA

1.1 Introduction to target tracking

Passive sonar is a system for detecting boats and ships at sea, (both of which are referred to as sonar targets), and works by 'listening' to the noise that these targets make. Hydrophones which act as underwater microphones are used to detect changes in pressure in several directions, allowing the operator to determine from which direction the detected sound originates, resulting in a bearing to the target. In addition to measurements originating from man-made sources, there is a relatively high level of background noise from both biological sources such as singing whales and snapping shrimps and atmospheric sources such as rain and waves. In addition, sound does not travel in straight lines in water, as the sound waves are refracted by changing density, temperature and salinity in the seawater. This combination of clutter and measurement uncertainty leads to very noisy data from which it is essential to extract time-series in order to estimate the positions of targets.

1.2 Sonar data processing

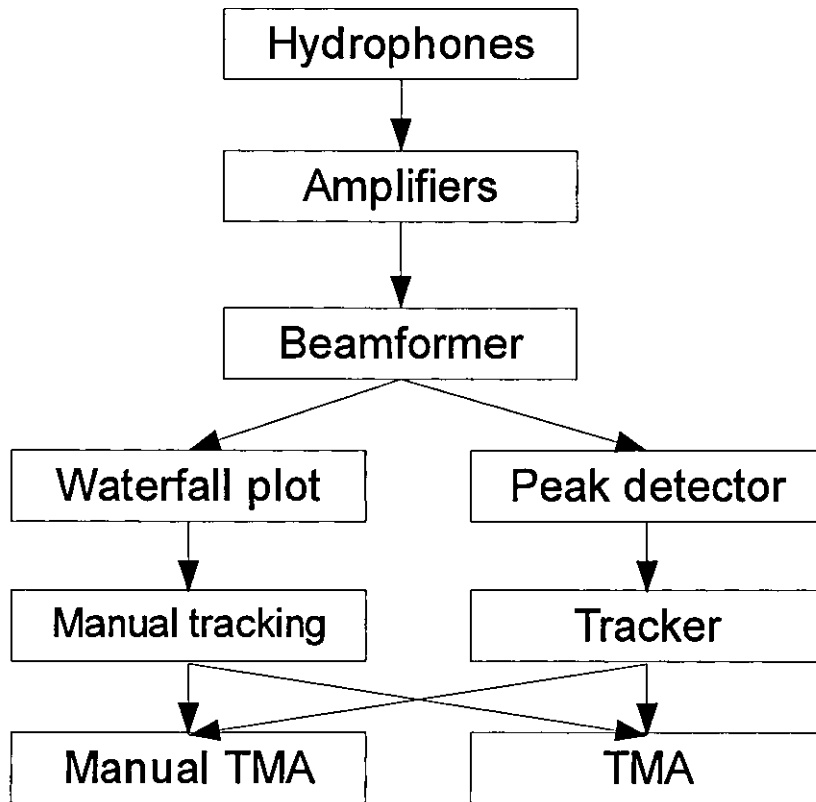


Figure 1 Data flow in a typical passive sonar system

Figure 1 shows a block diagram of a typical passive sonar system, the various components of the system are described in the following few paragraphs, summarised from [282].

1.2.1

Hydrophones

In passive sonar a hydrophone is essentially a microphone which has been designed to operate underwater, from the Greek hydro meaning water and phoni meaning sound or voice.

The hydrophones are usually arranged in a regular pattern to permit processing to establish the direction of the source sound. This pattern may be a straight line, or a 2d or 3d grid.

1.2.2

Amplifiers

The output of each of the hydrophones is fed through an amplifier to increase the signal to a level sufficient for later processing stages

1.2.3

Beamformer

The beamformer is the first level of data processing in a passive sonar system.

Amplified data from the hydrophones is analysed, taking advantage of interferometry to convert an array of omnidirectional hydrophones into a single, highly directional source. Using data processing the direction of the beam formed may be steered in any direction, allowing the sound intensity in each direction to be calculated. For each slice of time an array of intensity data may be obtained.

Figures 2 to 4 shows the pattern of responsiveness of beamformed data from different configurations of hydrophones. Figure 2 shows the output of a single hydrophone, and demonstrated a completely uniform responsiveness in all directions. Figure 4 shows the output of a line array it is capable of forming a central beam which can be used to find the bearing of the target, however it can be seen that it is incapable of discerning between port and starboard. Adding an extra dimension can be seen to overcome this shortfall in Figure 3. Adding a third dimension to the hydrophone array allows beamforming not just in the bearing to the target, but also in the elevation, allowing a truly three dimensional picture to be compiled.

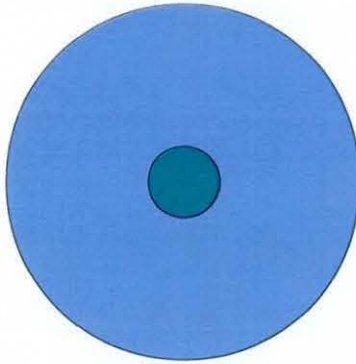


Figure 2 Omnidirectional response of a single hydrophone.

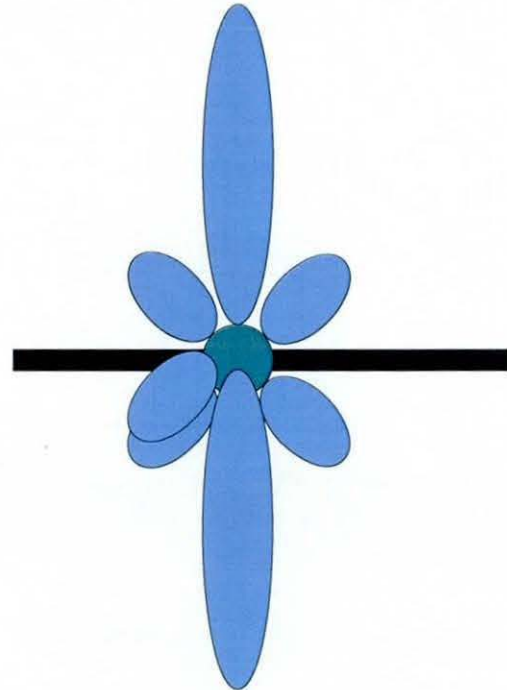


Figure 4 Unidirectional (though ambiguous) response of a beamformed 1d line array of hydrophones

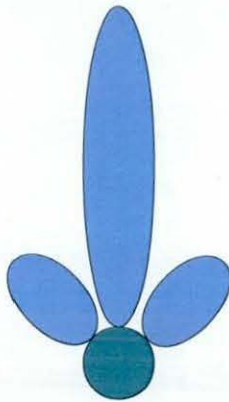


Figure 3 Unidirectional response of a beamformed 2d array of hydrophones

1.2.4

Waterfall plot

One traditional way to display this data is in a time bearing waterfall plot. On this plot different levels of intensity are shown by changing the brightness or colour of a pixel in the plot. Each horizontal line represents a slice of time, as each new array of intensity data arrives from the beamformer it is displayed at the top of the plot, displacing all other data which moves down the plot one place. The oldest data is discarded as it flows off the bottom of the plot.

Examples of waterfall plots are given in figures 30 and 31 (page 65). The first showing background noise, while the second shows a straight running target passing endfire. There are two endfire regions on a line array sonar, one at 0° and one at 180° . As a target passes either of these it changes from being on the port side of the detecting sonar to the starboard or vice-versa. As a line array sonar cannot

differentiate between the two directions, on the waterfall plot it looks as though the target has changed direction. Also note that beam width is wider at endfire than at broadside, or 90° .

1.2.5 Peak detector

For the data to be of use to later stages in the system, it is passed through a peak detector. At its simplest the peak detector applies a threshold, any point above the threshold is stored as a measurement for use by the tracker. Some of these points will have originated from a target, however many if not all will be background noise

1.2.6 Tracker

The tracker takes the collection of points, and attempts to group them together into sets, where each set is believed to originate from the same target. This enables the estimation of the position and trajectory of the target, and to help reject the clutter measurements. Given all previous measurements in a track, the tracker predicts the next measurement in order to aid isolating the true measurement from the noise. The tracker can have no prior knowledge of the kinematics of the target, or whether any single detection is a true target detection or clutter.

1.2.7 Target Motion Analysis (TMA)

Once the data has been processed by the tracker into a set of point measurements associated to a real world object, it is possible to start TMA. This is the process of establishing the kinematics of the original object from the measurements. At its simplest this may mean estimating the probable speed and calculating the best line fit for the original trajectory and position given the input measurements.

1.3 Motivation

This thesis focuses on the stage internal to the tracker which predicts the next bearing in the time series, as improvements here directly lead to improvements in the overall accuracy of the tracker. Improved accuracy in a target tracker allows more accurate localisation of targets through TMA, which is of crucial importance for military usage where a few tenths of degrees of accuracy may make the difference between life and death.

1.4 Previous work and limitations

Many different techniques have been proposed for forecasting values in a bearing values time series. This thesis focuses on three baselines. The first two; the Extended

Kalman Filter (EKF) and the Particle Filter (PF) are two widely used methods in passive sonar tracking, while the third a vanilla feed-forward backpropagation trained ANN is a commonly used Artificial Intelligence (AI) method which has achieved some limited success in the area.

Both of the first two baselines, the EKF and the PF require a model of the target motion in order to operate. Though in theory this model could be arbitrarily complex, in practice many assumptions are made in their construction which limits their achievable accuracy. Although the ANN's performance does not have this limit, it has been found to be equivalent in terms of output accuracy with the other two baselines.

1.5 Proposed work

The proposed technique in this thesis is to use a multi-objective genetic algorithm to create an ensemble of Artificial Neural Networks (ANNs) to perform bearing predictions on bearing time series. Each ANN in the ensemble is a classifier, used as a bearing predictor in a manner also created and demonstrated within this thesis. The aim is to create a predictor which like the vanilla ANN is not limited by model accuracy, while also outperforming the other baselines.

1.5.1

Contribution

There are many novel features of this thesis. Firstly a methodology is described for using a classification algorithm as a bearing predictor for sonar target tracking is new which allows the use of any number of classification algorithms to be used in a completely new way in sonar target tracking.

Further to this the work was extended to use ensembles of classifiers to enhance the predictions; not only have ensembles not previously been used to perform target tracking, but NCL has not previously been used to train a target tracking ensemble.

A Genetic Algorithm (GA) was created which can both design an ensemble of Artificial Neural Networks (ANNs) and train it in a single step. This is the first time that an ensemble has been constructed in such a way, and a multi-objective form of the algorithm is shown to be highly effective at creating optimal ensembles which, unlike ensembles trained with NCL, have structural as well as learned diversity.

Most importantly however, the largest source of novelty here is the discovery that as long as ensembles are created as whole entities rather than component parts, all of the advantages provided by techniques such as NCL which stimulate diversity to in turn increase accuracy may be obtained through use of evolution with multiple objectives.

This significantly reduces computational requirements of training and testing the ensembles, and greatly simplifies the process of creating such an ensemble when compared to approaches that require a learning algorithm to teach the ANN. This may be applied to any GA creating ensemble classifiers for any purpose, allowing a form of NCL to be applied not only to ANNs, but to any classifier or predictor which may be describe with a chromosome.

1.5.2

Objectives

The aim of this thesis is to evaluate the proposed approach against established baselines which represent the state-of-the-art using multiple measures of prediction accuracy and classifier diversity. Five data sets of varying degrees of realism are used to test the proposed solution and the baselines. The learning algorithm based tests make use of 10 fold cross validation to ensure reliable results, while the statistically based baselines are tested on the full data set in each case

It was hoped that the result will be a technique which is more accurate than the baselines, and not limited by the assumptions which must be made to construct a mathematical model of a system, as is the case for the EKF and the PF.

1.6 Structure

The structure of the rest of the document is as follows, the current standard approaches to target tracking are described and evaluated in chapter 2 Chapter 3 improves upon these approaches presenting a novel approach to bearing prediction using Artificial Neural Networks (ANNs) and K-Nearest Neighbour. Chapter 4 further improves the predictors by using ensembles of ANNs trained with Negative Correlation Learning. Chapter 5 attempts to both speed up and automate the process of designing the ensemble using a GA. This is further enhanced in chapter 6 with a multi-objective GA to mimic the NCL. The mimicking of NCL is further improved in chapter 7 to include the λ parameter which governs the balance between accuracy and diversity. Conclusions and recommendations for further work are given in chapter 8. Finally chapter 9 gives the references.

2 State-of-the-art in target tracking

2.1 Summary

Although the new techniques outperformed the baselines in chapter 3, it was only by a very narrow margin. This chapter provides a literature review of the state of the art in target tracking and data fusion, exploring what techniques have already been utilised to improve results. It is anticipated that from this it will be possible to create a novel approach which will improve on the results obtained.

2.2 Data fusion

One method of improving tracker estimates is to combine the results of multiple sensors. This can provide more accurate information than using a single sensor [284], this allows either improved accuracy from existing sensors or the same performance from smaller or cheaper sensors. This chapter covers all aspects of tracking, from single sensor tracking to tracking and fusion across multiple platforms. This literature review section has been written to complement the landmark survey paper on the subject [108], adding some of the notable breakthroughs in target tracking of the last decade in fields such as sensor management and distributed sensing. Target tracking and Multi Sensor Data Fusion (MSDF) are used in many diverse fields, although most of the literature addresses the fields of military target tracking or autonomous robotics [188].

Military distributed data fusion is used to facilitate Network Centric Warfare (NCW) [48][205] or Network Enabled Capability (NEC) [269]. If platforms such as warships and aeroplanes are networked together, and their data is shared, then they will be able to compile a more accurate picture of their environment than with just data from their own sensors. An NEC system contains three vital components [110]

1. A collection of sensors to generate observations
2. An automatic processing system to convert data into information and knowledge.
3. A high-speed communications network to enable the process.

Sensors may be clustered together such as on a submarine, which may have several sonars on-board, or may be carried individually by soldiers [259]. Henceforth the word “platform” will be used to describe any object that carries sensors. At any fusion

processing node, data may therefore come from one of three sources [185] (see Figure 5)

1. Data type 1: Data from a platform's own sensors, known as 'organic data'
2. Data type 2: Network connections to other platforms.
3. Data type 3: A database of data previously received, and of local track estimates

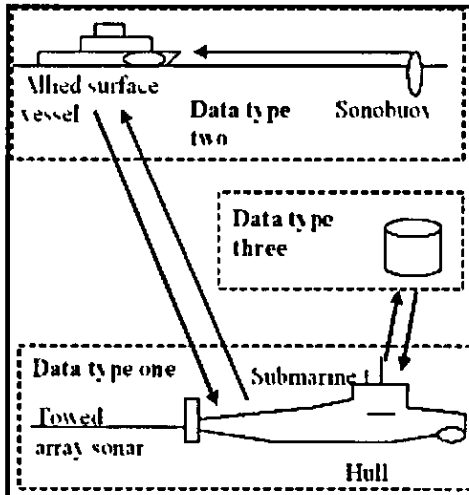


Figure 5 Three possible data sources for a network enabled submarine

Traditionally, military data fusion architectures have been centralised or hierarchical [63]. There are however many advantages to decentralised schemes, which include lighter processing load, no requirement for a single centralised database, lower communication load, reduced possibility of data flow bottlenecks, and high survivability as there is no longer a single point of failure [176].

To facilitate decentralised fusion, three main issues need to be addressed:

- 1 Architecture – The way in which nodes connect and share information. For a detailed coverage of this aspect of MSDF see [205], [164] and [95] for a military perspective, or [298] for autonomous systems.
- 2 Sensor management – The way in which sensors are placed to maximise coverage of an area for different tactical goals [299].
- 3 Algorithms – The way in which processing should be performed.

Although this chapter focuses on the military applications of MSDF, it is also readily applicable to robotics. Robots are required to move around autonomously in unknown environments. Due to factors such as cost, reliability and ease of use, the two most common sensors on this sort of mobile robot are ultra-sonic sonars and digital video cameras [26][70]. MSDF is required to combine and process the data. This has

traditionally been performed by some form of Kalman [96] or Bayesian filter, however in recent years there has been a trend towards the use of soft techniques such as fuzzy logic and artificial neural networks (ANNs) [209]

Although over thirty fusion architectures have been proposed [240], the most widely cited model for data fusion was created by the American Joint Directors of Laboratories Data Fusion Sub-panel [274]. This divided the data fusion process into four levels, which make up a hierarchy of processing. Although this is by no means the only hierarchy for data fusion, and is primarily focussed on military applications, it does provide a useful structure with which to classify fusion algorithms. Sections 2.3 to 2.6 are divided into the four levels of the JDL model to enable similar algorithms to be compared.

2.3 JDL Level 1 – “Object refinement”

Object refinement is usually partitioned into data registration, data association, position attribute estimation and identification [109]. These four categories and the algorithms that fit within them are outlined in sections 2.3.1 to 2.3.4. Some algorithms do not directly fit into a single category, [166][88] and [221] for example all created algorithms which estimated attributes and performed identification complementary processes by fusing the information from two or more sensors Association and state estimation has also been performed in a single step [155] to improve performance.

2.3.1

Data registration

Data registration functions align the data into a common frame of reference. This is often to change coordinate systems from self-centred Cartesian co-ordinates to latitude, longitude and height above sea level for example.

2.3.2

Data association

The association step compares measurements, and attempts to collect measurements originating from the same real world object into a single track. The difficulty is in distinguishing from which target, if any, each measurement originates. This is addressed by measurement-to-track association.

In a distributed system, association can also be the step where tracks from different processing nodes are compared, to combine tracks that are estimating the state of the same real world object. This is track-to-track association Sections 2.3.2.1 to 2.3.2.2 describe the various approaches for data association

2.3.2.1 Nearest neighbour

Nearest neighbour is the simplest form of association algorithm. In this algorithm, the nearest measurement to the established track is chosen to update the track. This algorithm is very simple, and capable of finding a viable solution with very little computational cost. However, in a dense environment this may lead to many pairings with a similar probability, so errors are typically large [28]. “All neighbour” is another related technique in which all measurements within a gated region are included in the track [28].

2.3.2.2 Artificial Neural Networks (ANNs)

A simple introduction to ANNs is given in appendix E. Track to track data association takes the tracks formed on multiple sensors and attempts to associate or group the tracks that correspond to the same target. With more than two targets, this problem is NP hard, and an approximation technique is required to find a solution. [295] proposed a way of using ANNs to solve this problem. It was shown by [295] that this neural network approach, based upon Hopfield neural networks always finds the optimal solution 17.4% of the time, and found a solution that approximates the true solution the rest of the time.

2.3.3

Position/attribute estimation

Position and attribute estimation is the process of taking the associated measurements and calculating the target’s state. An example is Target Motion Analysis (TMA) for passive sonar. Passive sonars can only measure the bearing of the target, not the distance. It is necessary to perform TMA to calculate the range and velocity of the target. In sections 2.3.3.1 to 2.3.3.3, we review the most popular methods for position/attribute estimation.

2.3.3.1 Kalman Filter (KF)

The Kalman Filter (KF) [153] was first proposed in the 1960s and it is the most commonly used technique in target tracking and robot navigation ever since. The basic KF has been shown to be a form of Bayesian filter [120], that is optimal estimator for linear Gaussian systems. Given a series of noisy measurements, the KF is capable of estimating the state of the system.

An extension to the KF is, the Extended Kalman Filter (EKF) [16]. This enables data such as bearings-only passive sonar data to be used in the KF. Due to the linearisation step, the EKF is suboptimal. The EKF is the most popular tool in the literature for sensor fusion in mobile robot navigation. The procedure for applying the EKF to target tracking is described in the rest of this section

The measurements are input to the algorithm as a series of scalar bearing values. The KF state at time i is described by the 4×4 covariance matrix P_i , and the state vector \hat{x}_i ,

$$\hat{x}_i = [x \quad y \quad \dot{x} \quad \dot{y}]^T \quad \hat{x}_{i|} = [x \quad y \quad x' \quad y']$$

where (x, y) is the target position relative to the target in metres, and (x', y') is the speed of the target in the north and east directions respectively. The start position of the target is unknown, so it is initialised to a position at a preset distance along the first bearing measurement received. As the input is in a different format to the state matrix a function to convert the state vector to an estimated measurement

$$f(\hat{x}) = \tan^{-1} \frac{x}{y}$$

Where x and y are the first and second elements of the state matrix

The state transition matrix Φ is

$$\Phi = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where Δt is the time difference between the most recent measurement and the preceding measurement. The EKF can be divided into two parts, the update and predict stages, described here in sections 2.3.3.1.1 and 2.3.3.1.2.

2.3.3.1.1 Predict

$$\hat{x}_{i|t-1} = \Phi x_{i-1|t-1}$$

$$P_{i|t-1} = \Phi_i P_{i-1|t-1} \Phi_i^T + Q_i$$

Where Q_i is the system noise matrix containing random numbers taken from a Gaussian distribution along the lead diagonal. The random values in Q_i are regenerated for each update.

$$Q_i = \begin{bmatrix} rand() & 0 & 0 & 0 \\ 0 & rand() & 0 & 0 \\ 0 & 0 & rand() & 0 \\ 0 & 0 & 0 & rand() \end{bmatrix}$$

2.3.3.1.2 Update

The Jacobian.

$$H = \begin{bmatrix} \frac{\hat{x}_{(i|l-1)}(2)}{\hat{x}_{(i|l-1)}(1)^2 + \hat{x}_{(i|l-1)}(2)^2} & \frac{-\hat{x}_{(i|l-1)}(1)}{\hat{x}_{(i|l-1)}(1)^2 + \hat{x}_{(i|l-1)}(2)^2} & 0 & 0 \end{bmatrix}^T$$

$$\text{Where } \hat{x}_{i|l-1} = \begin{bmatrix} \hat{x}_{i|l-1}(1) \\ \hat{x}_{i|l-1}(2) \\ \hat{x}_{i|l-1}(3) \\ \hat{x}_{i|l-1}(4) \end{bmatrix}$$

The measurement covariance: $z_i = x_i - f(\hat{x}_{i|l-1})$

The residual covariance: $S_i = H_i P_{i|l-1} H_i^T + \sigma_i$ where σ_i is the standard deviation of the input bearings

Optimal Kalman gain: $K_i = P_{i|l-1} + H_i^T S_i^{-1}$

The updated state estimate: $\hat{x}_{i|l} = \hat{x}_{i|l-1} + K_i z_i$

The updated covariance: $P_{i|l} = (I - K_i H_i) P_{i|l-1}$

Both the KF and EKF were originally used on the data from a single sensor. [294] first developed the idea of combining information from local sensors at a central fusion node to form a more accurate global estimate. The drawback of this algorithm was that each local sensor requires the global estimate, which required two-way communication, and negates some of the advantages of parallelisation.

It has also been proven [97] that when the KF is used at a central fusion node to fuse the results of multiple local KFs, the results may be improved by feeding the global estimate back to the local filters as their prior state for the next iteration. As the outputs of the local filters are correlated in time, the performance of such a system can be further improved by only outputting every n^{th} measurement to the global tracker, to obtain near optimal performance [186].

An information theoretic view of the KF and EKF has also been suggested [115]. The Information Filter (IF) or inverse covariance filter is a KF that estimates the

information state vector, y , defined $y \equiv P^{-1}x$ where x is the traditional state vector, and P is its covariance. The covariance of the information state vector is the inverse of the covariance of the state vector, also known as the Fisher Information Matrix or Information Matrix. In this way, the filter estimates the information matrix directly. This form of filter is especially beneficial when the state vector is larger than the measurement vector.

In cases where the measurement model is highly non-linear, even the EKF may diverge. In this situation, the Sigma Point Kalman Filter family of algorithms can be used [278]. Rather than circulating only the mean through the algorithm, SPKFs circulate a collection of precisely selected points around the mean, called sigma points. In using several points, the non-linearity is more accurately modelled. The use of several points may make this appear similar to a Particle Filter (see section 2.3.3.2), however SPKFs requires an order of magnitude fewer points, and are therefore far less computationally expensive. SPKFs include the Unscented Kalman Filter (UKF) [150] [277] observed however that even UKFs are still limited to Gaussian distributions.

2.3.3.2 Particle filter (PF)

Earlier attempts at improving upon the results of the EKF involved using an IMM. By setting the different models to represent different Gaussian distributions taking a weighted average of the Gaussian results, arbitrary distributions could be modelled. However, this method cannot be applied automatically [83].

The Kalman Filter does make assumptions that the noise on the data is Gaussian, and that the standard deviation is known. Unfortunately, other than in simulated experiments, the error is rarely either exactly known or Gaussian, so a method for filtering using arbitrary probability density functions (PDFs) is required.

A direct approach to modelling the PDF is to divide the search space into a grid, and using the spaces in the grid to represent points in the PDF. Choosing the grid is however a non-trivial task, and especially in multidimensional space a large number of grid points may become necessary.

The particle filter also known as the Bootstrap, Condensation or Monte-Carlo filter was developed to counter this very problem. Rather than having a fixed grid to represent the PDF, these used movable 'particles'. Early versions of the particle filter used a

fixed number of particles, which led to the particles collapsing to a single point and the filter diverging in the same way that a KF does with a poorly described Gaussian [46]. [100] developed the 'bootstrap filter' or Sequential Importance Resampling (SIR) PF. This introduced a resampling step required to prevent the filter diverging which removed the particles with the lowest weights at each step, and created new particles at points where the weight was the highest. The bootstrap filter was shown to be more accurate than the EKF for tracking in a system with non-linear measurements, such as bearings only tracking. Since then several variants of this bootstrap have been developed, such as versions for multitarget tracking [138][280] and for manoeuvring targets using an IMM PF approach [30][33]. PFs have been shown to be particularly effective in a distributed sensing environment [178] A thorough description of the different types of PF may be found in [12]

However even the PF has some reliance on an internal model; the step which calculates the weights of each of the particles inevitably must use some form of model to establish whether the particles accurately describe the input data.

PFs are extremely difficult to set up for optimum performance as there are many settings to alter, including the number of particles, bearing measurement noise σ_v , process noise σ_w and the initial area in which to distribute particles. Sampling Importance Resampling (SIR) (or alternatively Sequential Importance Resampling) is a commonly used form of the particle filter.

The SIR algorithm, as with other particle filters is very processor intensive, with a 5000 particle processing of 250'000 time cuts taking almost 50 hours on a 2.0 GHz PC using Matlab, the same predictions made by an Extended Kalman Filter takes only a few seconds. This leaves the particle filter many orders of magnitude slower than the EKF, though is generally known to produce more accurate results.

Each particle in this implementation of the filter contains a state vector which is the same as the state vector in the EKF used previously $\hat{X}_T^L = [x \ y \ \dot{x} \ \dot{y}]^T$, and an associated weight ω_T^L , where $\sum_{l=0}^P \omega_T^L = 1$ and p is the total number of particles. The area in which to distribute the particles initially was set as an ellipse along the bearing line. The major axis of the ellipse was set to the length of the known maximum target range, and the minor axis was set to the width required to enclose the target, taking account of the known bearing error at the mean target range. The ellipse was centred at the mean target range, which is known a priori, on the first input bearing.

The update step is again similar to the EKF, however for the particle filter the step must be applied to every individual in the population. As with the EKF, the state transition matrix Φ is

$$\Phi = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where Δt is the time difference between the most recent measurement and the preceding measurement. Multiplying the state vector by the transition matrix gives the new state vector. However unlike the EKF, some random noise is added at this step. The random numbers added are chosen from a normal distribution, with zero mean, and

a standard deviation of σ_w for the position elements, and $\frac{\sigma_w}{2}$ for the speed.

Using this update step the particles are propagated forward to the time of the next bearing input.

The weights are then calculated. In order to do this the bearing from the sensor to the estimated target position must be calculated and compared to the measured bearing, $\text{bearing}_T^L = \tan^{-1}(\hat{x}_T^L, \hat{y}_T^L)$ where \hat{x}_T^L and \hat{y}_T^L are the x and y components of X_T^L . Weight ω is calculated from the difference between the bearing estimate and the measured

bearing at a particular time,
$$\omega_T^L = \frac{1}{\sqrt{2\pi\sigma_v^2}} e^{-\frac{\hat{\text{bearing}}_T^L - \text{bearing}_T^L}{2\sigma_v^2}} .$$

Once the weights have been normalised to one to create probabilities, the cumulative probability distribution (CDF) is calculated which can be used to perform biased roulette wheel selection of the particles in much the same way that roulette wheel selection is performed in genetic algorithms (see appendix E). From the N original particles, N new particles are chosen to be used at the next time step. The process is then repeated, starting from the update step.

2.3 3.3 AI approaches

Sensor fusion with known statistics relies on well known techniques such as the Kalman Filter or Bayesian statistics. Where there is no specific statistical model of the

uncertainty, other techniques such as rule based sensor fusion, fuzzy logic, and neural networks must be used instead.

2.3.3.3.1 *Artificial Neural Networks (ANNs)*

A back propagation (BP) ANN has been used to give navigational abilities comparable to the state-of-the-art [231], however multi layer networks require a notoriously long training time, and alternatives are available to optimise network size. Radial basis function networks (such as those using localised receptive fields (LRF) [204]) train much faster than BP nets because only one layer of weights needs to be modified

A problem with ANNs is that determining the appropriate number of hidden units can be more of an art than science [13] proposed a system of dynamic node creation (DNC) which starts with a small network and increases the size one node at a time until the network is large enough to handle the task in hand DNC was later applied to data fusion by Ghosh et al [98], who found that given a large number of nodes, backpropagation networks were prone to overtrain very easily, while a network created using a combination of LRF and DNC did not suffer from this problem, although output encoding networks were found to be the most effective network type overall.

Target state estimation has also been performed using neural networks For example the Neurally Inspired Contact Estimator (NICE) [75] is a neural network based target motion analysis (TMA) algorithm. The NICE algorithm has an equivalent accuracy to the Maximum Likelihood Estimator (MLE), but is an order of magnitude faster

More recently genetic algorithms (GAs) have been used to design ANNs for data fusion [1] used such a technique to develop a data fusion system for an electronic nose.

Neural networks have been used as time series predictors [93], the output of the network was the value of the prediction, with one output node for each step ahead being predicted.

The baseline Artificial Neural Network (ANN) used was a non-recursive, feed-forward ANN trained with vanilla backpropagation. The last N values in the time-series were given as input with one bearing input directly mapping to one input node, and the output of the ANN is trained to be the next bearing in the series. This is the typical set-up of most of the previous attempts to use an ANN as a tracking algorithm [104][43][55][68][241][270].

2.3.4 Classification

2.3.4.1 Artificial Neural Networks (ANNs)

Section 2.3.3.3.1 introduced the concept of using ANNs for bearing prediction, however they may also be used for target classification. A neural network is a massive system of parallel-distributed processing elements, connected in a graph topography. Data is not stored separately from the processing ANN as they are intrinsically linked. One of the most difficult problems in ANNs is choosing the most appropriate network topology for the problem. The choice will depend upon the problem characteristics, the characteristics of the likely approach to solving the problem, and the characteristics of the neural networks to be built. There are also several types of learning rules. These are biologically inspired, and govern how the network learns

In one of the earliest examples of using ANNs to fuse multi-sensor data for identification, [56] used Back-propagation and Hopfield neural networks to identify targets. In backpropagation, the data is supplied to the network, and the difference between the input and output is calculated. Weights are changed to improve the result. Once the errors have been minimised for all of the data in the training set, the system is ready to use for test data. Hopfield networks have feedback from output to input, giving a dynamic response. They can be unstable but stability can be ensured by forcing the weight matrix to be symmetric with zeros along its main diagonal. A recurrent network forms an associative memory. Therefore, like human memory, if a part of the memory is supplied, the network will return the full memory. The associative nature of ANNs was utilised to identify targets given a limited amount of information. In the simple examples given, the networks did not make a single mistake in identifying the targets, showing that it is possible to use ANNs to recognise and identify targets

Neural networks have since been shown [261] to be an extremely simple, easy to apply method and they outperform other fusion techniques at low correlation levels

2.4 JDL Level 2 – “situation assessment”

Situation assessment (SA) fuses the kinematic and temporal characteristics of the data to create a description of the situation in terms of indications of warnings, plans of action, and inferences about the distribution of forces and information. A SA algorithm will decide whether and in what way an object is or is likely to act in a hostile manner.

Unfortunately, most research is on the lower levels of fusion, and therefore this area is less well understood [118]

[187] used a series of algorithms for situation assessment. First the uniform k-centralised mean (UKCM) algorithm clustered the detected targets into groups. Once these clusters have been formed, it is possible to assess their intent using a fuzzy belief network. The simple rule-set of the fuzzy belief network, and the simple experimental scenario show that this kind of technique is capable of making situational assessments, though a more complex belief network would be required to tackle any real problem. This is also relevant in non-military contexts, such as context aware processing in which the task is to develop a machine that is able to understand and react appropriately to its environment. Wu et al. [296] looked at multi-sensor data fusion from an omnidirectional camera and a microphone to detect the focus of attention of attendees at a meeting. In this study Dempster-Shafer logic was used to combine the processed outputs of the sensors, such as the location of the meeting and who was talking. This was used to improve the estimate of each attendee's focus of attention compared to the output of the individual sensors.

2.5 JDL Level 3 – “threat assessment”

The third level of refinement assesses the threat posed by the enemy being tracked. This may also include an assessment of the friendly force's ability to engage the enemy effectively. Fusion levels two and three are often referred to as ‘information fusion’, while level one is ‘data fusion’. Although this distinction is vague, it is useful as the higher levels tend to utilise symbolic rather than numerical reasoning, and tend to be more subjective [283]. In human factors research this is often referred to as ‘Situational Awareness’ (SA)

Level three of the JDL model has received far less attention in the literature than any of the other levels. Initial papers are starting to appear on the subject, though at present they are as much about understanding the challenges of the problem as solving it.

Salerno et al [240] provided the starting point for a framework for information fusion for SA; it also gives an example situation in which automated situational awareness would be of benefit. The paper concludes with a discussion of metrics that could be used to validate SA techniques.

[145] looked at the problem of threat assessment using cognitive fusion techniques.

This breaks the problem down into three areas:

1. Situation awareness, understanding the meaning of multi sensor data, recognising complex time-dependent patterns and determining threats and other activities that reveal intent.
2. Decision awareness, reasoning about situations and understanding the ramifications of suggested actions.
3. Knowledge awareness, learning and improving skills for fusion procedures, and utilizing historic data to create new fusion patterns and situation classes

The combination of real time Event Correlation (EC) and Case Based Reasoning (CBR) is suggested to produce a generic framework to perform threat assessment. When EC recognises a series of correlated events, CBR can be used to identify the events as a case, where a case adds further meaning to the set of events and infers a possible situation [145] provided a basis for a possible system, but recommend further work must be done before any such system could be used in a real problem domain.

2.6 JDL Level 4 – “process assessment”

The process management stage is an ongoing assessment of the other fusion stages to ensure that the data acquisition and fusion is being performed in a way that will give optimal results This could also improve results, by adjusting the parameters in the fusion process, establishing a target priority [284] or moving the sensors to give improved coverage of the search area [299]. The problem of optimal sensor deployments is closely related to both the alarm placement problem, which is known to be NP complete and the Knapsack problem, which is known to be NP complete [143]. Penny [226][227] found a strategy for locating a submarine as quickly as possible using passive sonobuoy sensors (buoys fitted with a sonar) which was shown to reduce the detection times up to a factor of four. Hernandez et al [115] generalised these results to create a framework for the systematic management of multiple sensors in target tracking in the presence of clutter.

[215] and [239] gave a method for optimally distributing the sensors in time; the results show that if the target has a high probability of detection and a medium or high manoeuvring index, then time-staggered sensors (sensors with updates arriving in turn) should be used. In other circumstances there is little between staggered and synchronised (arriving at the same time) sensor updates If two sensors have drastically different performance, then optimal results are obtained by keeping them synchronised. If they have similar or identical performance then they should be staggered uniformly.

Multi Sensor Management (MSM) was discussed by [299], who argued that multi-sensor management affected all levels of the JDL model. They described MSM as a top down approach, which begins at level 4, but continues down right to level one as follows:

- Level 4 (mission planning)
 - Which service to perform?
 - Which accuracy level?
 - What area of the environment to focus on?
- Level 3 (resource deployment)
 - What extra sensors are required?
 - Where to place the new sensors?
- Level 2 (resource planning)
 - Sensor selection for multi sensor tracking
 - Sensor cueing, handing tracks from one sensor to another
- Level 1 (sensor scheduling)
 - Timeline of commands for each individual sensor

2.6.1

Distributed sensing

Process assessment has also been covered in the distributed sensing literature; here it is a matter of dynamically selecting which sensors to use in order to gain the most information in the most efficient way. The idea of using Shannon information theory for this was first proposed by [119]; selecting sensors based on expected information gain was first suggested by [194]. [286] more recently showed a technique for dynamically selecting the sensor to request data from in order to maximise the information gain. [286] used greedy selection of the next sensor; of all of the unused sensors, the one predicted to give the largest information gain is used.

[206] developed a system that made the most of limited resources on distributed nodes by designing a mobile code daemon. This daemon allowed a node to download the classifiers or trackers it required as it found that it needed them, at the same time clearing out the ones that were no longer required. This allowed the system to configure itself dynamically. [94] extended [206] to create a system in which nodes form themselves into clusters or coalitions. This avoids the ‘curse of dimensionality’ problem that is troublesome in very large systems. Without this, each node would be forced to share information with every other, meaning that the processing and

communications burden increases with each node added to the network, while in the proposed scheme, nodes only share information with those in the same coalition.

In another coalition forming technique, [256] discussed how to form dynamic coalitions of autonomous nodes. Dynamic coalitions are teams that form to perform a task, when a single node would not have sufficient resources to perform the task. Nodes learn how to form coalitions that are more productive. Experimental results show that these cooperative agents can track targets far better than trackers that simply react individually, and are able to share computational resources, allowing faster and more efficient processing.

[136] and [300] discussed methods of creating a hierarchy in which the nodes are divided up geographically into coalitions, and each coalition is given a team leader. In [136], each track detected is allocated a track leader by the team leader, and this node instructs the other nodes in the coalition. The technique is made to work using a conflict resolution strategy, which is required when nodes are given two conflicting tasks. [300] investigated how the number of levels in an architecture may affect performance, and found that as the number of levels in the hierarchy increases, the number of targets it is capable of tracking decreases. However the amount of time required by an individual node to complete its mission decreases exponentially.

[218] proposed an auction based technique called Dynamic Mediation (DM) for forming and allocating work to cooperating teams of nodes. In DM, the bid is not simply an individual value bid from a particular node, but a bid from a team of nodes, which can include information such as positive or negative interactions with other jobs allocated to the team. Experimental results from [218] suggest that DM shows the largest performance improvement over a traditional auction where time is limited.

[178] resolved the curse of dimensionality by separating the processes of allocating data points to targets being tracked and position estimation. Targets far away from each other are tracked separately in the traditional way, while targets close together are tracked jointly. As more than one target may be tracked at the same time, the PDF will not be Gaussian. This led [178] to use a particle filter as a position estimation algorithm as it can estimate arbitrary distributions.

Akyıldız et al have written a comprehensive survey [5] on the subject from a networking perspective. This gives a description of the major network topologies and protocols available, and concludes that there remain many unsolved problems in sensor network research such as fault tolerance, scalability, node cost, and power

consumption. In another survey paper [6], the same authors outline the major applications for sensor networks, citing examples such as:

- Military applications, such as monitoring friendly forces and battle damage assessment
- Environmental applications, such as bird migration monitoring, or flood detection
- Health applications, such as tracking doctors within a hospital, or remotely monitoring patients' physiological data
- Home automation

2.7 Evaluating state-of-the-art approaches to target tracking

2.7.1

Summary

The rest of this chapter measures the performance of three of the commonly used bearing prediction algorithms already described. These three baseline techniques were chosen to reflect the state-of-the-art at the time of writing. Firstly the Particle Filter was chosen as it is recognised as the most accurate prediction algorithm on data from non-linear systems [310][137][79][45]. Secondly the EKF was chosen as it is the most widely used prediction algorithm [82][266][113]. [249] stated that in this area most applications are based on either the EKF or the PF. Finally the type of ANN which represents the state-of-the-art passive sonar target tracking was selected [104][43][55][68][241][270], though as with many areas of ANN research little has been written on this topic for a number of years. Data sets of varying degrees of complexity are constructed to test the algorithms in increasingly complex scenarios. Results show that all algorithms give roughly equivalent results on all data sets.

Three baseline techniques are outlined here;

- The Extended Kalman Filter (EKF) as it is the *de facto* algorithm in target tracking
- The Particle Filter is a newer technique which has proven to be more effective than the EKF in a range of applications
- A Neural Network with a single output node to output trained to predict the next bearing – this is the most widely adopted machine learning technique but has been found to be comparable to the other techniques at best.

The Extended Kalman Filter is shown to be the most accurate in the simple scenarios, while the Particle Filter is shown to be the most accurate in the most complex scenarios. The performance of the ANN is not considerably worse, but is worse than at least one of the other two in almost all of the scenarios

2.7.2

Extended Kalman Filter (EKF)

2.7.2.1 Kalman filter tuning

To ensure that the comparison with the baseline was fair, the Kalman Filter was carefully tuned to give best performance. This tuning can be divided into two broad categories; initialisation and running parameters. The methodology used to tune the filter is outlined in sections 2.7.2.2 and 2.7.2.3, while the results are given in Appendix D

2.7.2.2 Initialisation

When the first bearing arrives in the filter, all that it is possible to deduce about the target position is that it is along the bearing line. Prior knowledge about the data set, or the sonar performance also gives the minimum and maximum possible ranges. For the data sets used here, the target position was therefore set to be along the first bearing, at the median start range of targets. The speed components were both initialised to zero, which is the median speed in each direction for the data. The covariance matrix was all zeroes, except for the lead diagonal. These non-zero values represent the uncertainties in the x,y position and the x,y speed, and are set to the maximum target range and $\frac{s^2}{3}$ respectively, where s is the standard deviation of the target speeds in the data and was set to 10.0. The values reflected the information known a priori from the data.

2.7.2.3 Running parameters

The data was run through the Kalman Filter several times, and a number of different settings for the parameters were tried.

The standard deviation is a scalar value used in the filter's model, and the known standard deviation of the data was used at the start of the optimisation (in a real system this would generally be a known constant for the sonar). After a number of iterations, the optimal value for each of the filter's variables were found, as described in Appendix D.

2.7.3

Particle filter

As previously stated there are a number of different parameters to tune in order to ensure high performance in the Particle Filter. The bearing measurement noise σ_v was set to the known bearing measurement noise for each dataset. A series of experiments was performed to empirically calculate the other two parameters; number of particles and process noise σ_w . The results of these optimisations can be seen in Appendix D.

2.7.4

Single output ANN

The baseline Artificial Neural Network (ANN) used was a non-recursive, feed-forward ANN trained with vanilla backpropagation. The last N values in the time-series were given as input with one bearing input directly mapping to one input node, and the output of the ANN is trained to be the next bearing in the series. This is the typical set-up of most of the previous attempts to use an ANN as a tracking algorithm [104][43][55][68][241][270]

The ANNs were implemented with SNNS [320] in the form of input text files which store the structure and initial state of the ANN. The training algorithm implementation used was the Std_backpropagation training algorithm in SNNS which implements a vanilla backpropagation training algorithm. To automate the process of establishing the optimal structure for the ANN, a simple Java program was created with nested loops which could optimize a series of parameters; number of input nodes, number of hidden nodes and learning algorithm. The number of input nodes maps 1:1 to the number of previous values to read when predicting the next value in the series.

Results of this optimisation can be found in appendix D.

2.7.5

Experimental design

Three baseline techniques were chosen to reflect the state-of-the-art at the time of writing. Firstly the Particle Filter was chosen as it is recognised as the most accurate prediction algorithm on data from non-linear systems [310][137][79][45]. Secondly the EKF was chosen as it is the most widely used prediction algorithm [82][266][113]. [249] stated that in this area most applications are based on either the EKF or the PF. Finally the type of ANN which represents the state-of-the-art passive sonar target tracking was selected [104][43][55][68][241][270].

In order to test the performance of the prediction algorithms, a procedure was developed outlined in the rest of this section. Firstly a number of data sets of varying difficulties were developed, as described in section 2.7.6.

A flow chart detailing the experimental procedure can be seen as figure 6. All algorithms were set to a known state. For the EKF and PF this was achieved by setting parameters to known values whereas for the ANN this was by training with standard backpropagation. Each algorithm was fed a pre-specified number of inputs, and the ability of the algorithm to predict the next value in the series was measured. Between each run of the predictors the algorithm was returned to its initial state. For the EKF and PF this meant resetting the parameters, whereas the ANN carried no information between predictions and therefore no action was necessary. As each data set contained several thousand examples each algorithm had to be run thousands of times, resetting in between each run.

As the ANN required training on part of the data and testing on another, ten fold cross validation was used to ensure fairness of comparison. The ANN was run 10 times, each time using a different non-overlapping section of the data as the test set, with the rest of the data being used for training and validation, see table 1. The mean value across each of the ten runs was taken as the overall result of the experiment.

Run	Testing data	Training data
1	1	All others
2	2	All others
3	3	All others
4	4	All others
5	5	All others
6	6	All others
7	7	All others
8	8	All others
9	9	All others
10	10	All others

Table 1 The tenth of the data set used in each run of the ANN

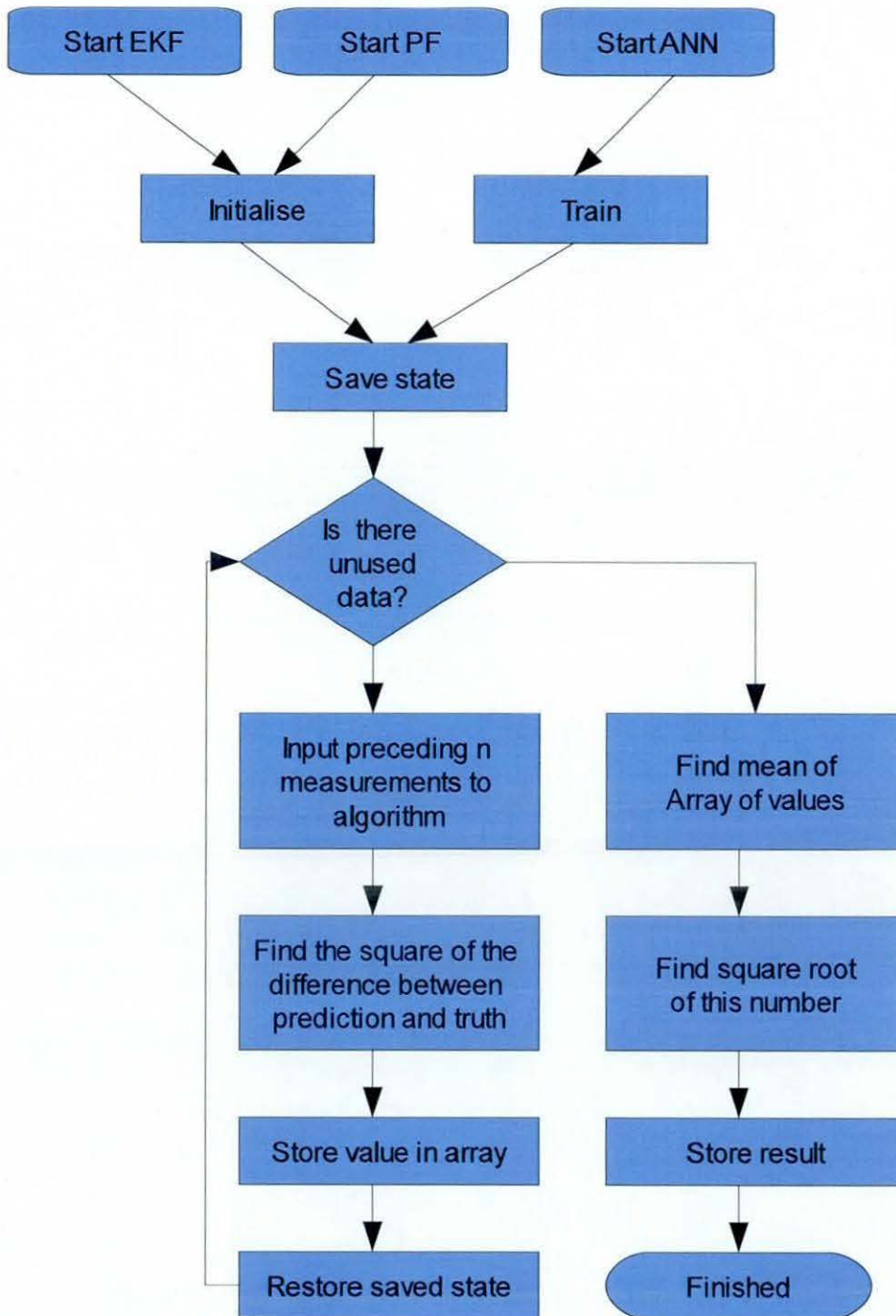


Figure 6 Flow chart of experimental plan

2.7.5.1 Accuracy measurement

Four accuracy measures are described in appendix E, RMS error, GMRAE, MdRAE and MdAPE. [10] recommended that the first of these should be avoided and the later three preferred as they are relative measures, while the first is absolute. The reason for

relative errors being preferred is because if, to take a financial example, one share loses value from £3.50 to £3.00, while another loses value from £350 to £300, then the real loss on each can be considered to be the same. This argument does not hold with sonar bearings where a change of 3.5° to 3.0° is much smaller than a change from 350° to 300°, an equivalent change would be 350° to 349.5°. Although these relative measures are questionable in this context, they are provided in some sections of this thesis for completeness.

2.7.6

Data

The datasets created for the experiments can be split into two broad categories. Firstly, smaller purely synthetic data sets with 500 points each, designed to test the abilities of the algorithms on sets of various levels of difficulty. Secondly much longer datasets comprising synthetic data overlaid on recorded background noise intended to test the algorithms' abilities on more realistic data.

2.7.6.1 Fully synthetic

Four smaller datasets were created, each of varying levels of difficulty to the tracker, ranging from a simple sine wave, through to a more complex pattern with large amounts of Gaussian noise added. These data sets are given in sections 2.7.6.1 to 2.7.6.1.4. The values are first shown in their original form to show the pattern, and then shown wrapped to be between -180° and 180°, to prevent for example the equivalent predictions of 182° and -178° from being seen as 360° away from each other. These small data sets were created in order to thoroughly investigate the capabilities of each of the algorithms under test. As the sets are short it is possible to plot the results of the algorithms and visualise comparative performance.

2.7.6.1.1 Set one

2.7.6.1.1.1 Summary

This is a deliberately simple set based upon a sine wave, designed to show nothing more than that all algorithms can follow the true values. No noise is added to the measurements.

2.7.6.1.1.2 Definition

$$y = 360 \sin\left(\frac{x}{50}\right)$$

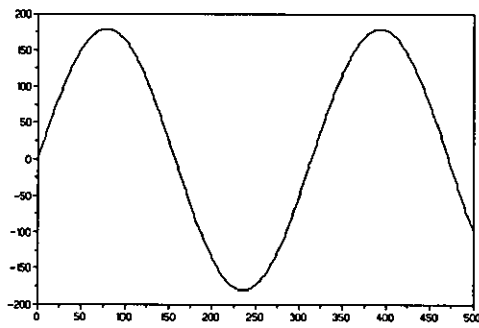


Figure 7 Small data set 1

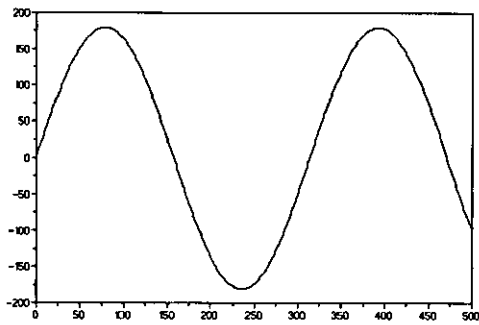


Figure 8 Small data set 1 restricted between -180° and $+180^\circ$

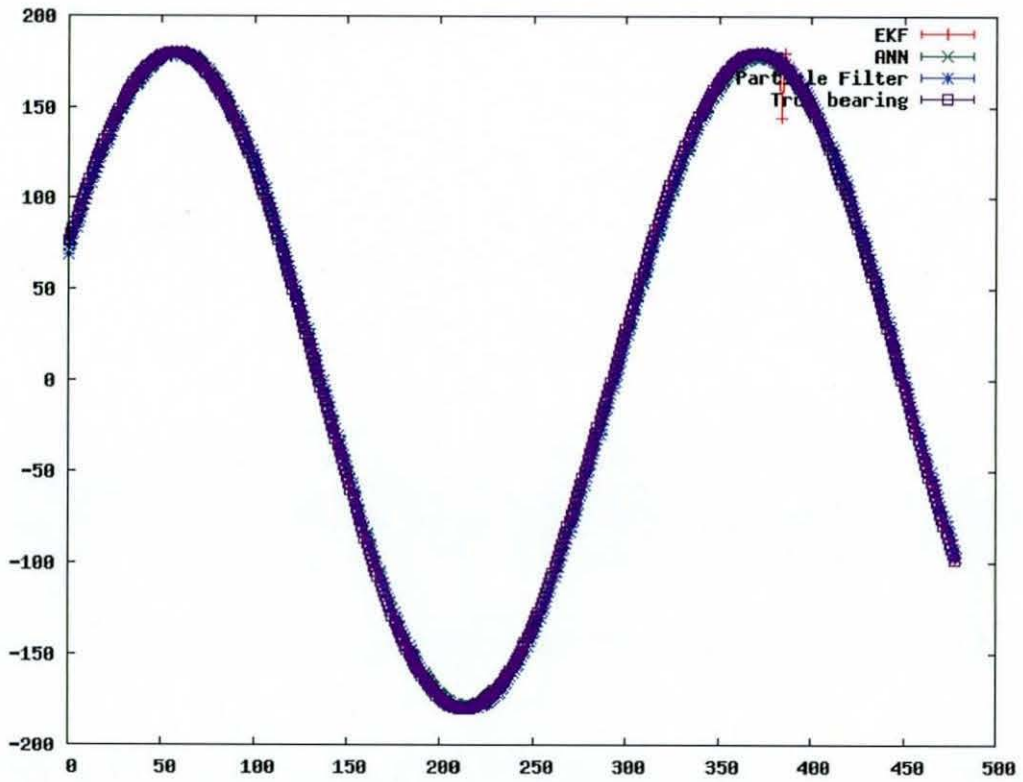


Figure 9 Output of the target tracking algorithms on simple data set 1

- All three trackers are shown to be effective on this extremely simple data set, with the exception of the large outlier on the EKF.

Figure 10 is a residual plot and shows the difference between the true bearing and the prediction made by each algorithm.

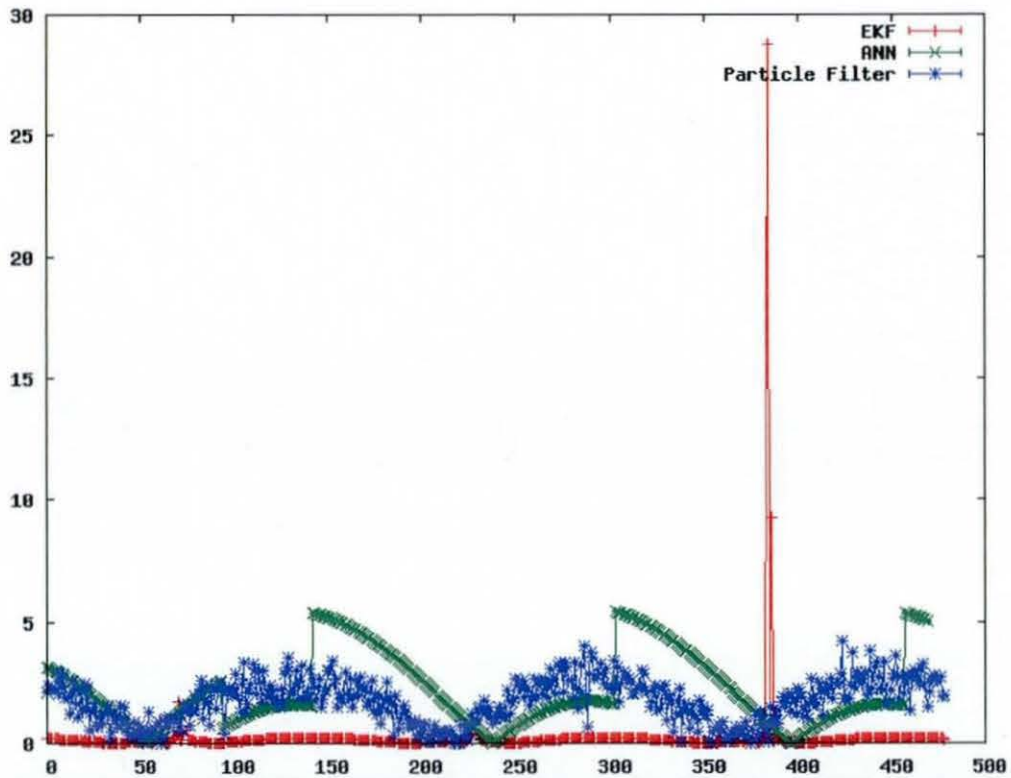


Figure 10 Bearing residuals of the target tracking algorithms on simple data set 1

- Figure 10 displays the results more clearly than figure 9
- On average the EKF is the most accurate predictor.
- The EKF is the most accurate of the predictors with the exception of two outlier points, one relatively small and only a little worse than the PF, while the other is very large.

At this point it is worth clarifying a point about how the algorithms are run. The EKF is known on occasion to diverge; it becomes overly confident, reducing the size of the error margins stored in the covariance to near zero. If the update is performed in this state, the filter interprets even a small difference between the prediction and the measurement as overly significant and overcompensates for it. This sends the estimate wildly wrong. From this point forward the filter gives wildly inaccurate predictions. In order to counter this here the filter is run on batches, rather than being run recursively. In order to predict each point the filter is given the previous N points. The two outliers represent times at which the EKF has diverged. Had the EKF been run in the normal manner its performance would have degraded for the rest of the run.

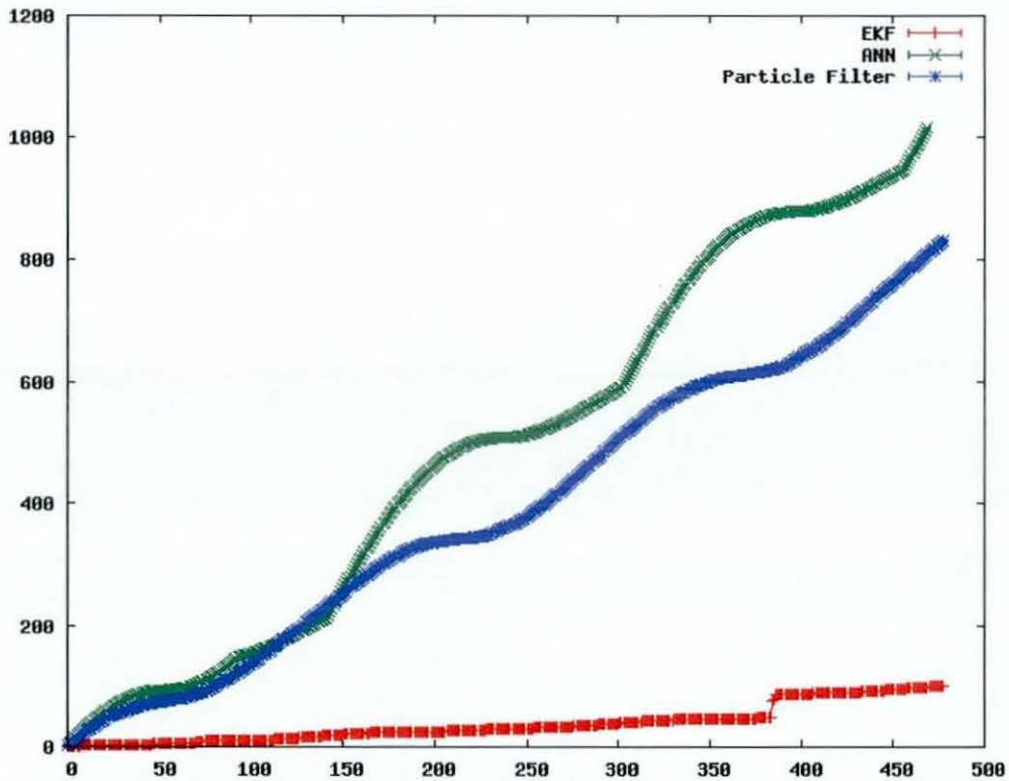


Figure 11 Bearing residuals of the target tracking algorithms on simple data set 1

This plot most clearly shows the relative performances of the algorithms;

- The EKF gives the lowest overall error.
- The PF and ANN have similar levels of performance.
- The ANN is the least accurate overall.

2.7.6.1.2 Set two

2.7.6.1.2.1 Summary

The second data set was constructed which was only a little more complicated. The intention was to test how the algorithms coped when given data whose gradient changes rapidly. This represents the by product of measuring data from non-linear systems such as passive target tracking.

2.7.6.1.2.2 Definition

$$y = 360 \left(\sin\left(\frac{x}{50}\right) + \cos\left(\frac{x}{20}\right) \right)$$

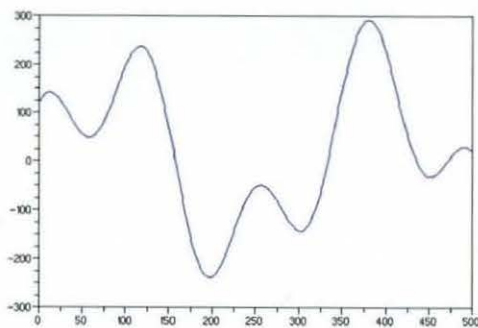


Figure 12 Small data set 2

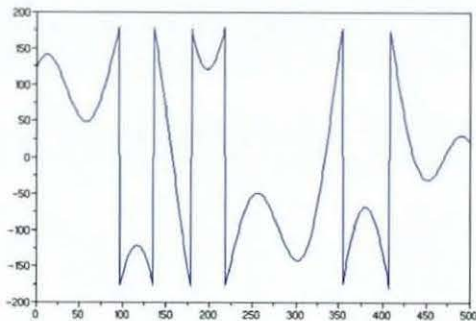


Figure 13 Small data set 2 restricted between -180° and $+180^\circ$

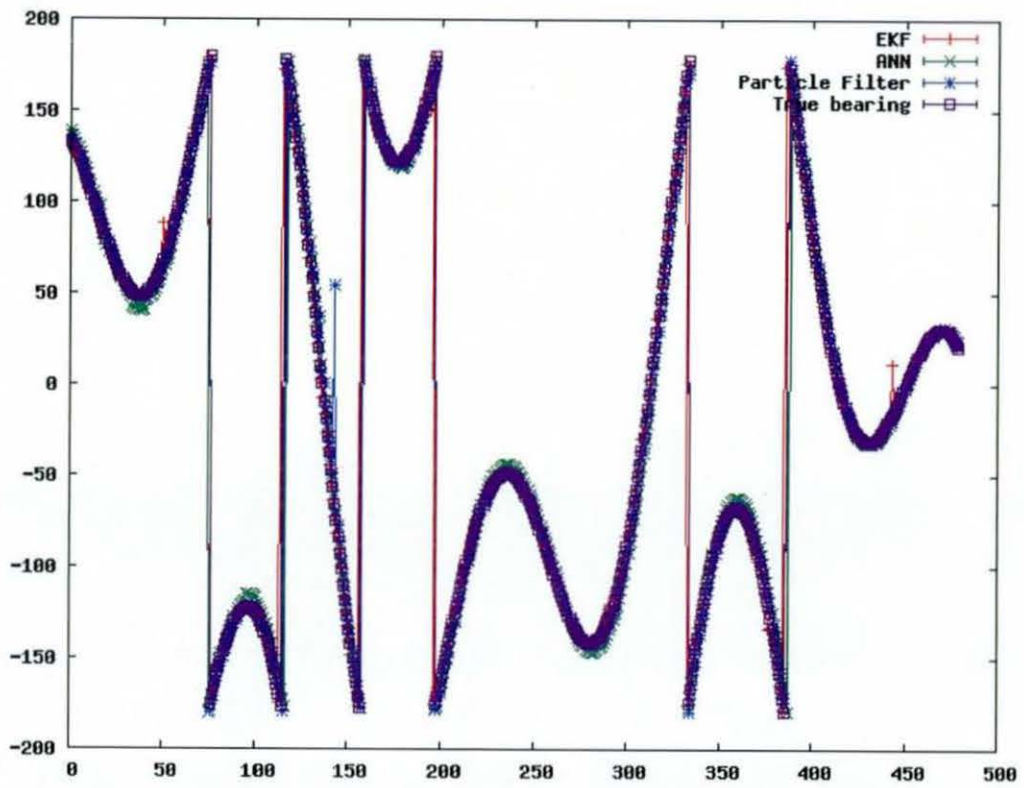


Figure 14 Output of the target tracking algorithms on simple data set 2

- Again, all filters can be seen to track the target well overall.
- Here there are outliers for both the EKF and PF.

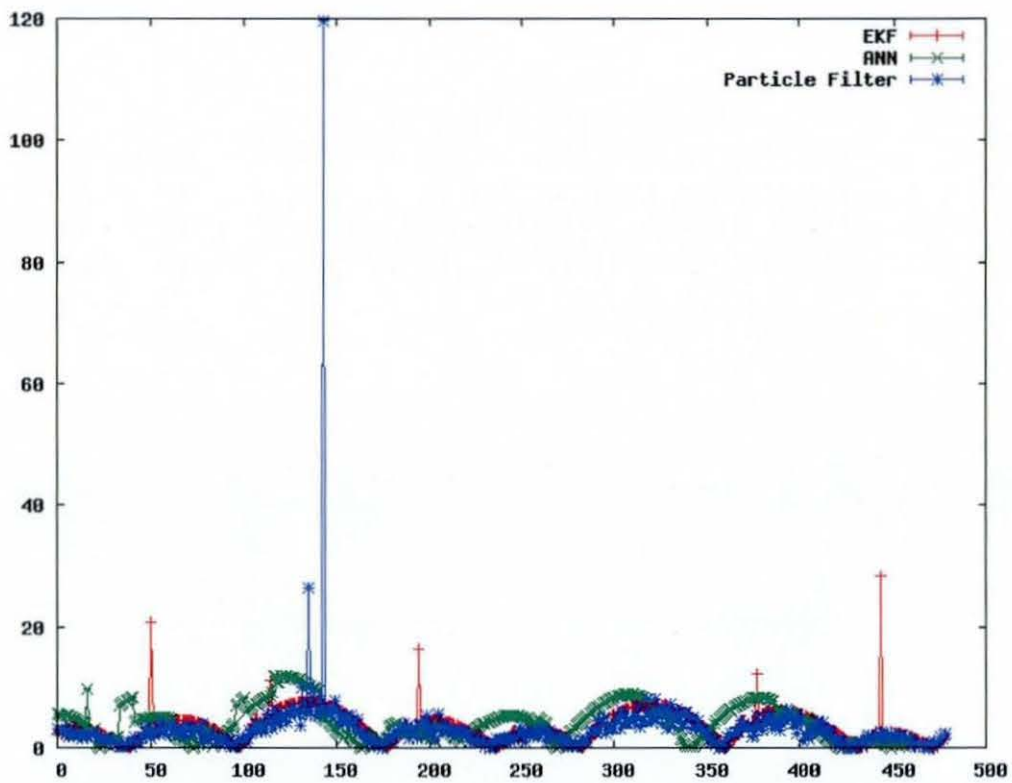


Figure 15 Bearing residuals of the target tracking algorithms on simple data set 2

- Again the residual plot shows the relative performances more clearly:
 - Here it is possible to see that the largest single outlier is the PF
 - The EKF gives more frequent, smaller outliers
 - Ignoring the outliers, the performance of the PF and EKF is nearly identical.
 - The performance of the particle filter does not seem to have been reduced by the increase in complexity, though the performance of the EKF has reduced considerably.

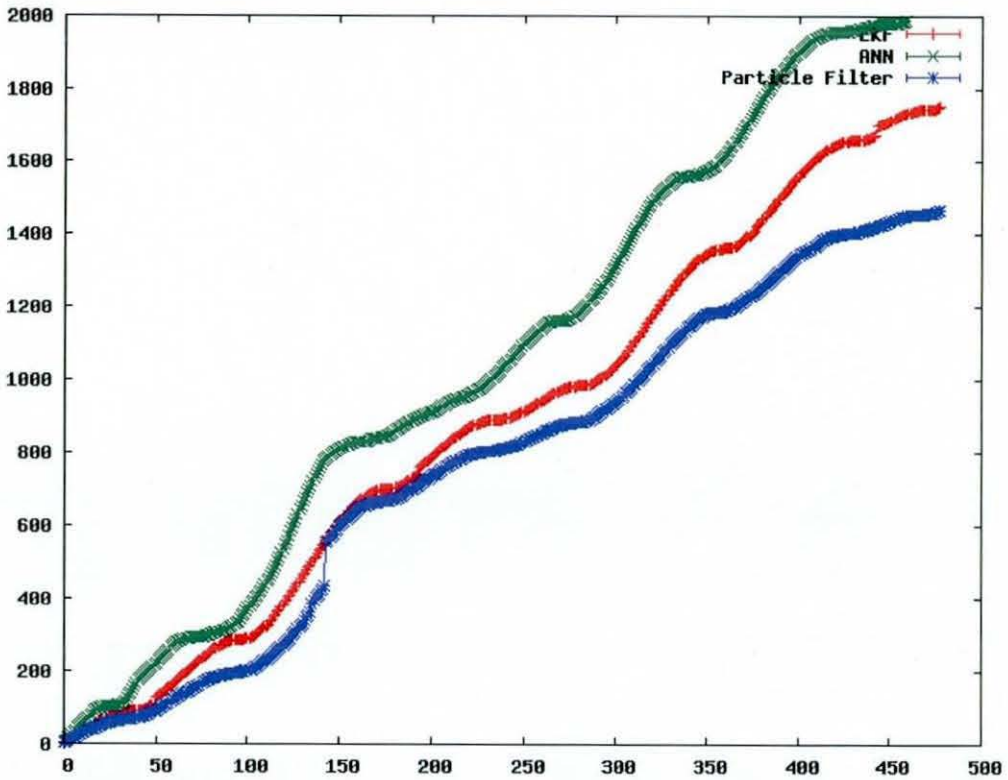


Figure 16 Cumulative bearing residuals of the target tracking algorithms on simple data set 2

- Here the performance of all of the algorithms is far closer than in figure 11.
- For this slightly more complicated data the PF is now the most accurate.
- The ANN is again the least accurate.

2.7.6.1.3 Set three

2.7.6.1.3.1 Summary

In the third data set a small amount of Gaussian noise was added to the measurements. The data given here is therefore far closer to real sonar data than the previous data sets.

2.7.6.1.3.2 Definition

$$y = 360 \left(\sin\left(\frac{x}{a}\right) + \cos\left(\frac{x}{b}\right) \right) + rand_{norm} 6.7$$

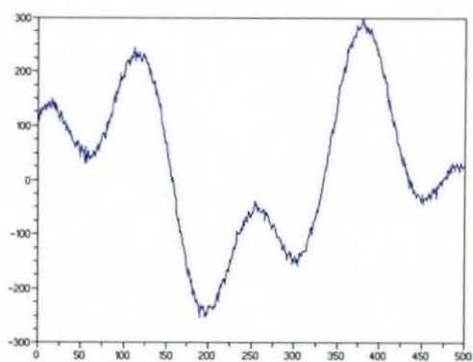


Figure 17 Small data set 3

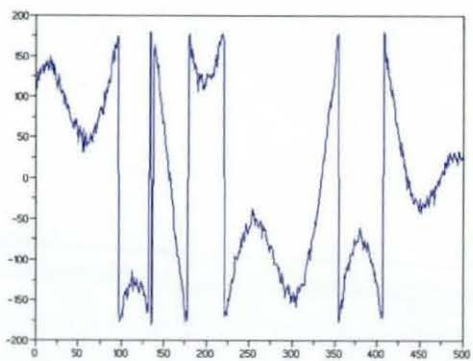


Figure 18 Small data set 3 restricted between -180° and $+180^\circ$

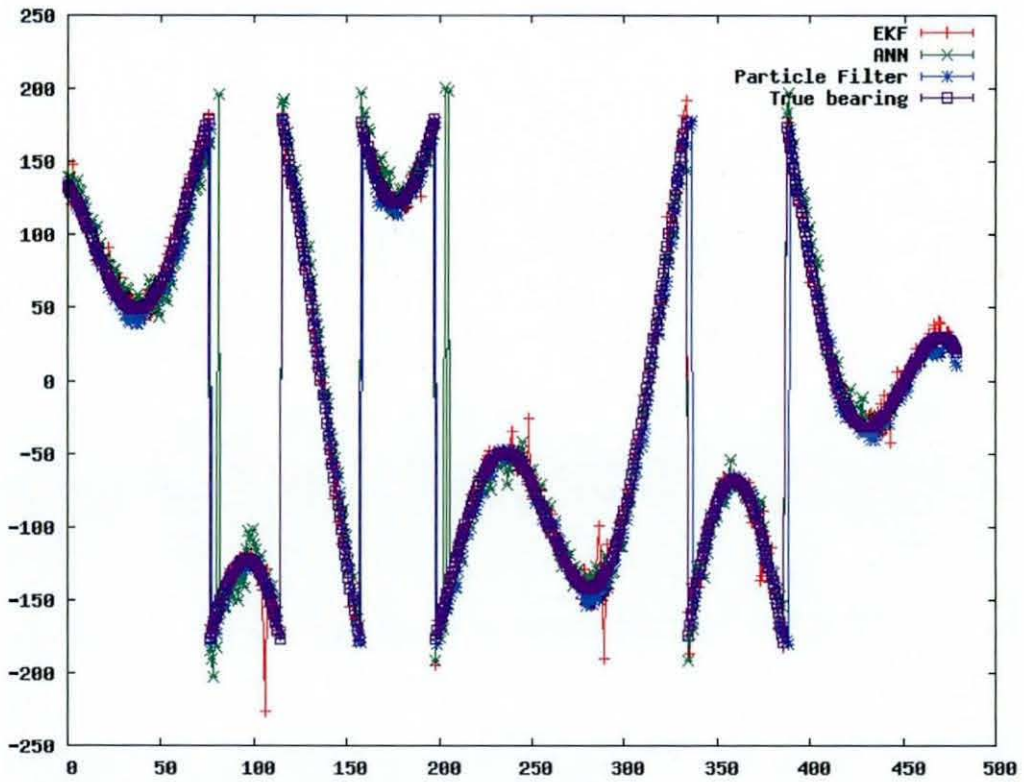
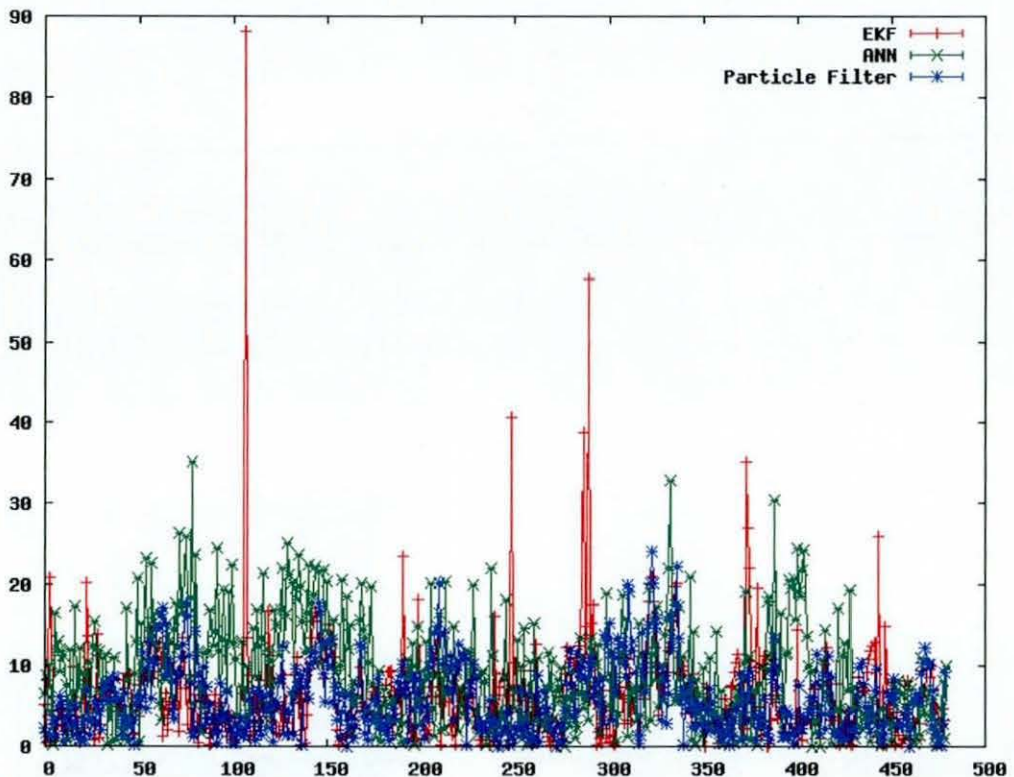


Figure 19 Output of the target tracking algorithms on simple data set 3

- The data has started to become complex enough to make the predictors less accurate, this can be seen by the fact that the lines on the chart are 'fuzzier' as predictions are becoming less accurate, this has resulted in enough differentiation in performance to be able to see each line.
- The outliers on the EKF are becoming larger and more frequent.



- From the residuals plot the EKF's frequent large outliers are clear to see.
- For the first time the PF is outperforming the KF.

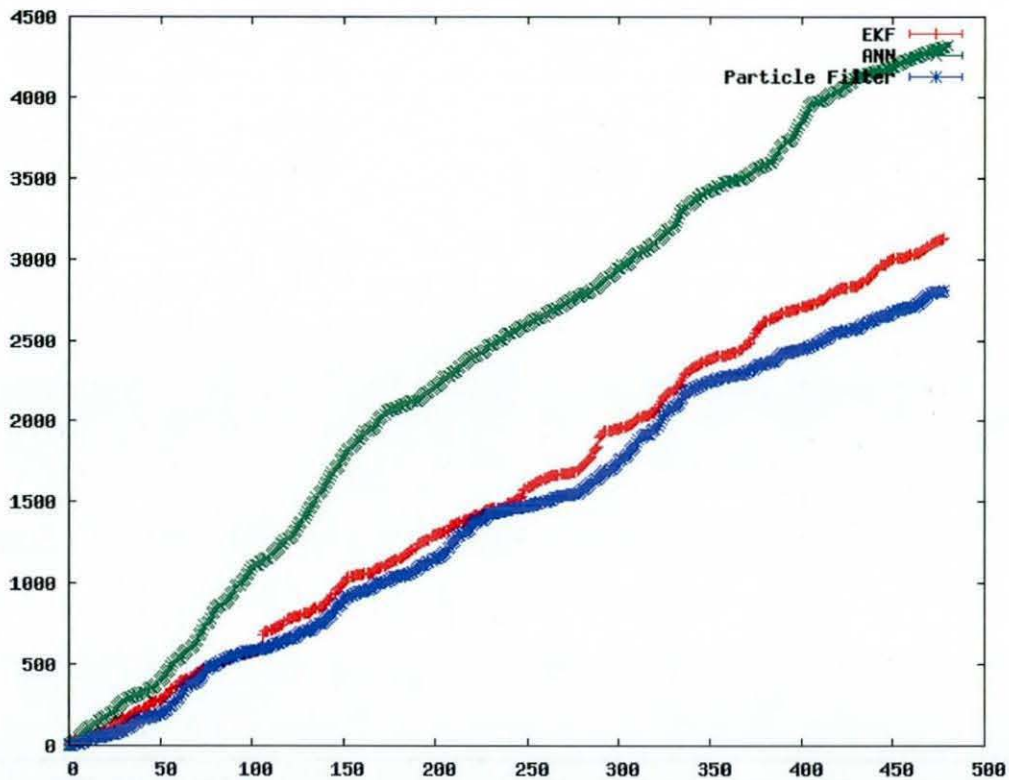


Figure 21 Cumulative bearing residuals of the target tracking algorithms on simple data set 3

- The performance of the EKF and PF are virtually identical
- The ANN is yet again far less accurate than the other two algorithms.

2.7.6.1.4 Set four

2.7.6.1.4.1 Summary

The fourth data set increased the level of noise to match the upper limit of the bearing binning function. With the parameters used here for the binning function, this noise level represents the limit of its performance. Higher levels of noise would cause a degradation in performance relative to the other predictors.

This is designed to be the hardest test for the algorithms, where the bearing changes being predicted are smaller than the noise.

2.7.6.1.4.2 Definition

$$y = 360 \left(\sin\left(\frac{x}{a}\right) + \cos\left(\frac{x}{b}\right) \right) + rand_{norm} 13.3$$

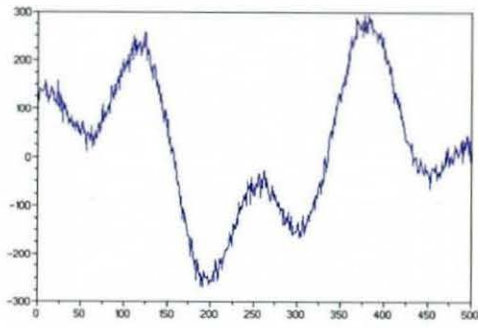


Figure 22 Small data set 4

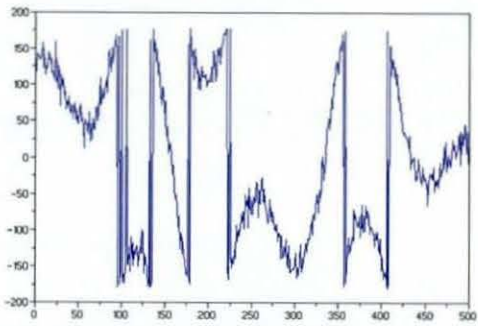


Figure 23 Small data set 4 restricted between -180° and $+180^\circ$

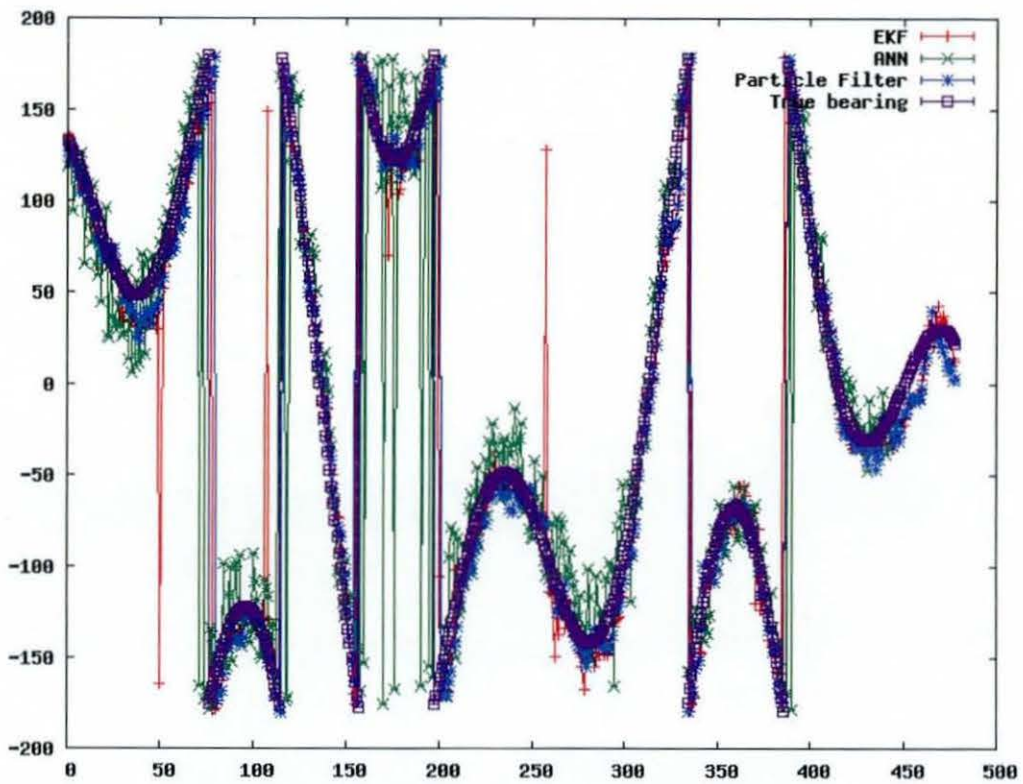


Figure 24 Output of the target tracking algorithms on simple data set 4

- The performance of all of the predictors is shown to have deteriorated significantly. The performance of all of the predictors appears poor.
- The EKF outliers are now very large, and easily noticeable even on the bearing plot.

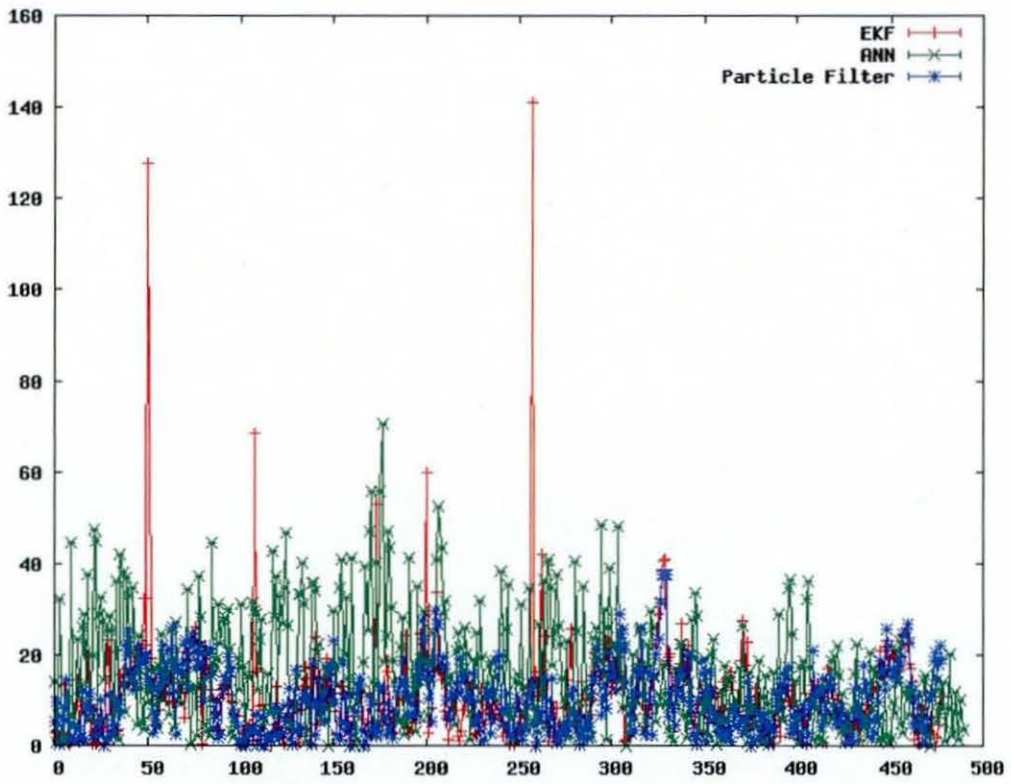


Figure 25 Bearing residuals of the target tracking algorithms on simple data set 4

- The enormous outliers on the EKF are even clearer on the residuals plot.
- Excluding the outliers, the performance of the EKF and the PF is very similar.

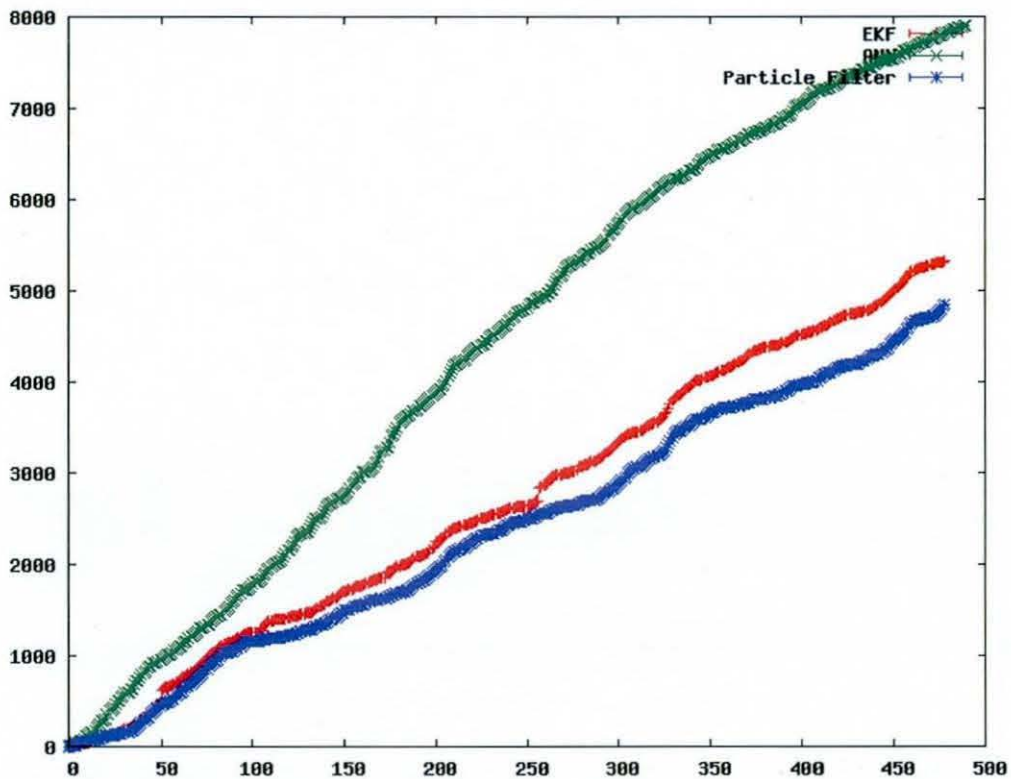


Figure 26 Cumulative bearing residuals of the target tracking algorithms on simple data set 4

- The cumulative plot is the clearest way to see the relative performance of the different algorithms; as in figure 21, the performance of the EKF and the PF are very similar, while the performance of the ANN is far less accurate.

2.7.6.2 Semi-synthetic

The second, considerably larger data set was created to be far more realistic. Ideally this would have consisted of real sonar recordings. However it was found to be difficult to obtain large enough volumes of data for which exactly one target was being detected and the true bearing to the target was known. This is especially difficult for the quantity required for completely training and testing a learning algorithm. By its nature it is considered secret, and even references that state that this data is difficult to obtain are impossible to find.

This necessitated the construction of simulated data. Here it was decided to make the time series as close to real data as possible by using a very low level sonar simulator, to generate target detections, overlaying this over recorded background noise.

[230] found that most sonar simulators for research purposes are not created to be a realistic enough simulation to replace real recorded data and tend to specialise into one

of the following roles; supporting research, assisting sonar acceptance testing or operator training. [230] described a new simulator called Sonar Data Generator (SDG) so detailed it could be used for all three, and was created with the goal of augmenting recorded data with high quality simulation. To ensure maximum realism in the semi-synthetic data set was created using the SDG described in [230].

This produces the closest simulation possible to real recordings, and having the advantage over real sonar recordings that the exact truth is known for every scenario. These were then run through a simple example of a sonar processing system to create a time series.

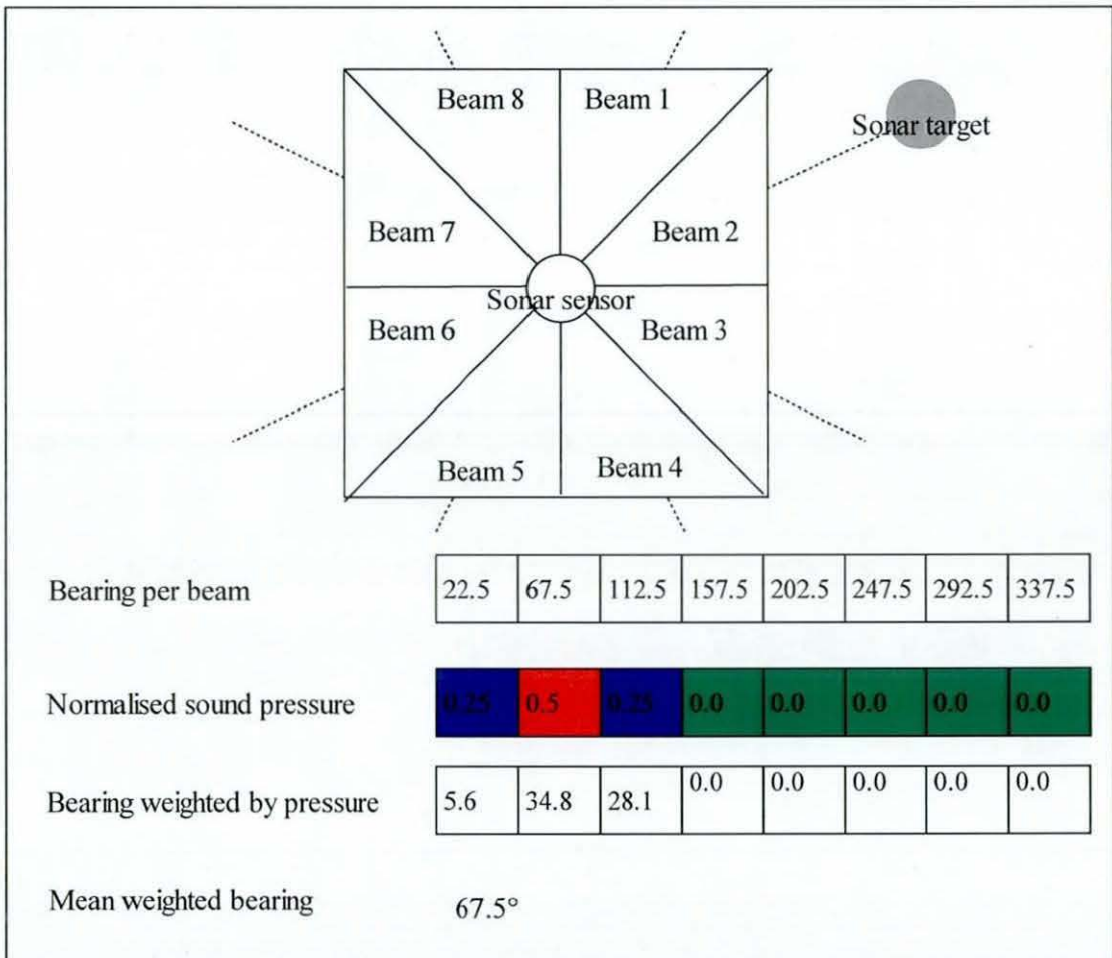


Figure 27 A simple example of processing beam data for a sonar with eight beams for a single instant

The data were constructed using a low-level passive sonar simulator which produced beam-level sonar data. This is a vector of sound intensities for a range of bearings, a simple example of which is shown in Figure 27. Here each beam represents a particular direction, which mapped to a bearing, and gave an energy level of the sound

received from that direction. As with real passive sonars the width of the beams was not uniform and altered from very narrow beams at broadside (around 90°) and widened to create more coarse bearing estimates at endfire (0° and 180° , the lower and upper bearing limits of the sonar). Another common feature of passive sonar that this simulated data shared is the inability to distinguish between port and starboard. All data, no matter the true direction of the target, had a range of between 0° and 180° . The ability to resolve the ambiguity between port and starboard was not a required feature of this tracker, so the 'true' bearings have also been normalised to between 0° and 180° . An example of this can be found in Figure 28 and Figure 29.

This data was then overlaid on looped background noise recordings from a real sonar, seen in Figure 30 and Figure 31. A form of thresholding was used to convert the beam data to a time-series. Each track was started on the nearest peak to the true bearing, and at each time step was continued with the nearest peak to the previous track bearing, this track can be seen as the black line in Figure 31.

As the target being tracked was simulated, the exact true bearing was known. The simulator was used to create 5000 independent data sets, each with 100 data points. The scenarios used consisted of a static sensor detecting a moving target with random start position within a pre-set range of the sensor, and random target kinematics. Only one target was simulated for each data set.

The output at this stage was a vector of intensities at each point in time. A simple peak detector was used to find a single bearing to use at each point. At this point the data had been converted to 5000 independent time series, each with 100 bearing values. A subset of this data is given in appendices B and C.

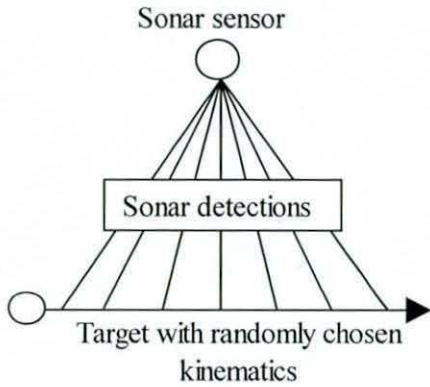


Figure 28 Plan view of simulated target over time

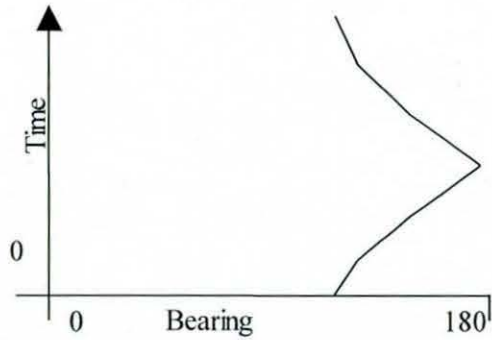


Figure 29 Time/bearing plot of sonar measurements

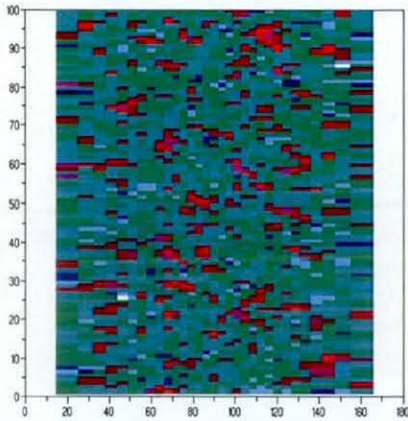


Figure 30 Plot of recorded noise, shown with time over bearing

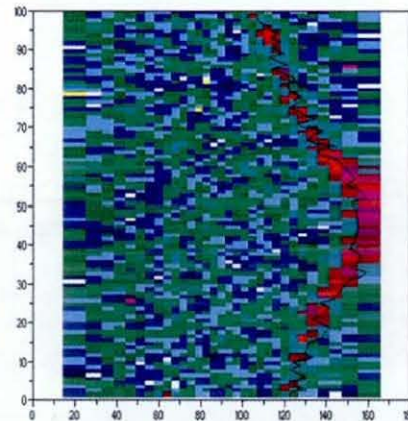


Figure 31 Plot of simulated data with recorded noise, overlaid with threshold track, shown with time over bearing

2.7.6.2.1 Conversion to time-series

The resultant time series produced for both the small and large data sets were then subdivided using a sliding window into a series of pattern recognition problems, a simple example of which is given in Figure 32. Applying the sliding window with between 1 and 50 inputs (the input number corresponding to the number of input nodes in the network being tested), resulted in 250'000 data sets (of the sort given in Appendix B) for each number of input nodes.

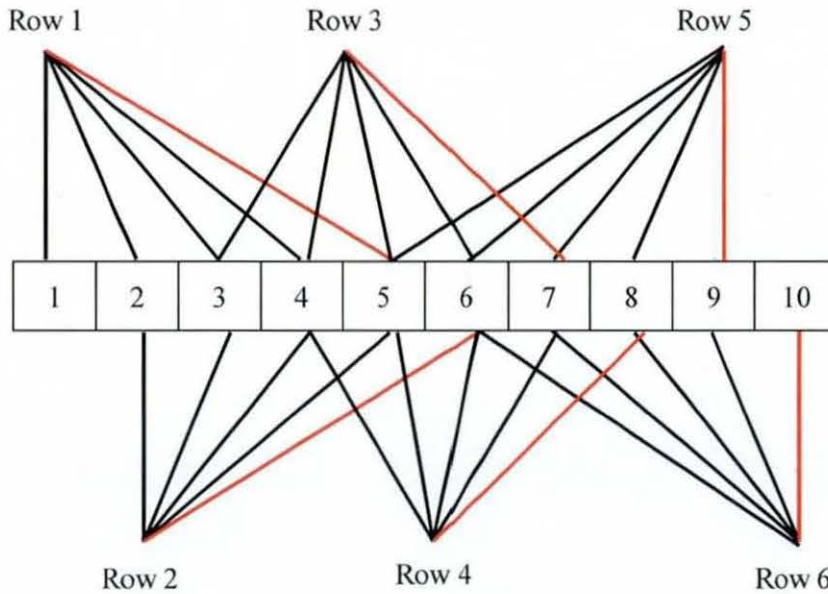


Figure 32 Sliding window data conversion, black lines represent input data, while the orange denotes the value to be predicted, converting a continuous time series into discrete samples.

All inputs and outputs were normalised so that the last input is always zero. Unless this normalisation had been performed, the network would see two identical input sets for which one of the sets was rotated as a completely different set of inputs. This normalisation reduces the amount of training required for the tracker to be accurate in a wide range of situations.

These examples were then collected into a single data table, which was then divided into tenths to perform cross-validation. This is so that the parts of the data used for training, validation and testing could be rotated for each experiment, ensuring that the data is properly tested against all input data, and data favourable to the algorithm is not inadvertently selected for testing.

2.7.6.2.2 Results

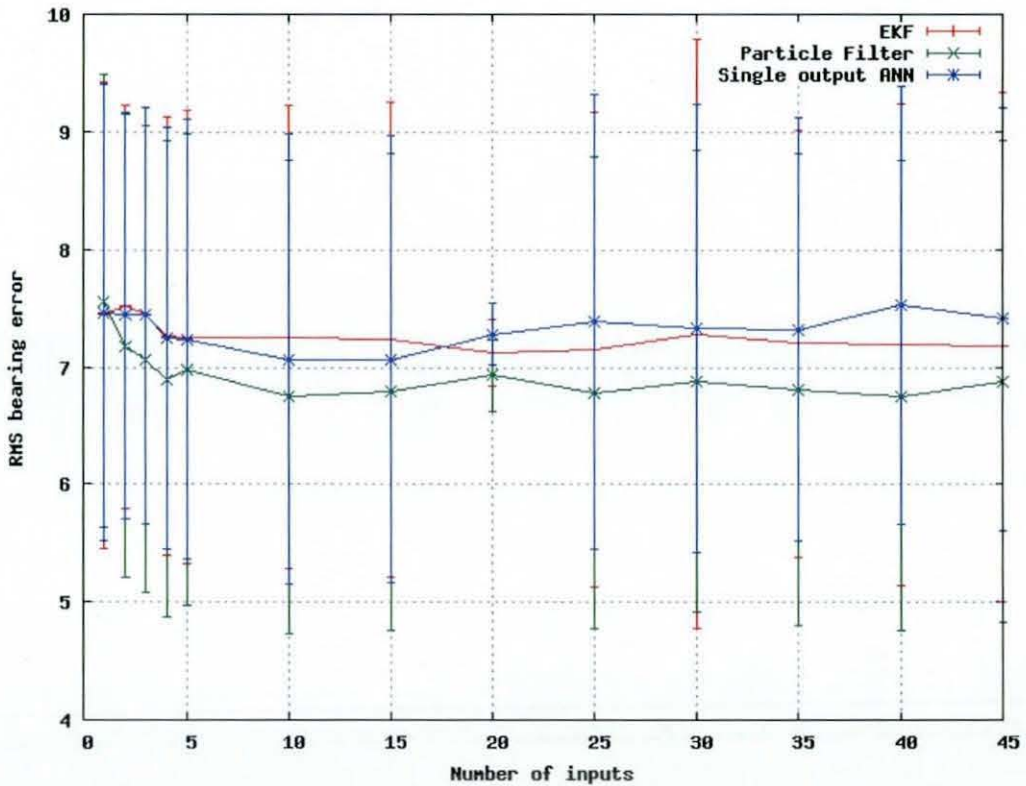


Figure 33 RMS bearing residuals for single output ANN, EKF and PF when run on the semi-synthetic data set with number of inputs varying from 1 to 45

- All algorithms are run in batches, with a set number of inputs to give one output.
- The PF is clearly the most accurate on this, the largest data set.
- The performance of the ANN is the poorest overall.
- The EKF is only more accurate than the ANN by a narrow margin.
- There is very little difference in performance between all three on this, the most realistic of the five data sets.

2.7.6.2.3 *Statistical comparison of algorithms*

Before judgements were made about which was the best algorithm it was important to establish a procedure for giving a confidence figure to the statement that one algorithm was more accurate than another. In order to achieve this the data was randomly divided into ten equally sized sets. Each of these sets would allow an independent test of the performance of the algorithms. For each of these probabilistic comparisons the null hypothesis is that the two algorithms being compared are actually equal in performance, and the probability that they could achieve this hypothesis is calculated. Table 2 gives the confidence that algorithm A is more accurate than algorithm B given the number of folds that algorithm A was more accurate than B out of a total number of ten independent runs or folds. This is calculated by working out the number of combinations of results which are equal to or better than the result obtained (such as eight out of ten folds), and then dividing this by the total number of combinations;

$$p_n = \frac{\sum_{i=n}^{10} \binom{J'}{i' \times (J' - i')}}{2^{J'}} \quad \text{where } j \text{ is the total number of runs or folds. Inside this}$$

formula $\frac{J'}{i' \times (J' - i')}$ calculates the number of possible combinations of i out of J being better than the other, while $2^{J'}$ gives the total number of combinations for j runs. Table 2 shows that in order to obtain a 90% confidence that algorithm A is more accurate than algorithm B it must outperform A in eight out of ten tests. For this thesis a target of 90% confidence or eight out of ten tests will be used as the target value for proving that an algorithm outperforms another.

Number of experiments in which a was better than b	Number of combinations of exactly this eventuality	Number of combinations of this or better	Odds for (assuming a and b are equal)	Odds for (assuming a and b are equal)
10	1	1	0.10%	99.90%
9	10	11	1.07%	98.93%
8	45	56	5.47%	94.53%
7	120	176	17.19%	82.81%
6	210	386	37.70%	62.30%
5	252	638	62.30%	37.70%
4	210	848	82.81%	17.19%
3	120	968	94.53%	5.47%
2	45	1013	98.93%	1.07%
1	10	1023	99.90%	0.10%
0	1	1024	100.00%	0.00%

Table 2 The confidence that algorithm a is more accurate than algorithm b given the number of folds that algorithm a was more accurate than b out of a total number of ten folds

Tables 3 to 5 show the performance per fold for all algorithms, the number of folds for which each was better than the others and the confidence factor associated with this respectively. This shows that we can have a 99.90% confidence in the ranking of the algorithms on this data set.

Fold	ANN	EKF	PF
1	8.13	7.13	6.84
2	7.67	7.02	6.36
3	7.81	7.1	6.5
4	7.64	6.93	6.41
5	8.07	6.98	6.78
6	7.71	7.01	6.43
7	7.37	7.05	6.13
8	7.81	6.99	6.66
9	7.32	7.2	6.04
10	7.72	7.53	6.43

Table 3 The outputs (per fold) of the three algorithms ANN, EKF & PF on the semi-synthetic data set.

B \ A	ANN	EKF	PF
ANN	x	10	10
EKF	x	x	10
PF	x	x	x

Table 4 The number of folds for which algorithm a is more accurate than algorithm b for the ANN, EKF & PF

B \ A	ANN	EKF	PF
ANN	x	99.90%	99.90%
EKF	x	x	99.90%
PF	x	x	x

Table 5 The confidence that algorithm a is more accurate than algorithm b for the ANN, EKF & PF

Table 6 shows the best performance of each algorithm when averaged across all ten folds.

Data set	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Semi-synthetic
EKF	0.21	3.66	6.53	11.12	7.09
PF	1.74	3.07	5.88	10.12	6.46
ANN	2.16	4.33	9.02	16.15	7.13

Table 6 The best RMS bearing error for each algorithm on each data set

- Although not limited by an internal model, the ANN is worse than both of the other techniques in all of the data sets.
- The EKF is more accurate on the simplest data set.
- The PF is more accurate on the more complicated sets
- There is a 99.90% confidence ranking on the ordering of the three algorithms, so in subsequent chapters comparisons will only have to be made against the particle filter to prove that an algorithm is more accurate than all three outlined in this chapter.

2.8 Conclusions

The performance of three standard target tracking algorithms has been evaluated, and performance is seen to degrade rapidly with increasing noise. These techniques are ideally suited to overly simplified data sets such as fully synthetic sets one and two. The performance of all of the tested algorithms was poor on the largest, most realistic data set, the semi-synthetic one. Here the ANN gave the worst overall performance, however it was not significantly worse than the other statistical techniques.

The only algorithm used which is not limited by assumptions made in an internal model is the ANN, however its performance was worse than the other two algorithms in almost all cases, with relative performance worsening as more input data was provided.

3 Improving upon existing predictors

3.1 Summary

In this chapter two new predictors are created, both based upon learning algorithms. A technique is developed in which a classification algorithm may be used as a bearing predictor. Two classification algorithms are tested; an ANN based classifier and a K-nearest neighbour classifier. Both are shown to operate effectively as trackers, though the ANN outperforms both the KNN and the baselines introduced in the previous chapter.

3.2 Introduction

The previous chapter has measured the performance of the three baseline methods. The first two baselines, the EKF and the PF require a model of the target motion in order to operate. Though in theory this model could be arbitrarily complex, in practice many assumptions are made in their construction which limits their achievable accuracy. Although the ANN's performance does not have this limit, it has been found to be worse in terms of output accuracy with the other two baselines. The aim of this chapter therefore is to start to develop a new predictor which, like the ANN could be immune to the limitations of having an overly simplified model, while also being capable of outperforming the other baselines.

This chapter presents the results of using two such machine learning algorithms as bearing predictors. Learning algorithms have a number of advantages over the more traditional statistical approaches. Provided that the system being predicted and its associated errors can be accurately modelled, the statistical techniques can be shown to be optimal. However, for reasons of computational efficiency and practicality of software development, these models are usually over-simplifications of the true system, leading to inaccuracies

Section 3.3 describes the problems of bearing prediction in passive sensors. The work done in this area is discussed in section 3.3. Our proposed solution is outlined in section 3.4. Sections 3.5 and 3.6 explain the experiments and the results respectively. Finally section 3.7 gives my conclusions based on this review.

3.3 Problem statement

The notation used in this thesis is based upon the notation used in [177]. Data is given as pairs called examples where each example $z_i = (x_i, y_i)$ consists of an object $x_i \in X$ and a label $y_i \in Y = \{1, 2, \dots, Y\}$. For the purposes of this thesis the object x_i comprises of the previous n bearing observations. The data consists of several bearing time series s_0, s_1, \dots, s_m each of which must be divided into sets using a sliding window approach before it can be used. First a window size w is selected, then a set of n examples is generated, each with w attributes, $x_i = [s_{i-w}, \dots, s_i]$.

The label is calculated from the difference between next bearing in the time series and the last bearing in the input $y_i = C(s_{i+1} - s_i)$ where C is a function that discretises the continuous real valued bearings into a set of categories.

The problem to be solved is therefore whether machine learning algorithms can compete with the baseline techniques in this context. In order to test this it will be necessary to select candidate learning algorithms and to define the binning function C . We will also experiment with the number of previous measurements to find the optimal number for predicting a future bearing.

3.4 Proposal and methodology

Our proposal is based upon two proposals for binning function, a uniform distribution as has been used before in the literature, and a Gaussian distributed binning function. The Gaussian was chosen as it is a common assumption that the distribution of the measurements is Gaussian about the true noiseless bearing. Here we make use of this assumption, and use the learning algorithms to model the difference between the Gaussian assumption and the true bearing, it is expected that using this function an approximately equal number of training sets will be classified into each of the bins

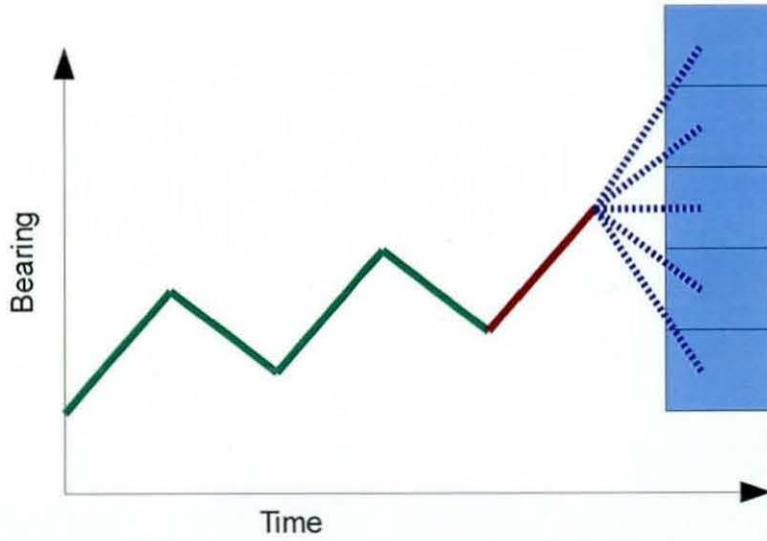


Figure 34 An illustration of uniform binning of prediction space

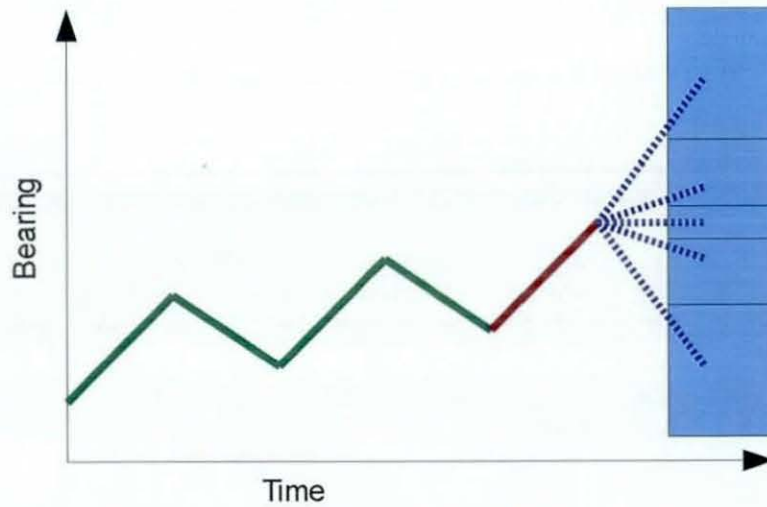


Figure 35 An illustration of Gaussian binning of prediction space

3.4.1

Binning function

In order to use any classification algorithm as a bearing probability predictor it is first necessary to specify the binning algorithm C to discretise the algorithm output space. Two possible functions for C are used in this thesis, the first producing uniformly distributed bins the second distributes the bins normally.

For the uniform binning function, two inputs are required, the first is the maximum

bearing rate change expected, i.e. $\Delta s_{max} = \max_{1 \leq i \leq m-1} (|s_{i+1} - s_i|)$, the second is the

number of categories Y . The $|Y|$ categories are uniformly distributed between

$-\Delta s_{max}$ and $+\Delta s_{max}$, each bin therefore is of size $s_{range} = \frac{2 \Delta s_{max}}{Y}$. As the

value for S_{max} is an estimation made with no a priori knowledge of the data, during experimentation it is possible for values of $s_{i+1} - s_i$ to be outside of the possible range of bins, in which case the value is rounded to the upper or lowermost category accordingly.

Once these bin sizes and ranges have been constructed it is necessary to convert the bearing data into arrays of outputs to facilitate classifier training. This was done by establishing which bin the required bearing would be in, setting the value of this bin to one, and the rest of the bins to zero, giving a simple probability density function (PDF) of the bearing

Initial experiments carried out with a uniform binning function, however, found that the range of training examples for each output bin was unevenly distributed. The distribution of training examples per bin was found to be approximately normal. In order to improve the classifier performance the number of training examples per classification should be approximately equal; therefore a distribution of bin sizes which gave close to equal numbers of examples per bin was also tested. Thus the second function used as the discretising function C is derived from a Gaussian rather than a uniform distribution for binning. The aim was to make the bins near the mean narrow, to decrease the number of examples, preventing overtraining. The bins further away from the mean are wider, to allow more examples to be observed. For this the inputs required are the standard deviation of the distribution σ_c in addition to the number of categories Y . The encoding here is achieved using the cumulative normal distribution for σ_c , with a mean of zero.

$$cdf = \frac{1}{2} \left(1 + \operatorname{erf} \frac{x}{\sigma \sqrt{2}} \right)$$

The cell to use is chosen by integrating between $-\infty$ and the bearing input s to give a number between 0 and 1, multiplying by the number of bins, and then finding the nearest whole number.

$$binnum = \lfloor Y \int_{-\infty}^s \frac{1}{2} \left(1 + \operatorname{erf} \frac{x}{\sigma \sqrt{2}} \right) dx \rfloor$$

The process for both binning functions can also be reversed to decode the outputs of a classifier into a bearing, denoted as $f^{-1}(X) = s$

3.4.1.1 Converting bin number to network training output

Both of the binning functions outlined above would result in an array in which all but one of the values would be zero, and a single element corresponding to the bearing would be one. However when this is used, a network which misclassifies the data into an adjacent bin to the truth is penalised as much as one which classifies into a bin distant from the one required. This is clearly incorrect as the latter would result in a far higher error in the application presented in this thesis.

To remedy this problem a second Gaussian distribution was used to alter the training arrays. The second Gaussian used the mean equal to the target bearing and a standard deviation of the bearing error, which is a known value for real-world sonar systems

The training value used for each bin is given in equation 6;

$$X_i = \int_{f^{-1}(i-1)}^{f^{-1}(i)} \frac{1}{2} \left(1 + \operatorname{erf} \frac{s}{\sigma_{bearing} \sqrt{2}} \right) ds \quad (6)$$

Where f^{-1} is the appropriate reverse formulation of either the uniform or Gaussian binning function, $f^{-1}(i)$ gives the bearing at the end of bin i . The resulting vector X is therefore the ideal output of the classifier network which is used to train the ANNs. When trained using this data, the array better approximates the true PDF of the data. An example of this type of data can be found in appendix C.

3.4.2

Learning algorithms

Two learning algorithms were chosen for the experiments, K-nearest neighbour (KNN) and artificial neural networks. KNN was originally chosen as it is very simple to implement, and was used to test quickly the theory that classifiers could be used as predictors. ANN was chosen to continue the work as it is known to be faster than

KNN, and better at generalising, though harder to implement. ANN and KNN are described in more detail in sections 3.4.2.1 and 3.4.2.2 respectively. Although only the KNN and ANN are used in this thesis, the ideas are generic and could be applied to any classification algorithm.

3.4.2.1 K-nearest neighbour (KNN)

In order to test the theory that a classification algorithm could be used as a bearing predictor, it was necessary to find a classification algorithm which would be simple to code, without any initial requirement for computational efficiency.

The KNN algorithm is such an algorithm that is commonly used in pattern recognition. It is most similar to the Nearest Neighbour classification algorithm, in which a given set of measurements are assigned to a particular class based on the nearest set of measurements in a database of known examples which is created during a training phase. The KNN only differs from this in that the K nearest classifications are considered, and the one that has previously been observed the most times is selected. Both NN and KNN are learning algorithms, and require a period of training before they can be used

The KNN used bins the input bearings uniformly, and the output bearings using one of the binning formulae described in section 3.4.1, and the database is built up by counting how many times each example is seen in the training data. The database stores each example and the count of how many times it has been seen. During the testing phase the input data is compared to values in the database, and the nearest k rows are selected, of those the one with the highest count is selected, and the classification associated with it is used as the prediction

3.4.2.2 Artificial neural networks (ANN)

The networks used in the ANN experiments were simple feed-forward, fully connected, multi layered networks with sigmoid activation functions in the hidden layers. The simplest possible form of multi layer ANN was chosen in order to demonstrate as easily and quickly as possible that ANNs could be used as classifiers to track targets.

The learning algorithm used is an altered form of backpropagation. Using standard backpropagation the network would be adjusted based on the classification error, so a misclassification resulting in a bearing error of 0.5° is treated as being as wrong as a misclassification resulting in a bearing error of 10° . An altered form of backpropagation that

takes the resultant bearing into account while training the network was developed. This effectively penalises solutions which are further away from the ideal solution in terms of output bearing, and promotes those which are closer.

The number of bearings to use is determined by the number of input nodes in the ANN. A network with n inputs will use the last n bearings to either predict the next bearing or estimate the range of the target. The output also has a varying number of nodes, each one representing a bearing bin. Values are assigned to the outputs according to the binning algorithm used.

A form of ANN which differs from that used in this new method has been used many times in the literature [93]. In this type of network the previous measurements are given as inputs to the network, but the ANN outputs each prediction as a value from a single output node, therefore no binning is used or required. This type of ANN was used in the experiments as the third baseline approach.

3.5 Experimental design

3.5.1

Experimental design

Cross validation and interleaving with 10 folds were used to validate the results in both the ANN and KNN based experiments. All 500'000 data points were used in total, each fold using 50'000 for testing and 450'000 for training, therefore every data point was used as part of the test set in one and only one of the 10 folds. The mean value was taken across all 10 folds to give the final result.

3.6 Results and comparison

The following results were generated with the 245000 patterns described in section 2.7.6.2, except for the KNN which proved too slow to practicably run on such a large data set, so the smaller 4998 pattern set was used, also described in section 2.7.6.2. For reasons of clarity the results for the KNN and EKF with uniform distribution binning are not shown in the following graphs. Since these are not classification algorithms running them through the binning process could only worsen their results. In every experiment the uniform binning function proved inferior to the Gaussian, the ANN with uniform binning function has been left in the results to demonstrate. Figures 36 to 47 give the results of the experiments. In each case, the chart is first given with all tested algorithms, followed by the same chart with only the best version of the proposed technique, shown against the best performing two benchmarks. This allows the performance increase to be seen more clearly. The Gaussian Binning ANN

Classifier is shown to be the optimal choice in each case, and the best performing baselines are the Particle filter and the EKF. The particle filter outperforms the EKF in every case.

Figures 36 to 39 require some explanation. These data sets were deliberately created so that it would be possible to plot every point in the set. This allowed a simple visual test to see if the algorithms were working as expected. Ten fold cross validation was still used; the results of the first fold going directly to the first ten percent of the chart, then the next fold to the second ten percent etc... until the entire chart was filled left to right. This relied on running through the dataset sequentially from start to finish, testing each 10% of the data in isolation, using the other 90% as training and validation data. This process would not work on *any* dataset, however due to the periodic property of these simple sets it can be expected that for each case in the testing data the learning algorithm will have seen a similar case during its training phase. Figures 36 to 39 show cumulative bearing residuals; that is at each time step the absolute value of the difference between the predicted bearing and the actual bearing is added to the total and the total to date is plotted. This allows the reader to see the performance of the algorithms over time. It is also important to note that this is not the case for the semi-synthetic data set, the patterns used in this set were randomized before it was divided into folds firstly as it did not contain the periodicity of the simple sets and secondly because this increases the reliability of the testing. The final result for time 500 on the far right of the plot shows the overall performance of the algorithms. As the line shows performance over time, jumps in the line represent outliers in the performance of the algorithm, points at which the predicted value had an unusually high error value

3.6.1

Synthetic data

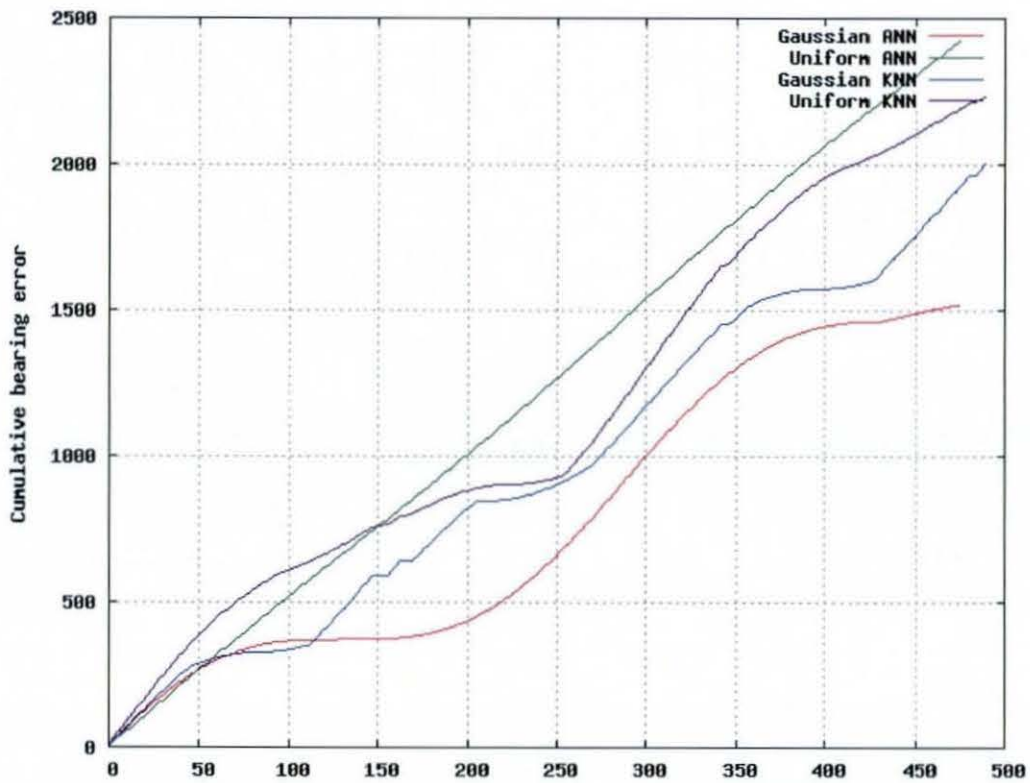


Figure 36 Cumulative bearing residual for Gaussian Binning ANN, Uniform Binning ANN, Gaussian Binning KNN and Uniform Binning KNN when run on synthetic data set 1

- The performance of the algorithms is similar, however the Gaussian binned ANN outperforms the others by a clear margin.
- The Gaussian binned KNN is the second best algorithm showing;
 - The KNN is capable of tracking targets
 - The Gaussian binning system is superior on this data set.

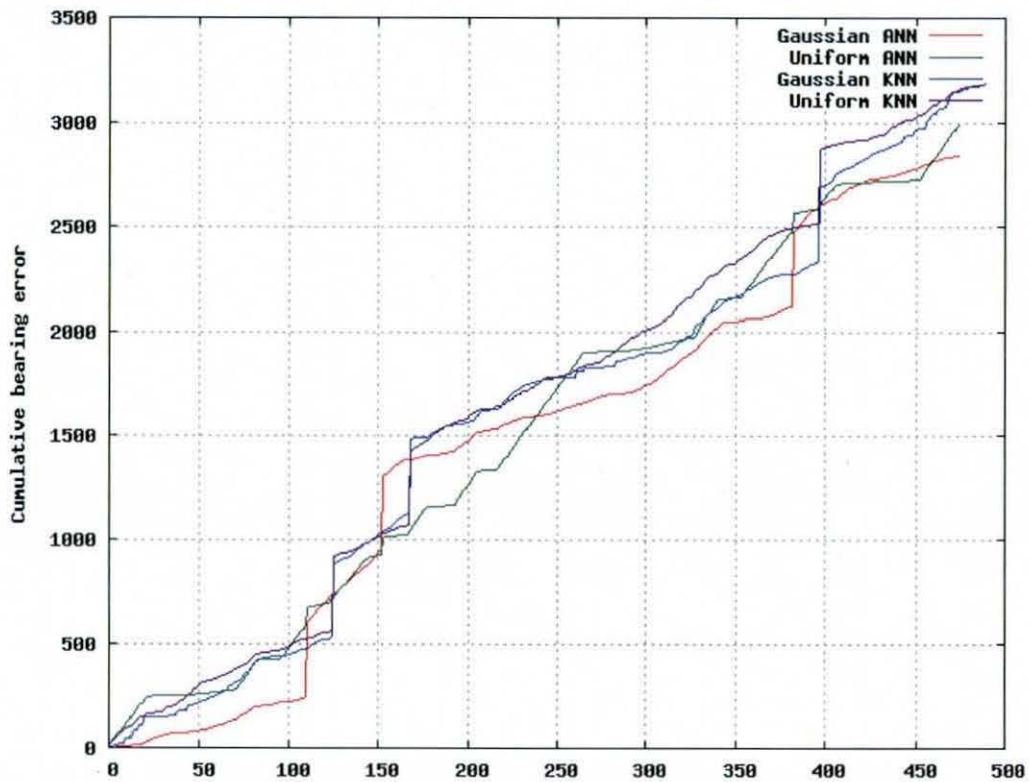


Figure 37 Cumulative bearing residual for Gaussian Binning ANN, Uniform Binning ANN, Gaussian Binning KNN and Uniform Binning KNN when run on synthetic data set 2

- On this marginally harder data set the performance of the four algorithms is more closely matched than on the first set.
- The ANN performs best overall, by a narrow margin.

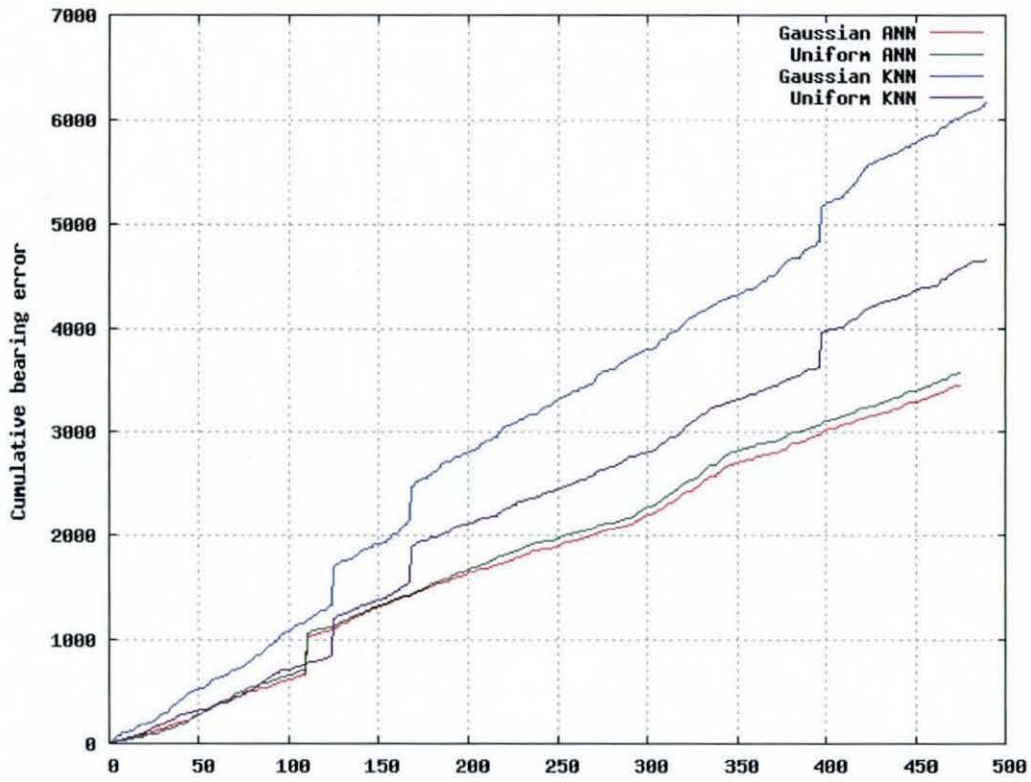


Figure 38 Cumulative bearing residual for Gaussian Binning ANN, Uniform Binning ANN, Gaussian Binning KNN and Uniform Binning KNN when run on synthetic data set 3

- As the data sets become more complex, the ANN is starting to have a clear advantage over the KNN.
- For this data set the ANN is superior regardless of the binning algorithm,
- The performance of the KNN shows more distinction between uniform and Gaussian binning.

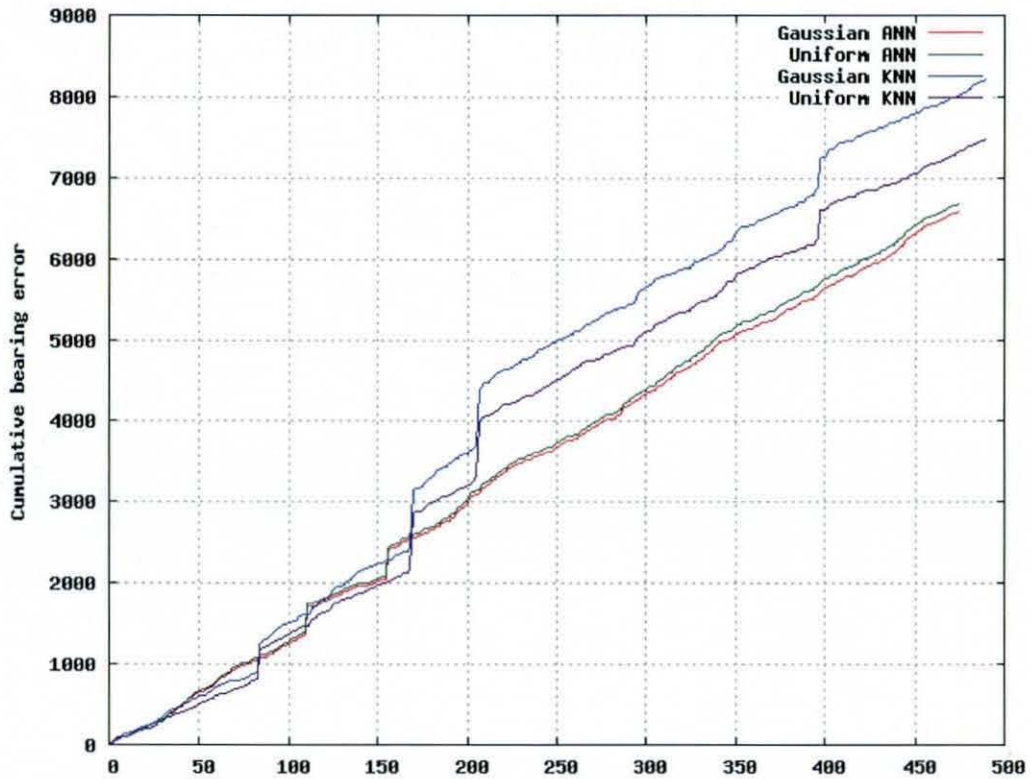


Figure 39 Cumulative bearing residual for Gaussian Binning ANN, Uniform Binning ANN, Gaussian Binning KNN and Uniform Binning KNN when run on synthetic data set 4

- Again, the performance of the two ANN based algorithms can be seen to be similar.
- The KNN based algorithms again show more distinction between the uniform and Gaussian binning versions than the ANN.

3.6.2

Semi-synthetic data

3.6.2.1 Binning parameters

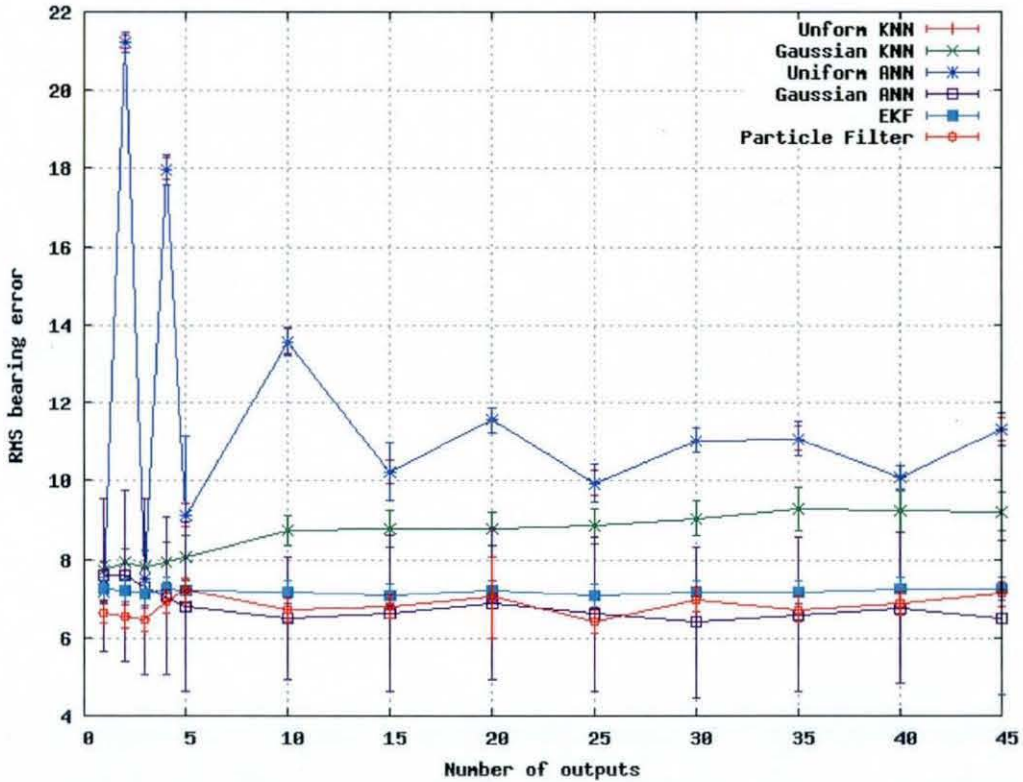


Figure 40 RMS bearing residuals against number of output bins (between 1 and 45) when run on the semi-synthetic data for the Uniform Binning KNN, Gaussian Binning KNN, Uniform Binning ANN, Gaussian Binning ANN, EKF, PF and Single Output ANN

- The Gaussian ANN performs best overall, giving the lowest error for virtually every outlier.
- The uniform binning KNN performs the worst overall, performing considerably worse than the other algorithms on almost every number of outputs.
- The Gaussian binning is only a little better, again performing worse than all of the non-KNN techniques.
- The performance of the ANN based classifier predictors is very similar to the baselines.

Figure 41 is a repeat of figure 40, showing only the best performing of the new algorithms, along with the two best baselines.

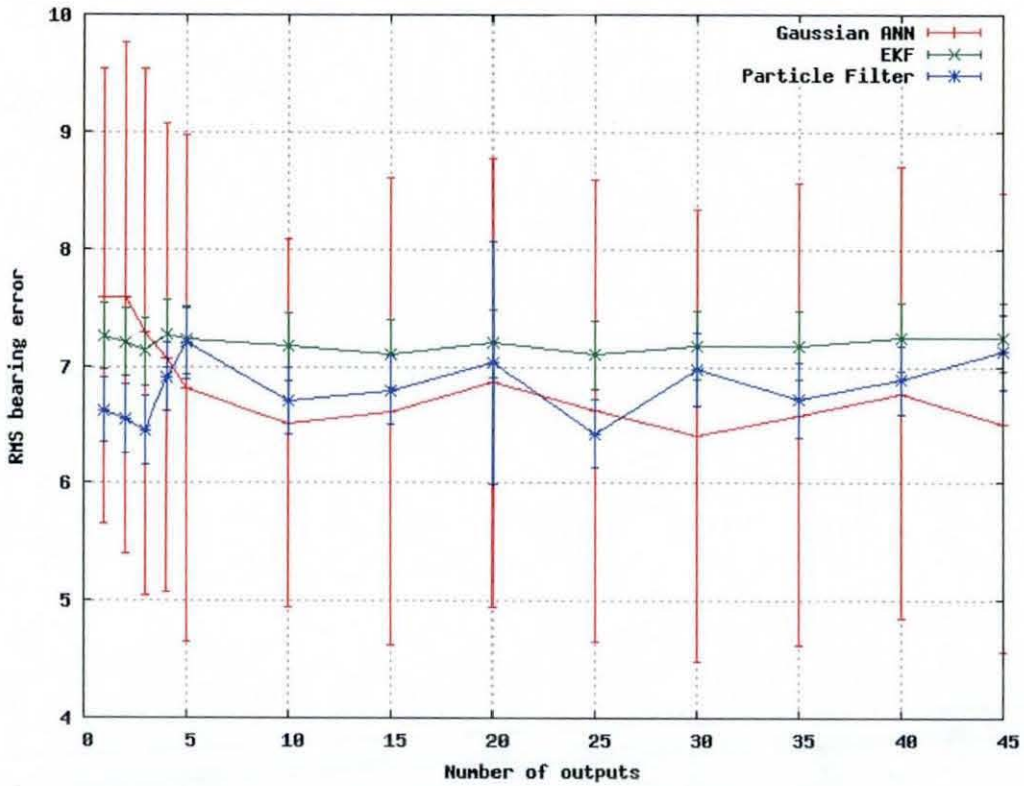


Figure 41 RMS bearing residuals against number of output bins (between 1 and 45) when run on the semi-synthetic data for the Gaussian Binning ANN, EKF and PF only

- The performance of the Gaussian binning ANN algorithm is shown to be very similar to the performance of the PF, however the ANN is generally better.

All of the algorithms were then tested as classifiers to see how well they predict in which bin the next bearing would fall. The predictions of the non-binning classifiers were run through the binning algorithms to allow direct comparison.

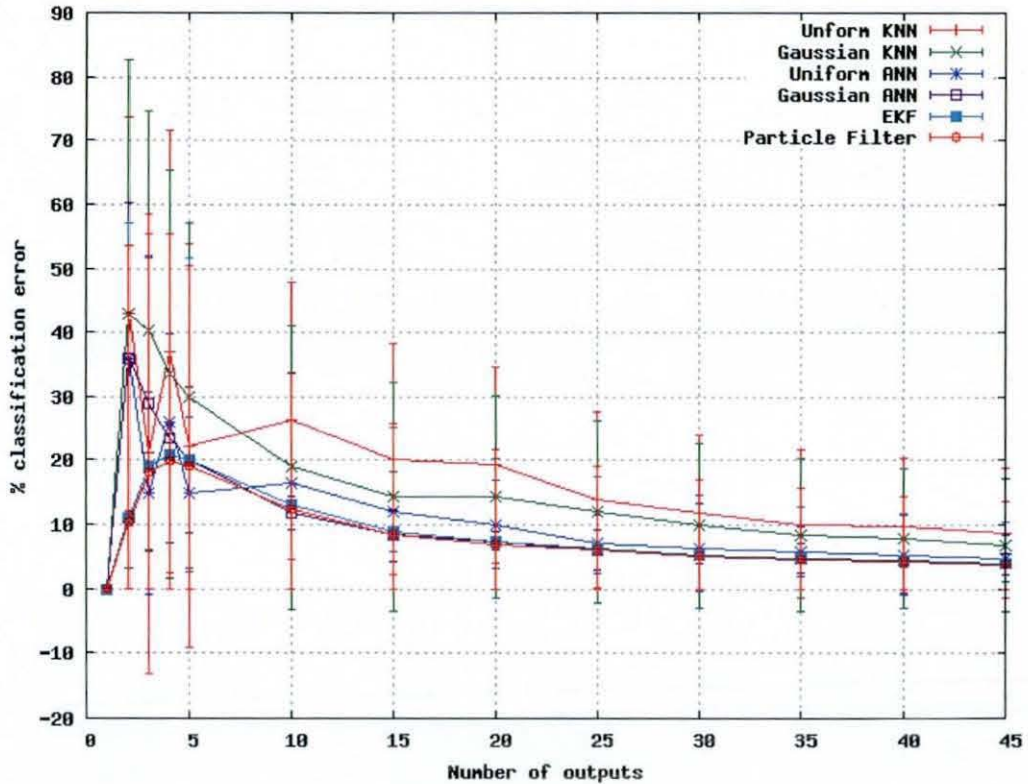


Figure 42 % classification error against number of output bins (between 1 and 45) when run on the semi-synthetic data for the Uniform Binning KNN, Gaussian Binning KNN, Uniform Binning ANN, Gaussian Binning ANN, EKF, PF and Single Output ANN

- As expected, with a single output bin, all algorithms have an error of zero.
- The relative performances mirror those observed on the RMS bearing error.

Figure 43 is a repeat of figure 42, showing only the best performing of the new algorithms, along with the two best baselines.

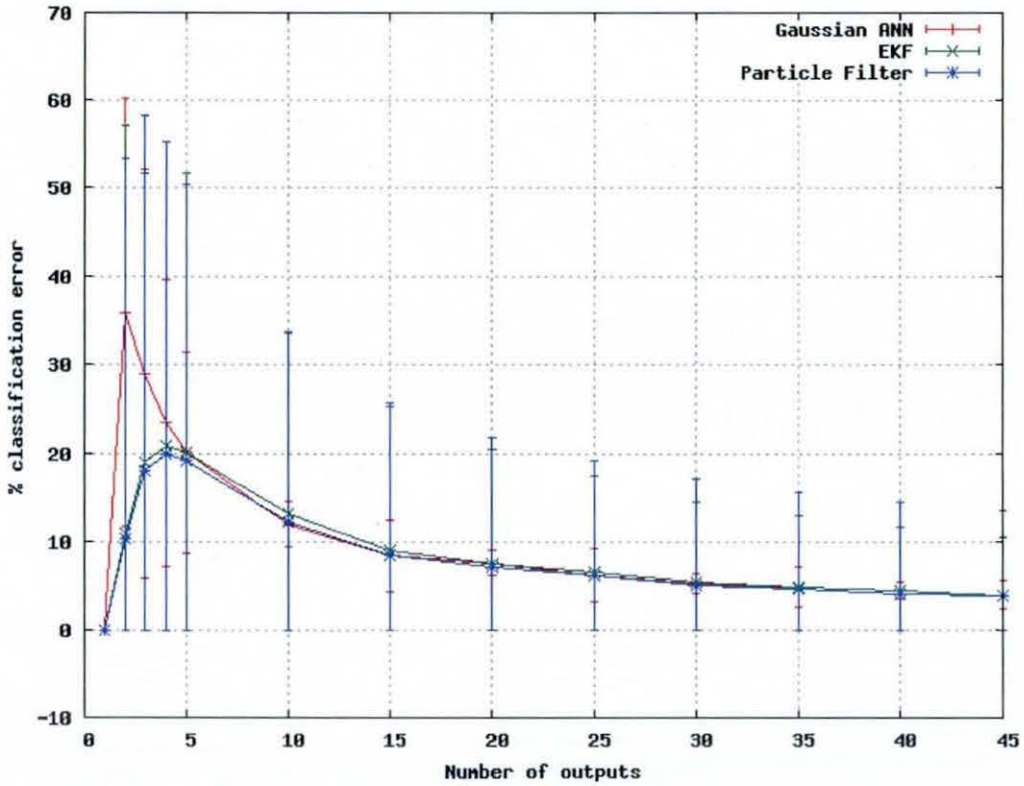


Figure 43 % classification error against number of output bins (between 1 and 45) when run on the semi-synthetic data for the Gaussian Binning ANN, EKF and PF only

- On this plot it is clear that with fewer outputs the new algorithm is less accurate than the baselines, however with larger number of bins, and therefore finer discrimination on predictions, the performance of the new algorithm is almost identical to the baselines.

3.6.2.2 Input parameters

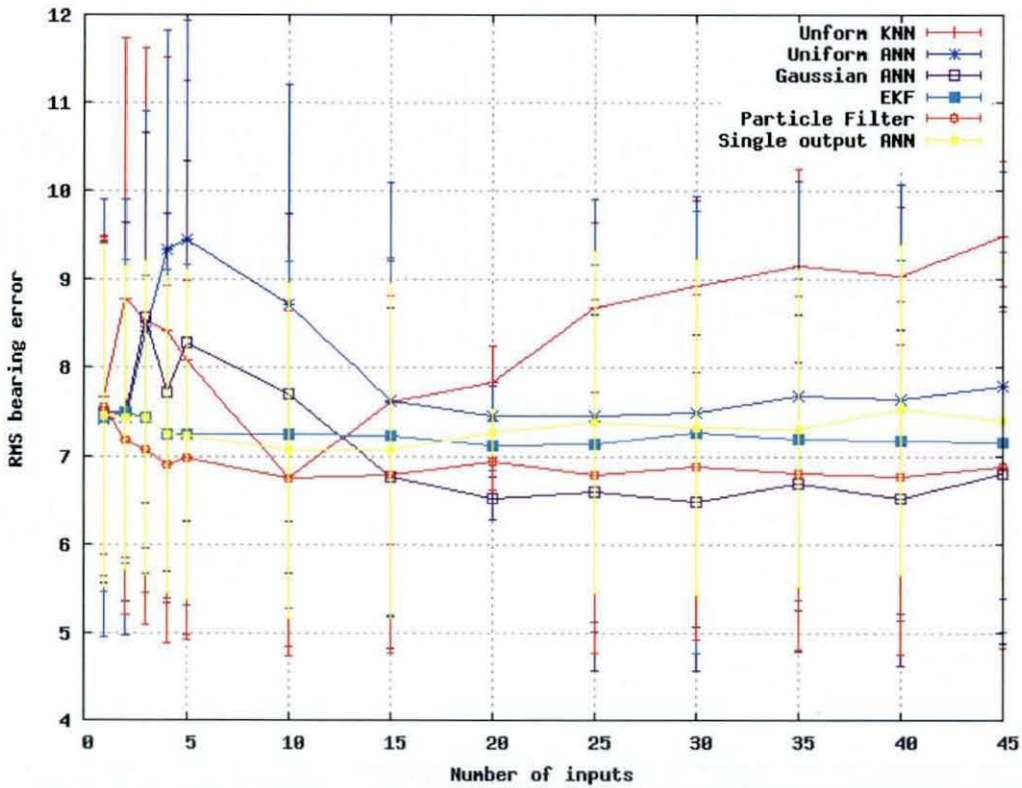


Figure 44 RMS bearing residuals against number of input bins (between 1 and 45) when run on the semi-synthetic data for the Uniform Binning KNN, Gaussian Binning KNN, Uniform Binning ANN, Gaussian Binning ANN, EKF, PF and Single Output ANN

- Again the KNN is poorer than all of the other techniques, with the uniformly binned KNN performing worse than the Gaussian binned KNN.
- The Gaussian binned ANN is again the best performing of all of the algorithms.
- As the number of inputs increases the performance of the KNN decreases. This may initially appear counter-intuitive; most algorithms improve with more data. However the KNN works by building up a database of examples with a count representing the relative probability of each example. As the input data is binned there are a finite number of examples, the number of which varies with the number of input bins. As the number of bins increases the number of examples in the database also increases, so for the same training data there are fewer instances seen of each example. For the best performance it is required that some examples will have been seen many times, while other similar examples have been seen far fewer times, which allows the KNN to

discriminate. When there were more bins the filter had actually seen only a single instance of many of the input patterns and non at all for many of the others.

It is possible to explain this with numbers. Using the Uniform Binning KNN as an example, if for the input and prediction data a fixed bin size b_f was used of 0.1 degrees, and a maximum bearing b_m of 10 degrees, for each input

there would be 101 combinations $\frac{2b_m}{b_f} + 1$. There would also be 100

combinations of the output (the predicted bearing). Therefore the total number

of examples in the KNN database would be $\frac{2b_m}{b_f} + 1$ ⁽ⁿ⁺¹⁾ for n inputs. This

would result in the ratios of training examples to possible combinations in the KNN database shown in table 7. This shows that the number of relevant examples decreases exponentially with the number of inputs. A scheme in which the bin size was dynamically altered to keep the database size constant would remedy this and allow the KNN to be effective with a larger number of inputs, this would however still have a limit to performance as the lowest possible useful number of bins is two, positive and negative. Even with this simple scheme with 100 inputs and 1 output this represents 6.18×10^{42} entries in the KNN database, approximately the same number as is the case with 20 inputs in table 7, a number for which figure 44 shows that performance is already noticeably degraded. The same also applies to the Gaussian binned version, as the number of bins per input is constant.

Number of inputs	Entries in KNN database	Examples per entry
1	40401	6.187965644
2	8120601	0.030785899
3	1632240801	0.000153164
4	3.2808E+11	7.62008E-07
5	6.59442E+13	3.79109E-09
10	2.1635E+25	1.15554E-20
15	7.09802E+36	3.52211E-32
20	2.32872E+48	1.07355E-43
25	7.64007E+59	3.27222E-55
30	2.50656E+71	9.97384E-67
35	8.22353E+82	3.04006E-78
40	2.69798E+94	9.2662E-90
45	8.8515E+105	2.8244E-101

Table 7 The number of entries in the KNN database and the number of examples per entry against number of inputs when using uniform binning with a maximum bearing of 10, a bin size of 0.1 and training with the semi-synthetic data set

Figure 45 is a repeat of figure 44, showing only the best performing of the new algorithms, along with the two best baselines.

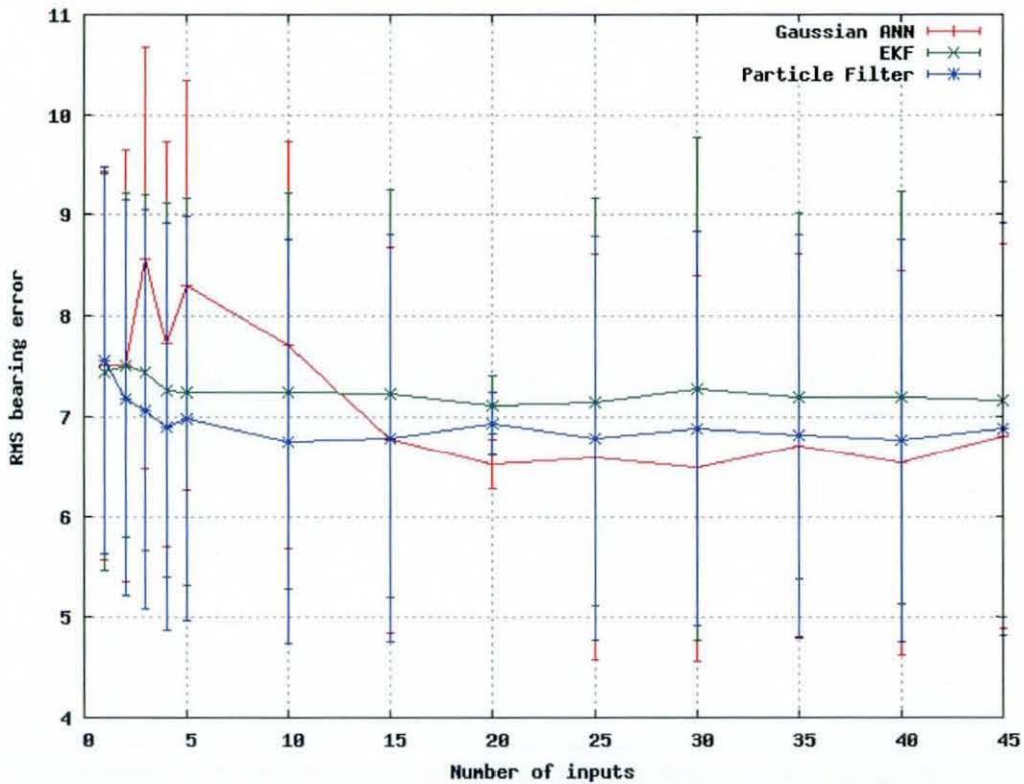


Figure 45 RMS bearing residuals against number of input bins (between 1 and 45) when run on the semi-synthetic data for the Gaussian Binning ANN, EKF and PF only

- For fewer inputs the Gaussian ANN performs the worst. Here the PF performs best, and the PF gives consistent level of error, regardless of the number of inputs.
- For more inputs the Gaussian ANN performs the best.
- The algorithm that gives the lowest error is the Gaussian ANN.

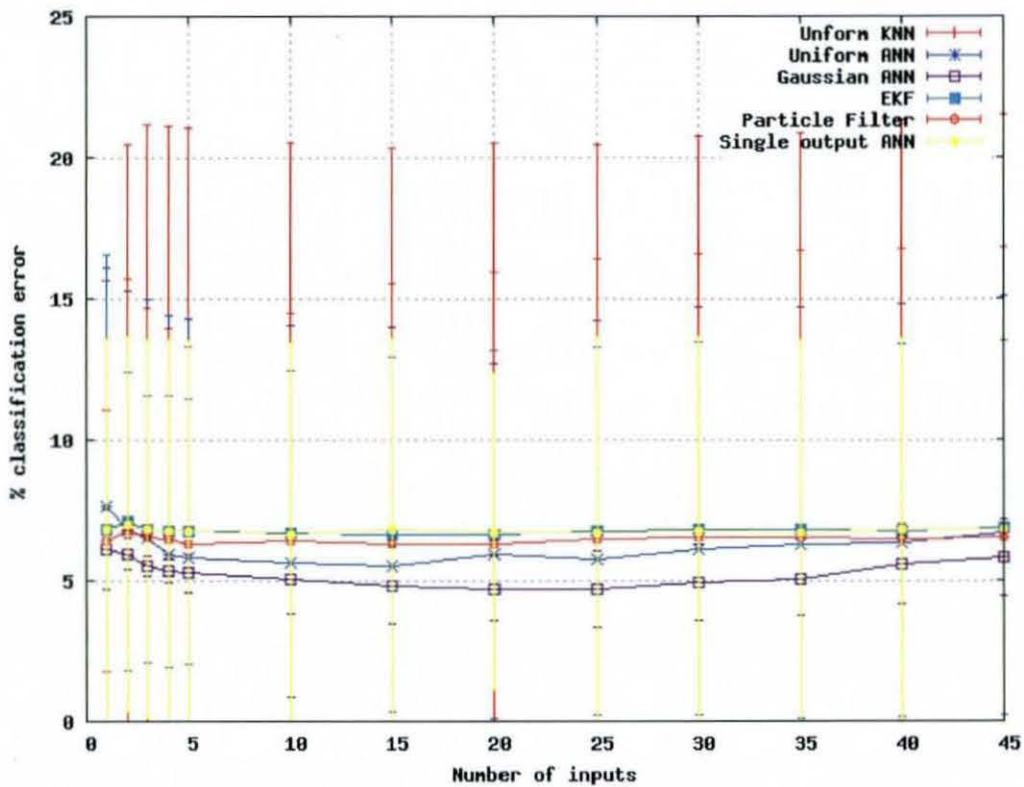


Figure 46 % classification error against number of input bins (between 1 and 45) when run on the semi-synthetic data for the Uniform Binning KNN, Gaussian Binning KNN, Uniform Binning ANN, Gaussian Binning ANN, EKF, PF and Single Output ANN

- Again the KNN performed the worst overall
- The Gaussian ANN performs the best, giving the lowest error at every point.
- Out of all of the new algorithms, only the Gaussian ANN outperforms the baselines.

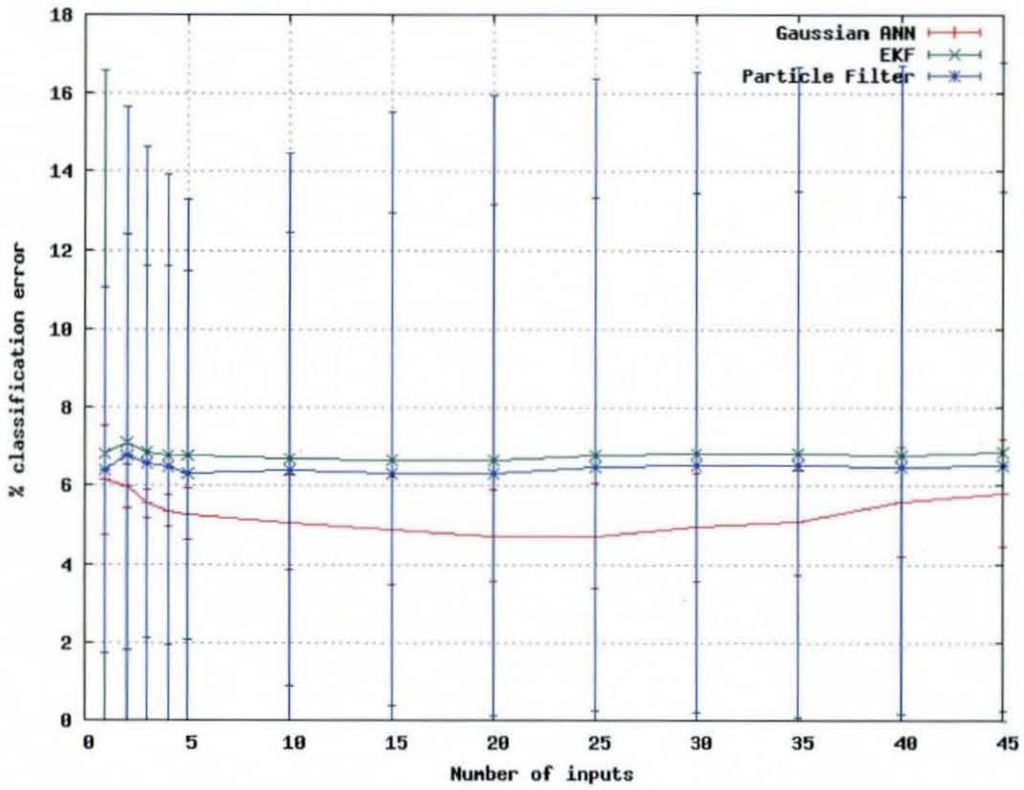


Figure 47 % classification error against number of input bins (between 1 and 45) when run on the semi-synthetic data for the Gaussian Binning ANN, EKF and PF only

Here the performance improvement made by the Gaussian binning ANN is even clearer. The mean improvement on the baselines is considerable.

Table 8 shows this broken down into the individual folds. Table 9 shows that this new algorithm the Gaussian Binning ANN (GANN) is more accurate than both the plain ANN and the EKF on all 10 of the folds, while it is more accurate than the PF on 7 of the ten folds. From table 10 it is possible to see that it can be said that the new Gaussian ANN is better than the ANN and the EKF with a confidence of 99.90%, however the confidence that the GANN is better than the PF is only 82.81%.

Fold	ANN	EKF	PF	GANN
1	8.13	7.13	6.84	6.59
2	7.67	7.02	6.36	6.46
3	7.81	7.1	6.5	6.32
4	7.64	6.93	6.41	6.35
5	8.07	6.98	6.78	6.46
6	7.71	7.01	6.43	6.43
7	7.37	7.05	6.13	6.33
8	7.81	6.99	6.66	6.56
9	7.32	7.2	6.04	6.18
10	7.72	7.53	6.43	6.42

Table 8 Results per fold for the ANN, EKF, PF & GANN

B \ A	ANN	EKF	PF	GANN
ANN	x	10	10	10
EKF	x	x	10	10
PF	x	x	x	7
GANN	x	x	x	x

Table 9 The number of folds for which algorithm A was more accurate than algorithm B for the ANN, EKF, PF & GANN

B \ A	ANN	EKF	PF	GANN
ANN	x	99.90%	99.90%	99.90%
EKF	x	x	99.90%	99.90%
PF	x	x	x	82.81%
GANN	x	x	x	x

Table 10 The percentage confidence that algorithm A was more accurate than algorithm B for the ANN, EKF, PF & GANN

Data set	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Semi-synthetic
EKF	0.21	3.66	6.53	11.12	7.09
PF	1.74	3.07	5.88	10.12	6.46
ANN	2.16	4.33	9.02	16.15	7.13
Gaussian ANN	3.2	5.97	7.28	13.86	6.41
Uniform ANN	5.1	6.28	7.53	14.09	7.15
Gaussian KNN	4.09	6.5	12.58	16.8	7.71
Uniform KNN	4.56	6.52	9.51	15.26	7.16

Table 11 The best RMS bearing error for each algorithm on each data set

- The new algorithms perform very poorly on the fully synthetic, very simple data sets
- The new algorithms, particularly the Gaussian ANN are very competitive with the baselines for the most realistic data set, the semi-synthetic data set.

3.7 Conclusions

The results show that in terms of RMS prediction error, as the number of inputs increases, the accuracy of the proposed technique improves dramatically. With fewer inputs the two best baseline techniques both outperform the proposed technique, however with more inputs the new technique is considerably more accurate than all of the baselines. Unsurprisingly, as the algorithm is classifier based, it performs better on the classification error metric than any of the baselines. These results initially appear contradictory, however there is a non-linear relationship between bearing accuracy and classification accuracy.

When the number of inputs is more than fifteen, by using the Gaussian ANN algorithm the bearing error can be reduced by as much as 16% over the EKF, and 6% over the PF. However there are many parameters such as the underlying algorithm (e.g. ANN or KNN), network size and structure and the learning algorithm that must be selected to find the most efficient and accurate network capable of outperforming the baseline techniques. Changing the parameters can significantly alter the performance, although the relationship between the parameters and the performance is clearly complex.

There are a number of examples in the literature of ANN ensembles outperforming individual ANNs, therefore one logical next step would be to experiment to find whether using an ensemble improves performance in this application

A set of parameters has been found that allows the ANN filter to outperform the EKF and Particle Filter. However although the average performance of the new algorithm is better than the state-of-the-art, this improvement was not shown to be statistically significant as it was an improvement in only seven out of the ten folds. This results in a confidence that the new algorithm is an improvement over the old of only 82% which is lower than the 90% target set for this thesis. As an improvement is required in at least eight out of the ten folds in order for the new algorithm's improvement to be considered statistically significant, ways to improve the algorithm further must be found.

4 Ensembles of predictors

4.1 Summary

This chapter expands the ANN classifier based predictor developed in chapter 3, taking inspiration from the ideas of fusion described in chapter 2. The outputs of an ensemble of ANN classifier based predictors are fused to give a more accurate output than was possible with a single ANN predictor. Negative Correlation Learning (NCL) is used to train the ensembles. Iterations of the experiment are performed to find the optimal structure for the ensemble. The result is an ensemble which can outperform all of the predictors previously described in terms of prediction accuracy.

4.2 Introduction

As noted in the previous section, there are many examples in the literature of ensembles which have been found to give superior performance over single ANNs. An ensemble is simply a collection of ANNs connected together with a form of fusion algorithm to combine the results, the first examples being proposed by [252] & [253]. Many examples can be found in the literature of ANN ensembles [238][252] and mixtures of experts [292][144][148] being used successfully. The idea that multiple combined classifiers should be expected to outperform a single classifier was formally proven in [273].

One simple method for creating an ensemble would be to train each internal ANN with standard backpropagation individually, and then take the mean output of the ANNs as the ensemble output. This however is likely to result in a collection of very similar networks, giving very similar outputs, negating the advantages of using an ensemble, as the mean would be virtually the same as the output of any of the components ANNs, while taking longer to train and test. Many researchers, including [112][114][167] & [220], have found that accuracy and diversity between its members are both required in order to create a successful ensemble.

One way in which learning algorithms have been combined to create an algorithm more accurate than its components is a technique known as Boosting [92] [251]. Boosting is a method for training ensembles of classifiers iteratively; all examples in the data set start with equal weight, but gain weight if they are misclassified by the algorithm. This weighting is rerun every time a new classifier is added to the ensemble. In this way the

examples on which the ensemble is the worst at classifying are shown the most during training. This however is not in itself a training algorithm, it is more a way of organising the input data prior to exposing it to the learning algorithm.

What is required is a technique for promoting diversity between the members of the ensemble. One method which has been widely found to be effective in the literature is Negative Correlation learning, as described in section 4.4. NCL is a good choice for a baseline for numerous reasons [41];

- It has been shown to be more effective than other learning algorithms at selecting ensemble members, through adjusting the balance between accuracy and diversity [39]
- One of the aims of its creators was to increase the amount of diversity in the networks constructed – which is also one of the objectives of the technique proposed here.
- It is the basis of many other techniques which have been seen recently in the literature, and is both widely used and effective (surveyed and categorised in [41]).

4.3 Ensembles in time-series forecasting

Recently a number of researchers have investigated ensembles of neural networks as time series predictors. In a recent survey paper on the area [312] described time series prediction as one of the four main areas of ensemble use, alongside multiple features, accuracy estimation and noisy data.

The two most important decisions when creating a time series forecasting algorithm are how much historical data must be used to make the prediction [154], and how far into the future to predict [149]. Both of these areas have been studied, and systems have been created to automatically tune the classifiers to the optimal values for the dataset.

- The individuals which make up the ensemble may be fuzzy predictors [158], ANNs [62] [210], or any algorithm already used to make predictions.

4.4 Negative Correlation Learning

[179][180] & [181] introduced Negative Correlation Learning (NCL) which is a method of training ensembles of ANNs which encourages the ANNs in the ensemble to become diverse. The goal of NCL is to negatively correlate the ANNs within an ensemble so that their errors cancel each other out. ANNs are typically trained using

backpropagation, here backpropagation is used, however the error term is altered to include the correlation of the ANN to the rest of the ensemble

The NCL error function is:

$$e_i = \frac{1}{2} (f_i(x) - d)^2 + \lambda (f_i(x) - f_{ensemble}(x)) \sum_{j \neq i} (f_j(x) - f_{ensemble}(x))$$

Where i is the ANN within the ensemble, d is the ideal output, $f_i(x_N)$ gives the result from running ANN i on data sample n of N and $f_{ensemble}(x_N)$ gives the output of the whole ensemble, d is the desired output, and λ is a parameter in the range $[0,1]$ which governs how much the correlation of the networks should be used in training e is the error value passed to the ANN for training using backpropagation.

For the experiments in this section NCL was used to train an ensemble, with the NCL error value calculated for each bin in the output

4.5 Experimental design

The experiments carried out for this section were designed to be as close to the experiments in sections 2.7 and 3 as possible to allow direct comparison. Again, the algorithm was optimised on the same manner as the EKF, the PF and the ANN used in section 2.7, a series of experiments was performed to empirically calculate the parameters which required adjusting in order to tune the algorithm to give its best performance for the data used.

The NCL parameter λ and the number of hidden nodes were all derived this way, looping through possible values for each, evaluating their performance against each data set. The learning parameter, number of input nodes and number of output nodes were set to the values found to be optimal in section 3. The number of ANNs in the ensemble was set to five, this is not expected to be optimal, however it was planned that if the experiments were successful, more experiments would be planned in order to find the optimal number.

As each parameter was tested, the RMS bearing error was evaluated, and the value of the parameter that gave the lowest RMS error was stored for use from that point on.

There were two iterations of the procedure

As the previous section identified optimal values for the number of input nodes and the number of output nodes, the values found in the previous section are utilised

4.6 Results

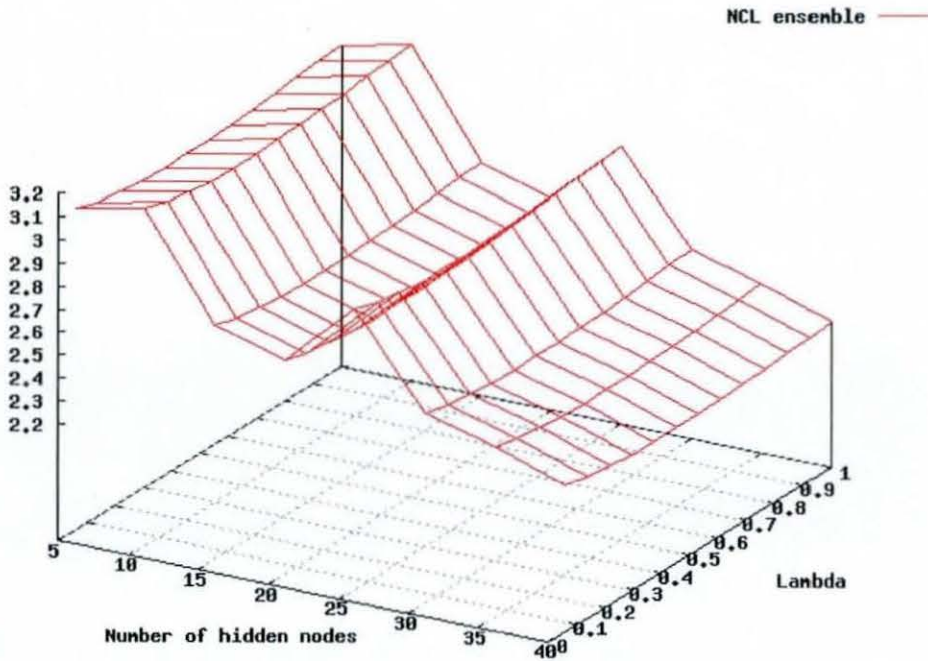


Figure 48 RMS bearing residual for λ between 0.0 and 1.0 and number of hidden nodes per ensemble between 5 and 40 for ensembles consisting of 5 ANNs trained with NCL using simple data set 1

- There is a 'smile' shaped curve showing lambda effecting the accuracy, with both no diversity and lots of diversity giving poorer results than moderate amounts of diversity.
- Unsurprisingly the most accurate result is for the largest number of hidden nodes, as there are five ANNs, there are a total of two hundred hidden nodes in the ensemble. This may be as there is no noise, and it is a simple data set, there is no penalty for over-fitting, which a larger network may be capable of.

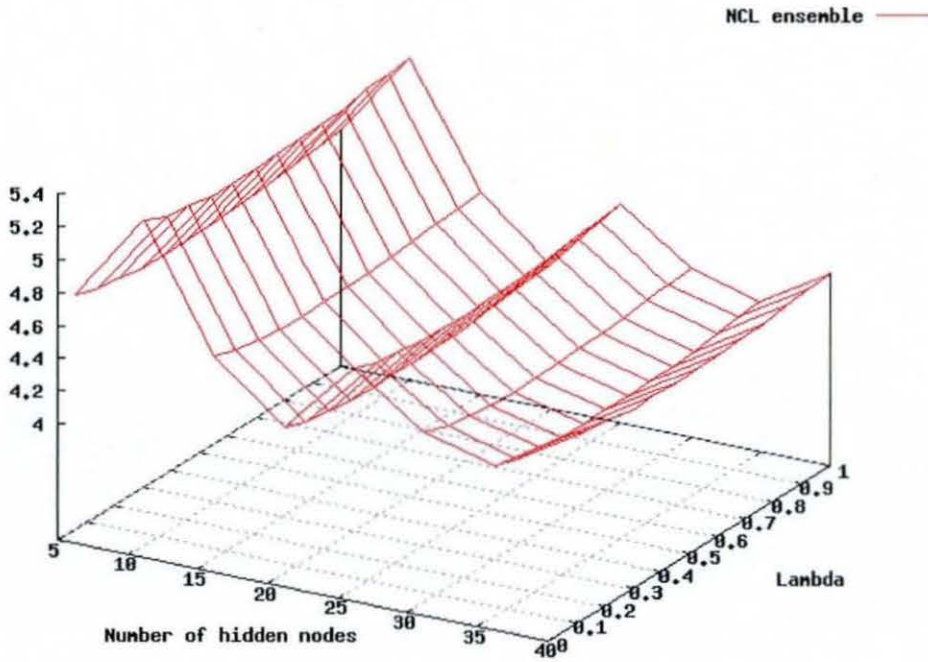


Figure 49 RMS bearing residual for λ between 0.0 and 1.0 and number of hidden nodes per ensemble between 5 and 40 for ensembles consisting of 5 ANNs trained with NCL using simple data set 2

- The smile shape is again noticeable, though subtle
- Here the most accurate result is from having twenty hidden nodes per ANN, here the ensembles based on larger ANNs may be being penalised for overfitting.

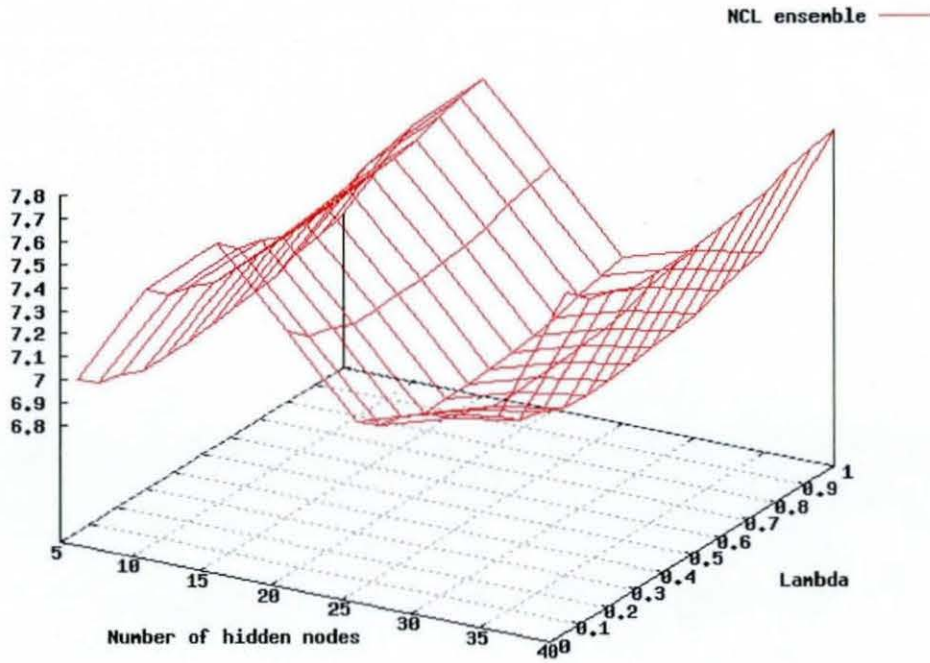


Figure 50 RMS bearing residual for λ between 0.0 and 1.0 and number of hidden nodes per ensemble between 5 and 40 for ensembles consisting of 5 ANNs trained with NCL using simple data set 13

- Here the smile shape is more noticeable, showing a clear advantage of moderate amounts of diversity..
- Again the largest networks are shown to be less accurate than both the medium sized and here the very small. The medium sized and smaller networks may be better at making generalisations about the data.

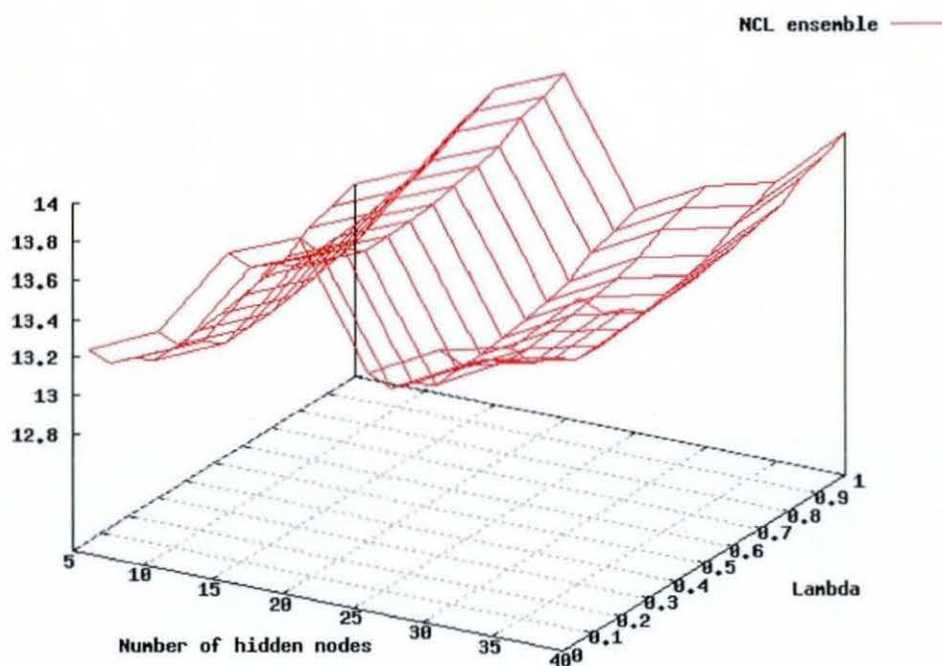


Figure 51 RMS bearing residual for λ between 0.0 and 1.0 and number of hidden nodes per ensemble between 5 and 40 for ensembles consisting of 5 ANNs trained with NCL using simple data set 4

- Here on the most complicated of the fully-synthetic data sets, the most accurate results come from the ensembles with the fewest hidden nodes.
- The smile shaped effect of varying degrees of diversity inclusion still more noticeable than on the first two sets.

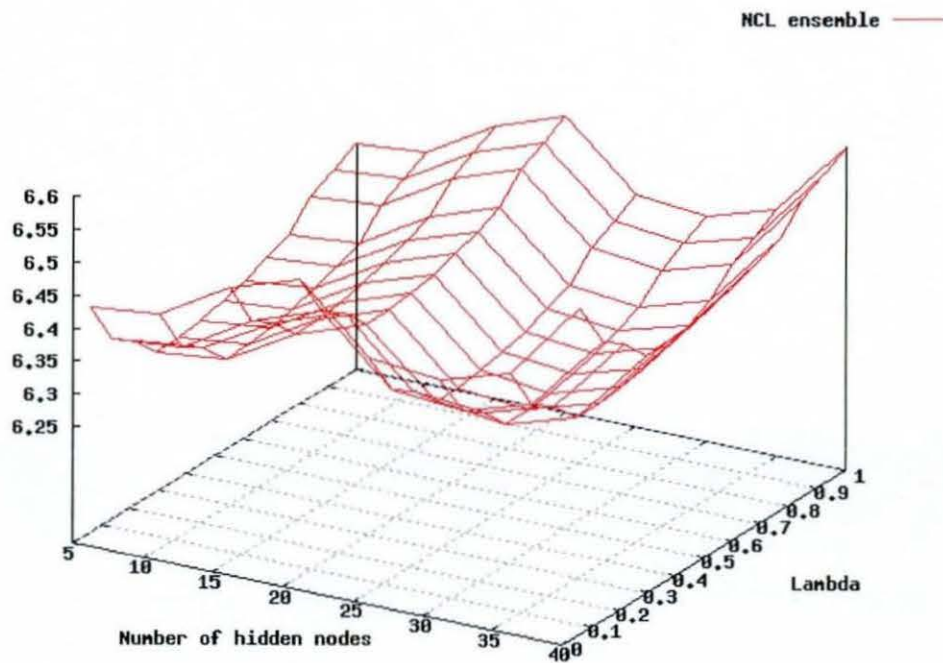


Figure 52 RMS bearing residual for λ between 0.0 and 1.0 and number of hidden nodes per ensemble between 5 and 40 for ensembles consisting of 5 ANNs trained with NCL using semi-synthetic data set

- NCL ensembles are shown to be more accurate than single ANNs.
- Again the lowest number of hidden nodes, and the medium number of hidden nodes give the best accuracy.
- The change in performance from finding the correct number of hidden nodes is far larger than that obtained by varying the amount of diversity introduced.

Data set	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Semi-synthetic
EKF	0.21	3.66	6.53	11.12	7.09
PF	1.74	3.07	5.88	10.12	6.46
ANN	2.16	4.33	9.02	16.15	7.13
Gaussian ANN	3.2	5.97	7.28	13.86	6.41
Uniform ANN	5.1	6.28	7.53	14.09	7.15
Gaussian KNN	4.09	6.5	12.58	16.8	7.71
Uniform KNN	4.56	6.52	9.51	15.26	7.16
NCL ensemble	2.97	4.21	7.03	13.2	6.29

Table 12 The best RMS bearing error for NCL trained ensembles on each data set

	RMS	Entropy	Kohavi Wolpert	Gen'd Diversity	MdRAE	MdAPE	GMRAE
NCL	6.293	0.685	0.122	0.175	5.390	1.148	32.680

Table 13 The best value for NCL trained ensembles on each metric for the semi-synthetic data set

Table 14 shows these RMS bearing error results broken down into the individual folds. Table 15 shows ensembles trained with NCL are more accurate than both the plain ANN and the EKF on all 10 of the folds, while they are more accurate than the PF and the Gaussian Binned ANN on 8 of the ten folds. From table 16 it is possible to see that it can be said that the ensembles are better than the ANN and the EKF with a confidence of 99.90%, and are more accurate than the PF and GANN with a confidence of 94.53%. Therefore the NCL trained ensembles meet this thesis' criterion of a confidence level of greater than 90% in it being superior to the baselines and it is therefore possible to say that the improvement in accuracy over the state-of-the-art provided by this new form of bearing predictor is statistically significant.

Fold	ANN	EKF	PF	GANN	NCL
1	8.13	7.13	6.84	6.59	6.24
2	7.67	7.02	6.36	6.46	6.29
3	7.81	7.1	6.5	6.32	6.27
4	7.64	6.93	6.41	6.35	6.31
5	8.07	6.98	6.78	6.46	6.25
6	7.71	7.01	6.43	6.43	6.3
7	7.37	7.05	6.13	6.33	6.33
8	7.81	6.99	6.66	6.56	6.29
9	7.32	7.2	6.04	6.18	6.29
10	7.72	7.53	6.43	6.42	6.29

Table 14 Results per fold for the ANN, EKF, PF, GANN & NCL

B \ A	ANN	EKF	PF	GANN	NCL
ANN	x	10	10	10	10
EKF	x	x	10	10	10
PF	x	x	x	7	8
GANN	x	x	x	x	8

Table 15 The number of folds for which algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN & NCL

B \ A	ANN	EKF	PF	GANN	NCL
ANN	x	99.90%	99.90%	99.90%	99.90%
EKF	x	x	99.90%	99.90%	99.90%
PF	x	x	x	82.81%	94.53%
GANN	x	x	x	x	94.53%

Table 16 The percentage confidence that algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN & NCL

4.7 Conclusion

NCL has proven to be an effective method for training ensembles to perform target tracking, outperforming both of the baseline techniques; the EKF and PF, and the ANNs produced in chapter 3. These new predictors have been shown to outperform every technique presented so far on every data set. The gains achieved in this application from utilising NCL and ensembles are statistically significant.

This is novel as not only have ensembles of classifiers not previously been used to perform target tracking, but NCL has not previously been used to train a target tracking ensemble.

Although successful, the training stage proved to be very slow, allowing relatively few combinations of parameters to be evaluated in a reasonable time-scale. Further work is required to find a method for creating and training ensembles which is less time consuming and therefore able to perform a more thorough search of the solution space.

One of the drawbacks of NCL is that as a training algorithm it does not have the capability to design the ANNs on which it is applied. A method is required to somehow automatically generate the optimal structure for the network which would provide the maximum level of accuracy, while taking inspiration from the NCL to enhance the ensemble's performance.

5 Genetic algorithms to create ensembles

5.1 Summary

A genetic algorithm is described which can design a whole ensemble; structure and weights. This automates the process of finding the optimal structure, and removes the requirement to train the ensemble. Although the process creates ensembles which are more accurate than the original baselines, and the improved versions developed in chapter 3, it does not manage to create an ensemble capable of outperforming the one created with NCL in chapter 4.

5.2 Introduction

Chapter 4 showed that it was possible to improve upon the earlier results using an ensemble of ANNs and training them with NCL. However due to the time consuming nature of ANNs, especially large ANNs being trained on large data sets, it is not possible to exhaustively test all possible parameters for the structure of the ANNs in the ensemble, or for the training algorithm. A technique was required to speed up this process and find the optimal tracking ensemble.

Genetic algorithms (GAs) can be used to discover a population of Pareto-optimal individuals in a single run [30], solutions for which no other is better in every way. The goal of this section is to further show that classifiers can be used as time series predictors, and identify a method for designing ensembles of predictors to enable them to outperform baseline techniques.

In this chapter a novel method is described for bearing prediction in target tracking using GAs to create ensembles of ANN classifiers, with the goal of creating a bearing predictor for target tracking which outperforms the traditional techniques.

Although the classifier chosen for these experiments is an ANN, the technique is generic to all classifiers and therefore any classifier could have been chosen, ANNs being selected as they were the best performing classifier for the task in previous experiments.

The rest of this chapter is organised as follows: section 5.3 gives a brief literature review, section 5.4 outlines the technique proposed, section 5.5 gives a description of the experiments, section 5.6 outlines the results, and finally section 5.7 provides the conclusions.

5.3 Evolving ANNs in the literature

ANNs are usually trained using example data, using supervised or unsupervised learning. Unsupervised techniques include backpropagation. There are a number of examples in the literature of ANNs which are in part constructed using an evolutionary algorithm. [305] surveys the field, dividing the current work into three broad categories, (evolution of weights, architecture and learning rules), outlined here in sections 5.3.1 to 5.3.3

5.3.1

Weights

Evolutionary algorithms and GAs have been used to discover the optimal weights for ANNs. This is less likely to be trapped in local minima, and more likely to find the global optimal values than gradient descent techniques such as backpropagation. The most important decision to be made is to establish the representation of the connection weights. Weights may be represented by binary strings [291][258][71][146], or by strings of real numbers [232] [302] [242] [267]. In the literature there are many examples of problems on which the performance of EANNs are superior to gradient descent training as they can train recurrent ANNs [293][248][297], fuzzy ANNs [297] [141], and on some problems train ANNs faster than backpropagation [232][233].

5.3.2

Architecture

5.3.2.1 Direct encoding

In direct encoding the chromosome describes the resultant ANN completely, giving an exact one to one mapping from genes to ANN. An $n \times n$ matrix is used to represent an ANN with n nodes, a one at element (i,j) represents a link from node i to node j , while a zero would mean there is none. An example of a matrix and the ANN it represents is given in Figure 53. This scheme can represent feedforward or recurrent networks. If it is known in advance that only feedforward networks are required, only the upper left triangle of the matrix is required. The size of the final ANN must be known in advance in this scheme. Direct encoding has been used by [290][247][195].

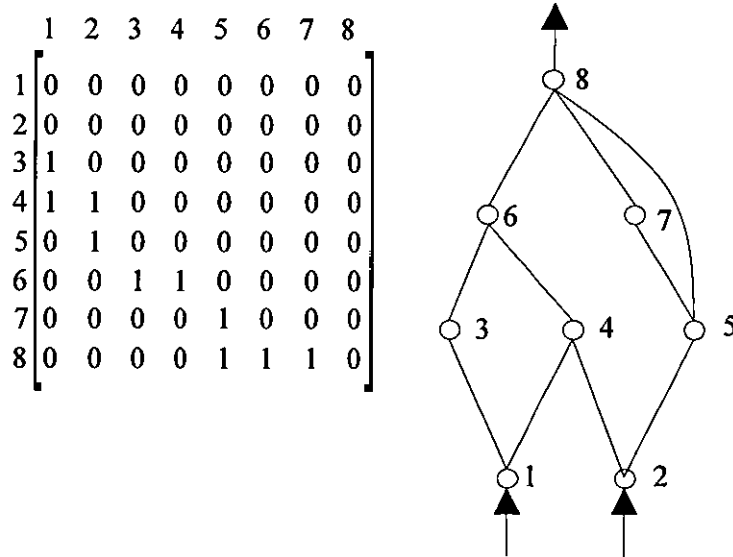


Figure 53 Direct encoding of an ANN structure

5.3.2.2 Indirect encoding

There are a number of different techniques which reduce the size of the search space by shortening the chromosome, only specifying the ANN indirectly [304][281][228][235]. One approach is to create a blueprint for ANN construction by specifying, for example, the number of hidden layers, and the number of nodes in each hidden layer. The chromosome would not completely describe the final ANN in the genes, rules must be established in advance to convert the chromosome to an ANN. For example, the architecture could be assumed to be feed forward and fully connected between layers. This would therefore only search a subset of possible networks. This approach has been used in [69][111].

Another method of indirect encoding is to use development rules [304][281]. Using development rules the resultant network would be fully described. Rather than evolving the network itself, a set of rules are evolved, an example of which can be found in Figure 54. The first three lines of Figure 54 show the evolved rules, while the rest of the figure shows how these rules would be applied to create the direct encoding matrix, resulting in the ANN shown in the bottom right of Figure 54.

$$S \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$A \rightarrow \begin{bmatrix} a & a \\ b & a \end{bmatrix} \quad B \rightarrow \begin{bmatrix} a & a \\ a & a \end{bmatrix} \quad C \rightarrow \begin{bmatrix} c & d \\ a & a \end{bmatrix} \quad D \rightarrow \begin{bmatrix} a & a \\ b & e \end{bmatrix}$$

$$a \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad b \rightarrow \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad c \rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \quad d \rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad e \rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$S \quad \begin{matrix} A & B \\ C & D \end{matrix} \quad \begin{bmatrix} a & a & a & a \\ b & a & a & a \\ c & d & a & a \\ a & a & b & e \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

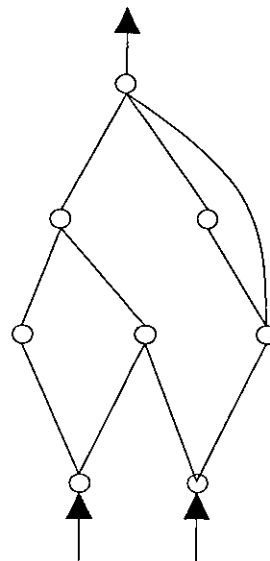


Figure 54 Using development rules to create an ANN

5.3.3

Architecture and weights

One drawback with evolving architectures alone is that the resultant ANNs must be trained before they may be used. The weights discovered during training depend upon the initial state before training, which is usually a set of random values, so there is a one-to-many mapping between chromosome and trained ANNs. As a result of this, each chromosome must be tested a number of times to accurately establish its fitness. Additionally different training algorithms may produce different results, even when the ANN starts with the same connection weights. This can be summarised as the genotype not having a one to one mapping to the phenotype. One way to alleviate this problem is to co-evolve the architecture and the weights. As this leads to a one-to-one mapping of genotype to phenotype, multiple evaluations of each chromosome are unnecessary to

establish fitness accurately. This approach has been widely used in the literature [258] [8] [193] [306] [182] [303] [199] [165] [195].

5.3.4

Evolving ensembles

The idea of evolving ensembles is not new, there are examples in the literature of evolving a population of ANNs, and then once evolution is finished, using the GA population as the ensemble [52][53][54][183]. In [183] the GA consists of a population of ANNs which are trained using NCL, and an evolutionary algorithm which selects individuals from the population to propagate to the next generation with Gaussian mutation

The DIVERse and ACCurate Ensemble Learning Algorithm (DIVACE) [53][54] and DIVACE II [52] algorithms are multi-objective evolutionary approaches to creating ANN ensembles. DIVACE [53][54] uses a multi-objective algorithm in which uses the NCL correlation penalty in addition to accuracy as objectives. As with [183] the GA consists of a population of ANNs which are combined in the last generation to form the ensemble. At each generation of the GA the individual ANNs are trained with backpropagation. DIVACE II [52] differs in that it also uses a variety of other methods for improving the performance of the algorithm including Boosting [92] [251] and Bootstrap Aggregating or Bagging [37].

Although not described as an evolutionary algorithm another algorithm created to iteratively construct ensembles was presented in [142] [142] started with an ensemble of very simple ANNS which was trained with NCL. If this gave errors within an acceptable limit then the algorithm was terminated. If not then for each ANN which is less accurate than a set threshold a new node is added, if none of the ANNs are below the threshold then a new ANN was added to the ensemble, and then the process is repeated.

5.4 Proposal

5.4.1

Original aim

Chapter 4 has shown that classifier ensembles can be used as predictors. In chapter 3, the ANN stood out as the best classifier for the purpose, therefore the ANN was chosen as the classifier to use in further experimentation. In the earlier experiments the ensembles were designed by hand, then trained with NCL. This is a time consuming method of producing ensembles, as many structures must be designed manually, trained and tested before one is chosen for use, a process involving much trial and error.

One problem with the evolutionary approach taken in [52][53][54][183] in which a GA is constructed which contains individual ANNs and the final generation of the GA is used as an ensemble is that although the ANNs have evolved to perform well individually, the evolutionary process does not select based upon the ability of a network to improve the whole ensemble. To resolve these issues, a new technique was proposed. An ensemble would be evolved by representing whole ensembles in each chromosome. A GA would then be run to discover the optimal ensemble. This was intended not to select ANNs individually that might make a good ensemble, but to evolve the best overall ensemble. The individuals were selected as parents for the next generation using the Non-Dominated Sorting Genetic Algorithm II (NSGA II) algorithm [77] a full description of the algorithm used to rank the individual ensemble's distance from the Pareto Front is given in appendix E

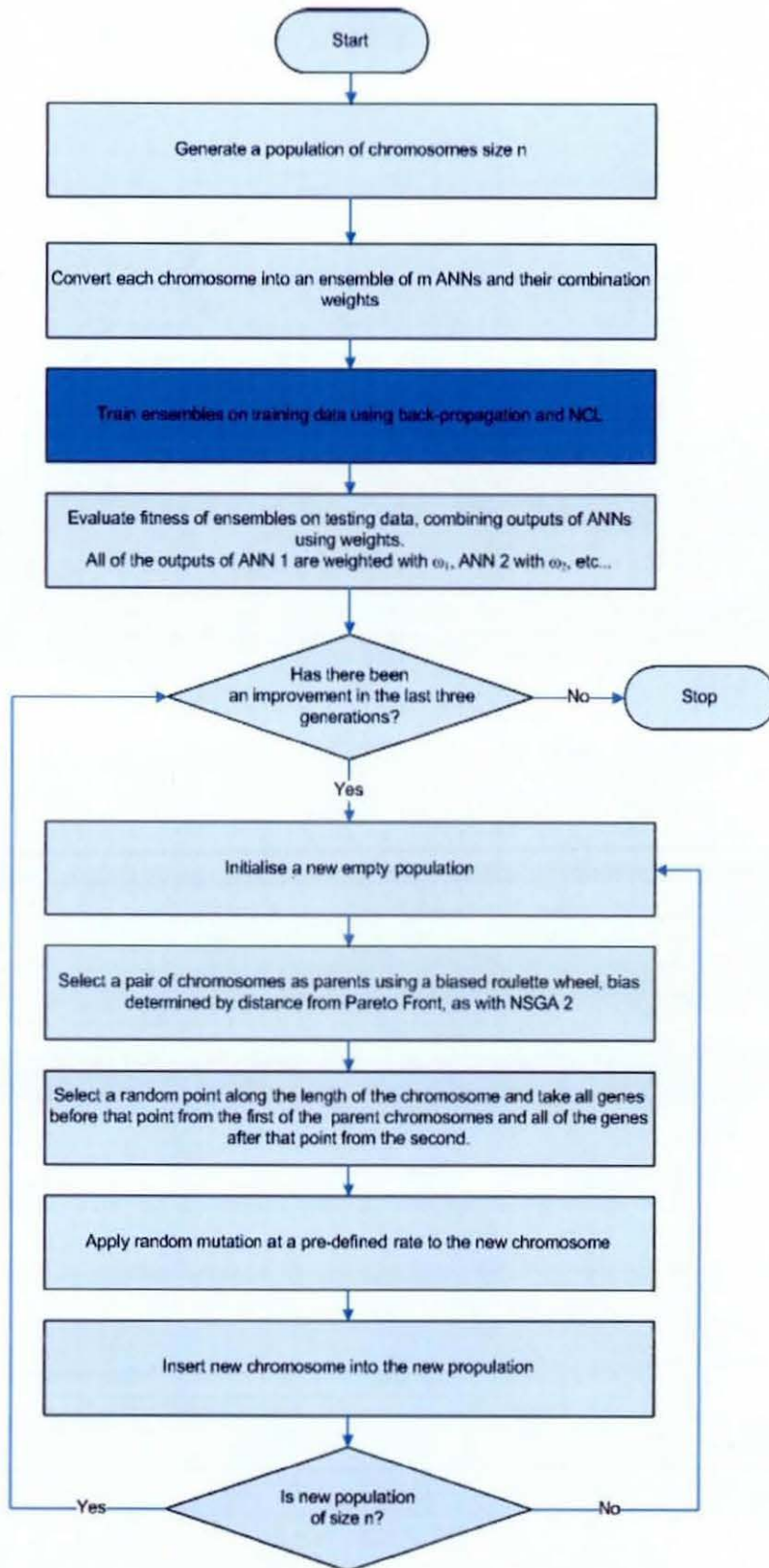


Figure 55 Flow chart of method originally proposed

Figure 55 is a flow chart detailing the steps required to run the experiment. Software was written to conduct this proposed experiment, and experiments began. However it soon became clear that the full run would take several years to complete, which was unfortunately far more time than was available for experimentation, and therefore the experiment was halted before any results were obtained. A solution was required to evaluate and evolve the ensembles faster.

5.4.1.1 Attempted hardware solution to running speed problem

Initially it was thought that the best way to improve the situation might be to leave the algorithm unchanged, and simply use more hardware. There are several distributions of the Linux operating system, including Mosix [319], OpenMosix [316], Chaos [318], Kerrighed [317] and OpenSSI [315], which allow the user to run distributed processing without having to specifically write any code in order to achieve it. The operating system runs across a number of networked computers, automatically distributing processing across the network, while giving the user at one of the computers the impression that they are using a single system. It is this feature that gives them the generic name of Single System Image (SSI) operating systems.

The only requirement on software in order to make use of these features is that any part of the program which should be distributed must be run in a separate process, as it is processes which migrate between the machines to balance the processing loads between them.

One of the SSI distributions was chosen, and the software was rewritten to evaluate each ensemble in a separate process, and write outputs to a file. The main program then parsed the text file to collect the results which it used in the evolutionary process. The experiments were then restarted with six dual processor machines sharing the processing, however even with twelve processors working on the problem it soon became clear that it would still require many months to finish the experiments, which was still longer than the time available. Therefore the experiment was again halted before any results were obtained.

5.4.2 Improved process

It was clear that the process used to this point was not efficient enough to evolve the ensembles as planned, and a new, more efficient technique was required to find the optimal ensemble. There was a clear inefficiency in the existing technique - the training must begin afresh when each new ensemble is tested - and this training is

discarded after the evaluation is complete. Training the ANNs in each ensemble was the most time consuming step of the whole process. With this in mind, the process was altered to co-evolve the network structure and weights. This had three advantages;

1. Training was not repeated for each generation, making the process orders of magnitude faster.
2. The weights were retained between generations, so knowledge acquired by an individual network was less likely to be lost.
3. As computationally expensive training was no longer required, the cost of evaluating the fitness of a chromosome was many orders of magnitude faster.

The technique involves evolving a group of neural networks to predict bearings as an ensemble, as shown in Figure 56. The earlier figure 55 showed the strategy used previously, one difference between figure 55 and figure 56 is highlighted in a darker shade of blue in figure 55 and has been removed in figure 56, while the other difference is highlighted in a darker shade of blue in figure 56. A GA is used to evolve ensembles of neural networks including the weights of the connections within the ANNs, in other words the ANNs are created completely trained. The GA is only required for the training phase of the algorithm. In order to apply a classification algorithm to the time-series prediction problem, the data is converted into a series of input/output sets using a sliding window approach; this was described in section 2.7.6.2.1.

Importantly, as the ANNs are no longer trained in this approach, it is not possible to use NCL to decorrelate the ANN outputs.

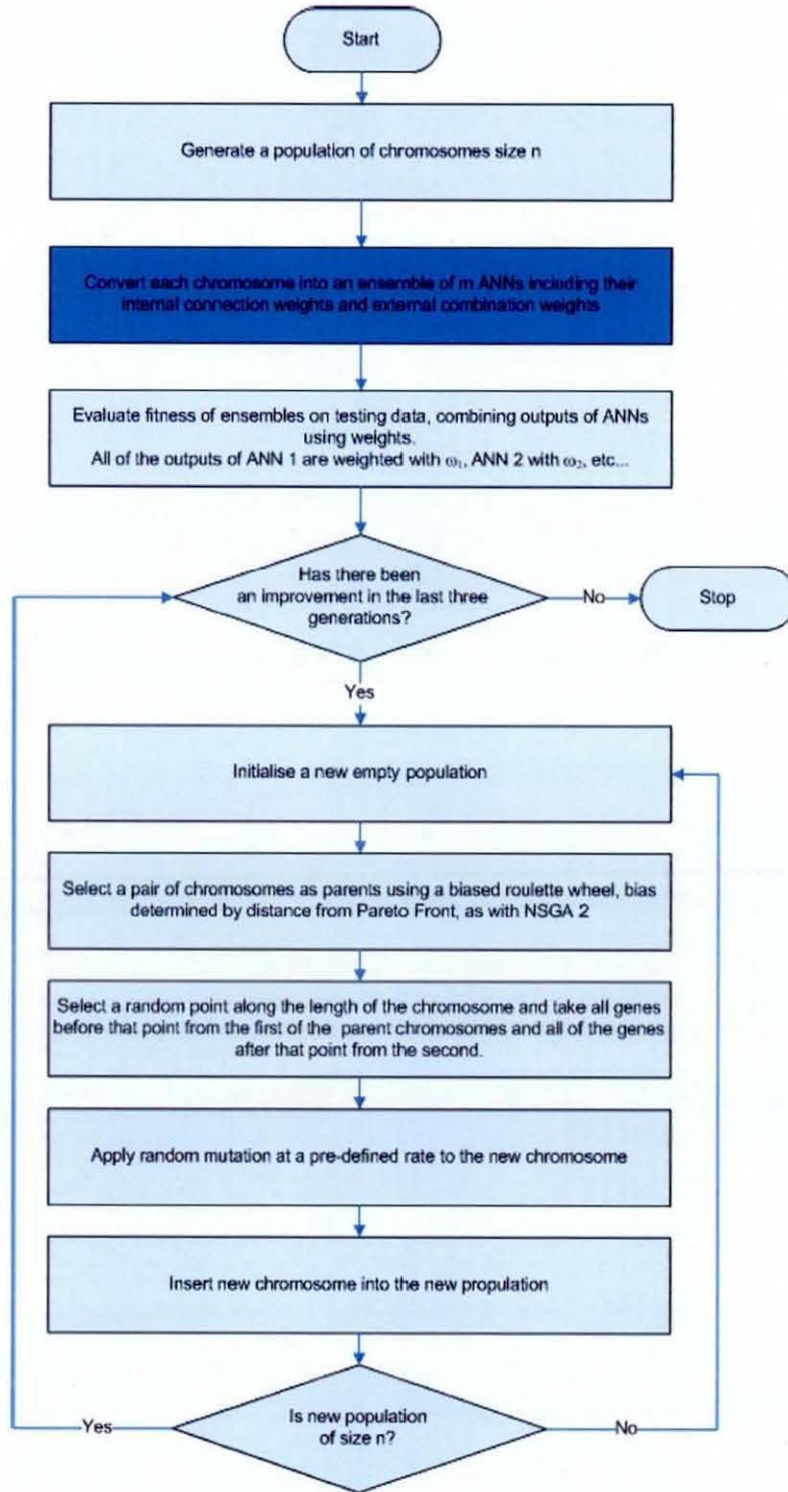


Figure 56 Flow chart of new methodology

Many aspects of this technique are novel; evolving whole ensembles of ANNs (weights and structure) where each individual in the GA represents an entire ensemble, using

ensembles of classifiers as time-series predictors and using both accuracy and diversity to evolve ensembles (rather than populations of ANNs to use as ensembles as used in [53][54] & [182]).

Section 5.4.3 outlines both the ensemble structure, and their component ANNs, while section 5.4.4 describes the Multiple-Objective Genetic Algorithm (MOGA) used to evolve them.

5.4.3 Artificial neural network used

The aim of the work is to produce the best possible ensemble of Artificial Neural Networks (ANNs) for performing target tracking through classification. Each member of each ensemble tested will be a bearing predicting neural network as in [41]. The output of the network is a series of binned probabilities, each representing the likelihood of the next bearing being in a particular direction. These probabilities can then be converted into a bearing using a binning function, as described in section 3.4.1.

5.4.3.1 Ensemble output fusion

Once the required binning function has been applied to each ensemble member in turn to produce a bearing prediction, the outputs are then combined using a weighted average, as shown in Figure 5.7. Although the weights $\omega_1, \omega_2, \dots, \omega_m$ are evolved, and can each be any value in the range [0,1], before they are used it is ensured that they sum to 1;

$$\omega_{i, new} = \frac{\omega_i}{\sum_{j=1}^m \omega_j}$$

5.4.4 Genetic algorithm

The ensemble was constructed using a Genetic Algorithm (GA). Each individual in the GA corresponded to a whole ensemble of predictors. The aim of the experiments was to evolve the most accurate ensemble possible by simultaneously minimising predictor errors and maximising classifier diversity.

5.4.4.1 Selection

To ensure accurate testing of the method, 10 fold cross-validation is used. Evolution is performed 10 times, each time with different data. The first time, the first 10% of the dataset is used as the test data. The next time the data between 10% and 20% is used as test data and so on until the whole data set has been used as test data in one of the runs.

A further 20% of the data is used for validation. The rest is used for training, all of the training being carried out as a by-product of evolution. Table 17 is a table of which data is used for which run.

Run number	Test Data	Validation Data	Training Data
1	1	2,3	All others
2	2	3,4	All others
3	3	4,5	All others
4	4	5,6	All others
5	5	6,7	All others
6	6	7,8	All others
7	7	8,9	All others
8	8	9,10	All others
9	9	10,1	All others
10	10	1,2	All others

Table 17 The tenth of the data set used in each run of the ensemble

The results of each ensemble on the training, validation and testing data are stored. The results from using the training data are used in the genetic algorithm to rank the ensembles. The GA is run until the validation results show no improvements for three generations. The results shown in this thesis are the mean of results from the testing data, in the final generation, across all 10 folds

If this technique was used in a tracker, the GA would be used when the tracker was designed to select the ideal ensemble based upon a sample of training data, which should be large enough to represent the data on which the tracker would be used. One of the resulting ensembles would be used in the tracker and would require no further training. This is an advantage with ANNs as they are slow during the training phase but very fast when used simply to find an output based on a given set of inputs.

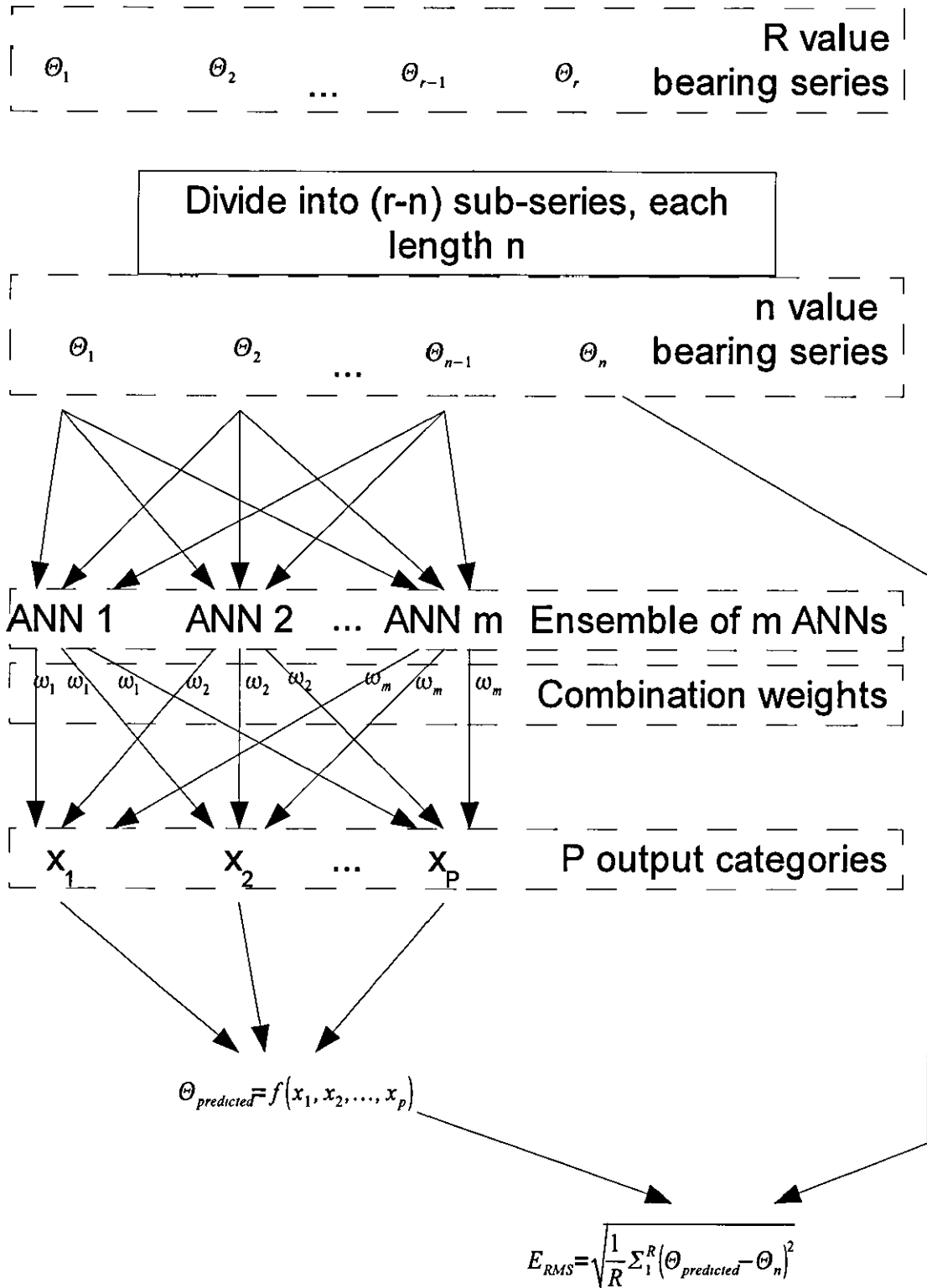


Figure 57 Evaluating ensemble accuracy

Selection for the next generation was based upon the RMS bearing error of each of the ensembles. The weighting used for ensemble 1 in the following generation is

$$w_i = \frac{\max(e) - e_i}{\sum_{j=1}^n (\max(e) - e_j)} \quad \text{where } e \text{ is RMS bearing error and } n \text{ is the population}$$

size, which ensures that fitness is inversely proportional to bearing error

To create a population size of n , n individuals were selected from the previous generation, based on their fitness value. Of these n individuals, 10% are carried through to the next generation unaltered, these are the elite individuals. The other 90% are divided into pairs, and crossed, as described in section 5.4.4.2

5.4.4.2 Crossing

The most commonly used GA method of simulating sexual reproduction is to divide the individuals selected by the roulette wheel into pairs of parents and then combine them to produce the offspring for use in the next generation as described in appendix E. Single point crossover is used to combine the parents, producing two child genomes. Mutation is applied to both of these at a rate of 5%, in other words one in 20 of the bits in the genome is inverted.

5.4.4.3 Genome

Each individual in the GA corresponded to a whole ensemble of predictors. The gene therefore contained a series of individual ANNs (Table 18), with each ANN section of the gene string storing not only all of the information required to create the network; its size, topology, binning function and activation functions but also a boolean which sets whether or not the network will be included in the ensemble, and a value ω which was the weight to use for the network in the weighted average combination (Table 19). This section was repeated M times within the gene string to allow it to represent an ensemble containing up to M classifiers. Using this representation it was possible simultaneously to evolve a whole ensemble of classifiers (without having to pre-set the ensemble size), and ensure the ensemble's diversity.

Table 18 to Table 20 gives the detail of the chromosome of the ensemble. In the experiments the maximum number of ANNs in the ensemble was set to 10, as this was empirically found to be a good balance between performance and time, however the exact number of ANNs used in the ensemble between 1 and the maximum was decided by the GA, as shown in Table 19, the ANN chromosome could turn individual ANNs on or off. The same is true for hidden nodes. Table 20 shows how although there is a

preset maximum to the number of hidden nodes, the GA chooses the number to use in each ANN in the ensemble, adding to the amount of diversity possible

Name	Type
Network 0	ANN chromosome
Network 1	ANN chromosome
...	
Network M	ANN chromosome

Table 18 The ensemble chromosome

Name	Type
ω (The weight used when fusing in ensemble)	float
Use this network	boolean
Input node 0 bias	
Input node 1 bias	
..	
Input node n_1 bias	
Hidden node 0	Hidden node chromosome
Hidden node 1	Hidden node chromosome
...	
Hidden node n_2	Hidden node chromosome
Output node 0 bias	
Output node 1 bias	
..	
Output node n_3 bias	

Table 19 The ANN chromosome

Name	Type
Use this node	Boolean
Weights input	Array of floats with as many values as there are input nodes
Weights output	Array of floats with as many values as there are output nodes
Bias	Float

Table 20 The hidden node chromosome

5.4.4.4 Worked example of ensemble construction

The chromosome described is a binary string which represents all of the parameters of the ANN. This section gives a worked example of a chromosome of the same type used in the experiments, but simplified considerably for clarity.

In this example all floating point numbers are represented as 5 bits and may take values between 0 and 1, Table 21 shows the type conversion used for numbers between binary and floating point. The maximum number of ANNs is three. The maximum number of hidden nodes is restricted to 2, and both the number of input nodes and output nodes are set to 2. Although in this example the last three of these numbers are the same, in practice they can be set to different numbers

Binary value	Decimal	Floating	Binary value	Decimal	Floating
	integer value	point equivalent		integer value	point equivalent
00000	0	0 000	10000	16	0 516
00001	1	0 032	10001	17	0 548
00010	2	0 065	10010	18	0 581
00011	3	0 097	10011	19	0 613
00100	4	0 129	10100	20	0 645
00101	5	0 161	10101	21	0 677
00110	6	0 194	10110	22	0 710
00111	7	0 226	10111	23	0 742
01000	8	0 258	11000	24	0 774
01001	9	0 290	11001	25	0 806
01010	10	0 323	11010	26	0 839
01011	11	0 355	11011	27	0 871
01100	12	0 387	11100	28	0 903
01101	13	0 419	11101	29	0 935
01110	14	0 452	11110	30	0 968
01111	15	0 484	11111	31	1 000

Table 21 Example of binary to floating point conversion (minimum value 0, maximum 1, 5 bits)

A random 204 bit binary string is shown below, while table 22 shows how the binary string is tokenised, parsed and converted to ensemble parameters. Finally Figure 58 shows the ANN ensemble that would result from these values

```
100110000000000000000100010001010011000000000110010000111010000000010
100000000000000001001010110111100000000101100011011000110010000001010
010000000000000100101011011110000000000110001101100011001000000
```

ANN	Component	Subcomponent	Binary	Integer	Fully parsed
			value	value	value

Ann 0	W		10011	19	0 612903226
	Use		0	0	TRUE
	Input bias		00000	0	0 000
			00000	0	0 000
	Hidden node 0	Use	0	0	TRUE
		Weights in	00001	1	0 032
			00010	2	0 065
		Weights out	00101	5	0 161
			00110	6	0 194
	Bias	00000	0	0 000	
	Hidden node 1	Use	0	0	TRUE
		Weights in	00011	3	0 097
			00100	4	0 129
		Weights out	00111	7	0 226
			01000	8	0 258
	Bias	00000	0	0 000	
Ann 1	W		10100	20	0 645
	Use		0	0	TRUE
	Input bias		00000	0	0 000
			00000	0	0 000
	Hidden node 0	Use	0	0	TRUE
		Weights in	01001	9	0 290
			01011	11	0 355
		Weights out	01111	15	0 484
			10000	16	0 516
	Bias	00000	0	0 000	
	Hidden node 1	Use	1	1	FALSE
		Weights in	01100	12	0 387
			01101	13	0 419
		Weights out	10001	17	0 548
			10010	18	0 581
	Bias	00000	0	0 000	
Ann 2	W		10100	20	0 645
	Use		1	1	FALSE
	Input bias		00000	0	0 000
			00000	0	0 000
	Hidden node 0	Use	0	0	TRUE
		Weights in	01001	9	0 290
			01011	11	0 355
		Weights out	01111	15	0 484
			10000	16	0 516
	Bias	00000	0	0 000	
	Hidden node 1	Use	0	0	FALSE
		Weights in	01100	12	0 387
			01101	13	0 419
	Weights out	10001	17	0.548	

			10010	18	0 581
		Bias	00000	0	0 000

Table 22 Example binary chromosome to ensemble parameters

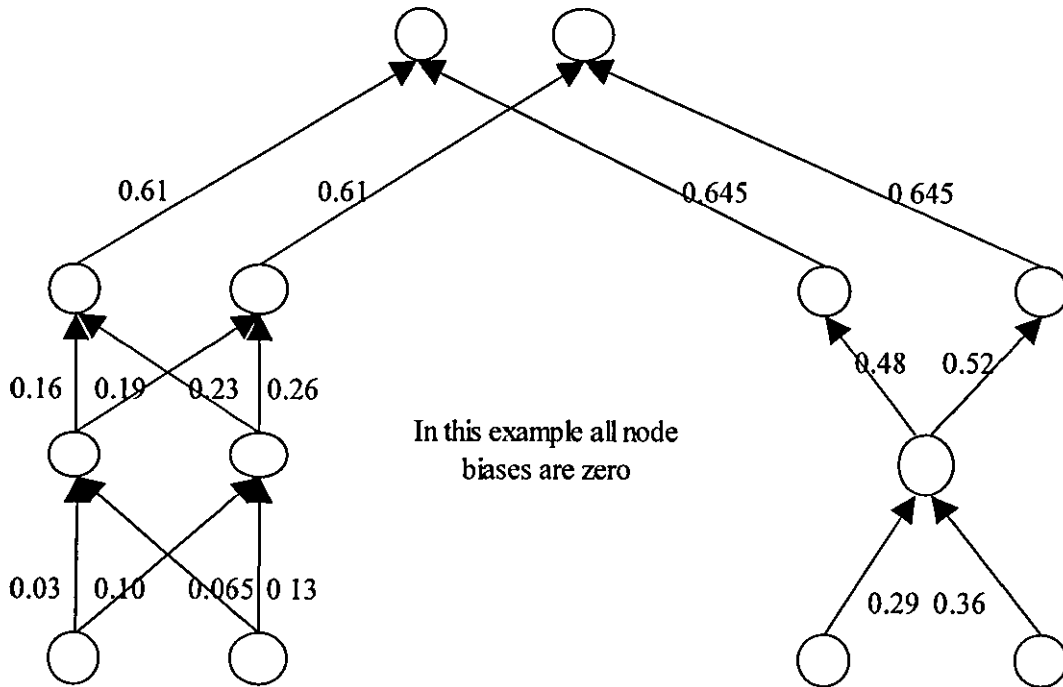


Figure 58 Ensemble consisting of two ANNs produced by parameters in Table 22

In the above example each ensemble was represented by a 192 bit long binary string. In the experiments conducted for this thesis, each ANN in the ensemble had 20 inputs, 10 outputs and up to 40 hidden nodes, each ensemble had up to 10 ANNs, and each floating point number was represented by 8 bits, representing a weight in the range [-2,2] This resulted in a binary string 101220 bits, approximately 98.85 Kb long. Figure 59 shows how the chromosome size would vary with the maximum number of ANNs it could represent, and this can be seen to be a linear relationship and directly proportional..

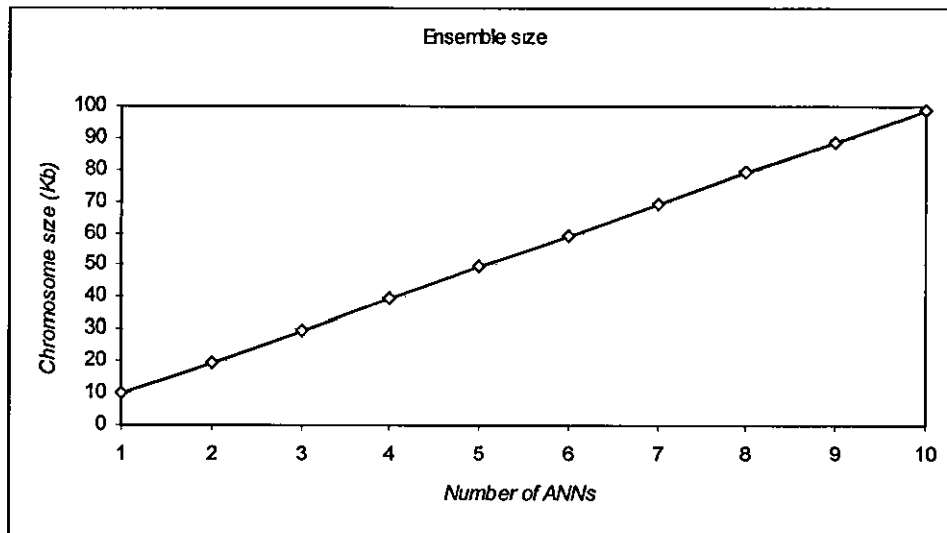


Figure 59 Size of chromosome given different maximum number of ANNs

5.5 Experiments

5.5.1

Objective

The goal of the experiments was to produce the most accurate ensemble of predictors by optimising the overall accuracy of the ensemble, rather than evolving accurate ensembles and combining them. It was expected that this would also perform better than NCL which penalises classifiers which give similar output to the ensemble by giving them a lower weight when the outputs are combined with a weighted average. To minimise computational effort required to explore the vast search space of possible solutions, the weights of the internal connections for the ANNs in the ensemble were co-evolved with the structure of the networks, eliminating the requirement to train networks or ensembles individually, resulting in training being preserved between generations.

5.5.2

Genetic algorithm

The GA was implemented in Java as described in section 5.4.4. Each generation contained 500 ensembles, containing at most 5 ANNs. A fixed number of generations was not used, the GA was run until the GA stopped giving an increased performance for three successive generations. The four measures of diversity along with the four measures of accuracy were calculated at the end of each generation.

5.5.3

Neural network ensembles

The ensembles were constructed as text files in the ANN format used by the Stuttgart Neural Network Simulator (SNNS) [320]. The input file for SNNS was constructed to create an extra layer of input nodes before the input nodes of the ANNs, this layer simply redirected the ensemble input unchanged to each ANN in the ensemble. An extra output layer was used to combine the ANNs using weights $\omega_1, \omega_2, \dots, \omega_n$ taken from the genome. The ensembles were evaluated using the SNNS

5.6 Results

RMS bearing error are given in this section for experiments using both the pure synthetic and semi-synthetic data sets.

5.6.1

Pure synthetic data

In order to compare the new technique against the baselines, a series of pure synthetic data sets were constructed, as described in section 2.7.6.1. These were designed to incrementally test the performance of the algorithms against increasingly 'difficult' data. The short nature of these sets allows the full algorithm output to be displayed here, along with a plot of the error residuals. The baselines used in these experiments are the two best performing baseline techniques in section 2.7, and the NCL trained ensemble developed in section 4; the Particle Filter and the EKF.

5.6.1.1 Experiment 1

The first data set used in testing is a deliberately simple set based upon a sine wave, described in section 2.7.6.1.1.

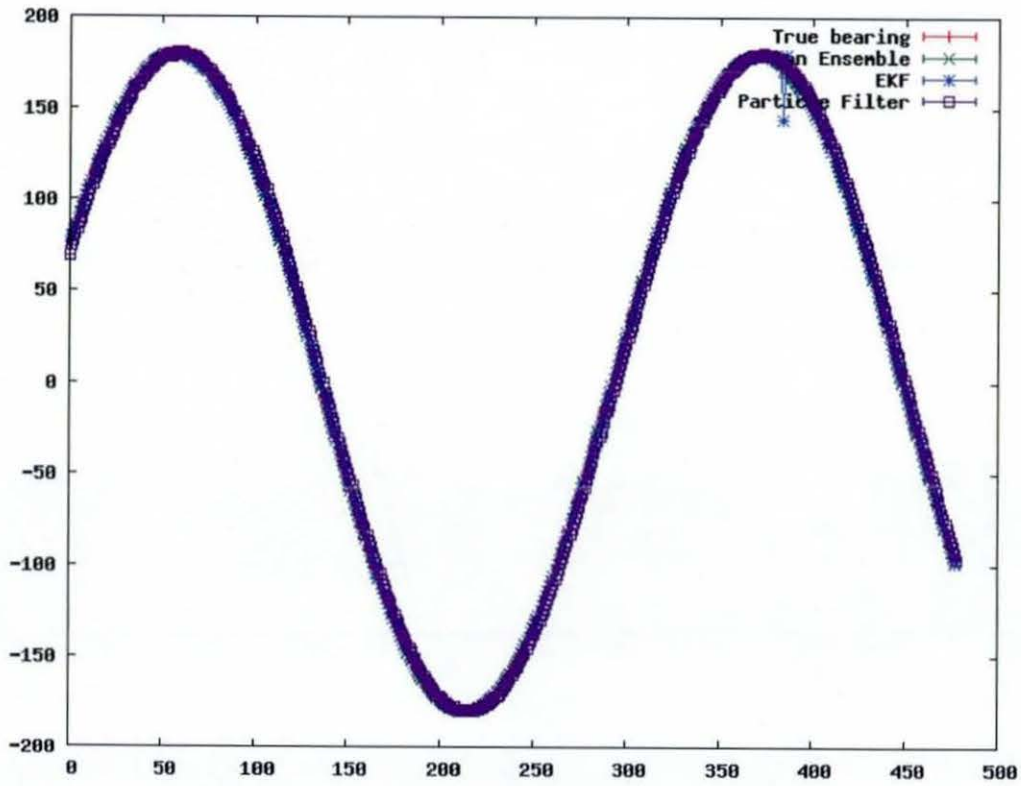


Figure 60 Output bearing values for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 1

- It is impossible to distinguish between the ANN ensemble, the baselines and the truth, showing that the new ensemble tracks relatively well on this simple set, though it is not possible to obtain any stronger conclusions from this plot.

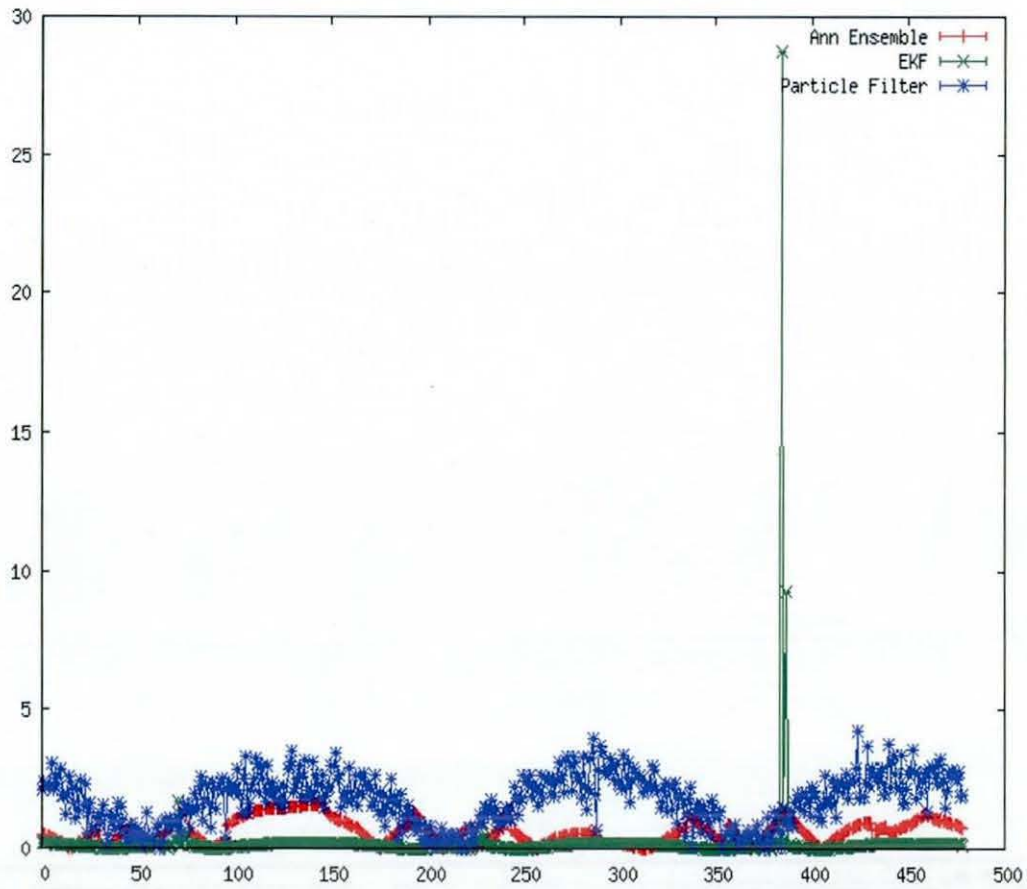


Figure 61 Output bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 1

- Here the performance is clearer to see; The newly created algorithm is more accurate than the PF, though less accurate than the EKF on this set.
- Excluding the outliers, the EKF can be seen to be the most accurate on this very simple data set, however the outliers are significant enough

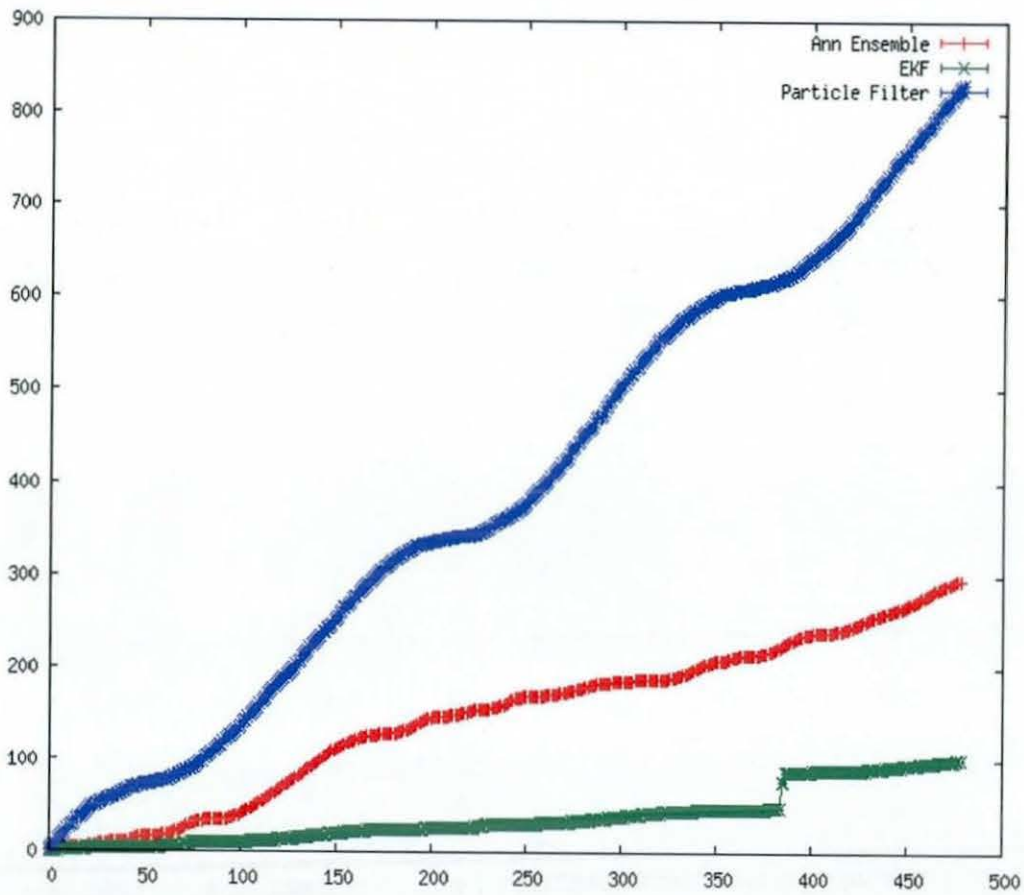


Figure 62 Cumulative bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 1

- Once again the cumulative plot best demonstrates the relative performance.
- Evolved classifier ensembles outperform the Particle filter, however the EKF is by far the most accurate on this very simple data set.

5.6.1.2 Experiment 2

As the first experiment showed that the new algorithm was capable of making bearing predictions, a second data set was constructed which was only a little more complicated. This data was originally shown in section 2.7.6.1.2.

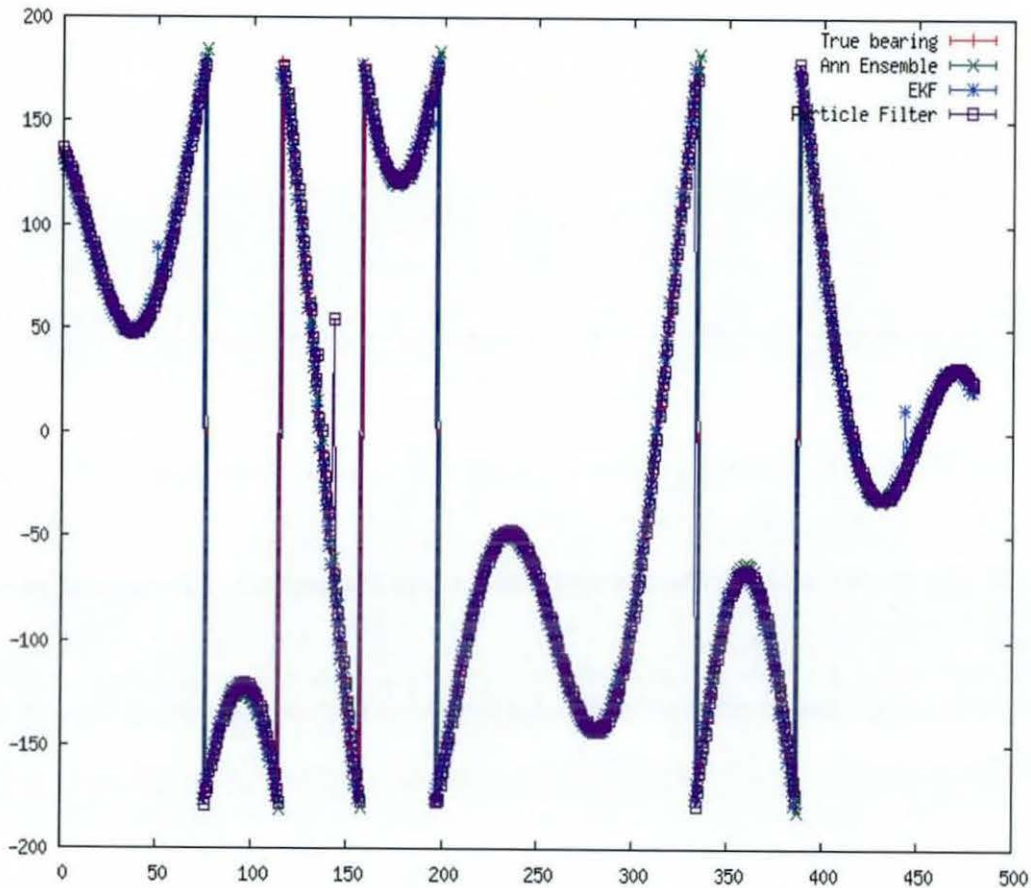


Figure 63 Output bearing values for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 2

- This plot has been included to show the original output of the algorithms to give a picture of the performance, but the differences between each algorithm are too small to draw conclusions.

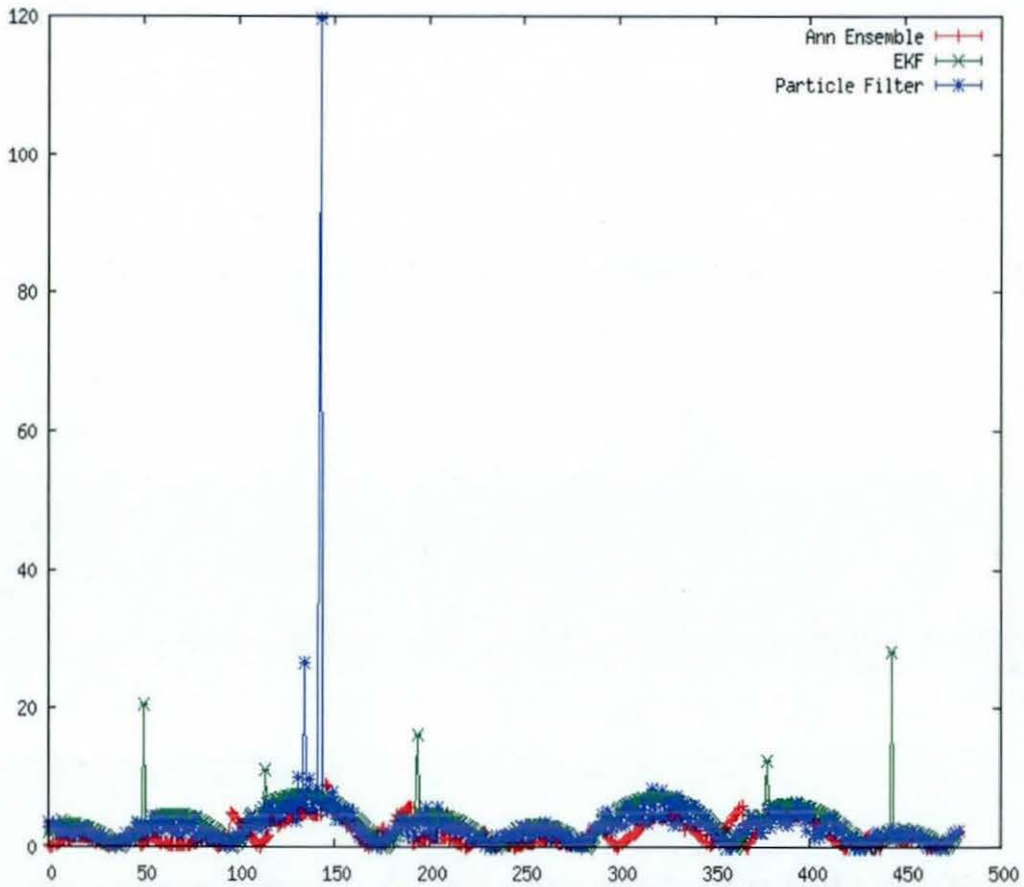


Figure 64 Bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 2

- Here the reliability and consistency of the new ANN ensemble is starting to show.
- The ANN ensemble gives no significant outliers, while the EKF and PF both do.
- Even disregarding the outliers, the performance of the ANN ensemble can be seen to be superior.

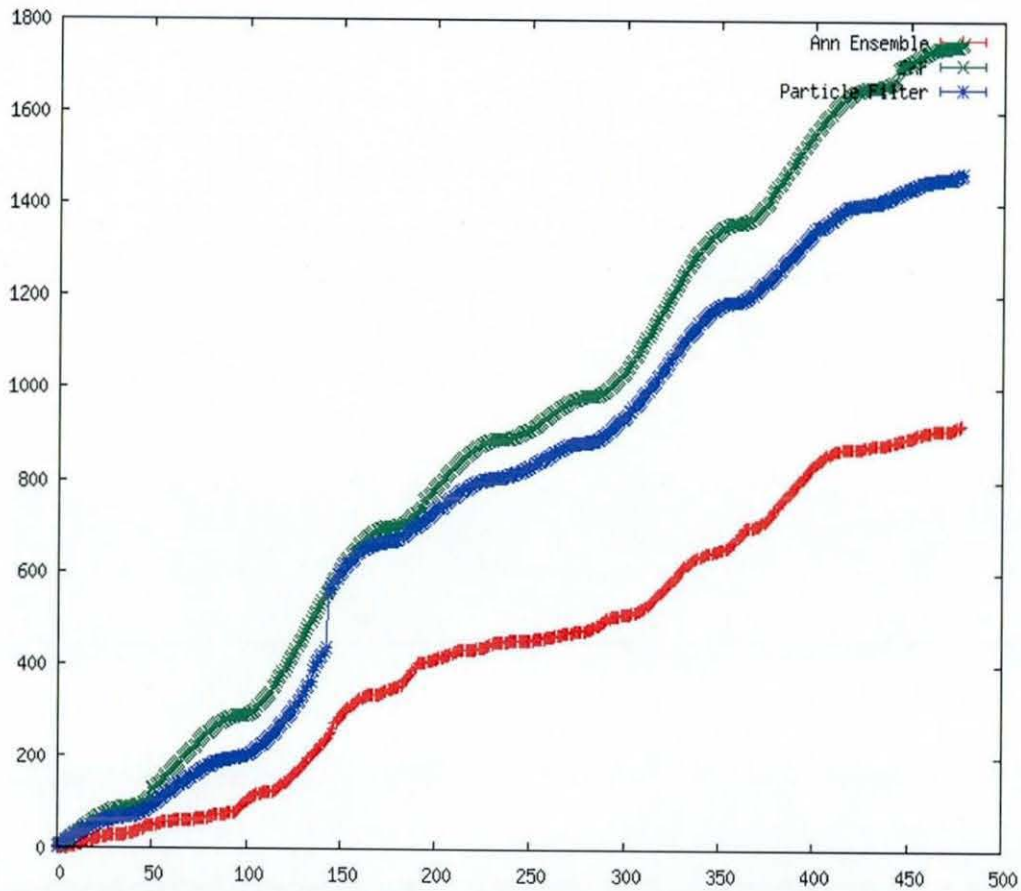


Figure 65 Cumulative bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 2

- With only a small increase in complexity the ANN ensemble has become the most accurate algorithm for tracking the measurements
- The numbers of outliers in the Kalman Filter have increased, meaning the probability of the Kalman Filter diverging on the data set has increased.
- The Particle Filter's ability to track in highly non-linear situations allows it to be more accurate than the EKF.

5.6.1.3 Experiment 3

A third data set was then tested which added a small amount of Gaussian noise to the measurements, as described in section 2.7.6.1.3.

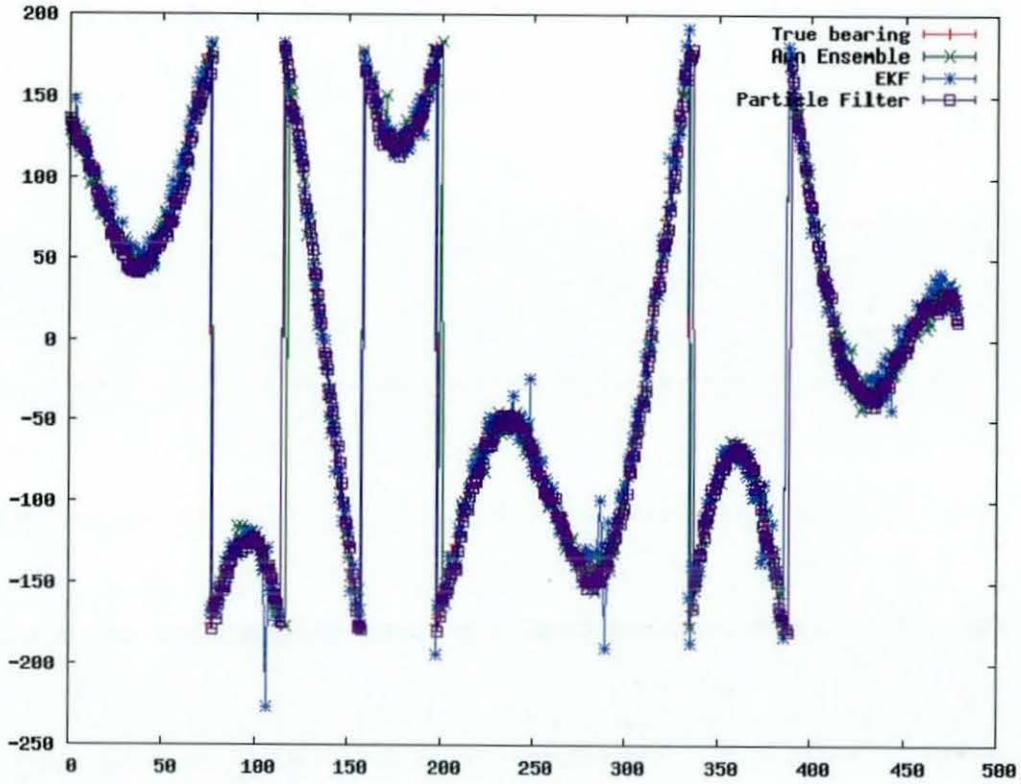


Figure 66 Output bearing values for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 3

- This plot has been included to show the original output of the algorithms to give a picture of the performance, but the differences between each algorithm are too small to draw conclusions.

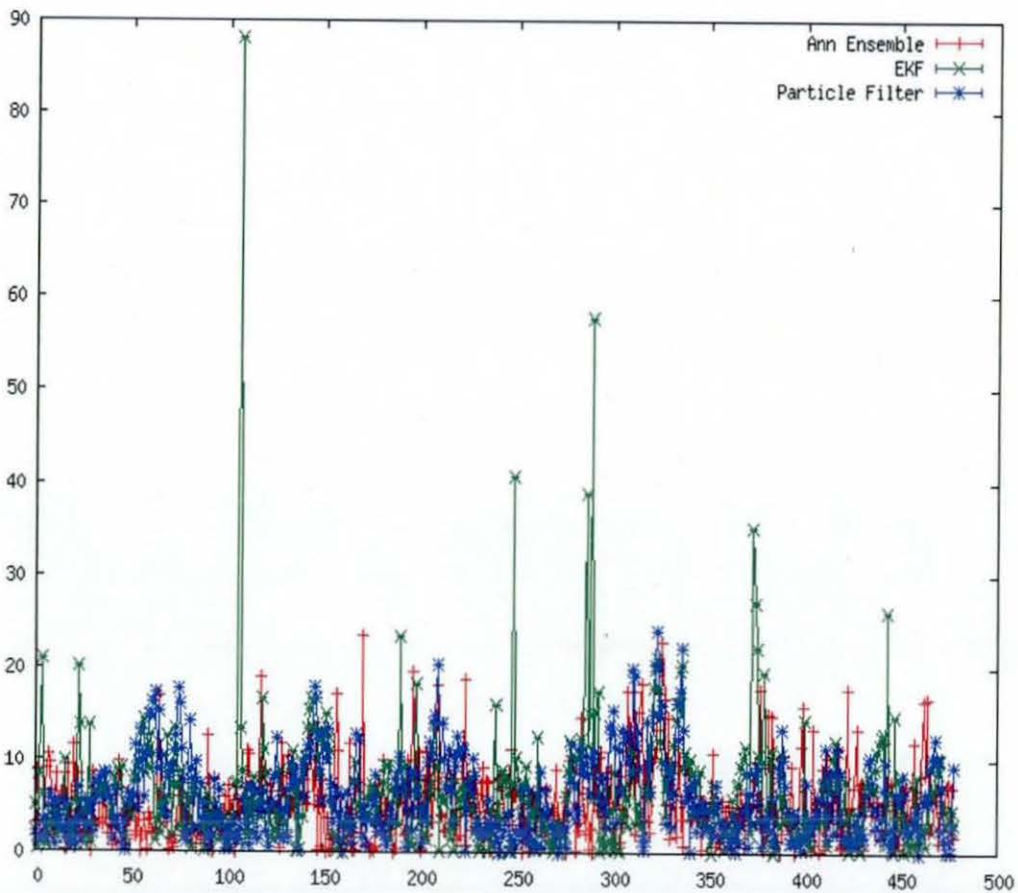


Figure 67 Bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 3

- The high level of clutter in this chart, caused by the random noise added to the data set makes it hard to draw conclusions from this chart.
- However, the numbers of outliers in the Kalman Filter have once again increased, meaning the probability of the Kalman Filter diverging on the data set has now considerably increased.
- The ANN is creating some small outliers, however only a similar number to the PF.

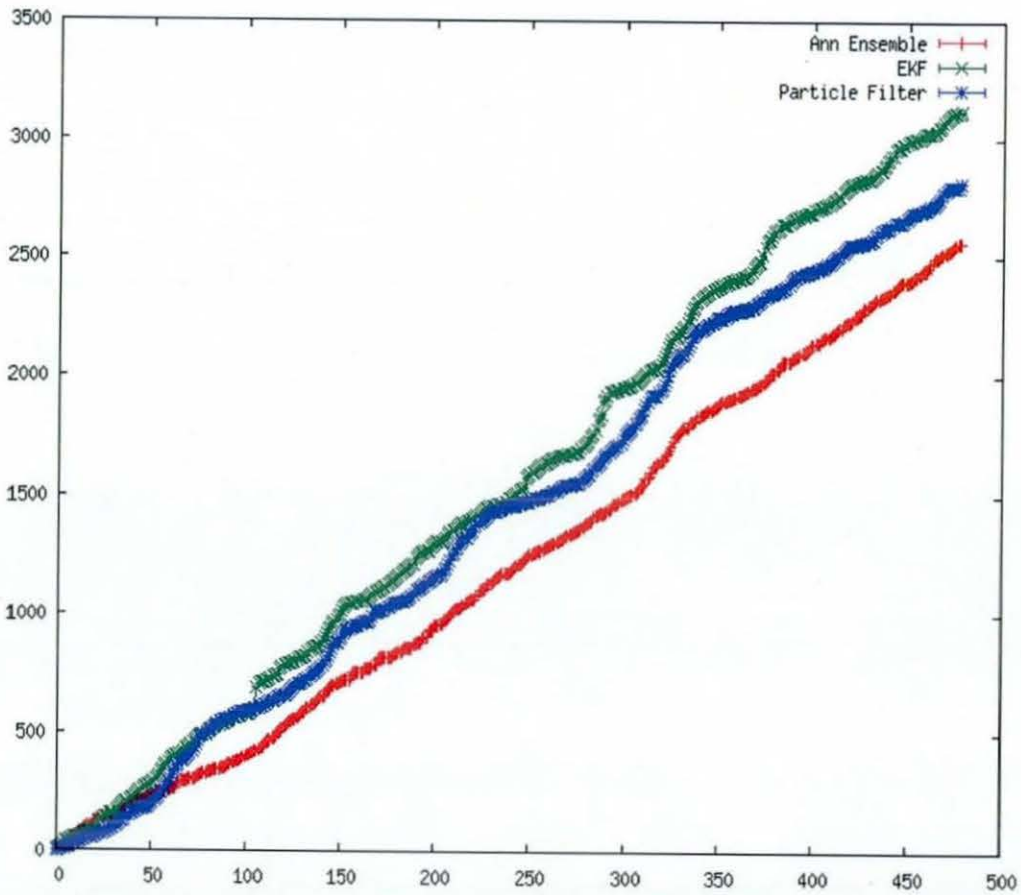


Figure 68 Cumulative bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 3

- Particle Filter and ANN ensemble performance is similar for a small amount of noise
- The ANN ensemble is once again the most accurate on this data set.

5.6.1.4 Experiment 4

The fourth data set increased the level of noise to match the upper limit of the bearing binning function, as described in section 2.7.6.1.4.

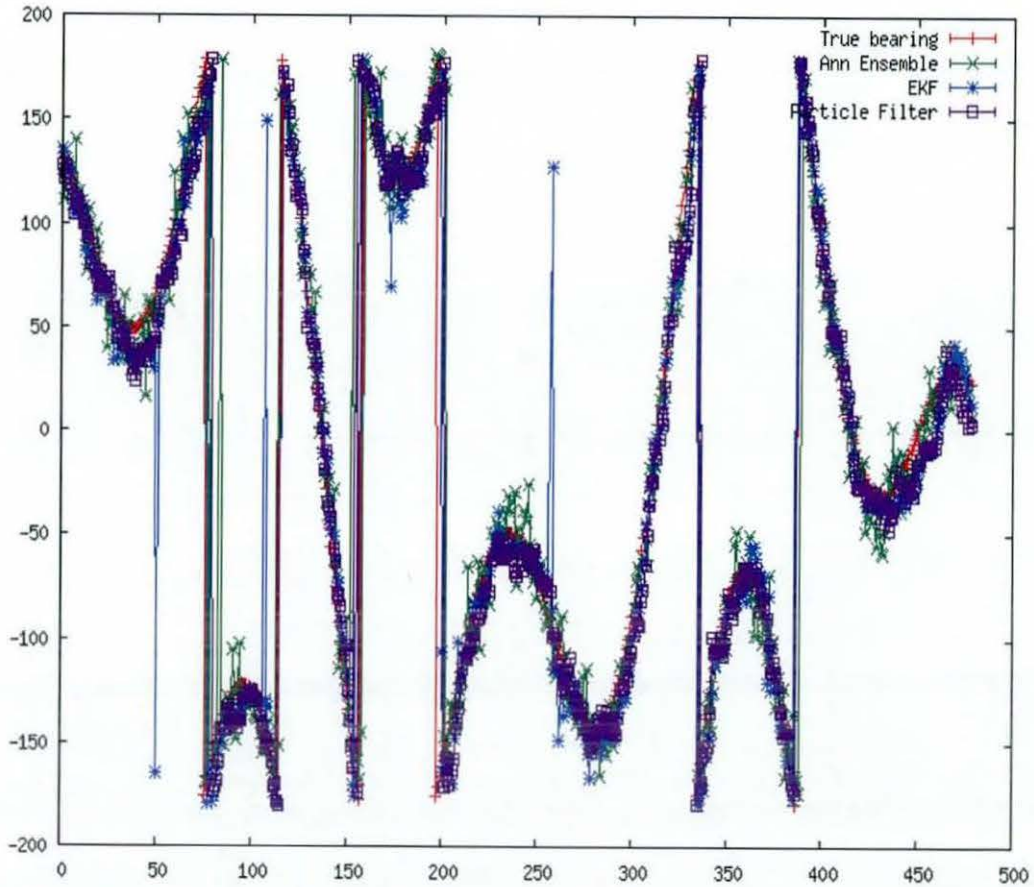


Figure 69 Output bearing values for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 4

- This plot has been included to show the original output of the algorithms to give a picture of the performance, but the differences between each algorithm are too small to draw conclusions.

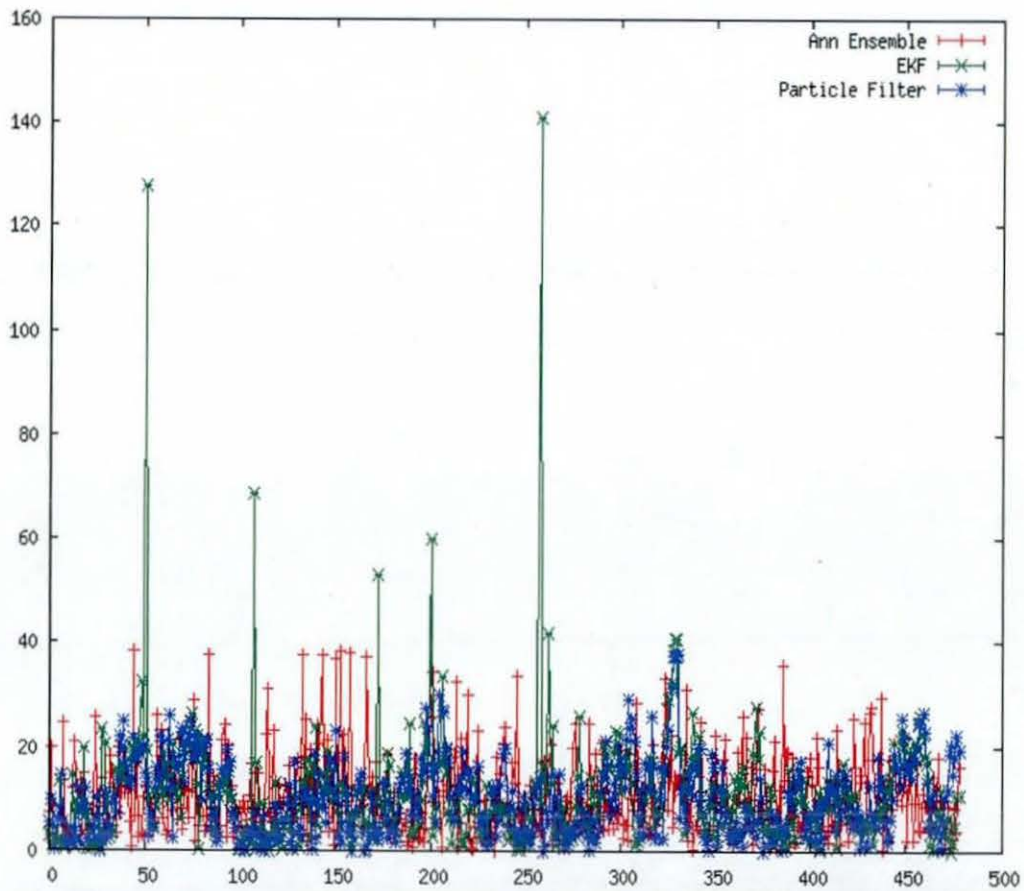


Figure 70 Bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 4

- As before, the clutter in this chart makes it difficult to draw strong conclusions.
- However, again the EKF produces large outliers.

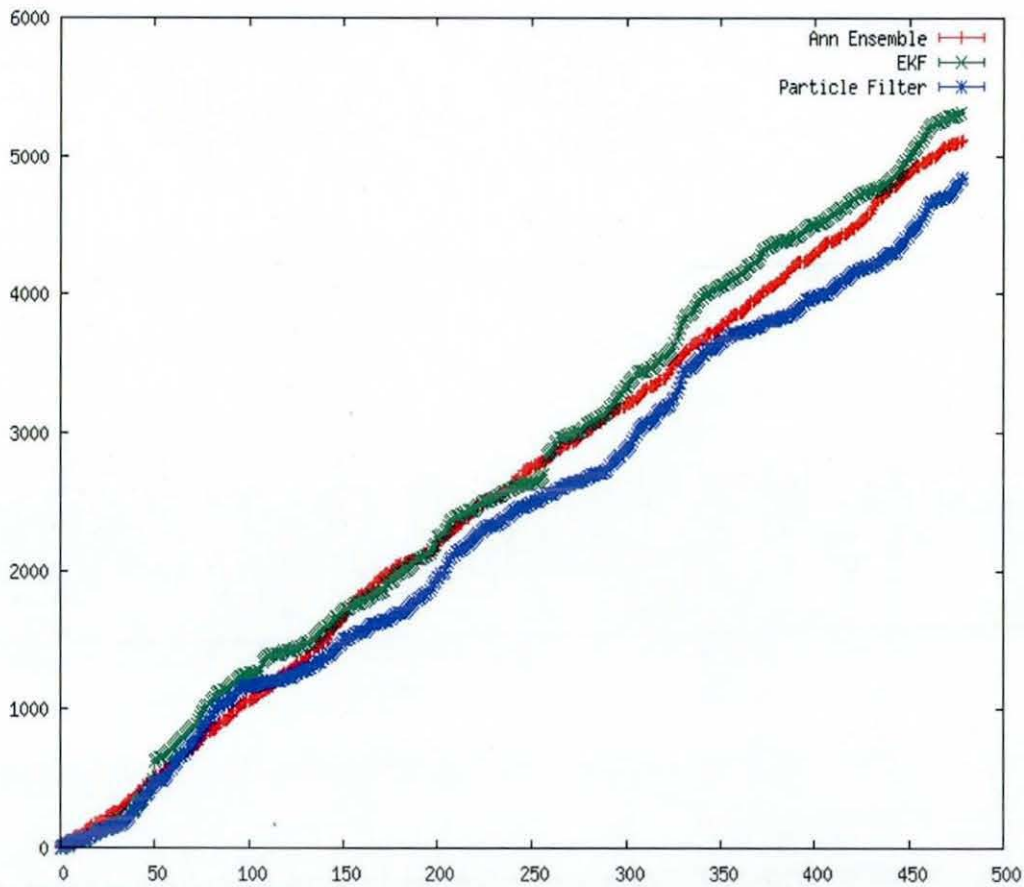


Figure 71 Cumulative bearing residuals for best ensemble created with GA, the EKF and the PF along with the true (expected) value for every data point in pure synthetic data set 4

- Figure 71 shows the performance of the new algorithm at its limit. At this point its performance begins to degrade.
- The ANN ensemble is marginally outperformed by the particle filter, however it still outperforms the EKF.

5.6.1.5 Conclusions

Only on pure synthetic set 2 did the EKF come close matching the performance of the ensembles in terms of RMS error, while the new algorithm outperforms the particle filter on all of the data sets. On data sets 2 and 3 the new technique was found to be the most accurate predictor in terms of RMS error.

Although these experiments have shown that on purely synthetic data the new algorithm is extremely effective, a more conclusive test would be to test it against a more realistic data set, with the difficulties, such as truly random noise, that this would present.

On this fully synthetic data the results clearly show that the new, ensemble based predictor is more accurate than both of the baselines for all of the data tested. Figure 72 shows that the newly created ensemble based predictor is more accurate than any of the baselines tested.

All of the data used here has uncertainties with a standard deviation of 13 degrees or less, which is within the maximum range set for the binning function here. This limit can be set to an arbitrarily large value, though this would either reduce precision or force an increase in the number of bins. Therefore data with noise of arbitrary levels of noise could be tracked with no tuning other than to set the maximum level of noise expected in the binning function.

5.6.2

Semi-synthetic data set

This section details the results of the experiments for which simulated data was overlaid on real recorded noise, described in section 2.7.6.2. For each of the following figures, the best individual ensemble was selected from each generation of the MOGA for the best line, and the mean result of all individuals in the MOGA population is shown as the mean line. As the ten folds are independent, the mean of the ten folds at each generation is shown in Figure 72 to Figure 79. A population of 500 ensembles in each generation was used, running for up to 50 generations.

The single ANN used here for comparison is the Gaussian binning ANN developed in chapter 3.

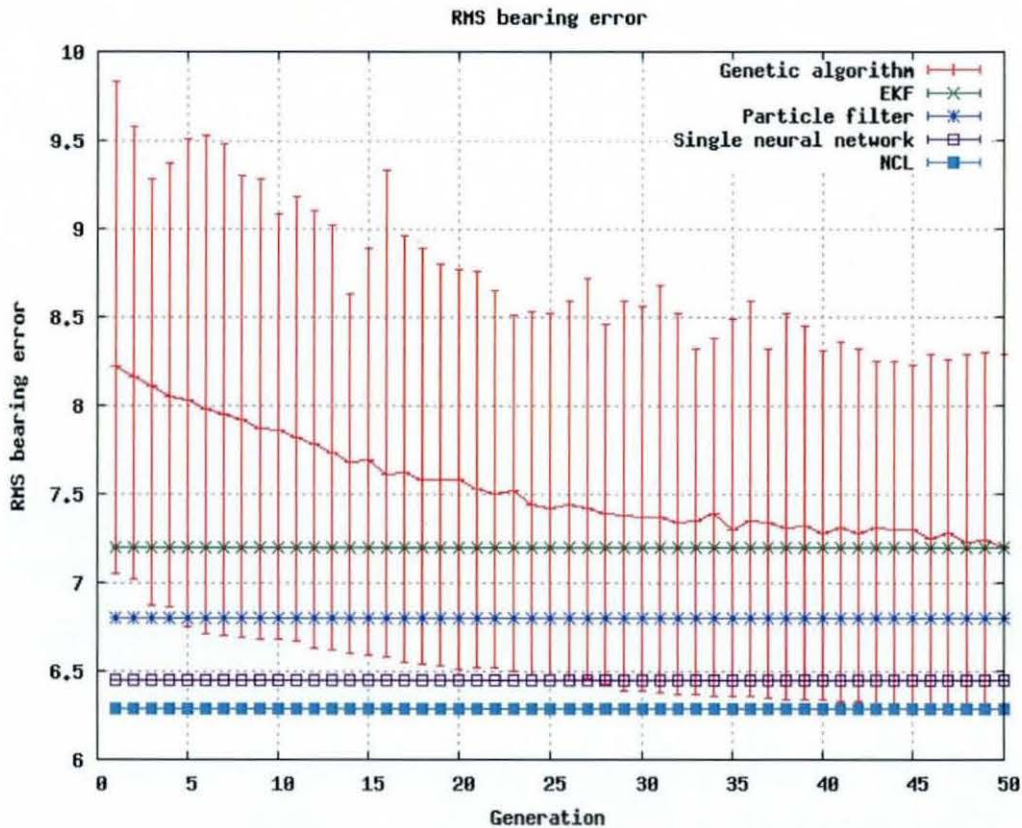


Figure 72 RMS bearing residuals of the ensembles in the GA population compared to the EKF, PF, Gaussian Binned ANN and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

- The error bars show the range of performance in the population.
- The line shows the mean performance of the GA population.
- The lowest end of the error bar is the individual ensemble in the population with the lowest error.
- Although the best ensemble in the GA population is more accurate than both the baselines and the predictors produced in chapter 3, it was no more accurate than the NCL trained predictor produced in chapter 4, only approaching the same level of performance in the last few generations.

Table 23 shows these results for RMS bearing error broken down into the individual folds, alongside the results for the NCL trained ensembles, the GANN, and the baseline algorithms from section 2.7. Table 24 shows ensembles created with a GA are more accurate than both the plain ANN and the EKF on all 10 of the folds, they are more

accurate than the PF and the Gaussian Binned ANN on 6 of the ten folds and more accurate than the NCL trained ensembles on only 5 out of the 10 folds. With the NCL ensembles being more accurate than the GA five times out of ten and the GA created ensembles being more accurate than the NCL the other five times, and from the fact that the mean performance is almost identical, it would appear that the algorithms are almost exactly matched in performance. From table 25 it is possible to see that it can be said that the ensembles are better than the ANN and the EKF with a confidence of 99.90%, more accurate than the PF and GANN with a confidence of 62.30%. There is only a 37.70% confidence in the new GA designed ensembles being an improvement on the NCL trained ones. It would appear that in this case the benefits obtained from being able to change the topology of the internal ANNs is countered almost equally by the reduction in performance from not using a backpropagation based learning algorithm.

Fold	ANN	EKF	PF	GANN	NCL	GA
1	8.13	7.13	6.84	6.59	6.25	5.98
2	7.67	7.02	6.36	6.46	6.29	6.77
3	7.81	7.10	6.50	6.32	6.28	6.67
4	7.64	6.93	6.41	6.35	6.32	6.09
5	8.07	6.98	6.78	6.46	6.26	6.18
6	7.71	7.01	6.43	6.43	6.31	6.21
7	7.37	7.05	6.13	6.33	6.34	6.39
8	7.81	6.99	6.66	6.56	6.29	6.33
9	7.32	7.20	6.04	6.18	6.29	6.38
10	7.72	7.53	6.43	6.42	6.29	6.00

Table 23 Results per fold for the ANN, EKF, PF, GANN, NCL & GA

B \ A	ANN	EKF	PF	GANN	NCL	GA
ANN	x	10	10	10	10	10
EKF	x	x	10	10	10	10
PF	x	x	x	7	8	6
GANN	x	x	x	x	8	6
NCL	x	x	x	x	x	5
GA	x	x	x	x	x	x

Table 24 The number of folds for which algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN, NCL & GA

B \ A	ANN	EKF	PF	GANN	NCL	GA
ANN	x	99.90%	99.90%	99.90%	99.90%	99.90%
EKF	x	x	99.90%	99.90%	99.90%	99.90%
PF	x	x	x	82.81%	94.53%	62.30%
GANN	x	x	x	x	94.53%	62.30%
NCL	x	x	x	x	x	37.70%

GA x x x x x x

Table 25 The percentage confidence that algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN, NCL & GA

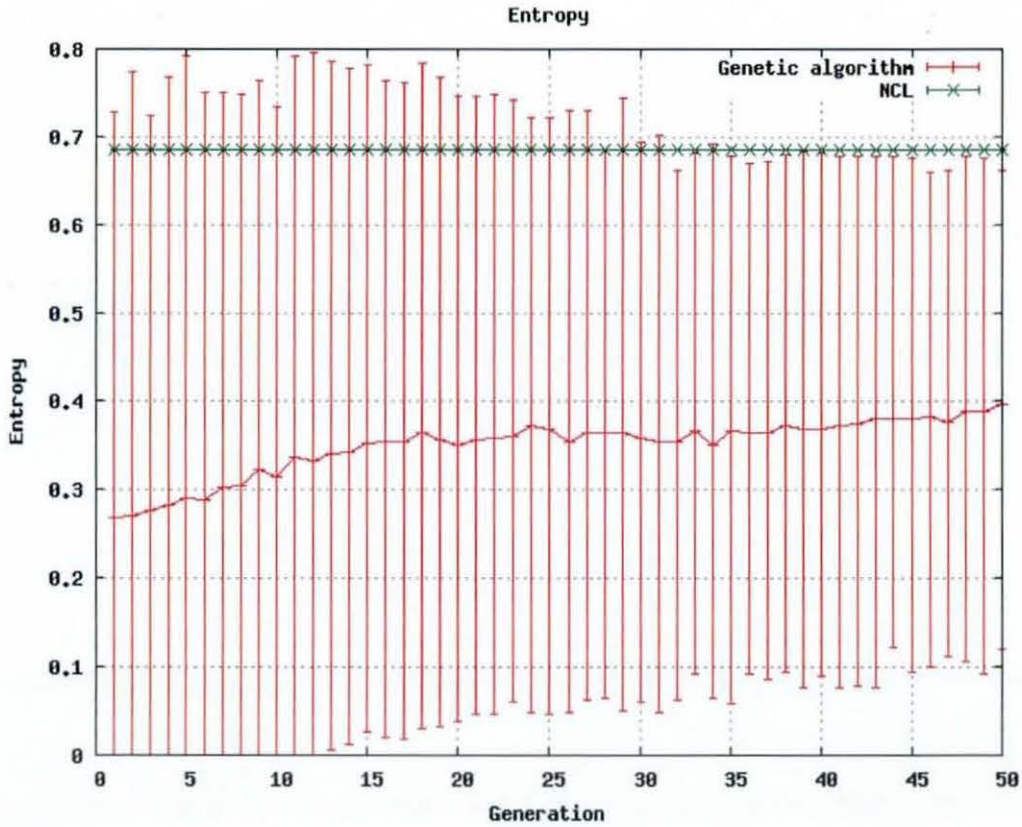


Figure 73 Entropy of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

- Entropy is a measure of diversity, therefore higher values are preferred
- Although at the beginning of the run the most diverse ensemble in the GA has a higher entropy value than the NCL produced ensembles, by the end of the run the most diverse individuals have less entropy than the NCL trained ensembles.
- As the GA is selecting based on RMS bearing error only, performance in terms of entropy is degraded throughout the run.

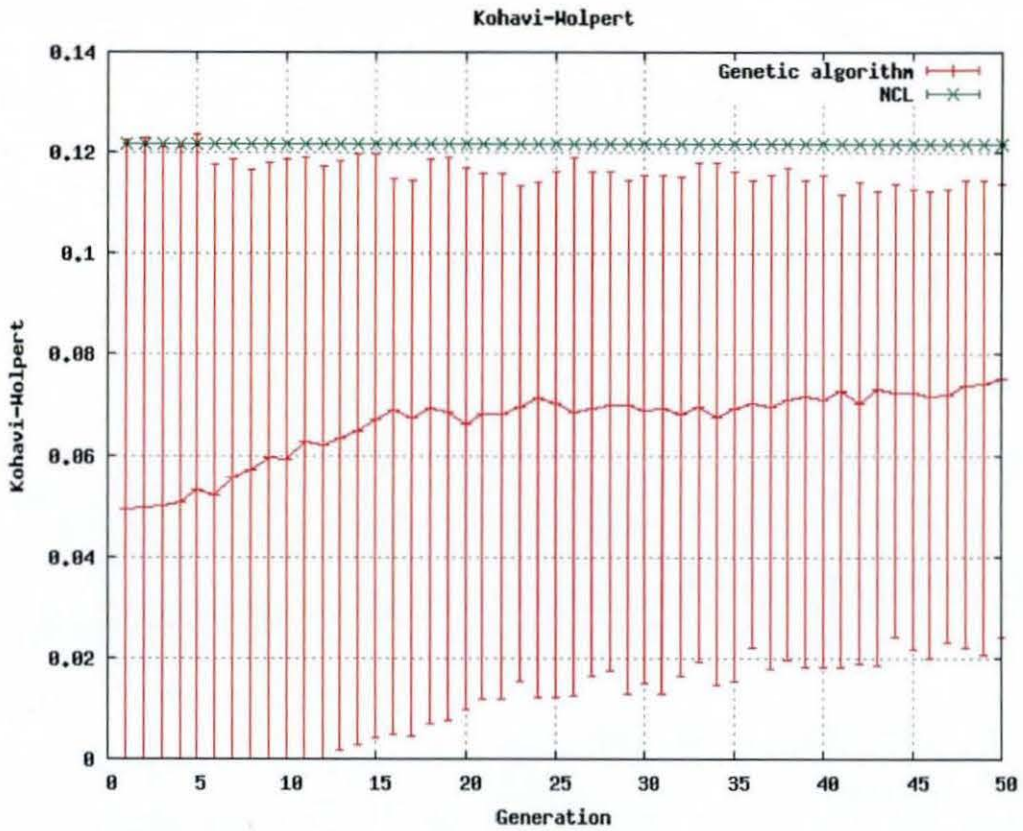


Figure 74 Kohavi-Wolpert diversity of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

- Kohavi-Wolpert (KW) is a measure of diversity, therefore higher values are preferred
- Although at the very beginning of the run the most diverse ensemble in the GA has a higher KW value than the NCL produced ensembles, by the end of the run the most diverse individuals have a lower KW value than the NCL trained ensembles.
- As the GA is selecting based on RMS bearing error only, performance in terms of KW is degraded throughout the run, as was the case with entropy.

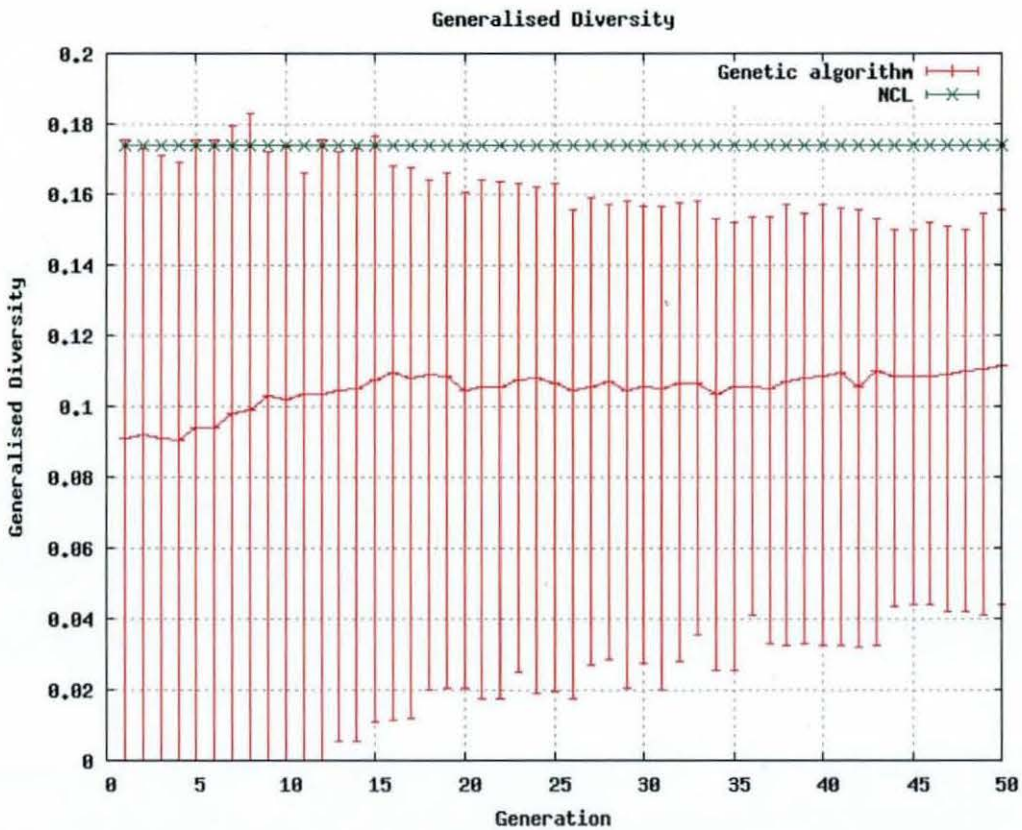


Figure 75 Generalised diversity of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

- Generalised Diversity (GD) is a measure of diversity, therefore higher values are preferred.
- Although towards the beginning of the run the most diverse ensemble in the GA has a higher GD value than the NCL produced ensembles, by the end of the run the most diverse individuals have a lower GD value than the NCL trained ensembles.
- As the GA is selecting based on RMS bearing error only, performance in terms of GD is degraded throughout the run, as was the case with entropy and KW.

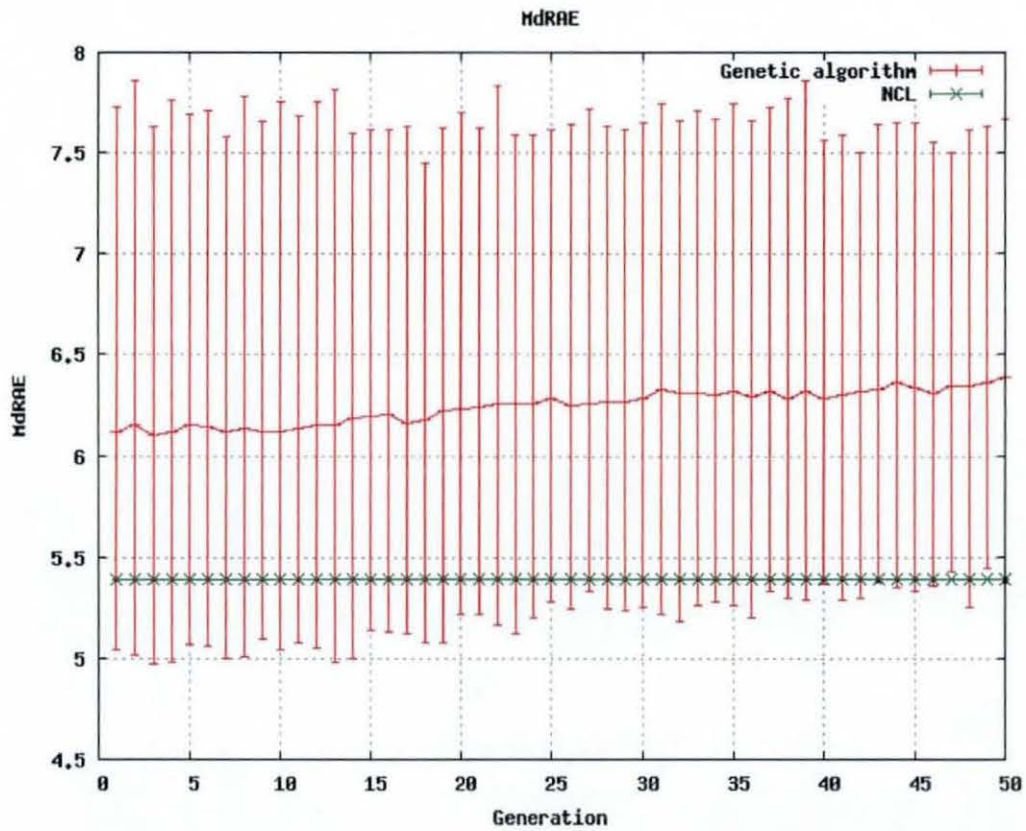


Figure 76 MdRAE of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

- According to this measure of accuracy, the performance of the algorithm actually degrades throughout the run.
- At the start of the run the best ensemble in the GA population outperforms the NCL trained ensemble. However, towards the end of the run this situation reverses.

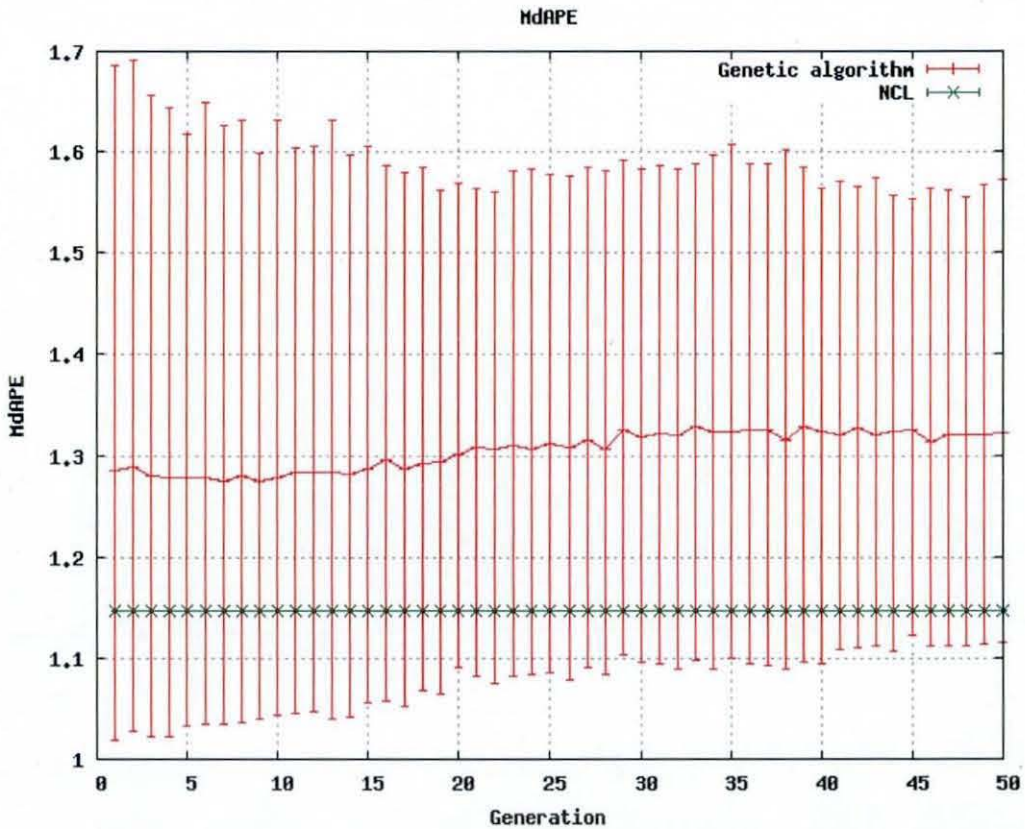


Figure 77 MdAPE of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

- From the beginning of the run the best ensemble in the GA population is more accurate by this metric than the NCL trained ensemble.
- Performance of the GA degrades as the GA progresses.
- As the GA performed better than the NCL on RMS bearing error (which is a the non-relative metric), and worse on MdAPE which is a relative metric, it would suggest that the GA created ensembles are worse when the value to be predicted is small, but better when the value to be predicted is larger.

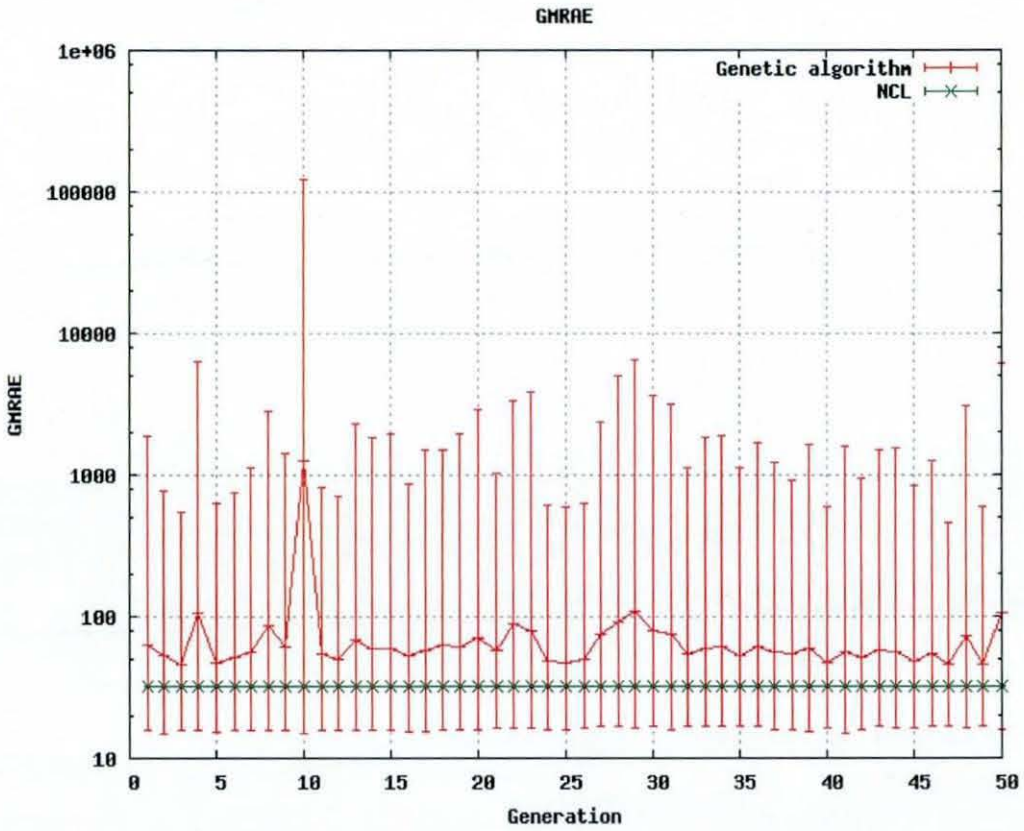


Figure 78 GMRAE of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

- Here the best GA produced ensemble can be seen to outperform the NCL trained ensemble.
- There is little noticeable improvement in performance in terms of GMRAE throughout the GA run.

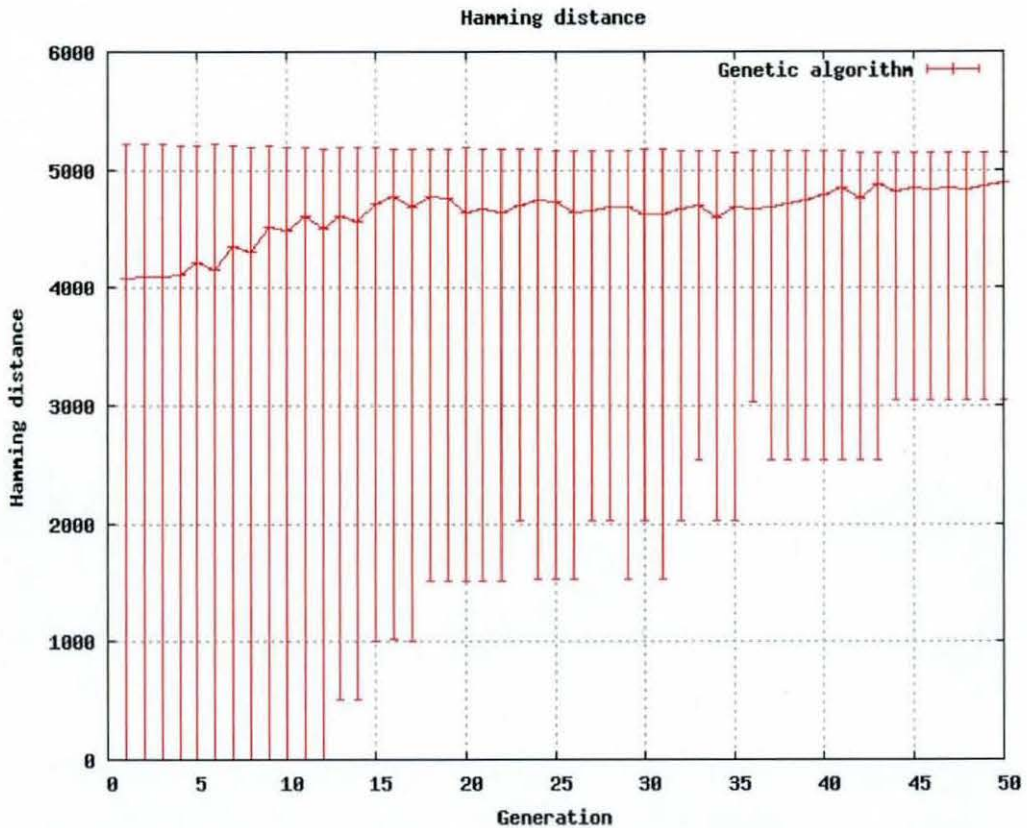


Figure 79 Hamming distance of the ensembles in the GA population compared to the NCL trained ensemble predictor tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

- As Hamming Distance is calculated here from the chromosome, it is impossible to calculate it for the NCL to compare.
- The chart shows the mean level of genotypic diversity increasing throughout the run.
- At the top end, the level of diversity in the most diverse individuals does not change significantly.
- This would suggest that even when selecting purely based upon accuracy, the most genotypically diverse ensembles perform better and are therefore selected for reproduction.

Data set	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Semi-synthetic
EKF	0.21	3.66	6.53	11.12	7.09
PF	1.74	3.07	5.88	10.12	6.46
ANN	2.16	4.33	9.02	16.15	7.13
Gaussian ANN	3.2	5.97	7.28	13.86	6.41
Uniform ANN	5.1	6.28	7.53	14.09	7.15
Gaussian KNN	4.09	6.5	12.58	16.8	7.71
Uniform KNN	4.56	6.52	9.51	15.26	7.16
NCL ensemble	2.97	4.21	7.03	13.2	6.29
GA ensemble	3.15	4.8	7.23	13.5	6.3

Table 26 The best RMS bearing error for GA generated ensembles on each data set

	RMS	Entropy	Kohavi Wolpert	Gen'd Diversity	MdRAE	MdAPE	GMRAE	Hamming distance
NCL	6.293	0.685	0.122	0.175	5.390	1.148	32.680	0.000
GA	7.300	0.768	0.123	0.160	4.970	1.080	16.180	5186.000

Table 27 The best value for GA generated ensembles on each metric on the semi-synthetic data set

5.7 Conclusions

A new form of bearing predictor has been discovered which outperforms the baseline techniques, the Extended Kalman Filter (EKF) and the Particle Filter (PF) on the most realistic data set.

Although the technique is successful in that it improves on the results from chapter 3, the new method of creating ensembles does not create ensembles more accurate than those trained with NCL as in chapter 4, although the performance is almost as good.

Here training the ensemble with NCL is shown to be better than the GA at both maximising diversity and minimising error. This would suggest firstly that as described in the literature, NCL is a very good technique for training ensembles to be both diverse and accurate. However it also shows that there is scope to improve the GA, and incorporating the idea of increasing diversity as used in NCL might be a way of improving performance of the GA.

The most novel aspect of this technique is that it evolves whole ensembles of ANNs (weights and structure) where each element in the GA population represents an entire

ensemble (unlike [52][53][54] & [182] where it is the whole GA population which forms a single ensemble)

6 Multi-objective GA

6.1 Summary

The previous section demonstrated a new method for automatically constructing and training an ensemble in one step through a GA. The resultant ensemble was not as accurate as one trained with NCL which uses diversity to improve the ensemble during training. This chapter expands the GA presented in the previous chapter to add diversity as a secondary objective of the GA. The results are an improvement upon those previously obtained from a GA, but still under-perform when compared to the NCL trained ensemble.

6.2 Introduction

Section 5 has shown that GAs were able to create ensembles for target tracking, but overall performance was not as good as training the ensemble with NCL. A view widely held in the literature is that increasing diversity in an ensemble will increase the ensemble's fitness. NCL does this by reducing the weighting of individual ANNs in the ensemble which do not increase the level of diversity in the ensemble. This section attempts to increase the diversity in the GA created ensemble by using diversity as a second objective in the genetic algorithm. As in [60], a multi-objective GA is adopted, using diversity and accuracy as objectives.

To utilise the benefits to create an ensemble of classifiers it is important to keep the classifiers in the population diverse. Premature convergence is defined as a situation in which the GA has reached a suboptimal state in which most of the genetic operators are unable to produce offspring of superior fitness to their parents [91]. It has unfortunately been shown that using standard GAs the probability of premature convergence is 100% [174]. Premature convergence is often caused by a loss of diversity in the population, as diversity is lost, the ability of the GA to search the entire search space is restricted.

One important addition to these experiments is that the GAs were evolved with multiple objectives (RMS bearing error for accuracy and Entropy for diversity). It was expected that the least accurate would be those evolved using only diversity as an objective, that those evolved with accuracy alone would be more accurate, but that the most accurate would be those trained with both diversity and accuracy as objectives.

A classifier is a function which accepts a set of inputs and assigns it to one of a number of classes. The inputs to the classifier are a subset of the features of the object to be classified. The two main types of computer classifier are supervised and unsupervised. When using supervised classifiers the operator defines the classes, while with unsupervised classifiers the system attempts to establish the structure of the input data. The latter is most useful when the classes are not known to the operator.

It would be reasonable to expect that if a group of classifiers were independently created then for easy data sets the classifiers would all produce the correct answer, while for a more difficult set of features the classifiers would produce different answers. Unfortunately independently produced classifiers often produce coincident errors. In order to ensure that the classifiers discovered produce non-coincident errors the diversity must be artificially increased in some way.

It should be expected that a diverse group of classifiers would be able to adapt to a wider range of situations, allowing them to be able to correctly classify data sets which were not present in the training set. This idea can anecdotally be considered to be similar to creating a consensus opinion of experts from a range of backgrounds rather than trusting the opinion of a group of experts with identical skills. Unfortunately as intuitive as the idea of using a diverse population of classifiers is, the benefits not yet been definitively proven, as many of the papers which claim to have done so do not measure diversity to ensure it is responsible for observed performance gains.

6.3 Previous work on diversity in GAs

After several generations of a GA it is inevitable that all of the individuals in a population will be very similar to each other [174]. Ideally the result of the GA will be that all of the individuals are grouped around the optimal answer, however if the individuals have converged in this way around a suboptimal solution this is known as premature convergence. Premature convergence is defined as a situation in which the GA has reached a suboptimal state in which the algorithms cannot produce a new generation which is a significant improvement over its parents [91]. In the case of wanting to find diverse individuals, as is the situation in this study, any form of convergence is premature. The rest of section 6.3 details approaches that have been taken in the literature to promote diversity and prevent premature convergence.

6.3.1 Genotype or phenotype diversity

There has been some debate in the literature as to whether measuring genotypic or phenotypic diversity leads to the best results. In measuring genotypic diversity the genes themselves are compared, while in phenotypic diversity measurement the decoded values from the gene are compared, in neither of the schemes are the true behaviour of the individual examined [301]. [76]demonstrated experimentally that that phenotypic fitness sharing consistently outperformed genotypic fitness sharing [173]concludes that genotypic measures are used most when increasing diversity, and although using the phenotype instead improves overall diversity more, it does not increase the diversity between the most fit individuals

Other measures that have been proposed include entropy [279]and chromosomal distance [216]

6.3.2 Restriction of selection procedure

The earliest example of changes to the GA introduced specifically to encourage diversity is the crowding operator [73], in which newly generated individuals replaced the individuals in the population most genotypically similar to themselves This therefore prevented 'crowds' of similar individuals forming.

More recently [301]used a similar scheme to increase diversity in a genetic programming context. In this scheme the population was divided into groups of similarly performing individuals, in each of these groups all but one was deleted, the one chosen to remain being picked arbitrarily. This leads to reduction of crowding at optima, and was found to increase diversity.

6.3.3 Restriction of mating

There are two opposing ways of restricting the pairs chosen to mate. The first, described in [203] in which genotypically similar parents are chosen to breed to create 'subspecies' [76]within the population, with the intention of creating niches implicitly. Associative mating algorithms described by [35] restrict crossover to phenotypically similar individuals. Although these techniques form subpopulations, they are unable to maintain them and convergence is not ultimately prevented [254]. The opposite approach was taken by [86], who prevented similar parents from mating in order to

prevent subspecies from forming, and ensure that the entire population remained diverse.

Another method of restricted mating is the parallel genetic algorithm. In this the population is divided into distinct subpopulations, each then finding a different niche. This method has been widely investigated [66][101][229][234][289]. To prevent these subpopulations from converging prematurely a few individuals are allowed to migrate from one subpopulation to another in each generation, however even a small amount of migration between subpopulations may cause premature convergence [254].

6.3.4 Altering mutation rate and population size

[257] created an Adaptive GA (AGA) which dynamically altered the probabilities of crossover and mutation to improve performance, maintain diversity and prevent premature convergence. In this scheme fit individuals remained unaltered, while the less fit were subject to higher mutation and an increased probability of crossover. However [174] found that increasing the population size had a more beneficial effect on diversity than altering the mutation rate.

6.3.5 Fitness sharing

[99] introduced the idea of similar individuals sharing fitness locally. The proximity of solutions is defined in both the gene space and the decoded gene space. The function used for sharing fitness is:

$$\varphi(d_{ij}) = \begin{cases} \frac{1 - \left(\frac{d_{ij}}{\sigma_{sh}}\right)^\alpha}{\sigma_{sh}}, & \text{if } d_{ij} < \sigma_{sh} \\ 0, & \text{otherwise} \end{cases}$$

where d_{ij} is a measure of the distance, σ_{sh} is the sharing parameter that governs the degree of sharing. Typically a value of 1 is used for α . Using this value the new fitness can be calculated as:

$$f_{si} = \frac{f_i}{\sum_{j=1}^M \varphi(d_{ij})}$$

where M is the number of individuals in the neighbourhood of the i^{th} individual.

[76] expanded upon this work by calculating an ideal value for the sharing parameter σ_{sh} . The formula used depended upon whether genotypic or phenotypic sharing was

required. In genotypic sharing the distance is defined as the Hamming distance between the two gene strings. σ_{sh} is the maximum number of bits difference that is allowed between strings in the same niche.

For phenotypic sharing the euclidian distance between the decoded parameters is used.

$$d_{ij} = \sqrt{\sum_{k=1}^p (x_{k,i} - x_{k,j})^2}$$

where $x_{1,i}, x_{2,i}, \dots, x_{p,i}$ and $x_{1,j}, x_{2,j}, \dots, x_{p,j}$ are the values decoded from the gene string σ_{sh} can then be calculated as:

$$\sigma_{sh} = \frac{\sqrt{\sum_{k=1}^p (x_{k,max} - x_{k,min})^2}}{\sqrt[p]{2q}}$$

where q is the desired number of subpopulations.

[76] showed experimentally both that sharing was superior to crowding, and that phenotypic sharing consistently outperformed genotypic sharing. However [173] showed that although phenotypic sharing improved diversity, the diversity of the fit individuals was not improved. Fitness sharing has since been widely used in the literature [237], [243], [4] & [244], and can therefore be accepted to have become the de facto standard.

[254] introduced a method of implicit sharing of fitness which was later used by [106] & [107]. The technique is a kind of multi-objective evolutionary algorithm (MOEA) in which each individual is a classifier capable of recognising a particular target. In order to recognise all possible classification targets the population must maintain diversity. The individual best able to recognise a particular target has its fitness increased. This is a form of emergent fitness sharing.

6.3.6

Diversity as a MOEA objective

Another way to encourage diversity is to use diversity as an objective in the evolutionary algorithm. Rather than attempt to combine several objectives (such as fitness and diversity) into one figure to use in a standard GA it is possible to create a Multi-Objective Evolutionary Algorithm (MOEA) (for a description of MOEAs, see appendix E).

[44] defined six objectives for multi objective evolutionary algorithms intended to increase diversity for GAs used in dynamic environments. For each experiment two objectives were used, standard fitness and one of the following:

- 1 Time stamp (simple counter)
- 2 Random number (previously used by [271])
- 3 Inverse of fitness
4. Distance to closest neighbour (DCN)
5. Average distance to all individuals (ADI)
6. Distance to best individual (DBI)

The first two objectives are intended to implicitly increase diversity by assigning random fitness, which will allow some of the less fit individuals to survive from one generation to the next. The third ensures the least fit individuals will survive from one generation to another for the sake of preserving diversity. Utilizing this extreme method of diversity preservation might be considered most useful in dynamic environments where the least fit in the current generation might contribute towards the most fit in the following generation if the environment changes. The last three objectives explicitly encourage diversity. [44] experimentally demonstrated that ADI and DBI outperformed the other objectives in final population fitness.

Diversity has also been used directly as an objective in several studies [255], [72] & [74]. [72] gave a MOEA which used as one of its objectives a weakened form of dominance. If standard dominance was represented as vector f dominating vector g if and only if,

$$1. \forall i \in 1, \dots, m \quad f_i \geq g_i$$

$$2. \exists i \in 1, \dots, m \quad f_i > g_i$$

The Pareto set is then defined as the set of vectors not dominated by any vector.

In [271] approximate dominance is introduced, defined as

$$1. \forall i \in 1, \dots, m \quad (1 + \varepsilon) \cdot f_i \geq g_i$$

where $\varepsilon > 0$ this approximation allows the individuals near to in addition to those actually on the Pareto Front to survive to the next generation, thereby preserving diversity.

[72] & [74] created the FOCUS algorithm which altered the calculation for finding non-dominated individuals, so out of a group of similar individuals occupying the same space on the Pareto-optimal front the first x_0 was given a domination number of zero to signify that it was not dominated, the second x_1 was given a value of one etc... The ordering of the individuals is arbitrary. It simply gives the GA a preference of one of the individuals in the vicinity. Individuals with no other local solutions are therefore non-dominated.

[255] used diversity as one of just two objectives, and found that using only fitness and diversity, rather than a number of objectives plus diversity outperformed all given baseline techniques.

6.4 Multi-objective Genetic algorithm

For this chapter, the ensemble was constructed using a Multi Objective Genetic Algorithm (MOGA). Each individual in the GA corresponded to a whole ensemble of predictors. The aim of the experiments was to evolve the most accurate ensemble possible by simultaneously minimising predictor errors and maximising classifier diversity.

There are similarities to the algorithm given in [52] in which a MOGA is used to select individual ANNs for use in an ensemble. [52] used the error function from NCL as the diversity objective for the ANNs. The key differences to this thesis are that in [52] the GA is used to evolve individual ANNs and combine them in the final generation to form an ensemble whereas in this thesis the individuals in the GA each represent an entire ensemble.

[60] introduced a multi-objective scheme which used objectives of minimising error, minimising the sum of network weights and most relevantly maximising the difference between the network output and the average output of the rest of the population. This does select individuals based on their own performance and their ability to complement the ensemble; however it does not design the overall ensemble to maximise diversity or ensemble performance. For example the technique cannot determine the optimal number of ANNs, or select the best of the ANNs, each individual, each ANN effectively determines its own inclusion in the ensemble

Although diversity and accuracy were expected to be complementary, it was expected that there would be a trade-off between obtaining the least error and the most diversity.

The MOGA was here being used to discover the Pareto Front of the two.

The goal of the experiments was to produce the most accurate ensemble of predictors by optimising both on overall accuracy and on the diversity of the ensemble. To minimise computational effort required to explore the vast search space of possible solutions, the weights of the internal connections for the ANNs in the ensemble were co-evolved with the structure of the networks, eliminating the requirement to train networks individually, resulting in training being preserved between generations.

6.4.1

Evaluating fitnesses

As in section 5 root mean squared bearing prediction error will be used as the accuracy measure in the MOGA, while here entropy will be also be used to measure diversity.

From these, the goal of the MOGA is to estimate the Pareto Front, the set of solutions which offer the best possible trade-offs between one objective and the other. The Pareto Front is described in appendix E, however in summary any solution located on this line cannot be improved for one objective without worsening it for another. The goal the MOGA is to estimate the Pareto Front, and incrementally improve the estimate over a series of generations

At the end of each generation the whole population is ranked on how close it is to the Pareto Front. All non-dominated individuals in the population are ranked $- (1^4) = -1$. These individuals are then set aside, and the PF of the rest of the population is found, the ensembles which make this up are ranked $- (2^4) = -16$. The next Pareto Front is ranked with $- (3^4) = -81$, and so on until all of the ensembles are ranked. The values are then scaled between 0.0 and 1.0.

Once the dominance ranking has been calculated the individuals are selected for the next generation. All of the non-dominated individuals are carried through to the next generation in order to preserve the details of the whole Pareto Front. The rest of the new population is selected using a roulette-wheel selection process to choose parents, which then through crossover and mutation become the individuals in the new generation.

The first three measures of diversity, entropy, Kohavi-Wolpert and Generalised Diversity all measure the phenotypic performance and show that the neural networks within the ensembles gave more diverse results as the evolution progressed. The fourth

diversity measure, the Hamming Distance, measured the genotypic diversity, and shows that not only is genetic diversity maintained within the ensembles, but it actually increases through the generations. This measures the genetic diversity between individuals within an ensemble, not the genetic diversity present in the MOGA population. During the experiments the mean hamming distance started to converge on the maximum hamming distance. This would suggest that while the genetic diversity within the ensembles was increasing, the genetic diversity within the MOGA population was actually decreasing.

6.5 Results

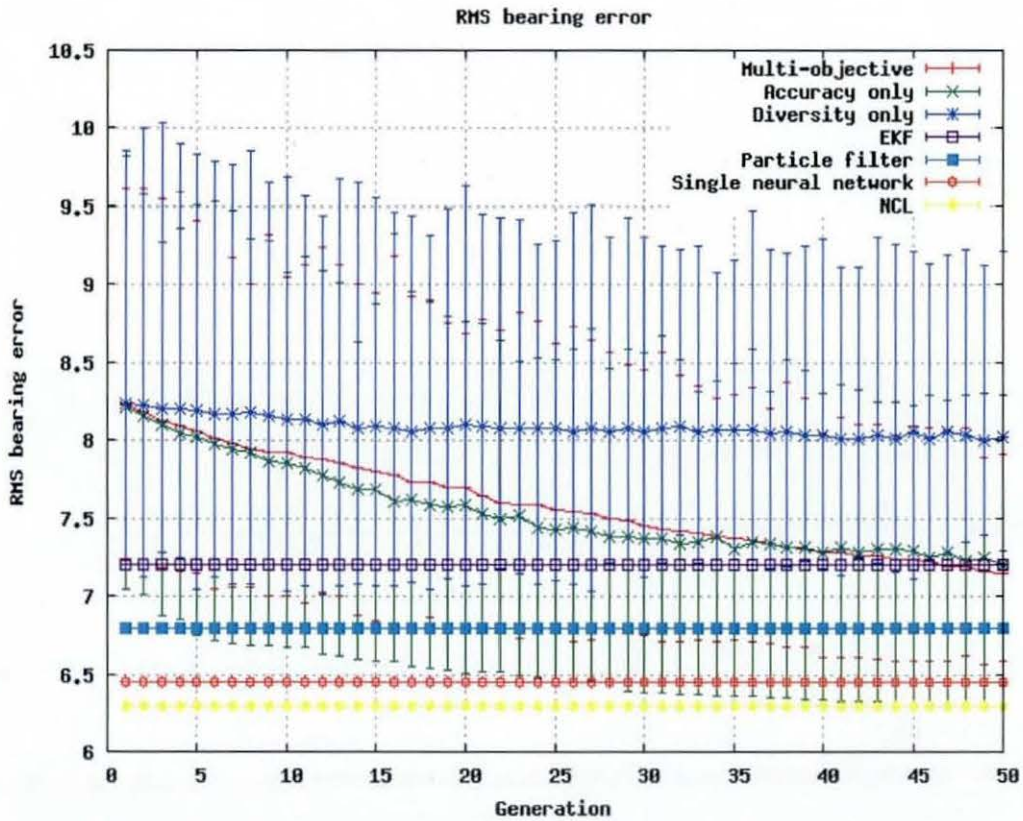


Figure 80 RMS bearing residuals of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only), EKF, PF, Gaussian Binned ANN and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

As the performance of the MOGA is worse than the algorithms tested in previous chapters a confidence factor was not calculated for this new algorithm.

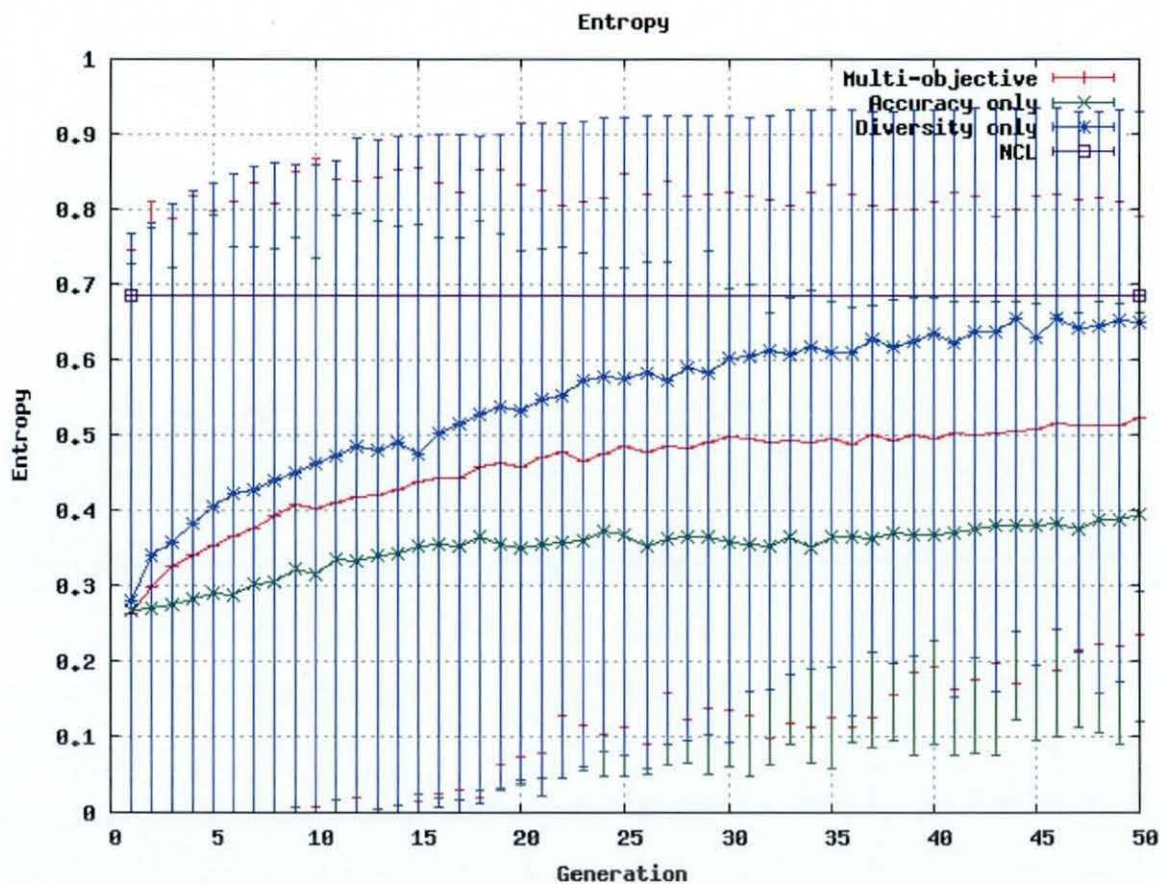


Figure 81 Entropy of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

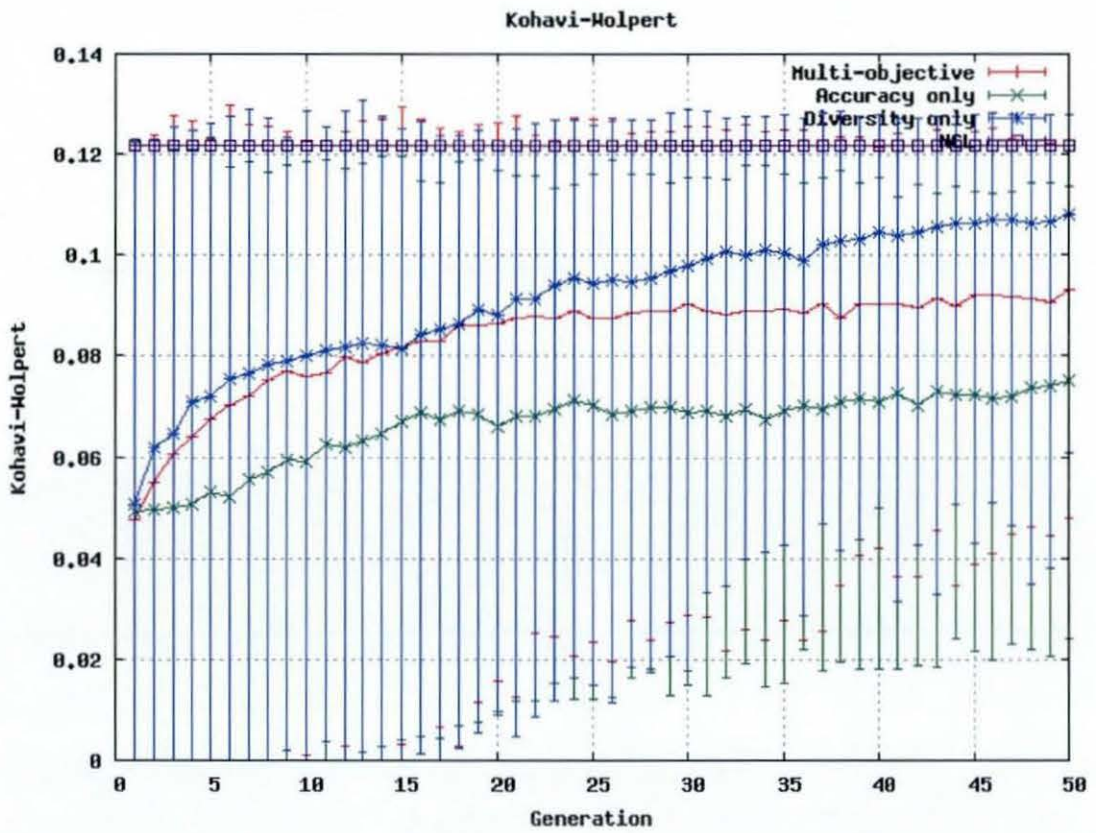


Figure 82 Kohavi-Wolpert diversity of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

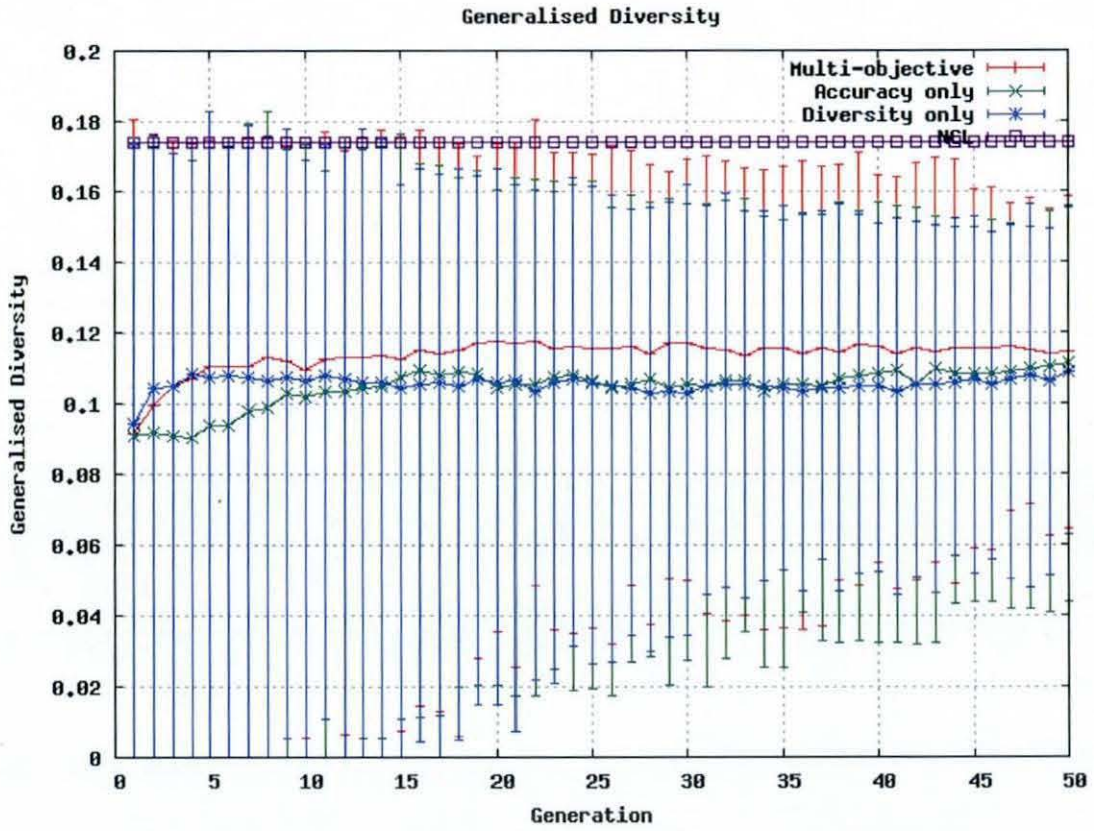


Figure 83 Generalised diversity of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

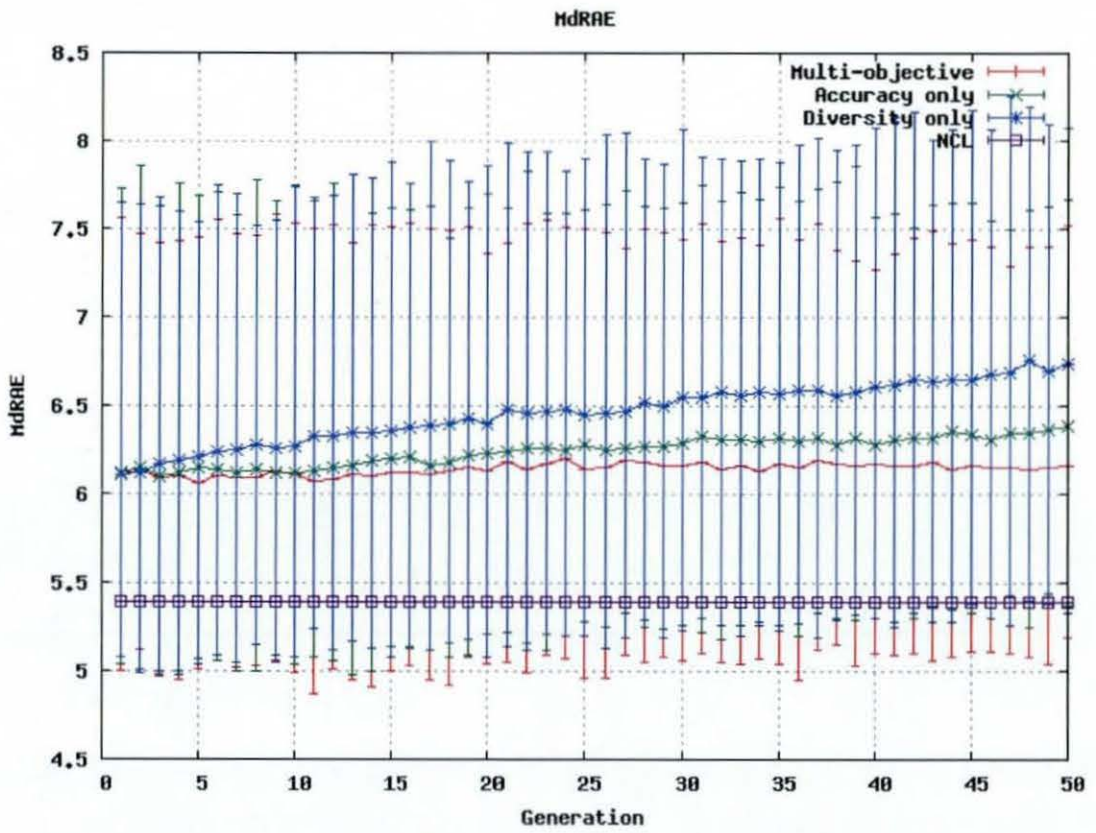


Figure 84 MdRAE of the ensemble with the highest entropy averaged across all folds

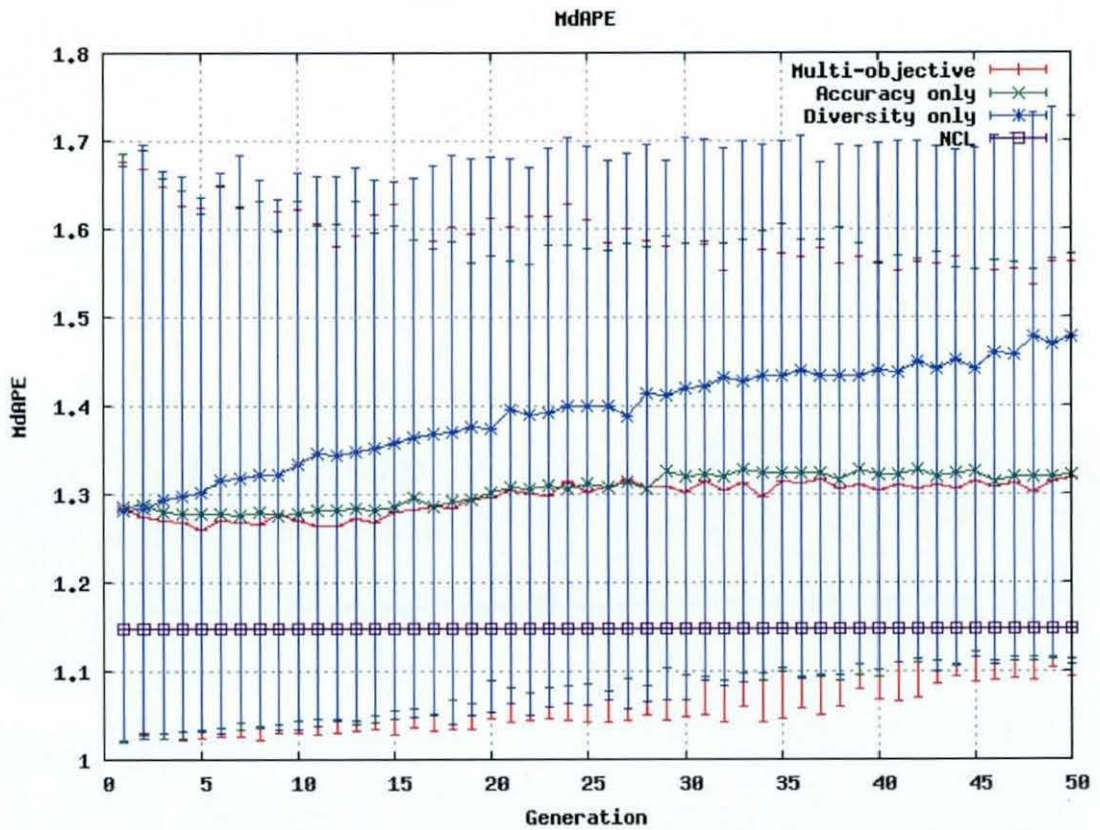


Figure 85 MdAPE of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

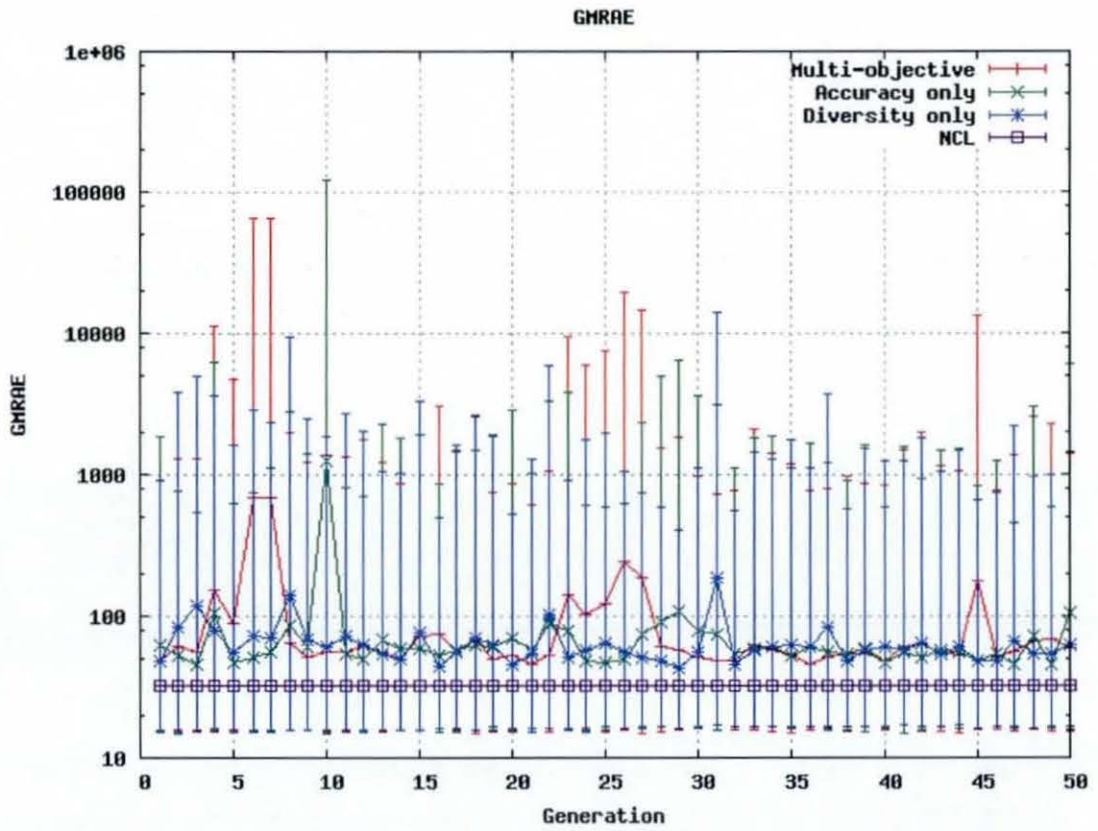


Figure 86 GMRAE of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

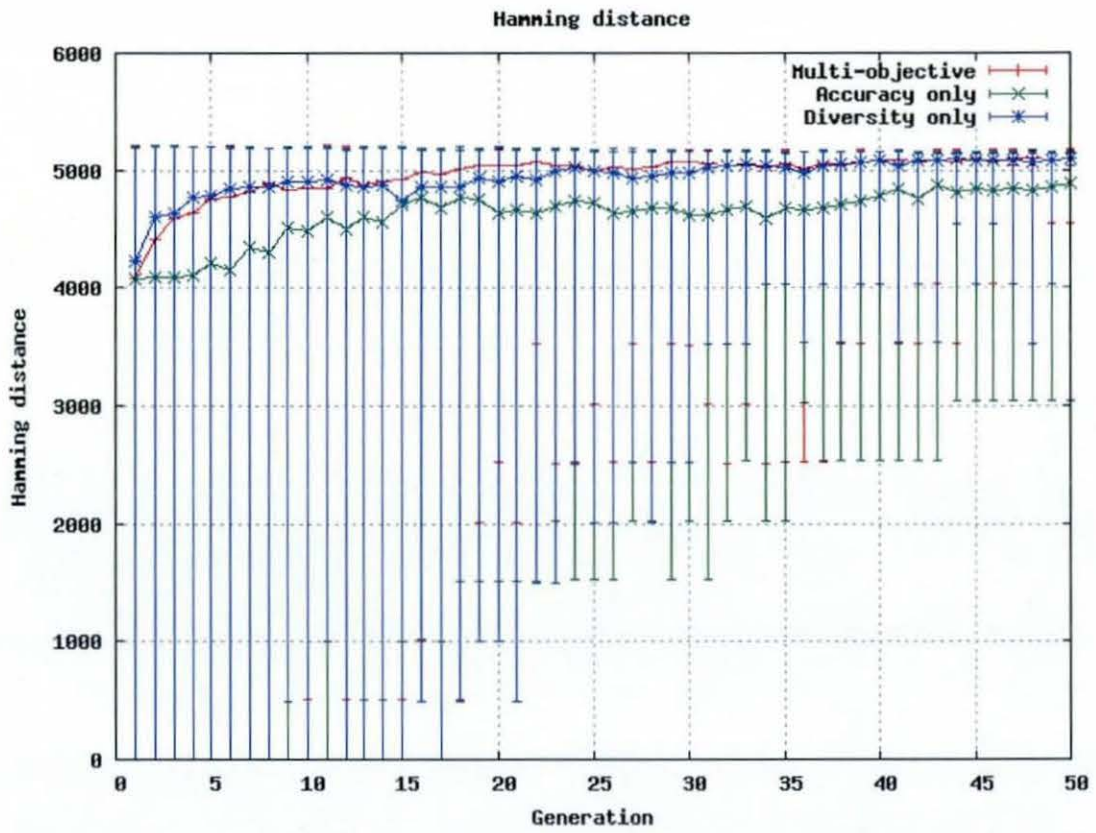


Figure 87 Hamming distance of the ensembles in the MOGA population compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most diverse and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

Data set	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Semi-synthetic
EKF	0.21	3.66	6.53	11.12	7.09
PF	1.74	3.07	5.88	10.12	6.46
ANN	2.16	4.33	9.02	16.15	7.13
Gaussian ANN	3.2	5.97	7.28	13.86	6.41
Uniform ANN	5.1	6.28	7.53	14.09	7.15
Gaussian KNN	4.09	6.5	12.58	16.8	7.71
Uniform KNN	4.56	6.52	9.51	15.26	7.16
NCL ensemble	2.97	4.21	7.03	13.2	6.29
GA ensemble	3.15	4.8	7.23	13.5	6.3
MOGA	3.25	6.01	7.4	13.96	6.58

Table 28 The best RMS bearing error for MOGA generated ensembles on each data set

	RMS	Entropy	Kohavi Wolpert	Gen'd Diversity	MdRAE	MdAPE	GMRAE	Hamming distance
NCL	6.293	0.685	0.122	0.175	5.390	1.148	32.680	0.000
GA	7.300	0.768	0.123	0.160	4.970	1.080	16.180	5186.000
MOGA	6.580	0.810	0.129	0.181	4.950	1.020	14.990	5168.000

Table 29 The best value for MOGA generated ensembles on each metric on the semi-synthetic data compared to the SOGA and the best NCL trained ensemble

6.6 Conclusions

Although the new MOGA based algorithm has demonstrated that it is more accurate than the other algorithms presented on several of the metrics. It did not outperform the NCL on the key objective of RMS bearing error. The MOGA however did give higher levels of diversity than NCL.

This is especially disappointing as [52] found that using a MOGA improved performance over the NCL. However unlike the technique used in this chapter in [52] the ANNs which are evolved with the MOGA are subsequently trained, which was found to be impossible in this thesis in section 5.4.1. Therefore it would appear that the benefits of the GA/MOGA structure proposed so far which was intended to create structural as well as learned diversity do not outweigh the penalty of not training the ensembles at each generation, although as [52] used different data to those used in this paper no definite conclusions can be made without further investigation.

This would suggest that just using random individuals along the Pareto Front is not a suitable way to introduce diversity to the ensembles. This could for example be because too much emphasis is being placed upon diversity, which here effectively has a 50% weighting, in that all ensembles are selected based on their proximity to the Pareto Front and are chosen with uniform probability along the length of the front. A more effective method for selecting ensembles might be to bias selection towards a particular end of the Pareto Front. A known bias would allow the optimal level of diversity to be found. Giving diversity a lower weighting in this way might be a way to improve upon the results of previous chapters.

7 Discovering accuracy/diversity balance which gives best overall accuracy

7.1 Summary

The previous section attempted to improve the GA by adding diversity as an objective. The way this was done however gave equal weighting to accuracy and diversity. NCL has a parameter λ which allows tuning of the balance between accuracy and diversity. This parameter is added to the GA for the same purpose; to find the balance between accuracy and diversity which minimises ensemble prediction error. The result is a predictor which outperforms all others presented so far. This clearly meets the original specification of a predictor which is not tied to an underlying (oversimplified) model which is capable of outperforming all of the baselines.

7.2 Introduction

Further to the work in section 6, in which an equal number of individuals were selected to continue to the next generation based upon their fitness, the aim of this chapter was to find the optimal balance between accuracy and diversity when creating ensembles of classifiers for target tracking.

A new parameter, λ is introduced, which has a similar purpose to the λ parameter in NCL. This is used to define the balance used between accuracy and diversity, with values in the range $0 \leq \lambda \leq 1$. A value of 1 represents using accuracy alone, while a value of zero represents using only diversity.

7.3 Changes to selection procedure

The GA is in almost all respects identical to the one used in chapters 5 and 6. The only change from section 6 is outlined in this section.

When the pool of individuals is selected for the next generation, it is chosen by its Pareto ranking, as in section 6, however here it is over-selected, so that ten times as many individuals are chosen than are needed to form an intermediate population. An individual's chances of being included in this population are proportional to its distance from the Pareto Front. The roulette wheel selection is then run a second time, to select individuals based upon their accuracy, selecting λn individuals from the intermediate population, where n is the required final population size. The process is rerun to select $(1-\lambda)n$ individuals from the sub-population based on their diversity

Of the individuals selected, 10% are carried forward to the next generation unaltered, while the rest are used as parents and crossed as in chapters 5 and 6.

7.4 Results

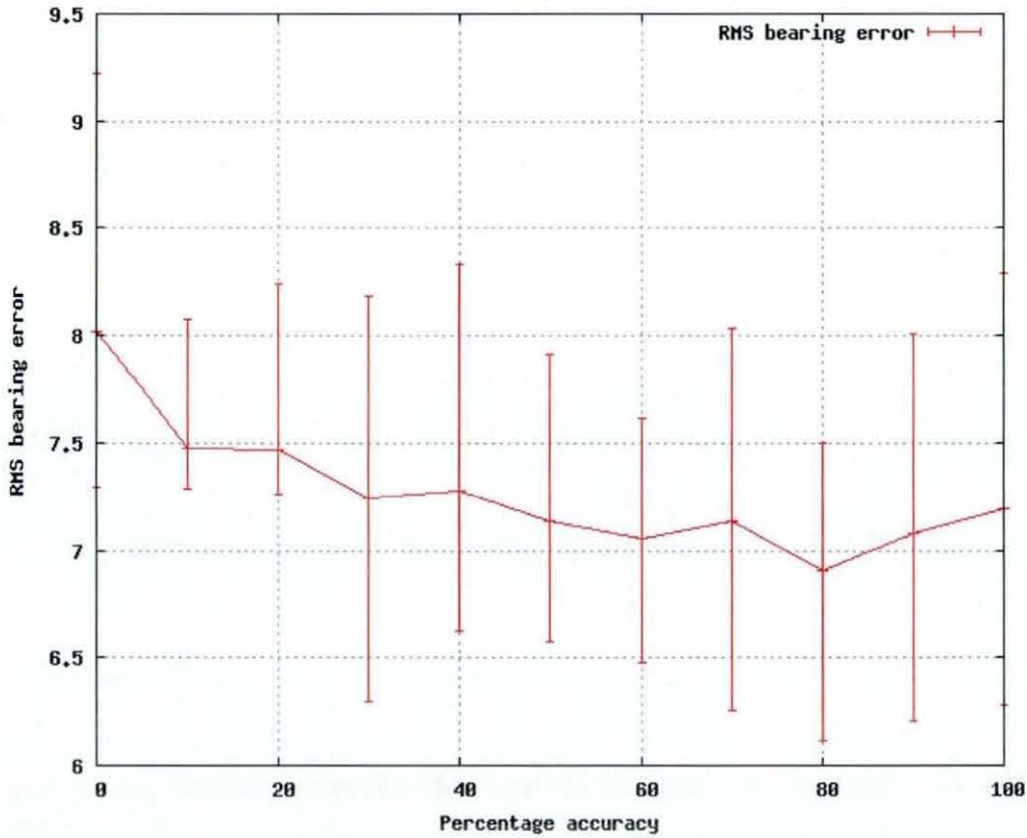


Figure 88 Final generation RMS error over percentage of individuals selected for their accuracy were the line represents the mean level of accuracy in the MOGA population in the final generation, while the error bars show the range between the most and least accurate individuals

As can clearly be seen from figure 88, the balance of accuracy and diversity found to give the lowest RMS bearing prediction was 80% accuracy, 20% diversity, with both pools selected from an intermediate population selected based on Pareto optimality. At this level, not only is the mean ensemble more accurate than at any level, the best ensemble in the population is more accurate than the best ensemble for any other balance. Additionally at 80%, the worst ensemble in the population is more accurate than the worst ensemble at the other levels of balance. This is an important measure of how tightly grouped the GA's population is around the optimal solution. Figures 89 to 96 show the full results of the experiments when selecting 80% of the population for accuracy and 20% for diversity.

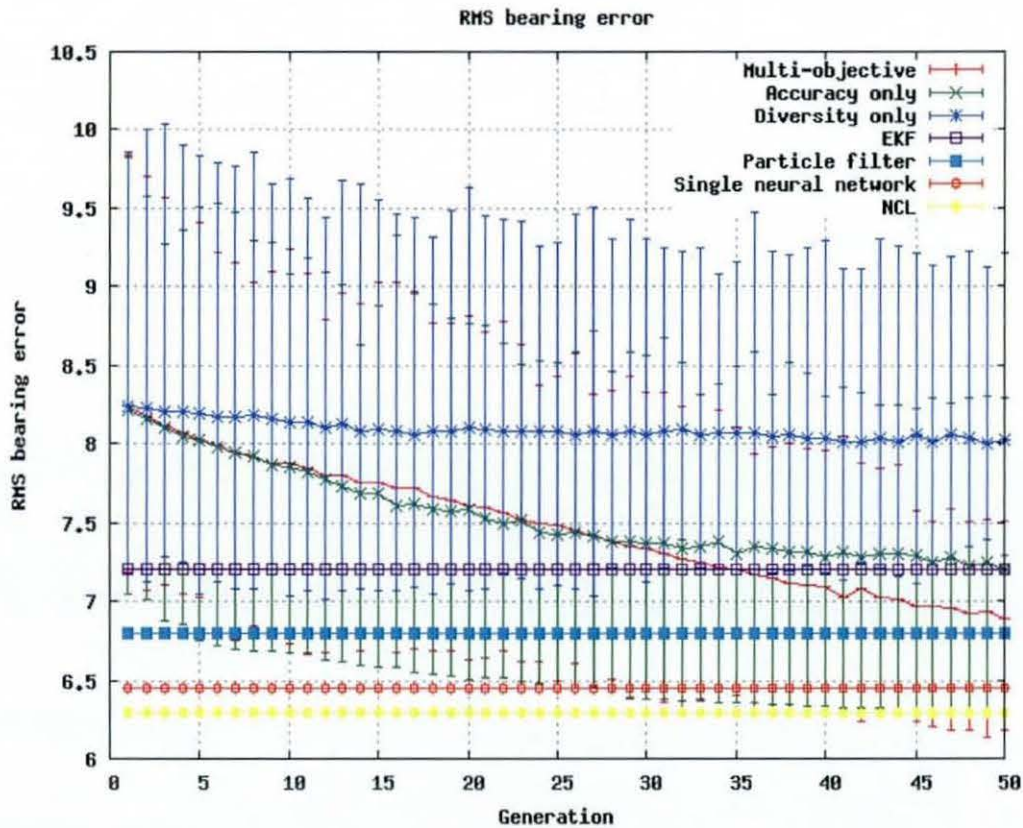


Figure 89 RMS bearing residuals of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only), EKF, PF, Gaussian Binned ANN and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most accurate and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

- Though the chart is cluttered, the 80% MOGA produced ensemble may be seen in red at the bottom right of the chart. This is where, at the end of the run, the best MOGA produced ensemble displays a clear improvement over the NCL trained ensemble.
- The population selected based purely on their diversity does not change significantly with respect to RMS bearing error throughout the run of the MOGA.
- Until around generation 30 there is little to distinguish between the performance of the single objective GA and the MOGA, at which point the single objective GA levels off, while the MOGA continues to improve..

Table 30 shows these results for RMS bearing error broken down into the individual folds, alongside the results for the GA, the NCL trained ensembles, the GANN, and the

baseline algorithms from section 2.7. Table 31 shows ensembles created with a GA are more accurate than both the plain ANN and the EKF on all 10 of the folds, they are more accurate than the PF on 9 of the ten folds and more accurate than the GANN, the NCL trained ensembles and the Single Objective GA created ensembles on 8 out of the 10 folds. From table 32 it is possible to see that it can be said that the ensembles are better than the ANN and the EKF with a confidence of 99.90%, more accurate than the PF with a confidence of 98.83%. There is also a 94.53% confidence in the new λ MOGA designed ensembles being an improvement on the GANN, the NCL trained ensembles and the Single Objective GA created ensembles. This means that there is a statistically significant improvement obtained by using the λ MOGA over all of the other algorithms tested.

Fold	ANN	EKF	PF	GANN	NCL	GA	λ MOGA
1	8.13	7.13	6.84	6.59	6.25	5.98	5.86
2	7.67	7.02	6.36	6.46	6.29	6.77	5.78
3	7.81	7.10	6.50	6.32	6.28	6.67	6.23
4	7.64	6.93	6.41	6.35	6.32	6.09	5.83
5	8.07	6.98	6.78	6.46	6.25	6.18	6.49
6	7.71	7.01	6.43	6.43	6.31	6.21	6.20
7	7.37	7.05	6.13	6.33	6.34	6.39	6.11
8	7.81	6.99	6.66	6.56	6.29	6.33	6.15
9	7.32	7.20	6.04	6.18	6.29	6.38	6.22
10	7.72	7.53	6.43	6.42	6.29	6.00	6.42

Table 30 Results per fold of the best of the new algorithms, the Gaussian binning Artificial Neural Network against the per-fold results for the baseline algorithms tested in section 2.7

B \ A	ANN	EKF	PF	GANN	NCL	GA	λ MOGA
ANN	x	10	10	10	10	10	10
EKF	x	x	10	10	10	10	10
PF	x	x	x	7	8	6	9
GANN	x	x	x	x	8	6	8
NCL	x	x	x	x	x	5	8
GA	x	x	x	x	x	x	8
λ MOGA	x	x	x	x	x	x	x

Table 31 The number of folds for which algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN, NCL, GA and λ MOGA

B \ A	ANN	EKF	PF	GANN	NCL	GA	λ MOGA
ANN	x	99.90%	99.90%	99.90%	99.90%	99.90%	99.90%
EKF	x	x	99.90%	99.90%	99.90%	99.90%	99.90%
PF	x	x	x	82.81%	94.53%	62.30%	98.93%
GANN	x	x	x	x	94.53%	62.30%	94.53%
NCL	x	x	x	x	x	37.70%	94.53%

GA	X	X	X	X	X	X	94.53%
λ MOGA	X	X	X	X	X	X	X

Table 32 The percentage confidence that algorithm A was more accurate than algorithm B for the ANN, EKF, PF, GANN, NCL, GA and λ MOGA

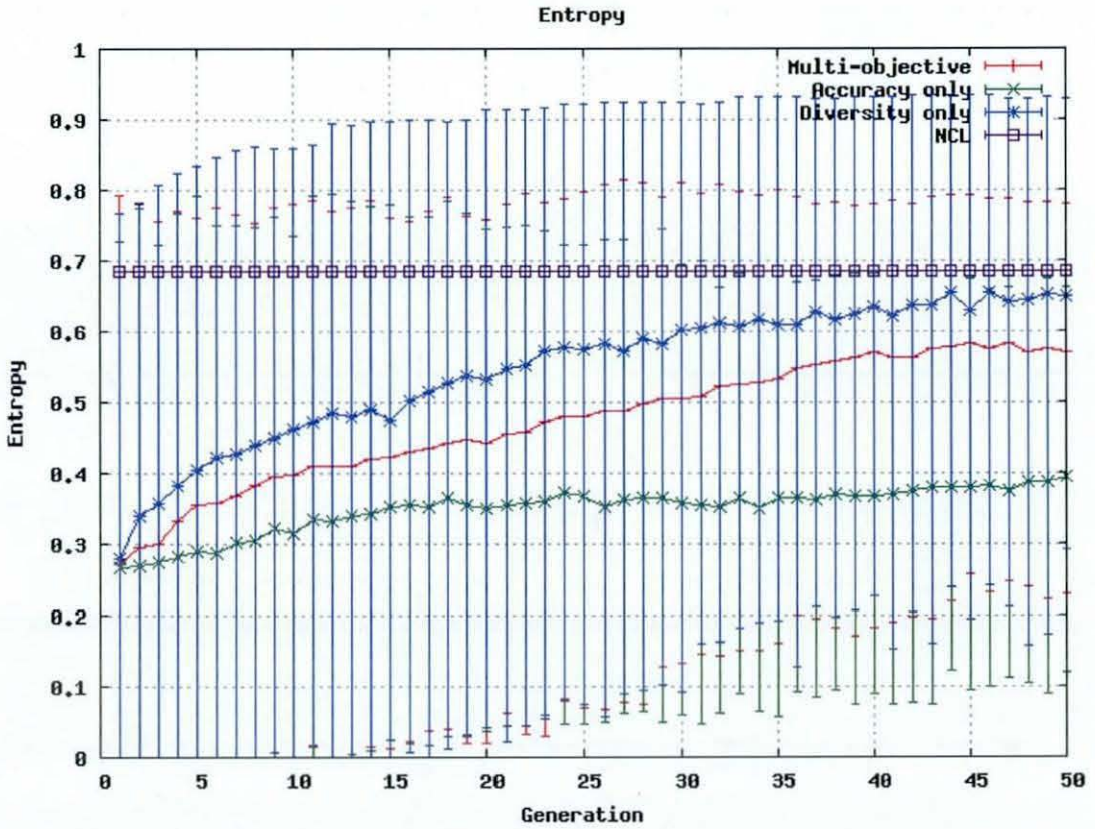


Figure 90 Entropy of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

- The highest levels of entropy are obtained by the single objective GA selected for diversity. Both the 80% MOGA and the single objective diversity GA outperform the NCL trained ensemble on this metric.
- The lowest level of entropy is given for the GA selected for accuracy.

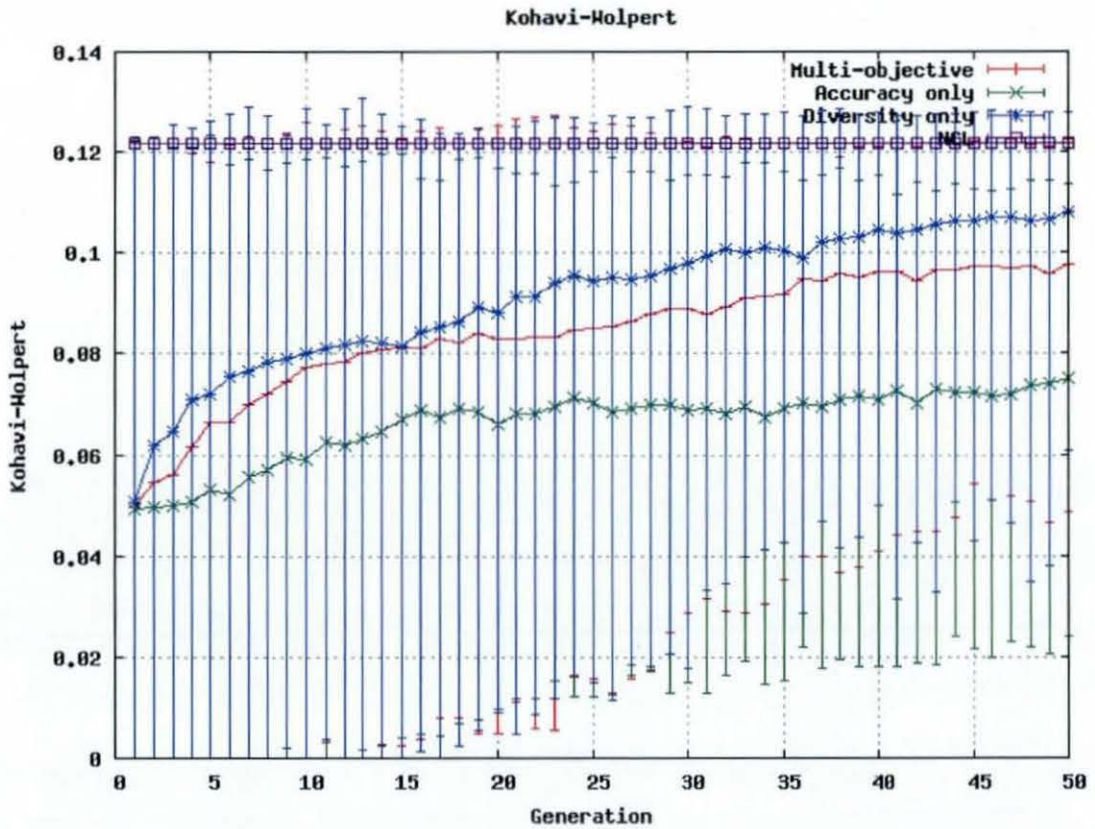


Figure 91 Kohavi-Wolpert diversity of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

- As with entropy, the highest level of diversity here is shown by the ensembles selected on diversity alone.
- The 80% MOGA also gives a higher level of diversity than the NCL trained ensembles
- The worst performance with respect to KW is from the ensembles produced by the GA selecting upon accuracy alone.

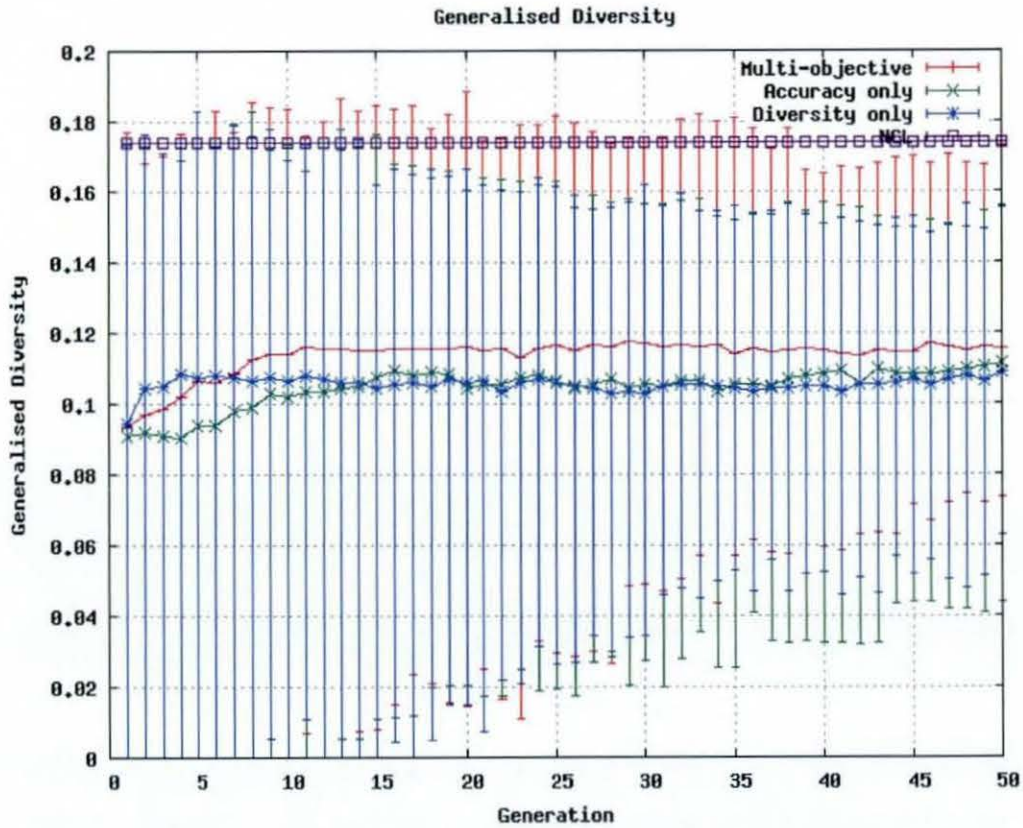


Figure 92 Generalised diversity of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

- This is one of the only metrics in which the NCL trained ensembles proved the most diverse.
- Although during the evolutionary process ensembles were created which were more diverse in terms of GD, by the end of the evolutionary run the best individuals were less diverse in terms of GD than the ones trained with NCL.

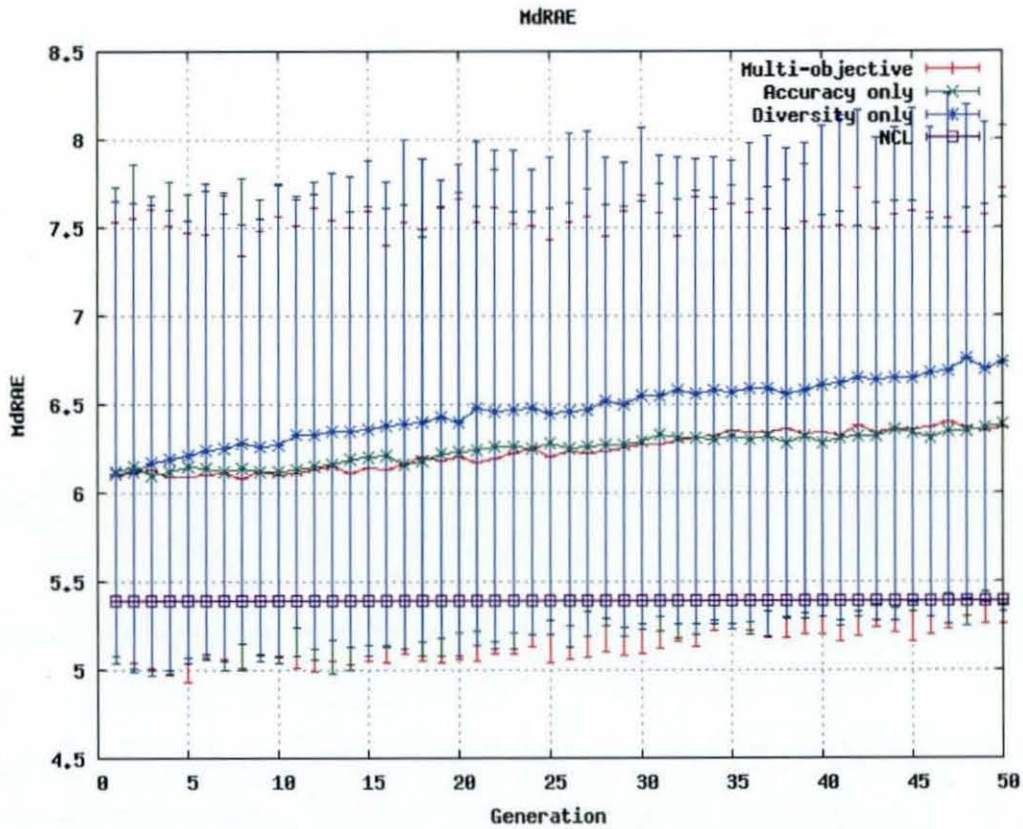


Figure 93 MdRAE of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

- At the end of the evolutionary run the best MOGA created ensembles are narrowly more accurate than those trained with NCL.
- The worst ensembles with respect to MdRAE are those produced by the diversity only GA.

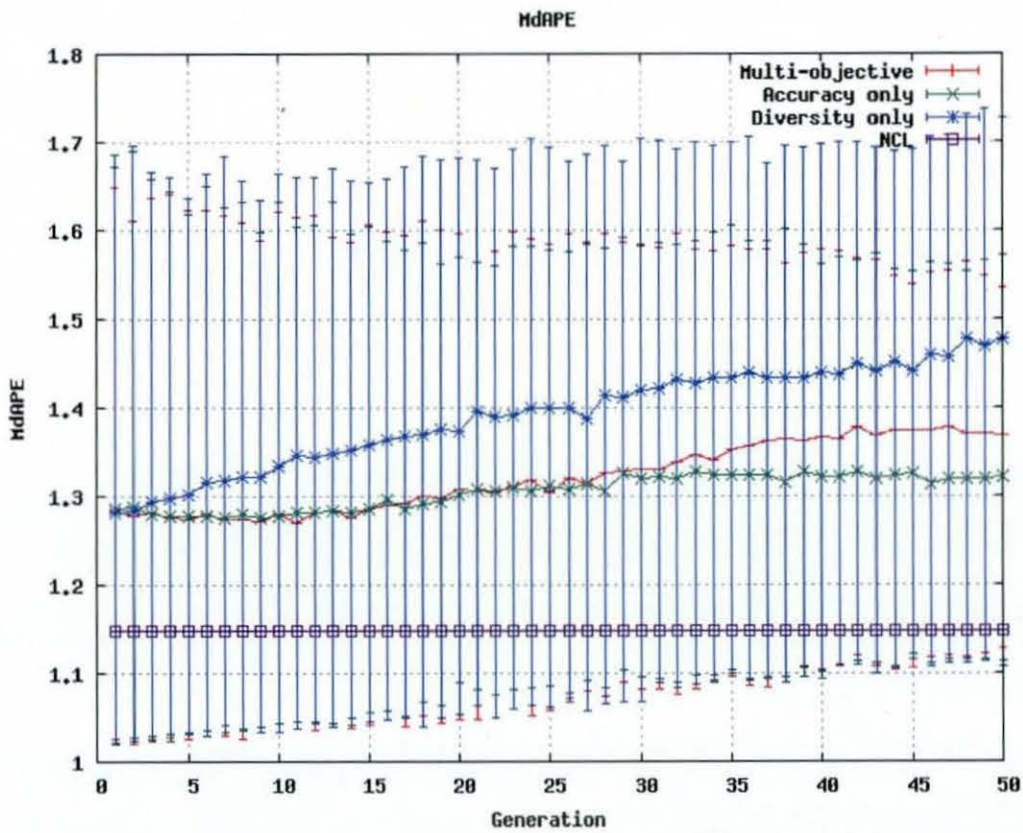


Figure 94 MdAPE of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

- The results for MdAPE are almost the same as those for MdRAE, except for this metric the most accurate individuals are those trained by the accuracy only GA.
- All three forms of the GA produce ensembles which are more accurate with respect to MdAPE than the NCL trained ensembles.
- The most accurate ensemble from the diversity only GA is more accurate than the most accurate ensemble from the MOGA.

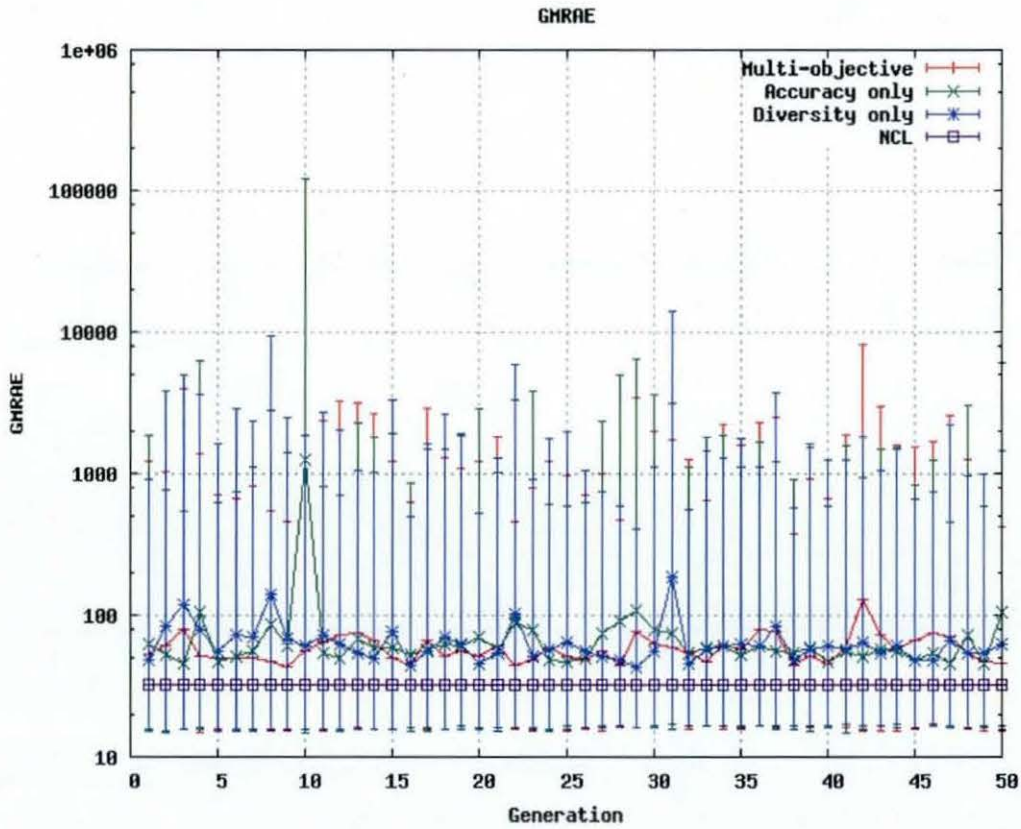


Figure 95 GMRAE of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least accurate ensembles in the GA population as the error bar and the mean accuracy shown as the line

- The performance of the three GA based algorithms are very similar with respect to GMRAE
- All three GA based algorithms produce ensembles which outperform the NCL trained ensemble on this metric.

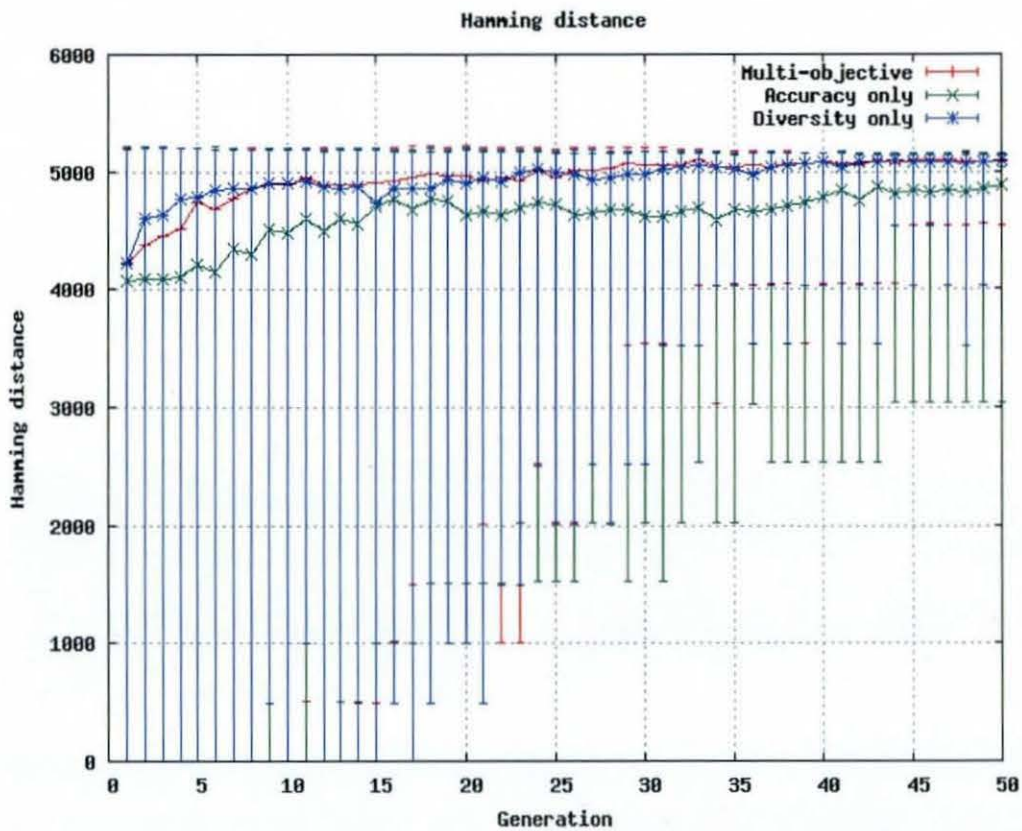


Figure 96 Hamming distance of the ensembles in the MOGA population for the MOGA with $\lambda=80\%$ compared to the SOGA (accuracy only), SOGA (diversity only) and NCL trained ensemble predictors tested on the semi-synthetic data set showing the range between the most and least diverse ensembles in the GA population as the error bar and the mean diversity shown as the line

- The Hamming Distance, and therefore the genotypic diversity can be seen to be very similar in the GA with diversity only and the 80% MOGA.
- The poorest performer with respect to the Hamming Distance is the GA with accuracy only.
- The mean level of diversity increases in all cases.

Data set	Synthetic 1	Synthetic 2	Synthetic 3	Synthetic 4	Semi-synthetic
EKF	0.21	3.66	6.53	11.12	7.09
PF	1.74	3.07	5.88	10.12	6.46
ANN	2.16	4.33	9.02	16.15	7.13
Gaussian ANN	3.2	5.97	7.28	13.86	6.41
Uniform ANN	5.1	6.28	7.53	14.09	7.15
Gaussian KNN	4.09	6.5	12.58	16.8	7.71
Uniform KNN	4.56	6.52	9.51	15.26	7.16
NCL ensemble	2.97	4.21	7.03	13.2	6.29
GA ensemble	3.15	4.8	7.23	13.5	6.3
MOGA	3.25	6.01	7.4	13.96	6.58
MOGA (80% diversity)	2.93	4.1	6.93	9.87	6.13

Table 33 The best RMS bearing error for MOGA generated ensembles on each data set

	RMS	Entropy	Kohavi Wolpert	Gen'd Diversity	MdRAE	MdAPE	GMRAE	Hamming distance
NCL	6.293	0.685	0.122	0.175	5.390	1.148	32.680	0.000
GA	7.300	0.768	0.123	0.160	4.970	1.080	16.180	5186.000
MOGA	6.580	0.810	0.129	0.181	4.950	1.020	14.990	5168.000
MOGA (80%) diversity	6.130	0.780	0.120	0.170	4.930	1.02	15.330	5215.000

Table 34 The best value for MOGA generated ensembles on each metric

7.5 Conclusions

The main conclusion of this chapter is that a small amount of diversity is beneficial to the population, as not only does it improve the performance of the best individual in the population, it also improves the fitness of the worst, indicating that the GA has been more successful in collapsing down to surround the optimal solution.

This would suggest that when GAs are used to design classifiers, a fraction of the population at each stage should be selected on the basis of the individual's diversity. This agrees with the theory behind NCL, and effectively extends its use to Genetic Algorithms.

The largest source of novelty is the discovery that as long as ensembles are created as whole entities rather than component parts, all of the advantages provided by techniques such as NCL which stimulate diversity to in turn increase accuracy may be obtained through use of evolution with multiple objectives. This significantly reduces computational requirements, and greatly simplifies the process of creating such an ensemble when compared to approaches that require a learning algorithm to teach the ANN. This may be applied to any GA creating ensemble classifiers for any purpose, massively simplifying the complexity of doing so, while also dramatically increasing performance

7.5.1

Comparison to NCL

The results produced by the technique in this thesis outperform NCL both in terms of accuracy of results and diversity of the ensembles, however as so many papers have been written on NCL and its ability to encourage diversity in ensembles, it is important to outline the differences between this technique and NCL, and to highlight some of the advantages of the new algorithm.

- Although the use of NCL does not preclude it, NCL itself has no mechanism for encouraging structural diversity between the ANNs in an ensemble. In most examples in the literature, the ANNs used are structurally identical, relying upon a combination of random weight initialization and NCL to create diversity using the connection weights.
- [183] gives an evolutionary approach to designing ANN ensembles, however unlike the approach used in this thesis [183] uses the GA to evolve ANNs where the whole GA population is simultaneously evolved using NCL at each generation. In this thesis each member of the GA population is an entire ensemble in its own right, and no form of learning algorithm is used to train the ensembles, except for the GA which co-evolves the structure and the weights.
- NCL is specific to ANNs, whereas the technique outlined in this paper is generic to any classifier, once the chromosome has been defined.
- When used in a GA setting, NCL learning must be repeated with each new generation to train the ensembles, whereas the technique outlined here does not, making it faster.
- Additionally although much of the NCL literature describes how it increases diversity in the ensembles, none of the papers available measure diversity to see

if it has actually increased as a result of applying the technique, or make any comparison to the amount of diversity that would be present had NCL not been used. This thesis provides the first quantification of the diversity generated in order to improve accuracy.

- However, this technique effectively is NCL but applied to MOGAs, so the benefits of NCL can be expected, without having to train the ensembles.

8 Conclusions and further work

A new form of time series predictor has been created, which has been shown to be far more accurate than any of the baseline techniques evaluated. The new family of algorithms created are shown to be both consistently reliable, and far less likely to produce outliers than the baselines. In addition to the novel way in which the ANNs and ensembles are applied, a GA based algorithm was created to both create the structure for and train the ensembles, further adding to the novelty presented.

Initially, in chapter 3, a group of new classification based predictors were produced. The results showed that in terms of RMS prediction error, as the number of inputs increases, the accuracy of the proposed technique improves dramatically. With fewer inputs the two best baseline techniques both outperform the proposed technique, however with more inputs the new technique is considerably more accurate than all of the baselines. When the number of inputs was more than fifteen, by using the Gaussian ANN algorithm the bearing error can be reduced by as much as 16% over the EKF, and 6% over the PF, the two most accurate baseline techniques tested. However there are many parameters such as the underlying algorithm (e.g ANN or KNN), network size and structure and the learning algorithm that must be selected to find the most efficient and accurate network capable of outperforming the baseline techniques. Changing the parameters can significantly alter the performance, although the relationship between the parameters and the performance is complex.

In chapter 4, the performance of the technique was further improved by changing from using ANNs trained with backpropagation to using ensembles of ANNs trained with Negative Correlation Learning (NCL). NCL was proven to be an effective method for training ensembles to perform target tracking, outperforming both of the baseline techniques; the EKF and PF, and the ANN based techniques already created. These new predictors were shown to outperform every technique presented so far on every data set. The gains achieved in this application from utilising NCL and ensembles are significant.

One of the drawbacks of NCL is that as a training algorithm it does not have the capability to design the ANNs on which it is applied. Chapter 5 tried to solve this

problem with a Genetic Algorithm (GA), with the aim of evolving the network structure to create an ensemble capable of improving upon the results obtained with NCL. Although the technique was successful in that it improved on the results from chapter 3, the new method of creating ensembles does not create ensembles more accurate than those trained with NCL as in chapter 4, although the performance was almost as good. Here training the ensemble with NCL was shown to be better than the GA at both maximising diversity and minimising error. This would suggest firstly that as described in the literature, NCL is a very good technique for training ensembles to be both diverse and accurate. However it also shows that there is scope to improve the GA, and incorporating the idea of increasing diversity as used in NCL might be a way of improving performance of the GA.

In order to improve upon the disappointing results from the GA, a Multi Objective GA (MOGA) was created in chapter 6 which took diversity as a second objective during the selection stage in the GA. Although the new MOGA based algorithm has demonstrated that it is more accurate than the other algorithms presented on several of the metrics It did not outperform the NCL on the key objective of RMS bearing error. The MOGA however did give higher levels of diversity than NCL. This would suggest that just using random individuals along the Pareto Front is not a suitable way to introduce diversity to the ensembles. This could for example be because too much emphasis is being placed upon diversity, which here effectively has a 50% weighting, in that all ensembles are selected based on their proximity to the Pareto Front and are chosen with uniform probability along the length of the front.

As a result of this, in chapter 7 the MOGA was rerun several times, each time using a different balance between accuracy and diversity to select the individuals for the next generation. A balance of 80% of the individuals selected for accuracy, with 20% selected for their diversity is found to be the most effective mix in this application. The results produced by the technique outperform all other techniques presented in this thesis both in terms of accuracy of results and diversity of the ensembles. The main conclusion of this experiment was that a small amount of diversity is beneficial to the population, as not only does it improve the performance of the best individual in the population, it also improves the fitness of the worst, indicating that the GA has been more successful in collapsing down to surround the optimal solution. This would suggest that when GAs are used to design classifiers, a fraction of the population at

each stage should be selected on the basis of the individual's diversity. This agrees with the theory behind NCL, and effectively extends its use to Genetic Algorithms.

There are many novel features of this thesis. Firstly the methodology for using a classification algorithm as a time series predictor in target tracking is new and allows the use of any number of classification algorithms to be used in time series prediction for target tracking.

Further to this the work was extended to use ensembles of classifiers to enhance the predictions; not only have ensembles not previously been used to perform target tracking, but NCL has not previously been used to train a target tracking ensemble. A GA was created which can both design an ensemble of ANNs and train it in a single step. This is the first time that an ensemble has been constructed in such a way, and a multi-objective form of the algorithm is shown to be highly effective at creating optimal ensembles which, unlike ensembles trained with NCL, have structural as well as learned diversity.

Most importantly however, the largest source of novelty here is the discovery that as long as ensembles are created as whole entities rather than component parts, all of the advantages provided by techniques such as NCL which stimulate diversity to in turn increase accuracy may be obtained through use of evolution with multiple objectives. This significantly reduces computational requirements of training and testing the ensembles, and greatly simplifies the process of creating such an ensemble when compared to approaches that require a learning algorithm to teach the ANN. This may be applied to any GA creating ensemble classifiers for any purpose, allowing a form of NCL to be applied not only to ANNs, but to any classifier or predictor which may be described with a chromosome.

8.1 Further work

This thesis gives results for use of the algorithm on four simple, generic data sets, and a very large passive sonar data set. However further work is required to firmly establish the assumed generality of the technique. Work is required to establish whether the results obtained on the amount of diversity required extend to other problem domains, and on other data sets. It would be extremely beneficial to find a way to predict in advance the proportion of individuals that must be selected from each objective in order to maximise performance.

Several papers are mentioned such as [52][53][54][142][183] which use a variety of techniques not tested in this thesis. For example [52][53][54][142] all use GAs to evolve ensembles (albeit with the GA population comprising of ANNs and combining the end population to form an ensemble), and these train to a greater or lesser extent during evolution. Although full training of the ensembles was shown to be impractical for this application in section 4.2, [183] & [142] for example only use partial training to improve results during the evolutionary processes, and then fully train the resultant ensemble. Also [52] uses Boosting and Bagging to improve results, neither of which were used within this thesis, mostly because they are more normally associated with learning algorithms, however they could be used in future work to enhance performance. Boosting could be used to increase the weighting of particular input patterns during the evolutionary process, allowing the GA to improve the ensembles' weaknesses. Bagging might be used to randomly subsample the extremely large data set to create simpler sets each of which could be used in a separate part of the GA population to create species of ensembles which could be combined in later generations to form an ensemble capable of predicting values from the whole data set.

Further work is also required to establish the measure of diversity which gives the biggest improvement in performance. Here four measures of diversity were tested, but only one was used in the evolutionary process. Experimentation is required to establish both the best one to use, and again to establish how this would change across different problem domains and data sets. Many other measures of diversity exist, so there is very wide scope for finding which is most suitable for this purpose.

Most, though not all, of the experiments performed for the classifier based predictors were done using an ANN as a classifier. Only ANNs and KNNs were used in this thesis. Another possible path for future work would be to experiment with other classifiers to find which ones worked in this situation. Although the ANN was shown to be the best of the classifiers tested, as only two different classifiers were tried it is unlikely the one most fit for purpose has been discovered.

Another area which time prevented exploring was the possibility of using the newly created algorithms in Target Motion Analysis. If the binning algorithm divided the output space by range, or into a grid, then the same algorithms could be used to estimate the distance to the target, or the target's position.

9 References

- [1] Abdel-Aty-Zohdy H S and Ewing R L, Intelligent information processing using neural networks and genetic algorithms, Proceedings of the 43rd Midwest Symposium on Circuits and Systems, Pages 840-845, August 2000
- [2] Ackerson G A and Fu K S, On state estimation in switching environments, IEEE transactions on automatic control, Volume 15, Issue 1, Pages 10 – 17, 1970
- [3] Adamidis P and Petridis V, Co-operating populations with different evolution behaviors, in Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, ICEC'96, Pages 188–191, 1996
- [4] Ahn J and Cho S, Speciated neural networks evolved with fitness sharing technique, Proceedings of the 2001 Congress on Evolutionary Computation, Volume 1, Pages 390-396, 2001
- [5] Akyildiz I F, Su W, Sankarasubramaniam Y and Cayirci E, A survey on sensor networks, IEEE communications magazine, Volume 40, Issue 8, Pages 102 – 114, August 2002
- [6] Akyildiz I F, Su W, Sankarasubramaniam Y and Cayirci E, Wireless sensor networks: a survey, Elsevier, Computer Networks, Volume 38, Issue 4, Pages 393-422, 2002
- [7] Alexander H L, State estimation for distributed systems with sensing delay, Proceedings of the International Society for Optical Engineering, Vol 1470, Pages 103-111, August 1991
- [8] Angeline P J, Sauders G M, and Pollack J B, An evolutionary algorithm that constructs recurrent neural networks, IEEE Transactions on Neural Networks, Volume 5, Issue 1, Pages 54–65, 1994
- [9] Arambel P O, Rago C and Mehra R K, Covariance intersection algorithm for distributed spacecraft state estimation, Proceedings of the 2001 American Control Conference, Volume 6, Pages 4398 – 4403, 2001
- [10] Armstrong J S and Collopy F, Error measures for generalizing about forecasting methods: Empirical comparisons, International Journal of Forecasting, volume 8, number 1, Pages 69-80, 1980
- [11] Arora A, Dutta P, Bapat S, Kulathumani V, Zhang H, Naik V, Mittal V, Cao H, Demirbas M, Gouda M, Choi Y, Herman T, Kulkarni S., Arumugam U.,

- Nesterenko M , Vora A , Miyashita M., A line in the sand - wireless sensor network for target detection, classification and tracking, Elsevier, Computer Networks 46 (2004) 605–634, 2004
- [12] Arulampalam M S, Maskell S, Gordon N, and Clapp T, A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking, IEEE Transactions on Signal Processing, Vol 50, No. 2, February 2002, 2002
- [13] Ash T, Dynamic node creation in backpropagation networks, Technical report, Institute for a cognitive science, University of California, San Diego, 1989
- [14] Aziz A M, Tummala M, Cristi R, Fuzzy logic data correlation approach in Multisensor-multitarget tracking systems, Elsevier Signal Processing 76, Pages 195-209, 1999
- [15] Bar-Shalom Y and Blom H A P, The interacting multiple model algorithm for systems with Markovian switching coefficients, IEEE transactions on automatic control, Vol.33, Issue 8, Pages 780-783, 1988
- [16] Bar-Shalom Y and Li X-R, Estimation and tracking: principles techniques and software, Artech House, 1993
- [17] Bar-Shalom Y and Li X-R, Multitarget-Multisensor tracking: Principles and techniques, YBS publishing, 1995
- [18] Bar-Shalom Y and Tse E, Tracking in a cluttered environment with probabilistic data association, Automatica, Vol 11, Pages 451-460, 1975
- [19] Bar-Shalom Y, Update with out-of-sequence measurements in tracking. Exact solution IEEE transactions on aerospace and electronic systems 38,3 July 2002, Pages 769-778, 2002
- [20] Bar-Shalom Y, Chen H and Mallick M, One step solution for the multistep out-of-sequence measurement problem in tracking IEEE transactions on aerospace and electronic systems, Vol.40, No 1, Pages 27-37, January 2004
- [21] Bar-Shalom Y, Extension of the probabilistic data association filter to multitarget tracking, Proceedings of the fifth symposium on nonlinear estimation, San Diego, Pages 16-21, September 1974
- [22] Bar-Shalom Y, Mallick M, Chen H, Washburn R, One-step solution for the general out-of-sequence measurement problem in tracking, Proceedings of the 2002 IEEE aerospace conference, Volume 4, Pages 1551 – 1559, 2002
- [23] Bar-Shalom Y, On the track to track correlation problem, IEEE transactions on automatic control, Vol. 26, Issue 2, Pages 571-572, 1981

- [24] Barak A and La'adan O, The MOSIX Multicomputer Operating System for High Performance Cluster Computing. *Journal of Future Generation Computer Systems*, Pages 361-372, 1998
- [25] Benaskeur A R, Consistent fusion of correlated data sources, *IEEE 2002 28th Annual Conference of the Industrial Electronics Society*, Vol 4, Pages 2652 – 2656, November 2002.
- [26] Benet G, Simo J E, and Martinez M, A multisensor robot distributed architecture, *Proceedings of Symposium on Information Control in Manufacturing*, Volume 2, Pages 583-8, 1999
- [27] Blackman S S and Popoli R, *Design and analysis of modern tracking systems*. Boston. Artech House. 1999
- [28] Blackman S S, Association and fusion of multiple sensor data, Pages 187 – 218 *Multitarget Multisensor tracking: advanced applications*, Norwood MA, Artech House, Yaakov Bar-Shalom (Ed), 1990
- [29] Bloem E A., Blom H A P, Joint probabilistic data association methods avoiding track coalescence, *Proceedings of the 34th IEEE Conference on Decision and Control*, Vol 3, Pages 2752 –2757, 1995
- [30] Blom H A P and Bloem E A, Tracking multiple manoeuvring targets by joint combinations of IMM and PDA, *Proceedings of the 42nd IEEE conference on decision and control*, December 2003
- [31] Blom H A P, Bloem E A, Combining IMM and JPDA for tracking multiple maneuvering targets in clutter, *Proceedings of the Fifth International Conference on Information Fusion*. Volume 1, Pages 705 - 712, 2002
- [32] Blom H A P, Bloem E A, Interacting multiple model joint probabilistic data association avoiding track coalescence, *Proceedings of the 41st IEEE conference on decision and control*. Volume 3, Pages 3408 – 3415, 2002
- [33] Blom HAP and Bloem E A, Joint IMM-PDA Particle filter, *Proceedings of the sixth international conference of information fusion 2003*, Volume 2, Pages 785 – 792, 2003
- [34] Blom HAP and Bloem E A, Probabilistic Data Association Avoiding Track Coalescence, *IEEE Transactions on Automatic Control*, Volume 45, Issue 2, Pages 247-259, 2000

- [35] Booker L B, Improving the performance of genetic algorithms in classifier systems, in Proceedings of an international conference on genetic algorithms and their applications, Pages 80-92, J. J. Grefenstette (Editor), 1985
- [36] Bracio B R, Horn W and Moller D P F, Sensor fusion in biomedical systems, Engineering in medicine and biology society, Proceedings of the 19th annual international conference of the IEEE, Volume 3, Pages 1387 – 1390, 1997
- [37] Breiman L, Bagging Predictors, Machine Learning, Pages 123-140, 1996
- [38] Brooks RR, Ramarathan P and Sayeed AM, Distributed target classification tracking in sensor networks, Proceedings of the IEEE, Volume 91, Number 8, August 2003
- [39] Brown G, Diversity in neural network ensembles, PhD thesis, Department of Computer Science, The University of Birmingham, 2004
- [40] Brown G, Wyatt J L and Tino P, Managing diversity in regression ensembles, Journal of machine learning research, 6, Pages 1621—1650, 2005
- [41] Brown G, Wyatt J, Harris R, Yao X, Diversity creation methods a survey and categorization, Journal of Information Fusion, Volume 6, Issue 1, Pages 5-20, 2005
- [42] Bruckner J M, Scott H R W and Rea G R, Analysis of multimodal systems, IEEE transactions on aerospace and electronic systems, Volume AES-9, Issue 6, Pages 883-888, 1973
- [43] Buchberger M, Jorg K W and Puttkamer, Laser radar and sonar based world modelling and motion control for fast obstacle avoidance of the autonomous mobile robot MOBOT-IV, IEEE Intl. Conference on Robotics and Automation, Pages 534-539, 1993
- [44] Bui L T, Branke J, Abbas H A, Diversity as a selection pressure in dynamic environments, Proceedings of the 2005 conference on Genetic and evolutionary computation, poster session, Pages 1557 – 1558, 2005
- [45] Cappe O, Godsill SJ, Moulines E, An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo, Proceedings of the IEEE, Volume 95, Issue 5, Pages 899-924, 2007
- [46] Carpenter J, Clifford P, Fearnhead P, Improved particle filter for non-linear problems, IEE proceedings – Radar, sonar and navigation, Volume 146, Issue 1, Pages 2-7, 1999

- [47] Caruso M J, Applications of magnetic sensors for low cost compass systems, IEEE 2000 Position, Location and Navigation Symposium, Pages 174 – 184, 2000
- [48] Ceruti M G, Data management challenges and development for military information systems, IEEE transactions on knowledge and data engineering, Vol. 15, No. 5, Pages 1059-1068, 2003
- [49] Challa S and Koks D, Bayesian and Dempster-Shafer fusion, Sadhana, Vol.29, Number 2, Pages 145-174, April 2004
- [50] Challa S, Robin JE, Wang X, A Bayesian Solution and Its Approximations to Out-of-Sequence Measurement Problems, Information fusion, Volume 4, Issue 3, Pages 185-199, 2003
- [51] Challa S, Wang X, Legg J, Track-to-Track Fusion using Out-of-Sequence Tracks, Proceedings of the International Conference on Information Fusion, Pages 919-926, 2002
- [52] Chandra A, Yao X, Evolving hybrid ensembles of learning machines for better generalisation, Neurocomputing, Volume 69, Issues 7-9, Pages 686-700, 2006
- [53] Chandra A, Yao X, DIVACE: diverse and accurate ensemble learning algorithm, Proceedings of the fifth International Conference on Intelligent Data Engineering and Automated Learning, Lecture notes in Computer Science, Volume 3177, Springer, Pages 619-625, 2004
- [54] Chandra A, Yao X, Ensemble learning using multi-objective evolutionary algorithms, Journal of Mathematical Modelling and Algorithms, Volume 5, Number 4, Pages 417-445, 2006
- [55] Chang C C and Song K T, Ultrasonic sensor data integration and its application to environment perception, Journal of Robotic Systems, 13(10), Pages 663-677, 1996
- [56] Chaudhuri S P and Das S, Neural networks for data fusion, IEEE international conference on systems engineering, Pages 327-330, 1990
- [57] Chen B and Tugnait J K, An interacting multiple model fixed lag smoothing algorithm for Markovian switching systems, IEEE transactions on Aerospace and Electronic Systems, Volume 36, Issue 1, Pages 243 – 250, 2000
- [58] Chen L, Arambel P O and Mehra R K, Fusion under unknown correlation Covariance intersection revisited, IEEE transactions on automatic control, Volume 47, Issue 11, Pages 1879-1882, 2002

- [59] Chen L, Arambel P O and Mehra R K, Fusion under unknown correlation – Covariance intersection as a special case, Proceedings of the Fifth International Conference on Information Fusion, 2002, Pages 905- 912, vol. 2, 2002
- [60] Chen H, Yao X, Evolutionary random neural ensembles based on negative correlation learning, IEEE Congress on Evolutionary Computation, Pages 1468-1474, 2007
- [61] Chen Y M, Huang H C, Fuzzy logic approach to multisensor data association, Elsevier, Mathematics and Computers in Simulation 52, Pages 399–412, 2000
- [62] Chen Y, Yang B, Dong J, Abraham A, Time-series forecasting using flexible neural tree model, Elsevier Information Sciences, Volume 174, Issues 3-4, Pages 219 – 235, 2004
- [63] Chong C Y, Mori S, Chang K C and Barker W H. Architectures and algorithms for track association and fusion, IEEE aerospace and electronic systems magazine, Volume 15, Issue 1, Pages 5 – 13, January 2005
- [64] Clouqueur T, Ramanathan P, Saluja K K, Wang K-C, Value-Fusion versus Decision-Fusion for Fault-tolerance in Collaborative Target Detection in Sensor Networks, Proceedings of the 4th Annual conference on information fusion, Pages TuC2/25 – TuC2/30, 2001
- [65] Coello C A, An updated survey of GA-based multiobjective optimization techniques, ACM Computing Surveys (CSUR), Volume 32 , Issue 2 (June 2000), Pages 109 – 143, 2000
- [66] Cohoon J, Hedge U, Martin W and Richards D, Punctuated equilibria. A parallel genetic algorithm, Proceedings of the second international conference on genetic algorithms, Pages 148-154, 1987
- [67] Coraluppi S, Carthel C, Recursive track fusion for multi-sensor surveillance, Information fusion 5, Pages 23-33, 2004
- [68] Courtney JD and Jain AK, Neural network learning of variable grid-based maps for the autonomous navigation of robots, IEEE Intl Conference on Robotics and Automation, Pages 40-45, 1994
- [69] Dasgupta D, McGregor D R, Designing application specific neural networks using the structured genetic algorithm, International workshop on combinations of genetic algorithms and neural networks, Pages 87 – 96, 1992

- [70] De Cubber G, Sahl H, Decroos F, Sensor Integration on a Mobile Robot, ISMCR 2002: 12th International Symposium on Measurement and Control in Robotics, Pages 89 – 92, 2002
- [71] de Garis H, GenNets: Genetically programmed neural nets—Using the genetic algorithm to train neural nets whose inputs and/or outputs vary in time, Proceedings of the 1991 IEEE International Joint Conference on Neural Networks (IJCNN'91 Singapore), Volume 3, Pages 1391 - 1396, 1991
- [72] De Jong K A and Pollack J B, Multi-objective methods for tree size control, Genetic programming and evolvable machines, 4(3), Pages 211-233, 2003
- [73] De Jong K A, An analysis of the behavior of a class of genetic adaptive systems, PhD thesis, University of Michigan, 1975
- [74] De Jong K A, Watson R A and Pollack J B, Reducing bloat and promoting diversity using multi-objective methods, in Proceedings of the genetic and evolutionary computation conference, GECCO 01, L. Spector et al (Editors), 2001
- [75] DeAngelis C M, Whitney J E, The neurally inspired contact estimator (NICE), OCEANS '98 Conference Proceedings, Volume 3, 28 Sept.-1 Oct. 1998
Page(s):1619 -1623 vol 3, 1998
- [76] Deb K and Goldberg D E, An investigation of niche and species formation in genetic function optimization, Proceedings of the 3rd international conference on genetic algorithms, Pages 42-50, 1989
- [77] Deb K, Pratap A, Agarwal S, Meyarivan T, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, Volume 6, Issue 2, Pages 182 – 197, 2002
- [78] Dempster A P, A generalisation of Bayesian Inference, Journal of the Royal Statistical Society, Vol.30, Pages 205-247, 1968
- [79] Doucet, A.; Johansen, A.M ; A tutorial on particle filtering and smoothing: fifteen years later, Technical report, Department of Statistics, University of British Columbia, 2008
- [80] Drucker H, Cortez C, Jackel LD, LeCun Y and Vapnik V, Boosting and other ensemble methods, Neural computation, Volume 6, Pages 1289-1301, 1994
- [81] Duarte M, Hu Y H, Vehicle classification in distributed sensor networks, Journal of Parallel and Distributed Computing 64 (7), 826–838, 2004

- [82] Ehrman LM, Lanterman AD, Extended Kalman filter for estimating aircraft orientation from velocity measurements, *Institution of Engineering and Technology: Radar, Sonar & Navigation*, Volume 2, Issue 1, Pages 12-16, 2008
- [83] Errington I, Non-linear estimation in bearings only tracking of manoeuvring targets, *IEE Colloquium on State Estimation in Aerospace and Tracking Applications*, 2/1 – 2/2, 1989
- [84] Escamilla-Ambrosio P J, Mort N, Hybrid Kalman filter – fuzzy logic adaptive multisensor data fusion architectures, *Proceedings of 42nd IEEE conference on decision and control* December 2003
- [85] Escamilla-Ambrosio PJ, Mort N, Multisensor data fusion architecture based on adaptive Kalman filters and fuzzy logic performance assessment, *Proceedings of the fifth international conference on information fusion*, 2002, Volume 2, Pages 1542 – 1549, 2002
- [86] Eshelman L J and Schaffer J D, Preventing Premature Convergence in Genetic Algorithms by Preventing Incest, *Proceedings of the 4th International Conference on Genetic Algorithms*, Morgan Kaufmann, Pages 115-122, 1991
- [87] Fan T, Yang C Y, Mao S Y, Li S H, Multi-resolution multiple-model target tracking based on model mixing, *IEEE international radar conference*, Pages 81-86, 2000
- [88] Farian A, Lombardo P and Marselia M, Joint tracking and identification algorithms for multisensor data, *IEE Proceedings on Radar and Sonar Navigation*, Vol.149, No. 6, Pages 271-280, December 2002
- [89] Fitzgerald R J, Development of practical PDA logic for multitarget tracking by microprocessor, in *Multitarget multisensor tracking Advanced applications*, Yaakov Bar-Shalom (Editor), 1989
- [90] Flynn A M, Combining sonar and infrared sensors for mobile robot navigation, *International Journal of robotics research*, Pages 5-14, December 1988
- [91] Fogel D B, An introduction to evolutionary optimization, *IEEE transactions on neural networks*, volume 5, Pages 3-14, 1994
- [92] Freund Y, Boosting a weak learning algorithm by majority, *Proceedings of the Third Annual Workshop on Computational Learning Theory*, 1990
- [93] Francis E. H. Tay, Cao L, Application of support vector machines in financial time series forecasting, *Omega (the international journal of management science)*, Elsevier, Volume 29, Pages 309-317, March 2001

- [94] Friedlander D, Griffin C, Jacobson N, Phoha S and Brooks R, Dynamic agent classification using ad hoc mobile acoustic sensor network, EURASIP journal on applied signal processing, 2003.4, Pages 371-377, 2003
- [95] Gad A and Farooq M, Data fusion architecture for maritime surveillance, Proceedings of the Fifth International Conference on Information Fusion, Vol.1, Pages 448-455, 2002.
- [96] Gao J B and Harris C J, Some remarks on Kalman Filters for the multisensor fusion, Elsevier Information fusion, Volume 3, Issue 3, Pages 191-201, 2002
- [97] Gao J B, Harris C J, Some remarks on Kalman filters for the multisensor fusion, Elsevier Information Fusion 3 (2002) 191–201, 2001
- [98] Ghosh J and Holmberg R L, Multisensor Fusion using Neural Networks, Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing, Pages 812-815, 1990
- [99] Goldberg D E and Richardson J, Genetic algorithm with sharing for multimodal function optimization. In proceedings of the second International conference on genetic algorithms and their applications, J. J. Grefenstette (Editor), Pages 41-49, 1987
- [100] Gordon N J, Salmond D J, Smith A F M, Novel approach to non-linear/non-Gaussian Bayesian state Estimation, IEE Proceedings-F Vol. 140, No 2, 1993
- [101] Gorges-Schleuter M, ASPARAGOS: and asynchronous parallel genetic optimization strategy, proceedings of the second international conference on genetic algorithms, Pages 422-427, 1989
- [102] Greenwood G W, Training partially recurrent neural networks using evolutionary strategies, IEEE Transactions on Speech Audio Processing, Volume 5, Issue 2, Pages 192–194, 1997
- [103] Grime S and Durrant-Whyte H F, Data fusion in decentralized sensor networks, Control Engineering Practice, Vol.2, Issue 5, October 1994, Pages 849-863
- [104] Barbera H M, Skarmeta A G, Izquierdo M Z, Blaya J B, Neural Networks for Sonar and Infrared Sensors Fusion, Proceedings of the third international conference on information fusion, Volume 2, Pages WEB4/18 – WEB4/25, 2000
- [105] Hadzagic M, Michalska H, Jouan A, IMM-JVC and IMM-JPDA for closely maneuvering targets, Conference Record of the Thirty-Fifth Asilomar

- Conference on Signals, Systems and Computers, Vol 2, Pages 1278 - 1282
2001
- [106] Hajela P and Lee J, Constrained genetic search via scheme adaption. An immune network solution. *Structural optimization*, 12, 1, Pages 11-15, 1996
 - [107] Hajela P, Yoo J and Lee J, GA based simulation of immune networks Applications in structural optimization, *Engineering Optimization*, 29, Pages 131-149, 1987
 - [108] Hall D L and Llinas J, An introduction to multisensor fusion, *Proceedings of the IEEE*, Volume 85, Issue 1, Pages 6 - 23, January 1997
 - [109] Hall D. *Mathematical Techniques in Multisensor Data Fusion*, Artech House, 1992
 - [110] Hammond N J, Revolution in military affairs- does hydrography have a part to play?, *United States Hydrographic Conference 2001*, May 22-24, 2001
 - [111] Hancock P J B, Smith L S, GANNET: Design of a Neural Net for Face Recognition, *Lecture notes in computer science*, *Proceedings of the 1st workshop on parallel problem solving from nature*, Volume 496, Pages 292 – 296, 1991
 - [112] Hansen LK, Salamon P, Neural network ensembles, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 12, Pages 993-1001, 1990
 - [113] Hartman R, Hawkinson W and Sweeney K, Tactical underwater navigation system (TUNS), *2008 IEEE/ION Position, Location and Navigation Symposium*, Pages 898-911, 2008
 - [114] Hashem S, Optimal linear combinations of neural networks, *Neural networks*, Volume 10, Number 4, Pages 599-614, 1997
 - [115] Hernandez M L, Kirubarajan T, Bar-Shalom Y, Multisensor resource management using posterior Cramer-Rao lower bounds, *IEEE Transactions on Aerospace and Electronic Systems*, Volume 40, Issue 2, Pages 399-416, 2004
 - [116] Hernandez M L, Marrs A D, Maskell S, Orton M R, Tracking and fusion for wireless sensor networks, *Proceedings of the Fifth International Conference on Information Fusion*, Vol.2, Pages 1023- 1029, 2002
 - [117] Hilton R D, Martin D A and Blair W D. Tracking with time delayed data in multisensor systems, Document Number NSWCDD/TR-9e/351, *Naval Surface Warfare Center*, Dahlgren, 1993

- [118] Hinman M L, Some computational approaches for situation assessment and impact assessment, Proceedings of Fusion 2002, 1, Pages 687-693, 2002
- [119] Hintz K and McVey E, A measure of the information gain attributable to cuing, IEEE Transactions on Systems, Man and Cybernetics, Volume 21, Issue 2, Pages 434-442, 1991
- [120] Ho Y C and Lee R C K, A Bayesian approach to problems in stochastic estimation and control, IEEE transactions on automatic control, Volume 9, Issue 4, 1964
- [121] Hong H , Chong-zhao H, Hong-Yan Z, Rong W, Multi-target tracking based on multi-sensor information fusion with fuzzy inference, Control and decision, Volume 19, Number 3, Pages 272 – 276, 2004
- [122] Hong L and Scaggs T, Real time optimal multiresolutional sensor/data fusion, Proceedings of the 1993 IEEE International Conference on Robotics and Automation, Vol 2, pages 117-122, 2-6 May, 1993
- [123] Hong L, An interacting multi-pattern probabilistic data association (IMP-PDA) Algorithm for target tracking, IEEE transactions on automatic control, Volume 46, Number 8, Pages 1223-1236, August, 2001
- [124] Hong L, Cong S and Wicker D, Distributed multi-rate interacting multiple model (DRIMM) filtering with out-of-sequence GMTI data, Proceedings of the fifth international conference on information fusion, 2002, Volume 2, Pages 1054 – 1061, 2002
- [125] Hong L, Cong S and Wicker D, Multirate interacting multiple model (MRIMM) filtering with out-of-sequence GMTI data, IEE proceedings radar, sonar and navigation, Volume 150, Issue 5, Pages 333 – 343, 2003
- [126] Hong L, Cong S, and Wicker D, Distributed Multirate Interacting Multiple Model Fusion (DMRIMMF) With Application to Out-of-Sequence GMTI Data, 102 IEEE Transactions on Automatic Control, Vol. 49, No. 1, Pages 102-107, 2004
- [127] Hong L, Fusing multiresolutional data using filter banks, Proceedings of the American Control Conference, Vol.2, Pages 1309-1313, June 1994
- [128] Hong L, Multirate Interacting Multiple Model Filtering for Target Tracking Using Multirate Models, IEEE Transactions on Automatic Control, Vol.44, Issue 7, Pages 1326- 1340, July 1999

- [129] Hong L, Multiresolutional distributed filtering, IEEE transactions on automatic control Vol. 39 No. 4, Pages 853-856, April 1994
- [130] Hong L, Multiresolutional filtering using the wavelet transform, IEEE transactions on aerospace and electronic systems, Volume 29, Number 4, Pages 1244-1251, 1993
- [131] Hong L, Multiresolutional multiple-model target tracking, IEEE transactions on aerospace and electronic systems, Volume 30, Issue 2, Pages 518-524, 1994
- [132] Hong L, Multiresolutional tracking using the wavelet transform, Proceedings of the 32nd conference on decision and control, Pages 924-929, 1993
- [133] Hong L, Optimal multiresolutional distributed filtering, Proceedings of the 31st conference on decision and control, Volume 4, Pages 3105-3110, 1992
- [134] Hong L, Two level JPDA-NN and NN-JPDA Tracking algorithms, Proceedings of the American control conference, Volume 1, Pages 1057-1061, 1994
- [135] Hong L, Werthmann J R, Bierman G S and Wood R A, A multiresolutional approach to target tracking, Proceedings of the IEEE 1993 national aerospace and electronics conference, NAECON 1993, Volume 1, Pages 388 – 392, 1993
- [136] Horling B, Mailer R, Shen J, Vincent R and Lesser V, Using autonomy, organizational design and negotiation in a distributed sensor network, In Distributed Sensor Networks: a multiagent perspective, Pages 139-184, Edited by Victor Lesser, Charles L. Ortiz, Jr and Milind Tambe, Kluwer Academic publishers group, 2003
- [137] Hu X, Schon TB and Ljung L, A Basic Convergence Result for Particle Filtering, IEEE Transactions on Signal Processing, Volume 56, Issue 4, Pages 1337-1348, 2008
- [138] Hue C, Le Cadre J P, Perez P, Sequential Monte Carlo methods for multiple target tracking and data fusion, IEEE transactions on signal processing, Volume 50, Number 2, Pages 309 – 325, 2002
- [139] Hurley M B, An information theoretic justification for covariance intersection and its generalization, Proceedings of the Fifth International Conference on Information Fusion, vol. 1, Pages 505- 511, 2002.
- [140] Hwang I, Balakrishnan H, Roy K, Tomlin C, Multiple-Target Tracking and Identity Management in Clutter, with Application to Aircraft Tracking, Proceedings of the 2004 control conference, Volume 4, Pages 3422 – 3428, 2004

- [141] Ichimura T, Takano T, and Tazaki E, Reasoning and learning method for fuzzy rules using neural networks with adaptive structured genetic algorithm, Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics, Part 4, Pages 3269–3274, 1995
- [142] Islam M M, Yao X and Murase K, A constructive algorithm for training cooperative neural network ensembles, IEEE Transactions on Neural Networks, Volume 14, Issue 4, Pages 820 – 834, 2003
- [143] Iyengar S S & Wu O, Computational Aspects of Distributed Sensor Networks, Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks, (ISPAAN.02), Pages 19-26, 2002
- [144] Jacobs R A, Jordan M I, Nowlan S J and Hinton G E, Adaptive mixtures of local experts, Neural Computation, Volume 3, Pages 79 – 87, 1991
- [145] Jakobson G, Lewis L and Buford J, An Approach to integrated cognitive fusion, Proceedings of the 7th International Conference on Information Fusion, Pages 1210-1217, 2004
- [146] Janson D J and Frenzel J F, Training product unit neural networks with genetic algorithms, IEEE Expert, Volume 8, Issue 5, Pages 26–33, 1993
- [147] Jiu J, Tan Y and Yang X, A framework for target identification via multisensor data fusion, proceedings of the 2003 international symposium on intelligent control, Pages 450-454, 2003
- [148] Jordan M I, and Jacobs R A, Hierarchical mixtures of experts and the EM algorithm, Neural Computation, Volume 6, Pages 181-214, 1994
- [149] Judd K, Small M, Towards long-term prediction, Elsevier Physica D, Volume 136, Pages 31–44, 1999
- [150] Julier S J and Uhlmann J K, A New Extension of the Kalman Filter to nonlinear Systems, Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management II, SPIE, Volume 3068, 1997.
- [151] Julier S J and Uhlmann J K, A non-divergent algorithm in the presence of unknown correlation, Proceedings of the American Control Conference, Volume 4, Pages 2369-2373, 1997
- [152] Kacalenga R, Erickson D and Palmer D, Voting fusion for landmine detection, IEEE Aerospace and electronic systems magazine, Volume 18, Number 8, Pages 13-19, August 2003

- [153] Kalman R E, A new approach to linear filtering and prediction problems, Transactions of the American Society of Mechanical Engineers – Journal of basic engineering, Volume 82, Pages 35-45, 1960
- [154] Kantz H, Schreiber T, Nonlinear time series analysis, Cambridge University Press, 1999
- [155] Kester L, Theil A, Fusion of radar and EO sensors for surveillance, Proceedings of the Third International Conference on Information Fusion, 2000. FUSION 2000, Vol.1, Pages TUD1/3- TUD1/9, July 2000
- [156] Khawsuk W and Pao L Y, Decorrelated state estimation for distributed tracking of interacting targets in cluttered environments, Proceedings of the American Control Conference, Anchorage, Volume 4, Pages 3208-3214, 2002
- [157] Khawsuk W, Pao L Y, Decorrelated state estimation for distributed tracking using multiple sensors in cluttered environments, Proceedings of the 2003 American Control Conference, Vol.4, Pages 3208-3214, 2003.
- [158] Kim D, Kim C, Forecasting time series with genetic fuzzy predictor ensemble, IEEE Transactions on Fuzzy Systems, Volume 5, Issue 4, Pages 523 – 535, 1997
- [159] Kittler J and Alkoot F M, Sum versus vote fusion in multiple classifier systems, IEEE transactions on pattern analysis and machine intelligence, volume 25, number 1, Pages 110-115, 2003
- [160] Kittler J and Messer K, Fusion of multiple experts in multimodal biometric personal identity verification systems, Proceedings of the 2002 12th IEEE workshop on neural networks for signal processing, Pages 3-12, 2002
- [161] Kittler J, Multi-sensor integration and decision level fusion, A DERA/IEE Workshop on Intelligent sensor processing (Ref. No. 2001/050) , Pages 6/1 – 6/6, 2001
- [162] Klein L A, A Boolean algebra approach to Multiple sensor voting fusion, IEEE transactions on aerospace and electronic systems, Volume 29, Number 2, Pages 317-327, 1993
- [163] Kohavi R and Wolpert D H, Bias plus variance decomposition for zero-one loss functions, Proceedings of the 13th international conference on machine learning, pages 275-283, 1996

- [164] Kokar M and Kim K H, Review of multisensor data fusion architectures and techniques, Proceedings of the 1993 IEEE International Symposium on Intelligent Control, Pages 261-266, 1993
- [165] Koza J R and Rice J P, Genetic generation of both the weights and architecture for a neural network, Proceedings of the IEEE International Joint Conference on Neural Networks, Volume 2, Pages 397-404, 1991
- [166] Krieg M L, Joint multi-sensor kinematic and attribute tracking using Bayesian belief networks, Proceedings of the sixth international conference of information fusion, Volume 1, Pages 17-24, 2003
- [167] Krough A and Vedelsby J, Neural network ensembles, cross validation, and active learning, Advances in Neural Information Processing Systems 7, Pages 231-238, 1995
- [168] Kuncheva L I, Combining pattern classifiers, methods and algorithms', Wiley, 2004
- [169] La Scala B, Farina A, Effects of cross-covariance and resolution on track association, Proceedings of the third international conference on information fusion, Volume 2, Pages WED1/10 – WED1/16, 2000
- [170] La Scala BF and Farina A, Choosing a track association method, Information fusion 3, Pages 119-133, 2001
- [171] Larsen T D, Andersen N A, Ravn O, Poulson N K, Incorporation of time delayed measurements in a discrete-time Kalman filter, Proceedings of the 37th IEEE Conference on Decision and Control, Vol.4, Pages 3972-3977, 1998
- [172] Laumanns M, Thiele L, Deb K, and Zitzler E, Combining convergence and diversity in evolutionary multiobjective optimization, Evolutionary Computation, Volume 10, Number 3, Pages 263 – 282, 2002
- [173] Legg S and Hutter M, Fitness Uniform Deletion: A Simple Way to Preserve Diversity, Proceedings of the 2005 conference on genetic and evolutionary computation, Pages 1271 – 1278, 2005
- [174] Leung Y, Gao Y and Xu Z-B, Degree of Population Diversity – A perspective on Premature Convergence in Genetic Algorithms and its Markov Chain Analysis, IEEE Transactions on Neural Networks, Volume 8, Number 5, Pages 1165-1176, 1997

- [175] Li T, Ishwar K S, Optimal multiple level decision fusion with distributed sensors, IEEE transactions on Aerospace and Electronic Systems, Vol 29, Issue 4, October 1993
- [176] Liggins M E, Chong C-Y, Kadar I, Alford M G, Vannicola V and Thomopoulos S, Distributed fusion architectures and algorithms for target tracking, Proceedings of the IEEE, Vol 85, No 1, January 1997
- [177] Linsday D and Cox S, Effective Probability Forecasting for Time Series Data Using Standard Machine Learning Techniques, S Singh et al (Eds.): ICAPR 2005, LNCS 3686, Pages 35–44, 2005.
- [178] Liu J, Chu M, Liu J, Reich J and Zhao F, Distributed state representation for tracking problems in sensor networks, Third international symposium on information processing in sensor networks, Pages 234-242, 26-27 April 2004
- [179] Liu Y and Yao X, Negatively correlated networks can produce best ensembles, Australian Journal of Intelligent Information Processing Systems, Vol 4, Pages 176-185, 1998
- [180] Liu Y and Yao X, A cooperative ensemble learning system, Proceedings of the 1998 IEEE international conference on Neural Networks (IJCNN '98), Pages 2202-2207, 1998
- [181] Liu Y and Yao X, Ensemble learning via negative correlation, Neural networks, 12(10), Pages 1399-1404, 1999
- [182] Liu Y and Yao X, Evolutionary design of artificial neural networks with different nodes, Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), pages 670–675, 1996
- [183] Liu Y, Yao X and Higuchi T, Evolutionary Ensembles with Negative Correlation Learning, IEEE Transactions on Evolutionary Computation, Volume 4, Number 4, Pages 380-387, 2000
- [184] Liu Y, Negative correlation learning and evolutionary design of neural network ensembles, PhD thesis, School of Computer Science, University College, University of New South Wales, 1999
- [185] Llinas J, Studying the complexities in distributed object tracking systems, 2003 IEEE International Conference on Systems, Man and Cybernetics, Volume 2, Pages 2035- 2041, 2003
- [186] Lobbia R and Kent M, Data fusion of decentralized local tracker outputs, IEEE Transactions on Aerospace and Electronic Systems, Vol.30, No. 3, July 1994

- [187] Looney C G and Liang L R, Cognitive situation and threat assessments of ground battlespaces, Elsevier Information Fusion, 4, Pages 297-308, 2003
- [188] Luo R C, Yih C C, Su K L, Multisensor fusion and integration: Approaches, applications, and future research directions, IEEE Sensors Journal, Vol.2, Issue 2, Pages 107-119, April 2002
- [189] Mallick M and Marrs A, Comparison on of the KF and particle filter based out-of-sequence measurement filtering algorithms, Proceedings of the sixth international conference of information fusion, 2003, Volume 1, pages 422 – 429, 2003
- [190] Mallick M, Coraluppi S and Carthel C, Advances in asynchronous and decentralized estimation. Proceedings of the 2001 IEEE Aerospace conference, Big Sky MT, Volume 4, Pages 1873-1888, March 2001
- [191] Mallick M, Kirubarajan T, Arulampalam S, Out-of-sequence measurement processing for tracking ground target using particle filters, IEEE aerospace conference proceedings, Volume 4, Pages 1809 – 1818, 2002
- [192] Mallick M, Krant S J, Bar-Shalom Y, Multi-sensor Multi-target Tracking using Out-of-Sequence Measurements, Proceedings of the Fifth International Conference on Information Fusion, Vol.1, Pages 135- 142, 2002.
- [193] Maniezzo V, Genetic evolution of the topology and weight distribution of neural networks, IEEE Transactions on Neural Networks, Volume 5, Pages 39–53, 1994
- [194] Manyika J and Durrant-Whyte H, Data fusion and sensor management a decentralized information-theoretic approach, Ellis Horwood, 1994
- [195] Marin F J and Sandoval F, Genetic synthesis of discrete time recurrent neural network, Proceedings of the International Workshop Artificial Neural Networks (IWANN'93), Lecture Notes in Computer Science, vol. 686. Berlin, Germany: Springer-Verlag, 1993, Pages 179–184, 1993
- [196] Marrs A D, Reed C M, Webb A R and Webber H C, Data incest and symbolic information processing, technical report, United Kingdom Defence Evaluation and Research Agency, March 1999
- [197] Maybeck P S, Stochastic Models, Estimation and Control, Volume 1, Mathematics in Science and Engineering, Volume 141-1, Academic Press, 1979

- [198] Mazor E, Averbuch A, Bar-Shalom Y, Dayan J, Interacting multiple model methods in target tracking. a survey, IEEE transactions on aerospace and electronic systems, Volume 34, Number 1, Pages 103-123, January 1998
- [199] McDonnell J R and Waagen D, Evolving recurrent perceptrons for time-series modeling, IEEE Transactions on Neural Networks, Volume 5, Issue 1, Pages 24–38, 1994
- [200] McGinnity S and Irwin G W, Multiple model bootstrap filter for manoeuvring target tracking, IEEE Transactions on aerospace and electronic systems Vol.36 no 3, July 2000
- [201] McLaughlin S P, Evans R J, Krishnamurthy V, Data incest removal in a survivable estimation fusion architecture, Proceedings of the Sixth International Conference of Information Fusion, 2003 Vol 1, Pages 229 – 236, 2003
- [202] McLaughlin S P, Krishnamurthy V, Challa S, Managing data incest in a distributed sensor network, Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol.5, Pages 269-272, 2003
- [203] Mitchell M, An introduction to genetic algorithms, MIT Press, 1996
- [204] Moody J and Darken C, Fast learning in networks of locally tuned processing units, Neural computation, Vol.1, Number 2, Pages 281-294, 1989
- [205] Moody S A, Challenges in building scalable network centric real time information dissemination systems, Proceedings of the 6th IEEE international symposium on object oriented real time distributed computing, Pages 203-210, 2003
- [206] Moore J, Keiser T, Brooks R, Phoha S, Friedlander D, Koch J, Reggio A and Jacobson N, Tracking targets with self-organising distributed ground sensors, Proceedings of the 2003 Aerospace conference, Volume 5, Pages 2113 – 2123, 2003
- [207] Mudigonda N R, Kacalenga R, Erickson D, The application of Dempster-Shafer theory for landmine detection, Multisensor, multisource information fusion, architectures, algorithms and applications, Proceedings of the SPIE, Volume 5099, Pages103-112, 2003
- [208] Murphy R R, Dempster-Shafer theory for sensor fusion in autonomous mobile robots, IEEE transactions on Robotics and Automation, Volume 14, Issue 2, Pages197-206, April 1998

- [209] Murphy R R, Sensor fusion, in Handbook of brain theory and neural networks, Bradford Book, 2003
- [210] Naftaly U, Intrator N and Horn D, Optimal ensemble averaging of neural networks, Network: Computation in neural systems, Volume 8, Issue 3, Pages 283-296, 1997
- [211] Nettleton EW and Durrant-Whyte H, Delayed and asequent data in decentralized sensing networks Proceedings of SPIE Conference, Vol.4571, Pages 1-9, 2001
- [212] Nicholson D, Lloyd C, Julier S, Uhlmann J, Scalable distributed data fusion, Proceedings of the Fifth International Conference on Information Fusion, Vol.1, Pages 630-635, 2002.
- [213] Niehsen W and Bosch R, Information fusion based on fast covariance intersection filtering, Proceedings of the Fifth International Conference on Information Fusion, Vol. 2, Pages 901- 904 Vol.2, 2002
- [214] Nilsson N J, Principles of artificial intelligence, Palo Alto, Tioga, 1980
- [215] Niu R, Varshney P, Mehrotra K, and Mohan C, Temporal fusion in multi-sensor target tracking systems, Proceedings of the Fifth International Conference on Information Fusion, Vol.2, Pages 1030- 1037, 2002
- [216] Nsakanda A L, Price W L, Diaby M and Gravel M, Ensuring population diversity in genetic algorithms: A technical note with application to the cell formation problem, European Journal of Operational Research, Volume 178, Issue 2, 16 April 2007, Pages 634-638
- [217] O'Neil S D and Pao L Y, Multisensor Fusion Algorithms for Tracking, Proceedings of the 1993 American Control Conference, 1993
- [218] Ortiz Jr C L, Rauenbusch T W, Hsu E and Vincent R, Dynamic resource-bounded negotiation in non-additive domains, In Distributed Sensor Networks: a multiagent perspective, Pages 61-107, Edited by Victor Lesser, Charles L. Ortiz, Jr and Milind Tambe, Kluwer Academic publishers group, 2003
- [219] Orton M and Marrs A, A Bayesian approach to multi-target tracking and data fusion with out of sequence measurements, IEE international seminar target tracking: algorithms and applications, Volume 1, Pages 15/1-15/5, 2001
- [220] Optiz D W, Shavlik J W, Actively searching for an effective neural-network ensemble, Connection science, Volume 8, Number 3/4, Pages 337-353, 1996

- [221] Pace D W, Mallick M, Eldredge W, Spectral feature-aided multi-target multi-sensor passive sonar tracking, Proceedings of IEEE Oceanic Engineering Society Conference 2003, Volume 4, Pages 2120-2126, 2003
- [222] Pao L Y and Kalandros M, Algorithms for a class of distributed architecture tracking, Proceedings of the American Control Conference, Volume 3, Pages 1434-1438, 1997
- [223] Pao L Y, Distributed multisensor fusion, American Institute of Aeronautics and Astronautics, Pages 82-91, 1994
- [224] Partridge D and Krzanowski W, Software diversity practical statistics for its measurement and exploration, Information and software technology, 39, Pages 707-717, 1997
- [225] Pattipati K R, Deb S, Bar-Shalom Y and Washburn R B, A new relaxation algorithm and passive sensor data association, IEEE transactions on automatic control, Volume 37, Issue 2, Pages 198-213, 1992
- [226] Penny D E, Multisensor management for passive target tracking in an anti submarine warfare scenario, Technical report, United Kingdom Defence Evaluation and Research Agency, 1999
- [227] Penny D E, Sensor management in an ASW data fusion system, Sensor fusion: Architectures, Algorithms and Applications III, Proceedings of the SPIE, Volume 3719, 1999
- [228] Perez C A and Holzmann C A, Improvements on handwritten digit recognition by genetic selection of neural network topology and by augmented training, Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics, Part 2, Pages 1487-1491, 1997
- [229] Pettey C, Leuze M and Grefenstette J, A parallel genetic algorithm, Proceedings of the Second International Conference on Genetic Algorithms, Pages 151-161, 1987
- [230] Phillips I, Data Synthesis for Passive Sonar, Journal of Defence Science, Vol 11, Num 4, October 1996
- [231] Pomerleau D A, ALVINN. An Autonomous Land Vehicle in a Neural Network, Advances in Neural information processing Systems I, (D. S. Touretzky – editor), Morgan Kaufmann, 1988
- [232] Porto V W, Fogel D B, and Fogel L J, Alternative neural network training methods, IEEE Expert, Volume 10, Pages 16-22, 1995

- [233] Prados D L, Training multilayered neural networks by replacing the least fit hidden neurons, Proceedings of the IEEE SOUTHEASTCON' 92, Volume 2, Pages 634–637, 1992
- [234] Tanese R, Distributed genetic algorithms, in J. D Schaffer (Editor) Proceedings of the third international conference on genetic algorithms, Pages 257-263, 1989
- [235] Ragg T and Gutjahr S, Automatic determination of optimal network topologies based on information theory and evolution, Proceedings of the 23rd EUROMICRO Conference, IEEE Computer Society, Pages 549–555, 1997
- [236] Raghavendra C S, Meesookho C, Narayanan S, Collaborative classification applications in sensor networks, Second IEEE Sensor Array and Multichannel Signal Processing Workshop, August 2002.
- [237] Rietman E A, Maintaining population diversity in a genetic algorithm: an example in developing control schemes for semiconductor manufacturing, Proc. SPIE Vol. 3165, p. 25-35, Applications of Soft Computing, Bruno Bosacchi; James C. Bezdek; David B Fogel; Eds., 1997
- [238] Rosen B E, Ensemble learning using decorrelated neural networks, Connection science, special issue on combining artificial neural networks, 8 (3&4), Pages 373-384, 1996
- [239] Ruixin Niu, Pramod Varshney, Kishan Mehrotra, and Chilukuri Mohan, Sensor staggering in multi-sensor target tracking systems, 2003 IEEE Radar conference, Volume 2, Pages 354-361, 2003
- [240] Salerno J, Hinman M and Boulware D, Building a framework for situational awareness, Proceedings of the 7th International Conference on Information Fusion, Pages 219-226, 2004
- [241] Santos V, Gonzalves J and Vaz F, Perception maps for the local navigation of a mobile robot: a neural network approach, IEEE Intl. Conference on Robotics and Automation, Pages 2193-2198, 1994
- [242] Saravanan N and Fogel D B, Evolving neural control systems, IEEE Expert, Volume 10, Pages 23–27, 1995
- [243] Sareni B, Krahenbuhl L, Fitness sharing and niching methods revisited, IEEE Transactions on Evolutionary Computation, Volume 2, Issue 3, Pages: 97-106, 1998
- [244] Sasaki D, Yang G and Obayashi S, Automated Aerodynamic Optimization System for SST Wing-Body Configuration, Transactions of the Japan Society

- for Aeronautical and Space sciences, volume 46, issue 154, Pages 230 – 237, 2004
- [245] Sasiadek J Z, Hartana P, Sensor data fusion using Kalman filter, Proceedings of the Third International Conference on Information Fusion, 2000. FUSION 2000, Vol.2, Pages WED5/19- WED5/25, July 2000
- [246] Sasiadek J Z, Sensor fusion, Annual reviews in control, Volume 26, Issue 2, Pages 203-228, 2002
- [247] Schaffer J D, Curuana R A, Eschelmann L J, Using genetic search to exploit the emergent behavior of neural networks, Physica D, Volume 43, Issue 1-3, Pages 244 – 248, 1990
- [248] Schultz A and Wechsler H, Data fusion in neural networks via computational evolution, Proceedings of the 1994 IEEE International Conference on Neural Networks. Part 5, Pages 3044–3049, 1994
- [249] Seignez E, Kieffer M, Lambert A, Walter E and Maurin T, Real-time Bounded-error State Estimation for Vehicle Tracking, The International Journal of Robotics Research, Volume 28, Number 1, Pages 34-48, 2009
- [250] Shafer G, A mathematical theory of evidence, Princeton University Press, 1976
- [251] Schapire R, Strength of Weak Learnability, Journal of Machine Learning, Volume 5, Pages 197-227, 1990
- [252] Sharkey A J C, On combining artificial neural nets, Connection science, Volume 8, Number 3/4, Pages 299-314, 1996
- [253] Sharkey A J C and Sharkey N E, Combining diverse neural nets, Knowledge Engineering review, Volume 12, Number 3, Pages 1 – 17, 1997
- [254] Smith R E, Forrest S, Perelson A S, Searching for Diverse, Cooperative Populations with Genetic Algorithms, Evolutionary Computation, Vol. 1, No 2, Pages 127-149, 1993
- [255] Snijders P, de Jong E D, de Boer B, Weissing F, Multi-Objective Diversity Maintenance, Genetic and Evolutionary Computation Conference (GECCO'06), poster session, Pages 1429 – 1430, 2006
- [256] Soh L, Tsatoulis C, Sevay H, A satisficing, negotiated and learning coalition formation architecture, In Distributed Sensor Networks: a multiagent perspective, Pages 109-138, Edited by Lesser V, Ortiz C L and Tambe M, Kluwer Academic publishers group, 2003

- [257] Srinivas M and Patnaik L M, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE transactions on system man and cybernetics, Volume 24, Issue 4, Pages 656-667, 1994
- [258] Srinivas M and Patnaik L M, Learning neural network weights using genetic algorithms—Improving performance by search-space reduction, Proceedings of the 1991 IEEE International Joint Conference on Neural Networks (IJCNN'91 Singapore), volume 3, Pages 2331–2336, 1991
- [259] Stein F, Garska J and McIndoo P, Network-centric warfare: Impact on army operations, EUROCOMM 2000 Information Systems for Enhanced Public Safety and Security. IEEE/AFCEA, Pages 288-295, 2000
- [260] Stelios CA, Thomopoulos and Zhang L, Distributed filtering with random sampling and delay, Proceedings of the 27th IEEE Conference on Decision and Control, Vol.3, Pages 2348 – 2353, 7-9 Dec. 1988
- [261] Storm S A, An investigation into the effects of correlation in sensor fusion, Thesis, Department of the air force, Air University, United States Air Force Institute of Technology, March 2003
- [262] Sugihara G and May R M, Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series, Nature 344, Pages 734-741, 19th April 1990
- [263] Sun S-L, Deng Z-L, Multi-sensor optimal information fusion Kalman Filter, Elsevier Automatica 40 (2004) Pages 1017 -1023
- [264] Sun S-L, Multi-sensor Optimal information fusion Kalman filter for discrete multichannel ARMA signals, Proceedings of the 2003 IEEE International symposium on intelligent control, Pages 377-382, 2003
- [265] Sun S-L, Multi-sensor optimal information fusion Kalman filters with applications, Elsevier, Aerospace Science and Technology 8 (2004) Pages 57–62
- [266] Tamas L, Lazea G, Robotin R, Marcu C, Herle S and Szekely Z, State estimation based on Kalman filtering techniques in navigation, IEEE International Conference on Automation, Quality and Testing, Robotics 2008, Volume 2, Pages 147-152, 2008
- [267] Tang K S, Chan C Y, Man K F, and Kwong S, Genetic structure for NN topology and weights optimization, Proceedings of the first IEE/IEEE

- International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95), Pages 250–255, 1995
- [268] Tang Z, Fishwick P A, Feed-forward neural nets as models for time-series forecasting, *ORSA journal of computing*, Volume 5, pages 374-385, 1993
- [269] The secretary of state for defence, Delivering security in a changing world, defence white paper, UK Ministry of Defence, December 2003.
- [270] Thrun S, Learning metric-topological maps for indoor mobile robot navigation, *Artificial Intelligence*, Volume 99, Issue 1, Pages 21-71, 1998
- [271] Toffolo A and Benini E, Genetic diversity as an objective in multi-objective evolutionary algorithms, *Evolutionary computation*, Volume 11, Number 2, Pages 151 – 167, 2003
- [272] Trailovic L, Pao L Y, Variance Estimation and Ranking of Gaussian Mixture Distributions in Target Tracking Applications, *Proceedings of the 41st IEEE Conference on Decision and Control*, Vol.2, Pages 2195-2201, 2002
- [273] Tumer K and Ghosh J, Analysis of decision boundaries in linearly combined neural classifiers, *Pattern Recognition*, 29(2), Pages 341-348, 1996
- [274] U.S. Department of Defense, Data Fusion Subpanel of the Joint Directors of Laboratories, Technical Panel for C3, *Data fusion lexicon*, 1991.
- [275] Uhlmann J K, Covariance consistency methods for fault-tolerant distributed data fusion, *Information Fusion* 4, Pages 201–215, 2002
- [276] Uhlmann J K, Julier S, Durrant-Whyte H F, A culminating theory in the theory and practice of data fusion, filtering and decentralized estimation, Technical report, Covariance Intersection Working Group, 1997
- [277] van der Merwe R, Doucet A, de Freitas N and Wan E, The Unscented Particle Filter, *Advances in Neural Information Processing Systems (NIPS13)*, MIT Press, Eds. T. K. Leen, T. G. Dietterich and V. Tresp, December 2000
- [278] van der Merwe R, Wan E A, and Julier S, Sigma-Point Kalman Filters Nonlinear Estimation and Sensor Fusion - Applications in Integrated Navigation, *AIAA Guidance Navigation and Controls Conference*, March, 2004
- [279] Venugopal V and Narendran T T, A genetic algorithm approach to the machine-component grouping problem with multiple objectives, *Computers and Industrial Engineering* 22 (4), Pages 469–480., 1992

- [280] Veres G V and Norton J P, Improved particle filter for multitarget-multisensor tracking with unresolved applications, IEE target tracking: algorithms and applications, Volume 1, Pages 12/1 – 12/5, 2001
- [281] Vonk E, Jain L C, and Johnson R, Using genetic algorithms with grammar encoding to generate neural networks, Proceedings of the 1995 IEEE International Conference on Neural Networks, Part 4, Pages 1928–1931, 1995
- [282] Waite A D, Sonar for practising engineers, Third edition, John Wiley and Sons, 1998
- [283] Wallenius K, Support for Situation Awareness in Command and Control, Proceedings of the 7th International Conference on Information Fusion, Pages 1117-1124, 2004
- [284] Waltz E and Llinas J, Multisensor data fusion, Artech House, 1990
- [285] Wang G H, Mao S Y, Liu Y J, Triple threshold radar-to-ESM correlation algorithm when each radar track is specified by a different number of measurements, IEE Proceedings on Radar Sonar Navigation, Volume 147, Issue 4, Pages 177-181, 2000
- [286] Wang H, Pottie G, Yao K and Estrin D, Entropy-based sensor selection heuristic for target localisation, Proceedings of the third international symposium on information processing in sensor networks, Pages 36-45, 2004
- [287] Wang X and Challa S, Augmented state IMM-PDA for OOSM solution to maneuvering target tracking in clutter, Proceedings of the International Radar Conference, pages 479- 485, 2003.
- [288] Weigend A S, Huberman B A, Rumelhart D E, Predicting the future: A connectionist approach, International Journal of Neural systems, Volume 1, No 3, Pages 193-209, 1990
- [289] Whitley D and Starkweather T, Genitor II: A distributed genetic algorithm, Journal of Experimental and theoretical artificial intelligence, 2(3), Pages 189-214, 1990
- [290] Whitley D, Starkweather T, Bogart C, Genetic algorithms and neural networks: Optimizing connections and connectivity, Parallel computing, Volume 14, number 3, Pages 347 – 361, 1990
- [291] Whitley D, The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best, in Proceedings 3rd International

- Conference on Genetic Algorithms and Their Applications, Pages 116–121, 1989
- [292] Whitworth C C, Cross-entropy based pruning of the hierarchical mixtures of experts, Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing, Pages 375-383, 1997
- [293] Wieland A P, Evolving neural network controllers for unstable systems, Proceedings 1991 IEEE International Joint Conference on Neural Networks (IJCNN'91 Seattle), Volume 2, Pages 667–673, 1991
- [294] Willner D, Chang C B and Dunn K P, Kalman Filter algorithms for a multisensor system, in Proc 15th IEEE conference on decision control, December 1976, Volume 15, pages 570-574, 1976
- [295] Winter M and Favier G, A neural network for data association, Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol.2, Pages 1041 - 1044, 15-19 March 1999
- [296] Wu H, Siegel M, Stiefelhagen R, Yang J, Sensor Fusion Using Dempster-Shafer Theory, IEEE Instrumentation and Measurement Technology Conference Anchorage, 2002
- [297] Wu K H, Chen C H, and Lee J D, Cache-genetic-based modular fuzzy neural network for robot path planning, Proceedings of the 1996 IEEE International Conference on Systems, Man and Cybernetics, Part 4, Pages 3089–3094, 1996.
- [298] Xinhan H and Min W, Multi-sensor data fusion structures in autonomous systems: a review, 2003 IEEE International Symposium on Intelligent Control, Pages 817-821, 2003
- [299] Xiong N and Svensson P, Multi sensor management for information fusion: issues and approaches, Elsevier Information fusion 3 (2002) Pages163-186
- [300] Yadgar O, Kraus S and Ortiz C L, Scaling-up distributed sensor networks. cooperative large-scale mobile-agent organizations, In Distributed Sensor Networks: a multiagent perspective, Pages 185-218, Edited by Lesser V, Ortiz, C L and Tambe M, Kluwer Academic publishers group, 2003
- [301] Yan W, Clack C D, Behavioural GP diversity for dynamic environments: an application in hedge fund investment, Proceedings of the 8th annual conference on Genetic and evolutionary computation, Pages 1817 – 1824, 2006

- [302] Yan W, Zhu Z, and Hu R, Hybrid genetic/BP algorithm and its application for radar target classification, Proceedings of the 1997 IEEE National Aerospace and Electronics Conference, NAECON. Part 2, Pages 981–984, 1997
- [303] Yao X and Liu Y, Ensemble structure of evolutionary artificial neural networks, Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, Pages 659–664, 1996
- [304] Yao X and Shi Y, A preliminary study on designing artificial neural networks using co-evolution, in Proceedings of the IEEE International Conference on Intelligent Control and Instrumentation, Pages 149–154, 1995
- [305] Yao X, Evolving Artificial Neural Networks, Proceedings of the IEEE, Volume 87, Number 9, Pages 1423 – 1447, 1999
- [306] Yao X, The importance of maintaining behavioral link between parents and offspring, Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, Pages 629–633, 1997
- [307] Yardim C, Gerstoft P and Hodgkiss WS, Tracking Refractivity from Clutter Using Kalman and Particle Filters, IEEE Transactions on Antennas and Propagation, Volume 56, Issue 4, 2008
- [308] Yu L, Wang S, Lai K K, A novel nonlinear ensemble forecasting model incorporating GLAR and ANN for foreign exchange rates, Elsevier Computers and Operations Research, Volume 32, Issue 10, Pages 2523 – 2541, 2004
- [309] Yu N-H and Yin Y, Multiple level parallel decision fusion model with distributed sensors based on Dempster-Shafer evidence theory, Proceedings of the second international conference on machine learning and cybernetics, Volume 5, Pages 3104-3108, 2003
- [310] Yua Y, Cheng Q, Particle filters for maneuvering target tracking problem, Science Direct, Signal Processing, Volume 86, Issue 1, Pages 195-203, 2006
- [311] Zadrozny B, Elkan C, Transforming classifier scores into accurate multiclass probability estimates, Conference on knowledge discovery in data, proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and mining, Pages 694-699, 2002
- [312] Zemke S, Ensembles in practice. Prediction, estimation, multifeature and noisy data, Proceedings of HIS-2002, Page 1, 2002

- [313] Zhang G P, Berardi V L, Time series forecasting with neural network ensembles: an application for exchange rate prediction, The journal of the operational research society, Volume 52, Number 6, Pages 652 – 664, 2001
- [314] Zhang KS, Li XR and Zhu YM, Optimal update with out of sequence measurements, IEEE transactions on signal processing, Volume 53, Issue 6, Pages 1992-2004, 2005

9.1 World Wide Web references

- [315] <http://openssi.org>
- [316] <http://sourceforge.net/projects/openmosix/>
- [317] <http://www.kerrighed.org>
- [318] <http://www.midnightcode.org/projects/chaos/>
- [319] <http://www.mosix.org/>
- [320] <http://www.ra.cs.uni-tuebingen.de/SNNS/>

Appendix A – Description of the genetic code

Name	Type
Network 0	ANN chromosome
Network 1	ANN chromosome
..	
Network M	ANN chromosome

The ensemble chromosome

Name	Type
ω_n (The weight used when fusing in ensemble)	float
Use this network	boolean
Input node 0 bias	
Input node 1 bias	
..	
Input node n bias	
Hidden node 0	Hidden node chromosome
Hidden node 1	Hidden node chromosome
...	
Hidden node n	Hidden node chromosome
Output node 0 bias	
Output node 1 bias	
..	
Output node n bias	

The ANN chromosome

Name	Type
Use this node	Boolean
Weights input	Array of floats with as many values as there are input nodes
Weights output	Array of floats with as many values as there are output nodes
Bias	Float

The hidden node chromosome

Appendix B – Example data after sliding window

This is a section of the data set for a network with five inputs, and shows the format of the data after applying the sliding window to extract the data for each training iteration.

Network input data (degrees)	True output (degrees)
- 0 0259897 - 3 6105299 - 3.6326396 - 0.0245261 0.	- 0.0150279
- 3.5677671 - 3.5940702 0.0098501 0.0301830 0.	0.0151300
- 0.0032573 3.5949016 3.609473 3.5735285 0.	3 5812125
0.7403398 1.469366 2.1478765 - 0.7111974 0	4 2967362
3.5504568 3.5236094 - 0.0408219 - 0.0349821 0.	3 562428
2 9008305 - 1 3955204 - 2.1215999 - 2 8185372 0	- 0 7151830
- 4.3504057 - 5 0629716 - 5.7463951 - 2.9143443 0.	- 4 3313174
2.1655235 0 7625777 2.8751063 5.0699282 0.	- 1.4177823
- 1 4201748 0.6966611 2.8957903 - 2.1698308 0.	- 5.0182729
- 0 6906021 2.2103865 - 2.1533749 0 7183154 0	- 4.3339977
- 0.0311357 - 3 6618662 - 0.0571447 - 0.0424290 0.	- 3.6493998
- 3.6715751 - 0 0566425 - 0.0317156 0.0209246 0	- 3 6175592
3 6963682 3.7009361 3.7332175 3.6919339 0.	0.0305168
- 5 840281 - 4 3467875 - 2 9268589 - 5.1575804 0.	- 2.2323039
1 420717 2 85884 0.6463125 5.8220873 0.	5.0552101
4 3246026 1.3904552 5 8446097 - 0 6990970 0.	4.3552918
2.9959674 5.9675932 - 2.0586424 - 2.8420739 0	- 0.7361527
0.0066305 - 7.2783566 - 7.320539 - 3.7372162 0.	- 3 7631221
- 4.5046263 - 5.2418995 - 2.3536665 0.6884596 0.	- 4 5015211
- 0.7371413 2.1510587 5.193152 4.5046592 0.	- 0.7346394
3.01721 6.0270505 5.3063059 0.7693939 0	- 0 7736980
- 3.0837958 - 2.2811317 - 5.2946343 - 4.5406189 0.	- 3 0501225
- 2 2588835 - 4 5069995 - 2 987597 2.318409 0.	1.5331851
6.8483529 6 0936379 9.1255264 4.5330005 0.	3.0105968
- 0.7973959 2.2451632 - 2.3366928 - 6.8590231 0.	- 4.6156149
- 3.0617313 - 6.1175146 - 9 1137724 - 0.7286766 0.	- 3 0772533
- 0.0342346 - 3 7858796 3.8438289 3.8171184 0.	- 0.0373515

2.3586707	8.4608002	6.9065108	1.5618136	0.	- 1.5600936
- 6.2673321	- 4.7292562	- 6.9815879	- 5.4510365	0	- 2.3910739
4 6294379	1.6042653	2.3619769	7.0401726	0.	4 6232653
3.2366092	2.4288752	5.5416255	- 3.0639923	0.	- 0.8325605
- 0 8144906	2 2999489	- 6 3039799	- 3.2382984	0.	- 4.9019256
- 6 4251671	- 12.644195	- 7.1936111	- 1.5704111	0.	- 2.5434217
3 110132	6 228425	9 5193357	8.7574568	0.	3.0852442
- 0 0469103	4.0353007	4.064723	- 3.9014328	0.	- 0.0383736
- 2.3924761	- 0.7443820	- 7 091866	- 1.5717615	0.	1.6053838
8 1517763	0 1783720	4 072556	4 0183969	0.	3.9604015
- 8.2471333	- 4.2845173	- 4.2702446	- 8.2202091	0.	- 4.1688166
4 0905948	4.0728731	0.0909139	8.2791281	0.	4.0604353
3.415092	- 1.4250705	5.9049406	- 3 2323911	0	- 0.914698

Appendix C – Example data with binning function applied

Here the same output bearings as appendix B are repeated. Here however, the bearings converted by the Gaussian binning function are also given. The binning function was a Gaussian with $\sigma=6.71$, and the bearing error Gaussian distribution had $\sigma=2.0$. The first column gives the true output bearings while the second column gives the data in the format provided to the ANN during training.

True output (degrees)	Ideal output network training
- 0.0150279	0.002, 0.197, 0.605, 0.193, 0.002
0.0151300	0.002, 0.193, 0.605, 0.197, 0.002
3.5812125	0.000, 0.004, 0.169, 0.676, 0.151
4.2967362	0.000, 0.001, 0.096, 0.653, 0.250
3.562428	0.000, 0.004, 0.172, 0.675, 0.149
- 0.7151830	0.007, 0.304, 0.575, 0.113, 0.001
- 4.3313174	0.255, 0.650, 0.093, 0.001, 0.000
- 1.4177823	0.017, 0.427, 0.497, 0.059, 0.000
- 5.0182729	0.377, 0.575, 0.048, 0.000, 0.000
- 4.3339977	0.256, 0.650, 0.093, 0.001, 0.000
- 3.6493998	0.159, 0.676, 0.161, 0.004, 0.000
- 3.6175592	0.155, 0.676, 0.165, 0.004, 0.000
0.0305168	0.002, 0.191, 0.605, 0.199, 0.002
- 2.2323039	0.044, 0.561, 0.370, 0.025, 0.000
5.0552101	0.000, 0.000, 0.046, 0.570, 0.384
4.3552918	0.000, 0.001, 0.091, 0.649, 0.259
- 0.7361527	0.007, 0.308, 0.573, 0.111, 0.001
- 3.7631221	0.173, 0.676, 0.148, 0.003, 0.000
- 4.5015211	0.283, 0.636, 0.080, 0.001, 0.000
- 0.7346394	0.007, 0.308, 0.574, 0.111, 0.001
- 0.7736980	0.007, 0.314, 0.570, 0.107, 0.001
- 3.0501225	0.097, 0.653, 0.241, 0.009, 0.000
1.5331851	0.000, 0.053, 0.480, 0.447, 0.020

3 0105968	0 000, 0.009, 0.247, 0.650, 0.094
- 4 6156149	0 303, 0 624, 0.072, 0.001, 0.000
- 3.0772533	0.099, 0 655, 0.237, 0 008, 0 000
- 0.0373515	0.003, 0 200, 0.605, 0 190, 0 002
- 1.5600936	0.021, 0.452, 0 476, 0.051, 0 000
- 2.3910739	0.052, 0 583, 0.344, 0.020, 0.000
4.6232653	0.000, 0 001, 0.071, 0.624, 0 304
- 0.8325605	0.008, 0.324, 0 565, 0.102, 0 001
- 4.9019256	0.355, 0.590, 0 054, 0 000, 0.000
- 2.5434217	0.060, 0 603, 0 320, 0.017, 0 000
3 0852442	0 000, 0.008, 0 236, 0 656, 0.100
- 0 0383736	0.003, 0.201, 0 605, 0 190, 0 002
1 6053838	0 000, 0.049, 0 470, 0 460, 0.022
3.9604015	0.000, 0 002, 0.127, 0.671, 0 200
- 4 1688166	0.230, 0 662, 0.107, 0 002, 0.000
4.0604353	0.000, 0.002, 0.117, 0.667, 0 214

Appendix D - Optimisation results

EKF

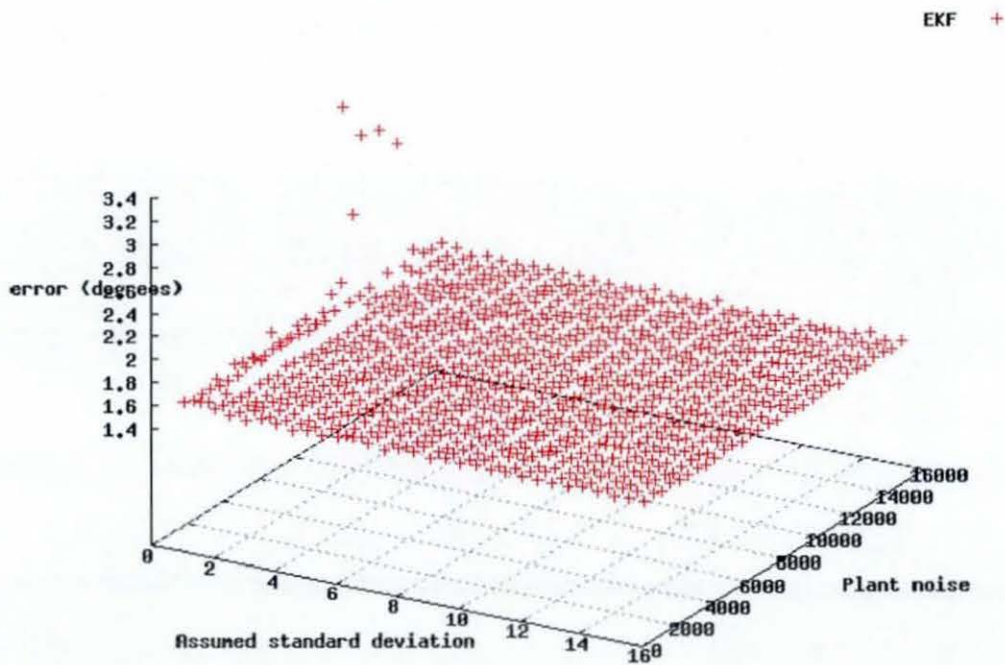


Figure 97 RMS bearing error for the EKF with different values for parameters; standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on pure synthetic data set 1

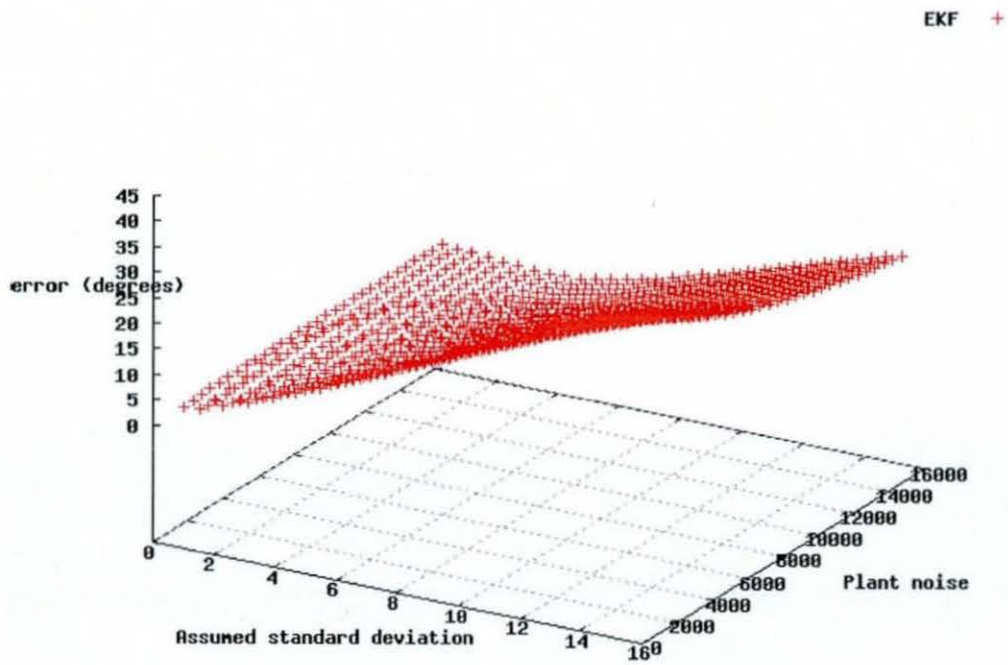


Figure 98 RMS bearing error for the EKF with different values for parameters; standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on pure synthetic data set 2

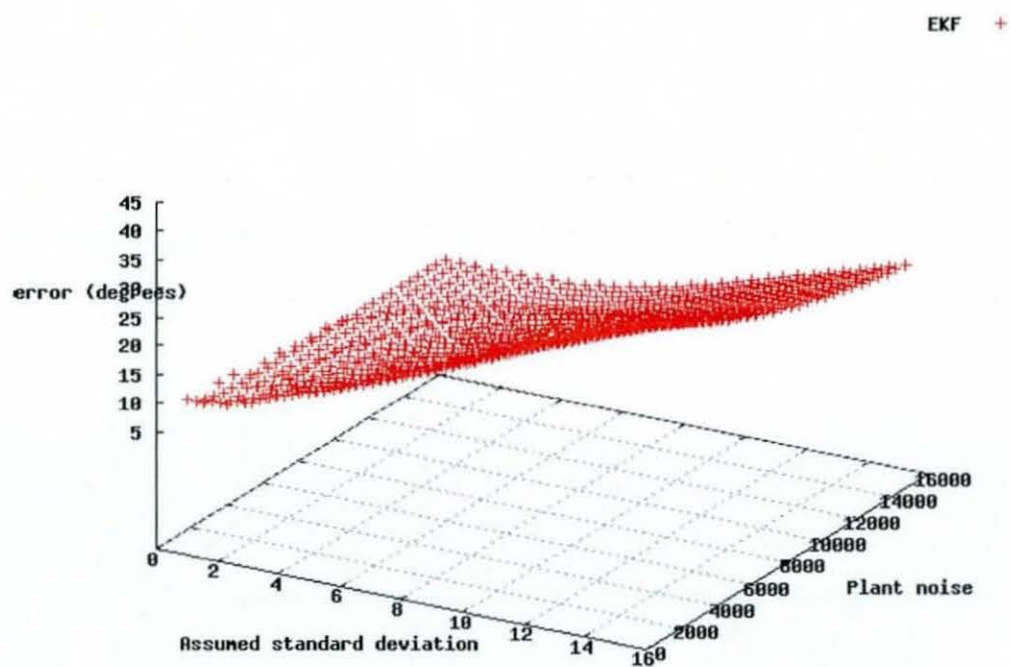


Figure 99 RMS bearing error for the EKF with different values for parameters; standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on pure synthetic data set 3

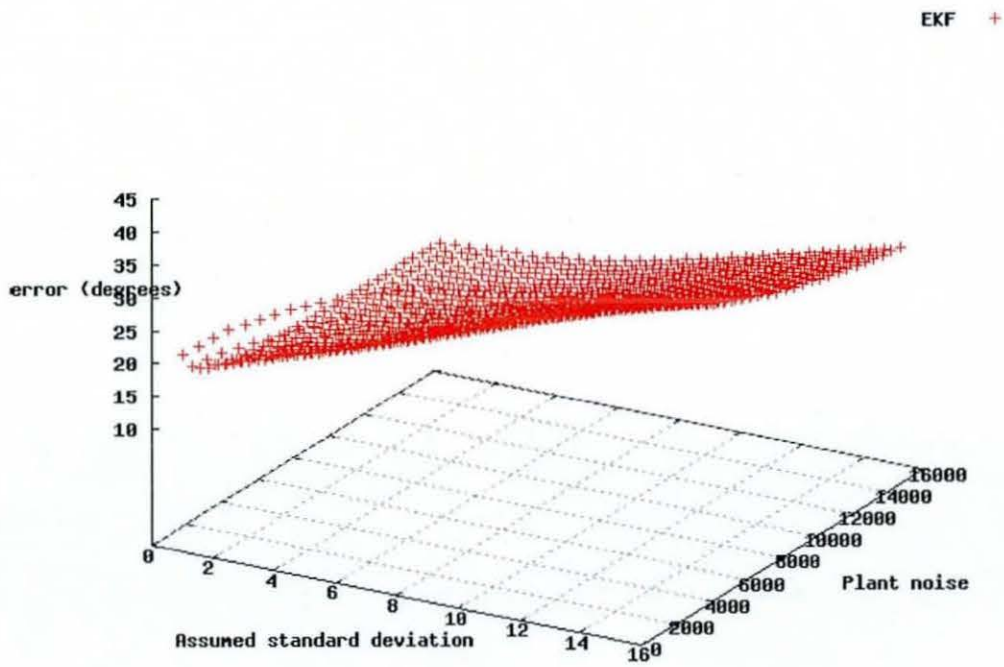


Figure 100 RMS bearing error for the EKF with different values for parameters; standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on pure synthetic data set 4

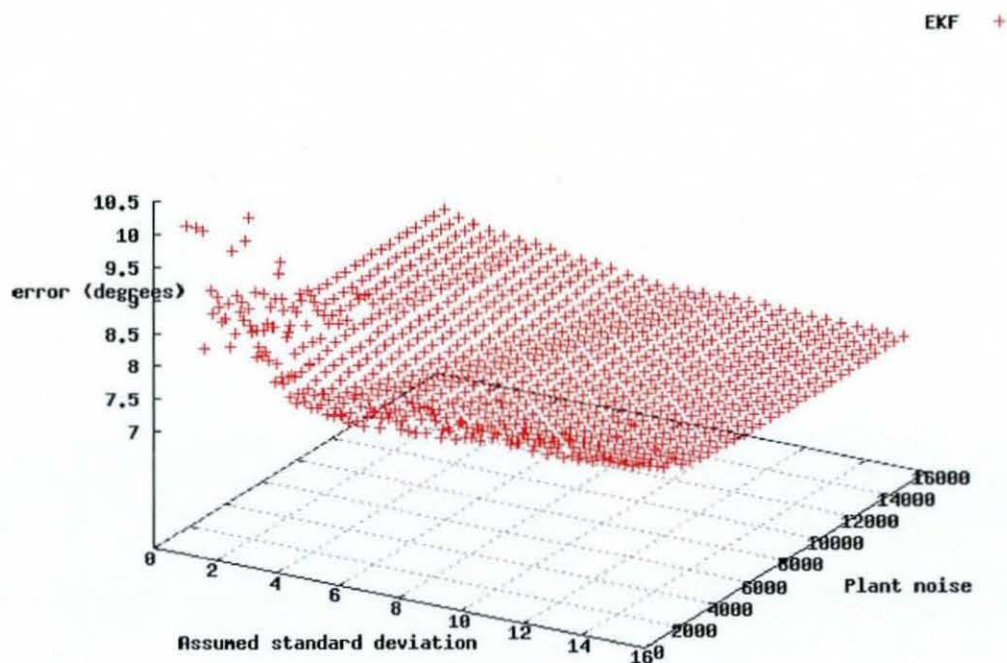


Figure 101 RMS bearing error for the EKF with different values for parameters; standard deviation between 0 and 16 and plant noise between 0 and 15000 as tested on semi-synthetic data set

Particle Filter

Various values were used as the parameters for the PF to determine the optimal set for use in the main experiments. Figure 102 to Figure 109 show the results of these experiments. First a variety of values were tried for the system noise parameter, in particle filters with 50 and 100 particles. The optimal system noise values can be seen to be similar in both, however in general the filter with the most particles is most accurate.

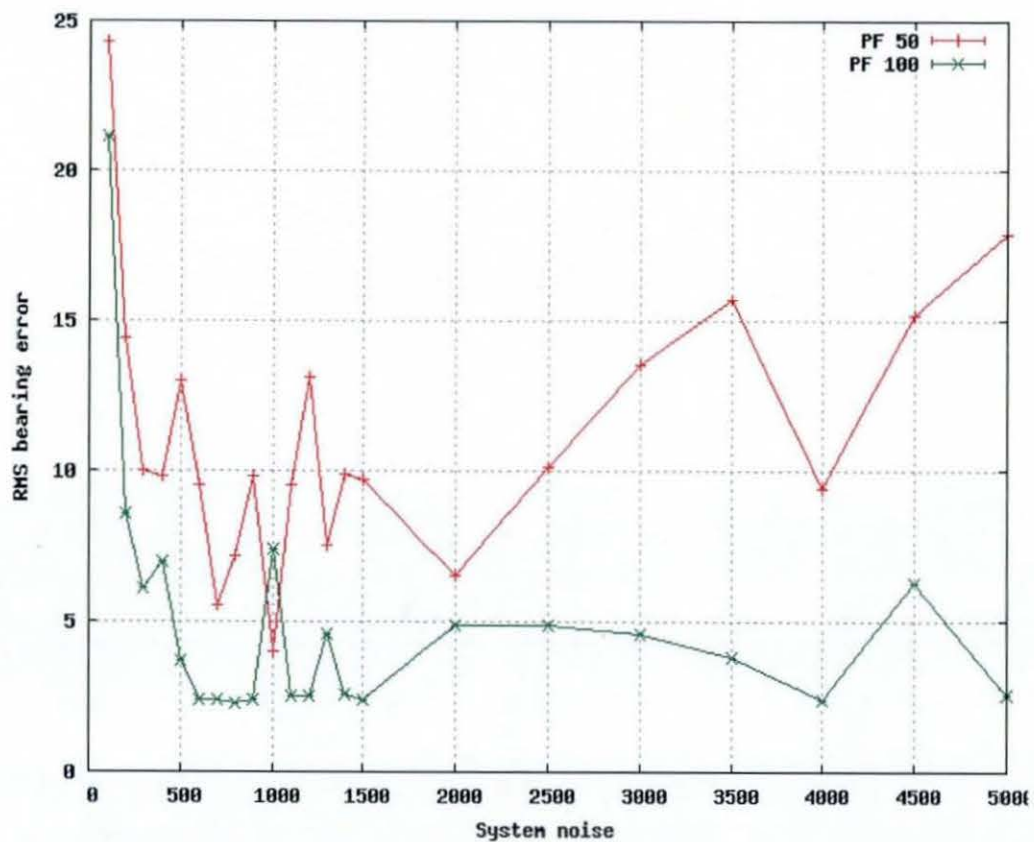


Figure 102 RMS bearing error for the PF with different values for parameter plant noise between 0 and 5000 as tested on pure synthetic data set 1

System noise	Particle filter with 50 particles	Particle filter with 100 particles
100	24.3	21.2
200	14.4	8.6
300	10	6.1
400	9.8	7
500	13	3.7
600	9.5	2.4
700	5.5	2.4
800	7.2	2.3
900	9.8	2.4
1000	4.0	7.4
1100	9.5	2.5
1200	13.1	2.5

1300	7.5	4.6
1400	9.9	2.6
1500	9.7	2.4
2000	6.5	4.9
2500	10.1	4.9
3000	13.5	4.6
3500	15.7	3.8
4000	9.4	2.4
4500	15.2	6.3
5000	17.9	2.6

Table 35 RMS bearing error for the PF with different values for parameter plant noise between 0 and 5000 as tested on pure synthetic data set 1

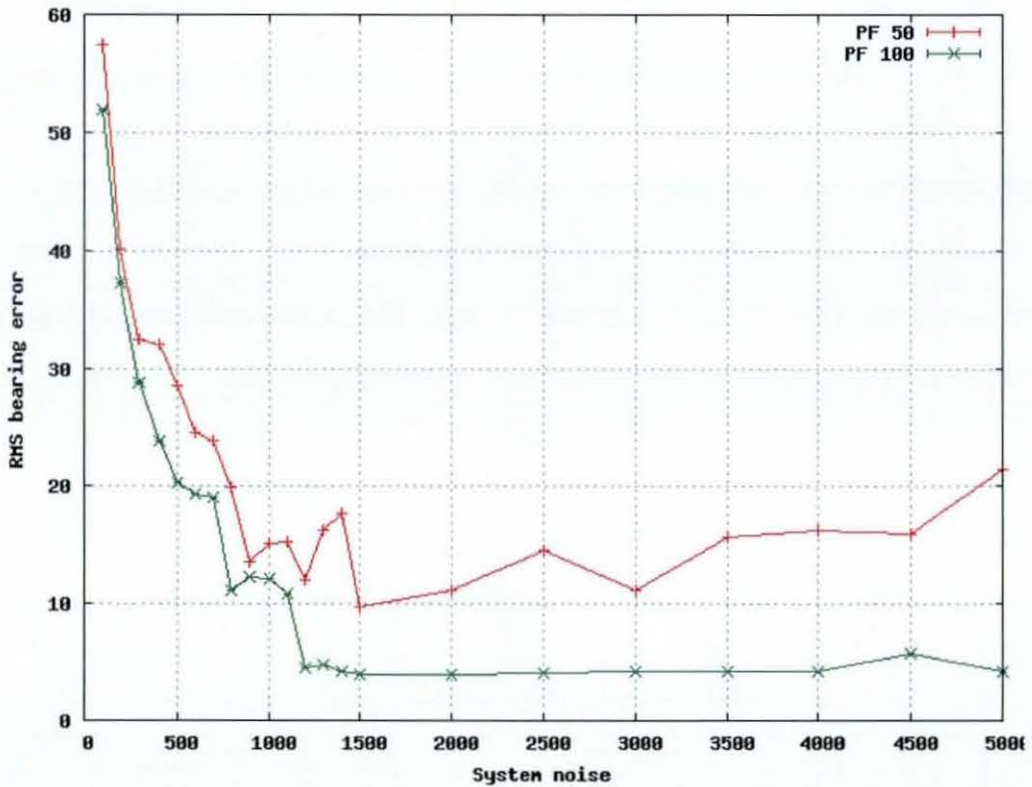


Figure 103 RMS bearing error for the PF with different values for parameter plant noise between 0 and 5000 as tested on pure synthetic data set 2

System noise	Particle filter with 50 particles	Particle filter with 100 particles
--------------	-----------------------------------	------------------------------------

100	57.4	51.9
200	40.1	37.3
300	32.5	28.8
400	32.0	23.9
500	28.5	20.4
600	24.5	19.4
700	23.9	19.0
800	19.9	11.2
900	13.6	12.3
1000	15.1	12.1
1100	15.2	10.9
1200	12.0	4.5
1300	16.2	4.8
1400	17.6	4.2
1500	9.8	3.9
2000	11.1	3.9
2500	14.5	4.1
3000	11.2	4.3
3500	15.7	4.3
4000	16.3	4.3
4500	16.0	5.8
5000	21.4	4.3

Table 36 RMS bearing error for the PF with different values for parameter plant noise between 0 and 5000 as tested on pure synthetic data set 2

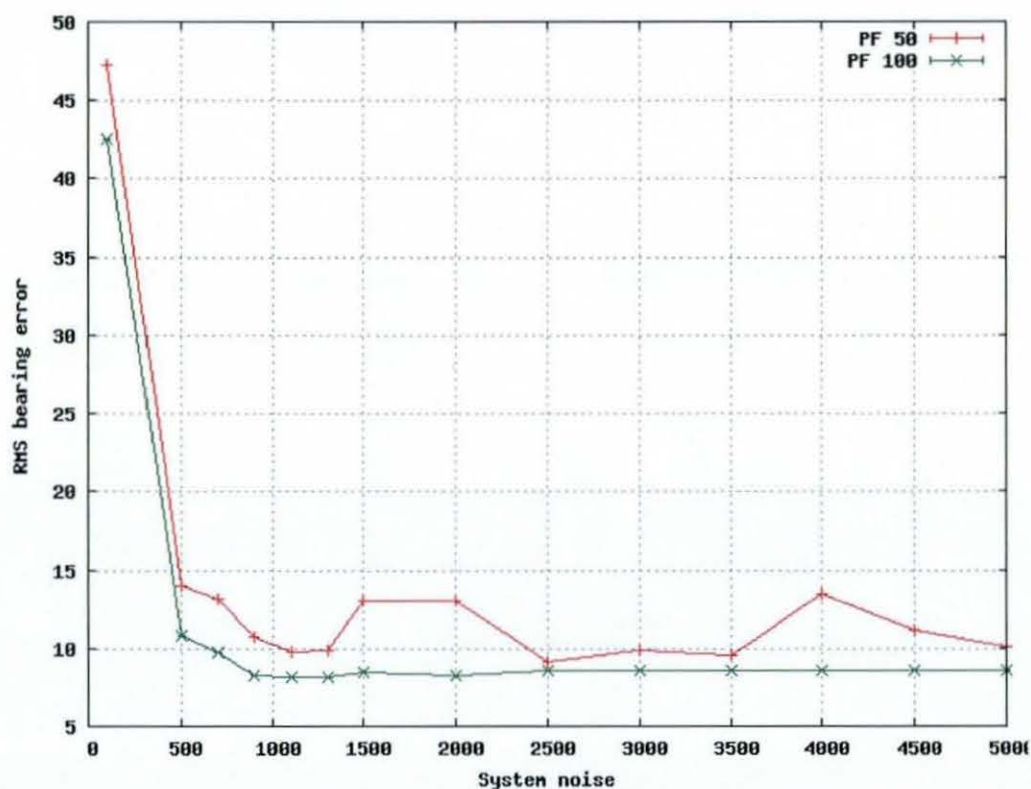


Figure 104 RMS bearing error for the PF with 200 particles with plant noise values between 0 and 5000 as tested on pure synthetic data set 3

System noise	Particle filter with 50 particles	Particle filter with 100 particles
100	47.3	100
500	14.0	500
700	13.2	700
900	10.7	900
1100	9.8	1100
1300	9.9	1300
1500	13.0	1500
2000	13.0	2000
2500	9.1	2500
3000	9.9	3000
3500	9.5	3500
4000	13.5	4000
4500	11.1	4500
5000	10.05	5000

Table 37 RMS bearing error for the PF with 200 particles with plant noise values between 0 and 5000 as tested on pure synthetic data set 3

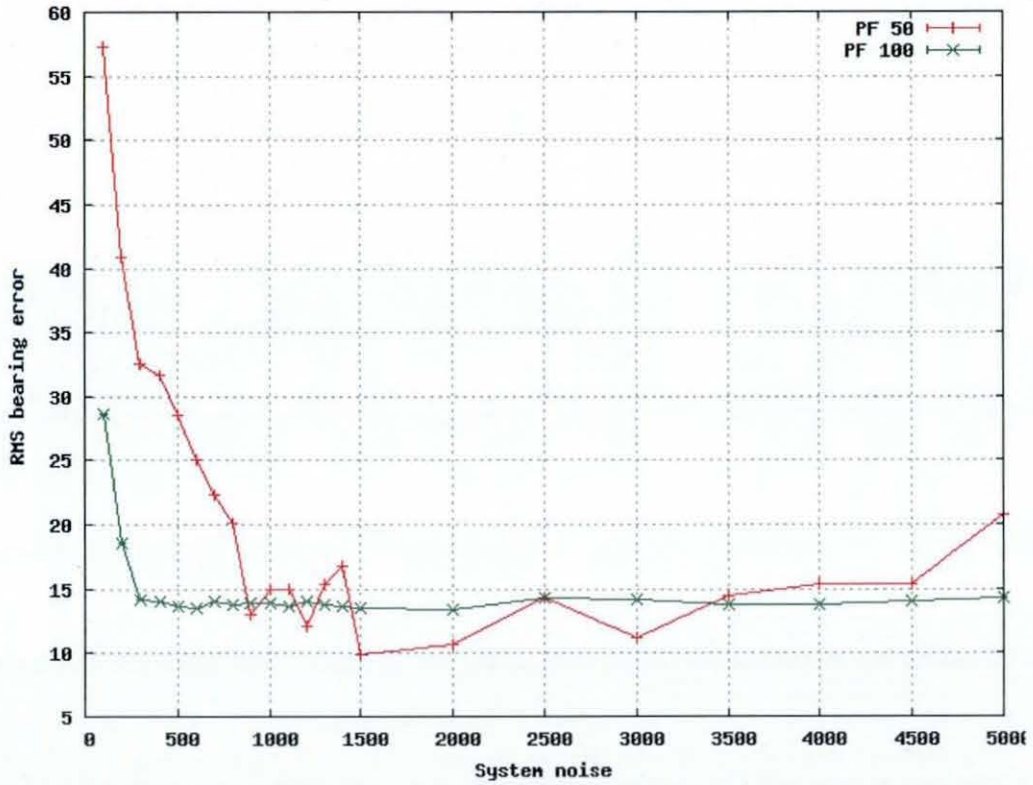


Figure 105 RMS bearing error for the PF with 200 particles with plant noise values between 0 and 5000 as tested on pure synthetic data set 4

System noise	Particle filter with 50 particles	Particle filter with 100 particles
100	100	100
200	200	200
300	300	300
400	400	400
500	500	500
600	600	600
700	700	700
800	800	800
900	900	900
1000	1000	1000

1100	1100	1100
12001	1200	1200
1300	1300	1300
1400	1400	1400
1500	1500	1500
2000	2000	2000
2500	2500	2500
3000	3000	3000
3500	3500	3500
4000	4000	4000
4500	4500	4500
5000	5000	5000

Table 38 RMS bearing error for the PF with 200 particles with plant noise values between 0 and 5000 as tested on pure synthetic data set 4

Varying the number of particles, while keeping the process noise, σ_w set to the optimal value of 4000, discovered above. It can be seen that in most cases little improvement is available with more than 200 particles. Using 1000 particles rather than 200 gives a mean RMS error improvement of less than 0.5%, however it leads to an eight fold increase in run time (Figure 110).

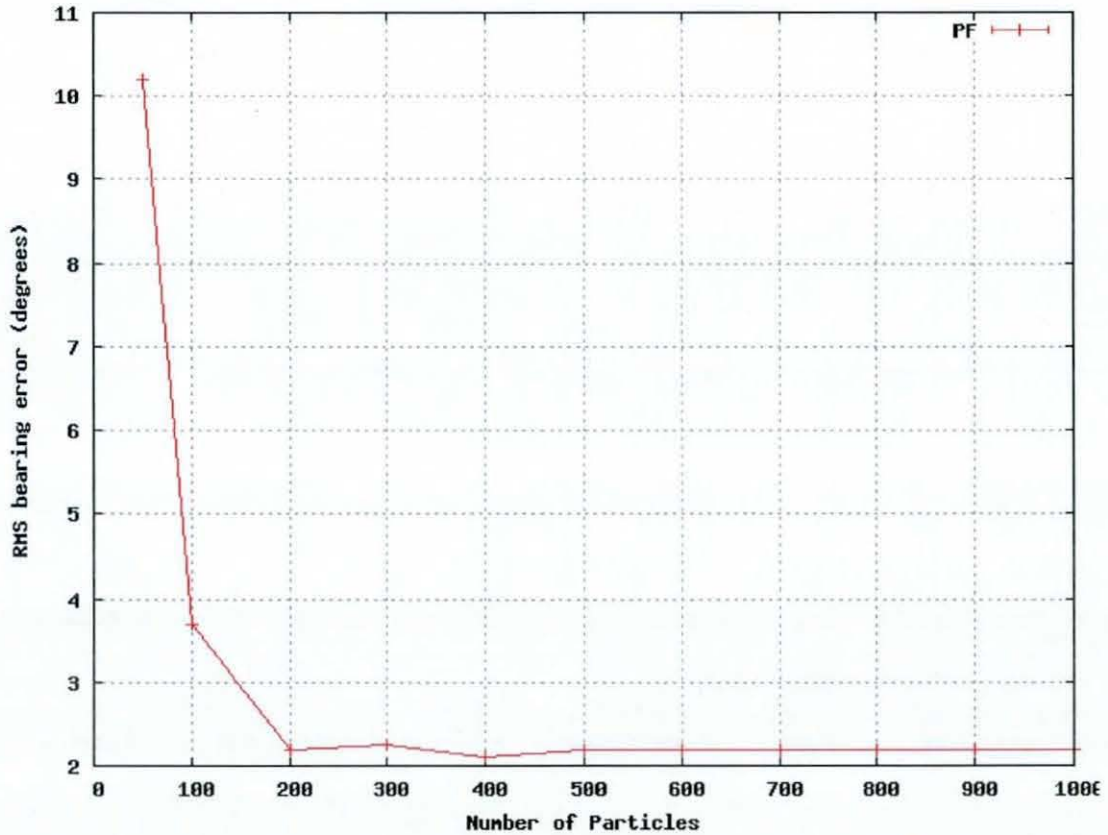


Figure 106 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 1

Number of particles	RMS bearing error
50	10.2
100	3.7
200	2.2
300	2.25
400	2.1
500	2.2
600	2.2
700	2.2

800	2.2
900	2.2
1000	2.2

Table 39 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 1

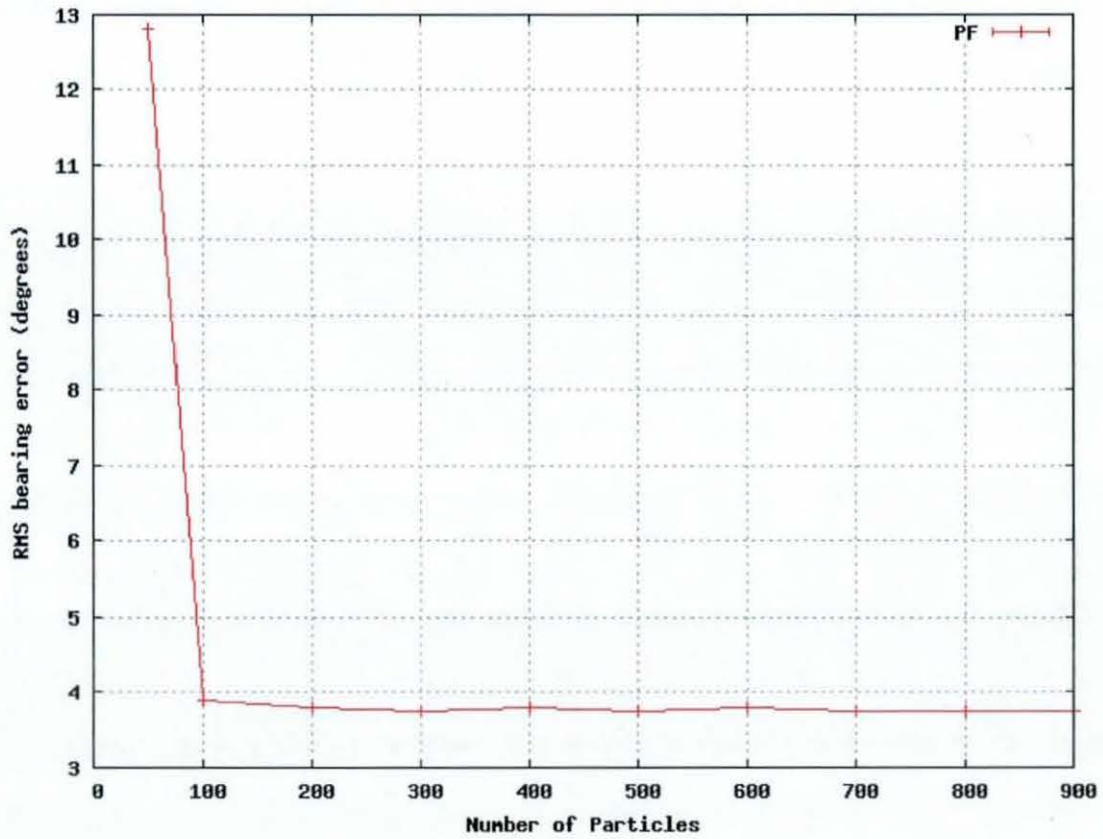


Figure 107 Pure synthetic data set 2

Number of particles	RMS bearing error
50	12.8
100	3.9
200	3.8
300	3.75
400	3.8
500	3.75
600	3.8
700	3.75

800	3.75
900	3.75

Table 40 Pure synthetic data 2

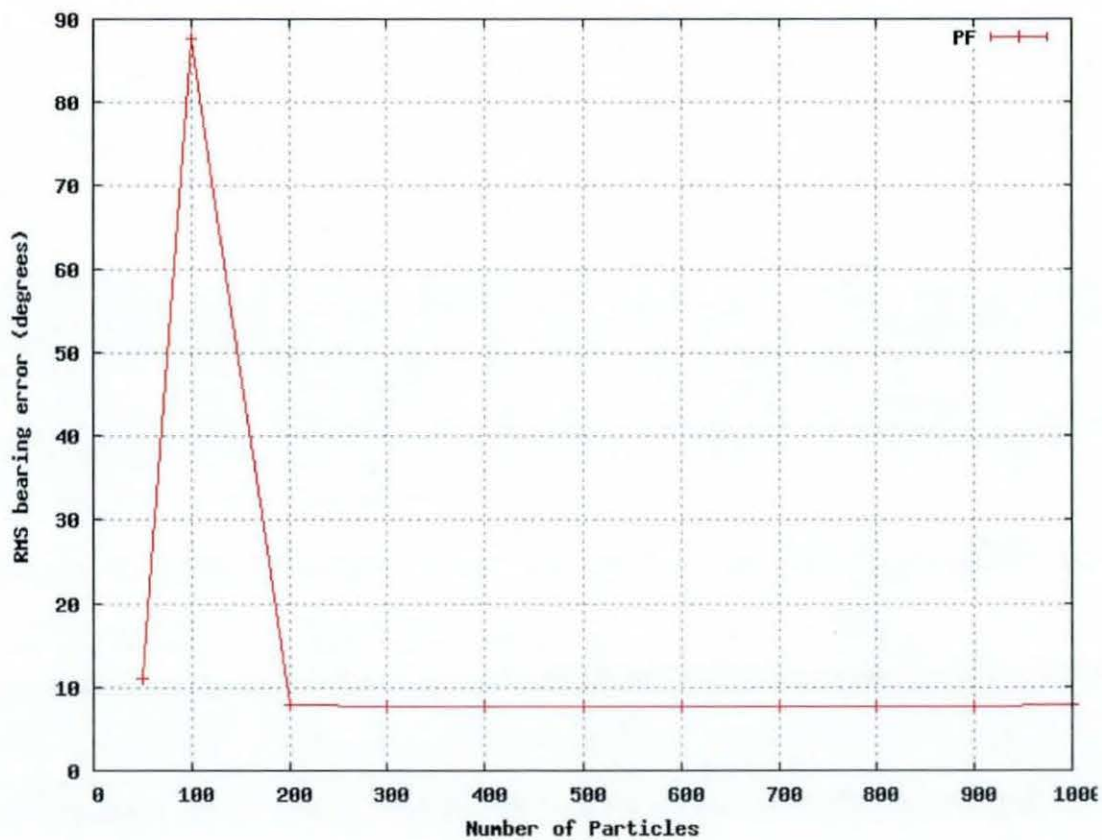


Figure 108 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 3

Number of particles	RMS bearing error
50	11.1
100	87.7
200	7.8
300	7.7
400	7.7
500	7.7
600	7.7

700	7.7
800	7.6
900	7.6
1000	7.8

Table 41 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 3

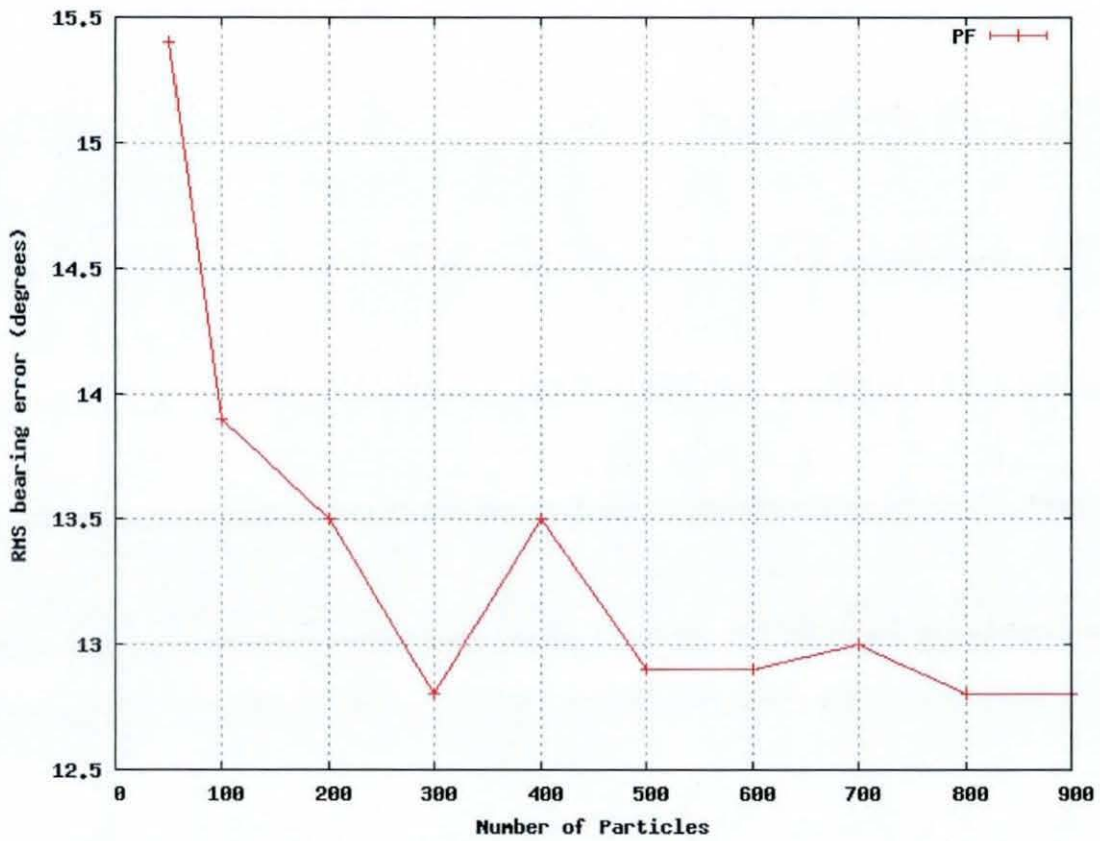


Figure 109 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 4

Number of particles	RMS bearing error
50	15.4
100	13.9
200	13.5
300	12.8
400	13.5
500	12.9

600	12.9
700	13.0
800	12.8
900	12.8

Table 42 RMS bearing error for the PF with different numbers of particles between 50 and 1000 as tested on pure synthetic data set 4

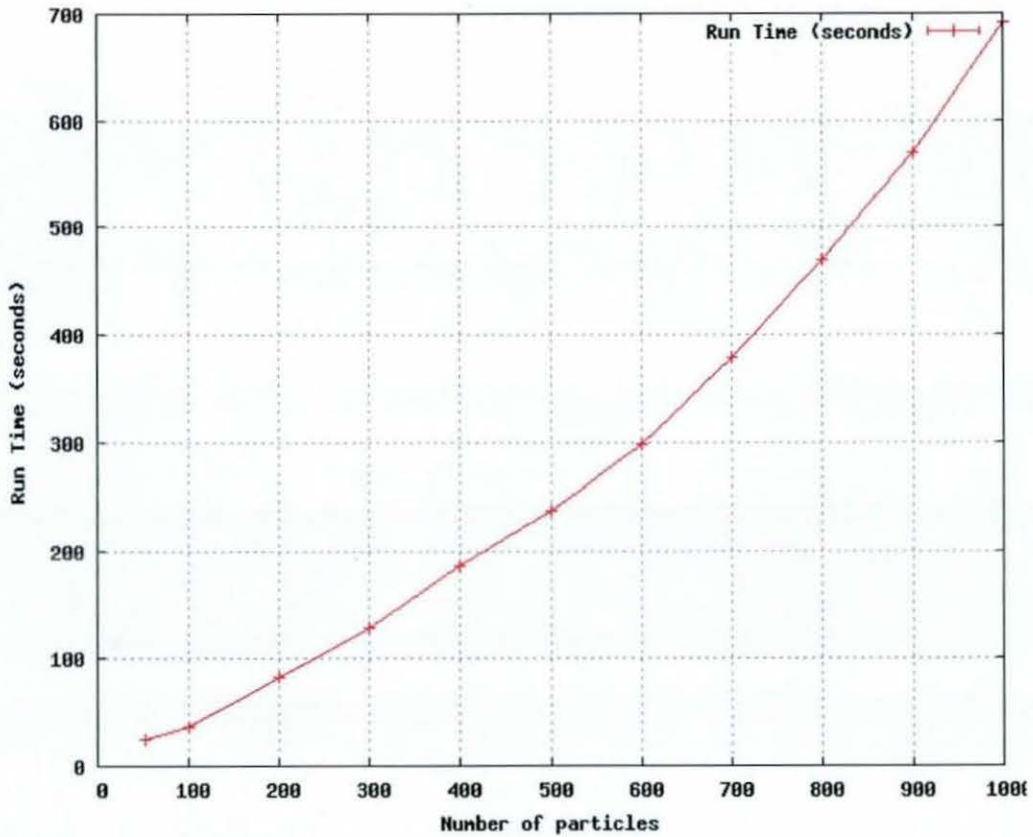


Figure 110 Run time in seconds on pure synthetic data over number of particles in particle filter between 50 and 1000 particles

Number of particles	Run time (Seconds)
50	24.99
100	37.05
200	82.77
300	129.12

400	186.17
500	237.86
600	297.56
700	378.19
800	469.98
900	569.28
1000	692.49

Table 43 Run time in seconds on pure synthetic data over number of particles in particle filter between 50 and 1000 particles

Single Output Neural Network

The single output ANN is described in section 2.7.4. The result of the optimisation used to establish the parameters is given here. A Java program was written which could repeatedly run the algorithm through a series of nested loops to evaluate every combination of parameters. This program was run on each of the five data sets in turn.

Pure synthetic 1

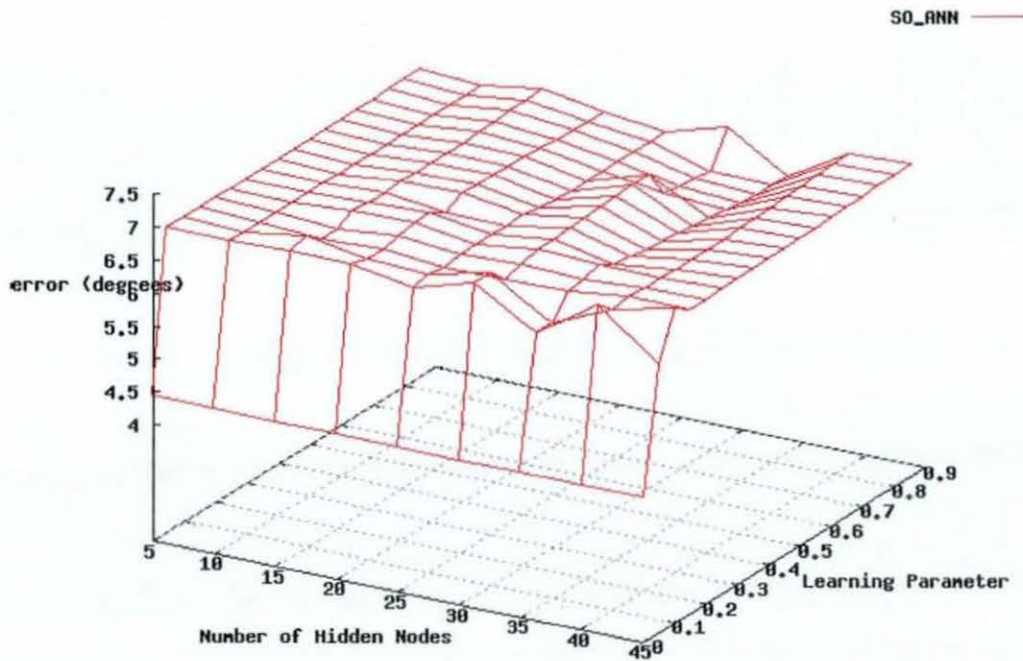


Figure 111 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 5 input nodes trained with backpropagation on synthetic data set 1

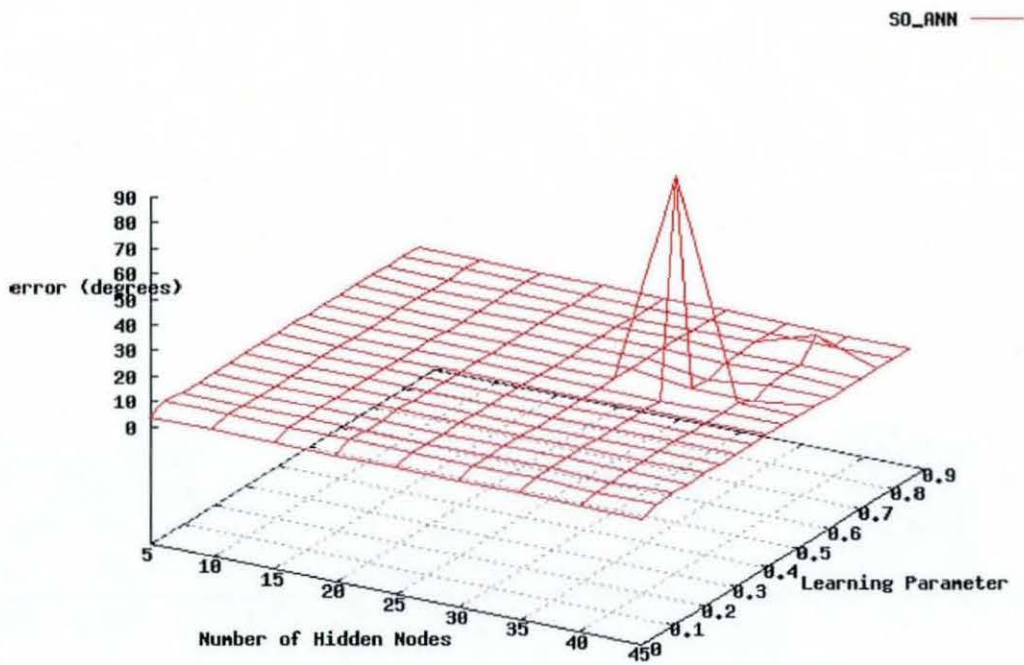


Figure 112 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 10 input nodes trained with backpropagation on synthetic data set 1

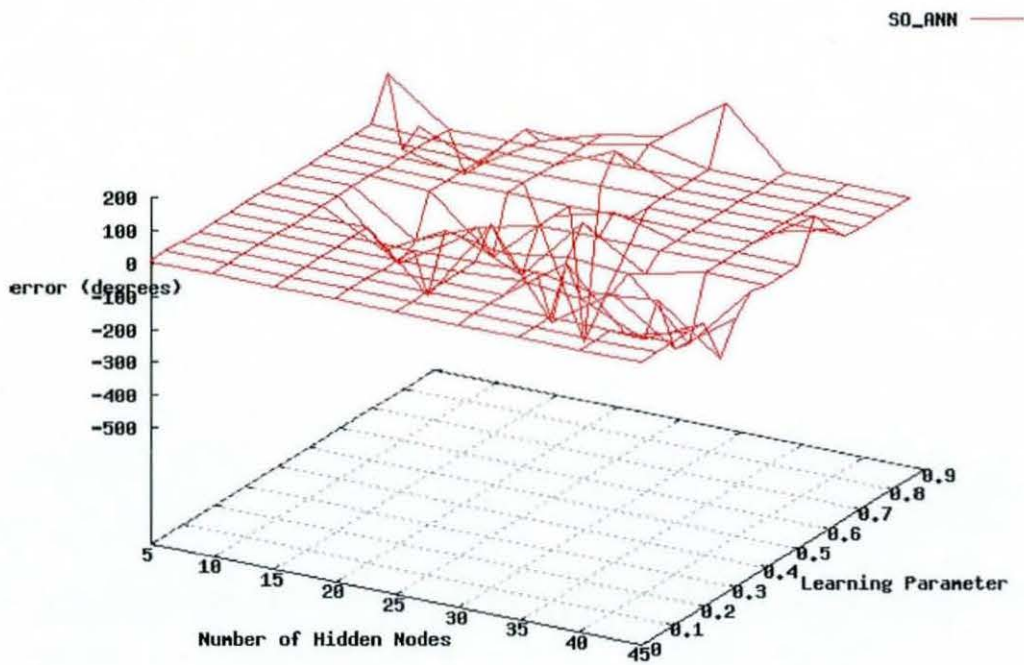


Figure 113 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 15 input nodes trained with backpropagation on synthetic data set 1

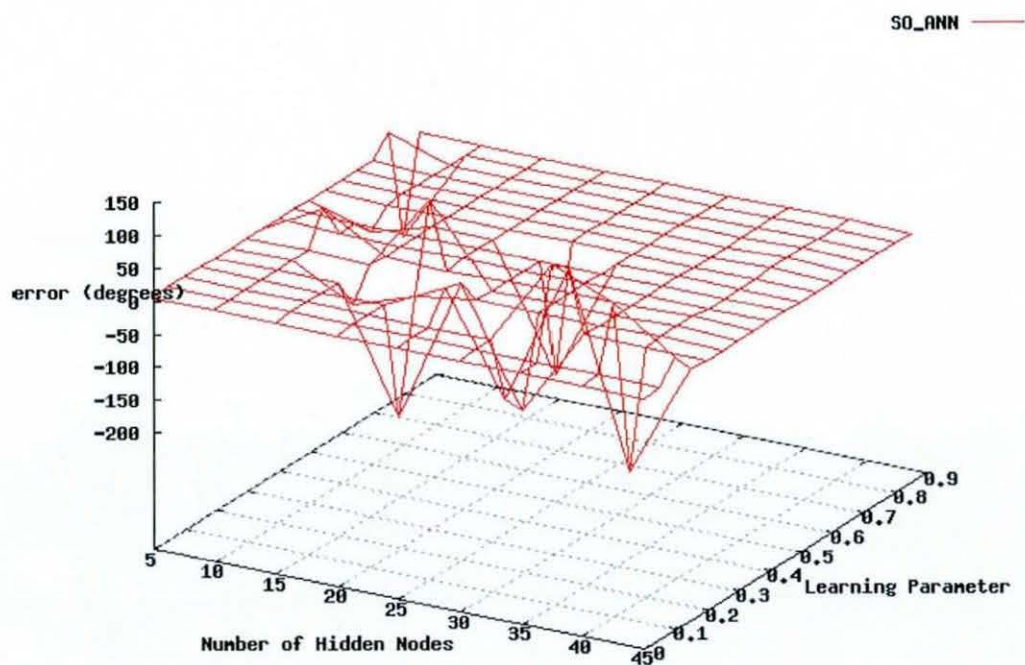


Figure 114 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 20 input nodes trained with backpropagation on synthetic data set 1

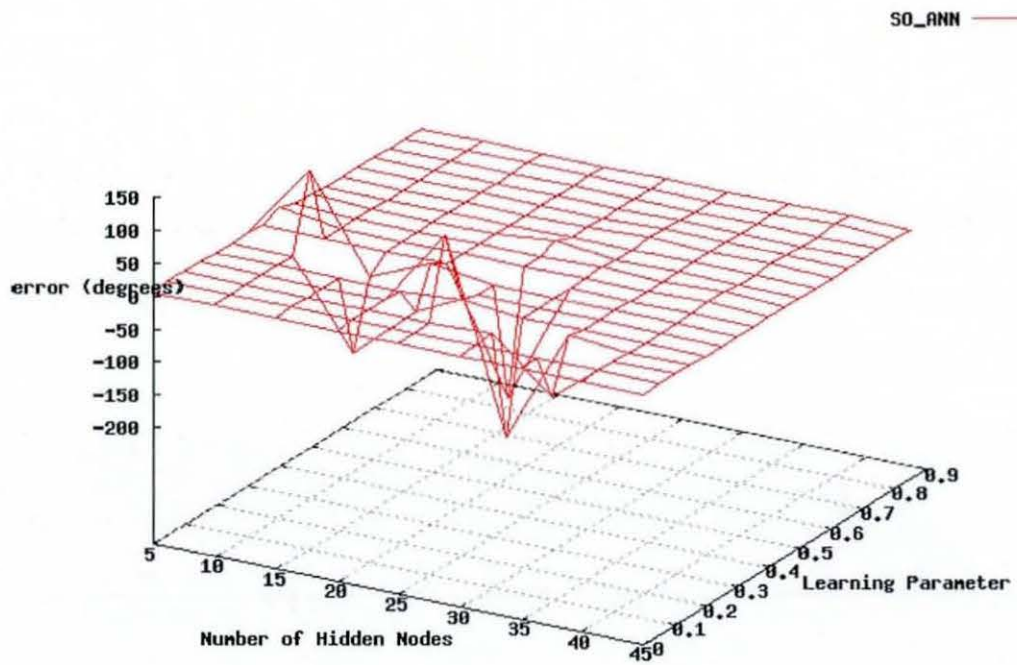


Figure 115 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 25 input nodes trained with backpropagation on synthetic data set 1

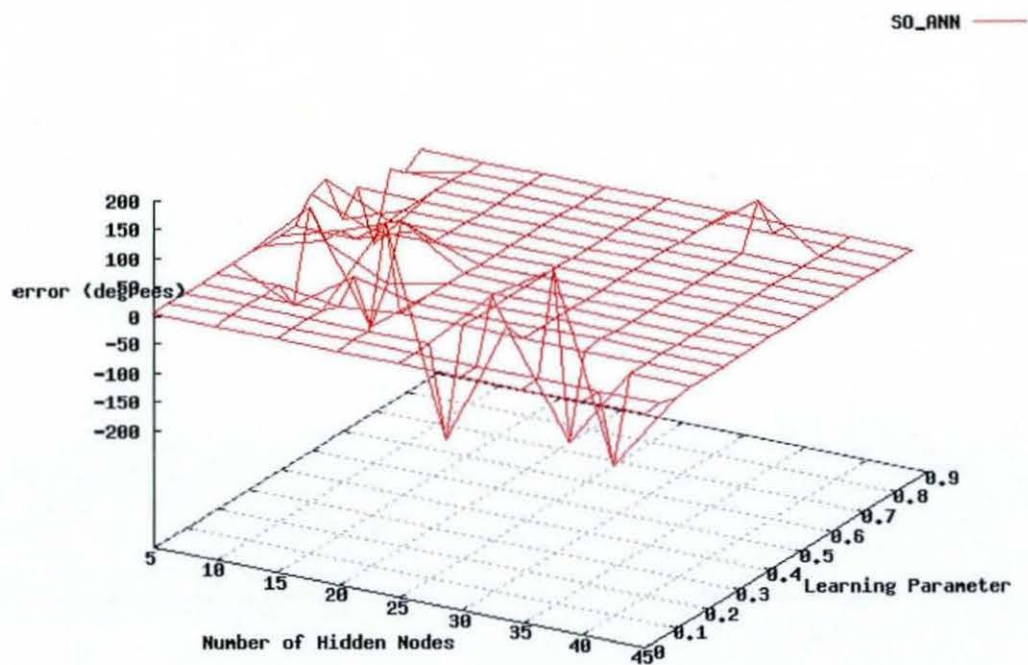


Figure 116 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 30 input nodes trained with backpropagation on synthetic data set 1

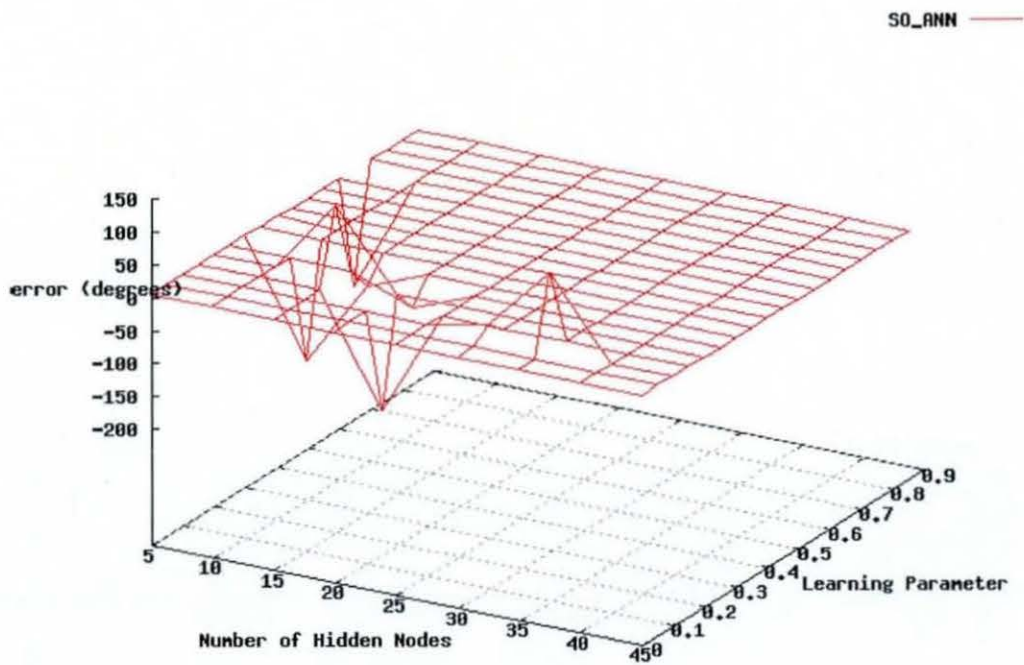


Figure 117 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 35 input nodes trained with backpropagation on synthetic data set 1

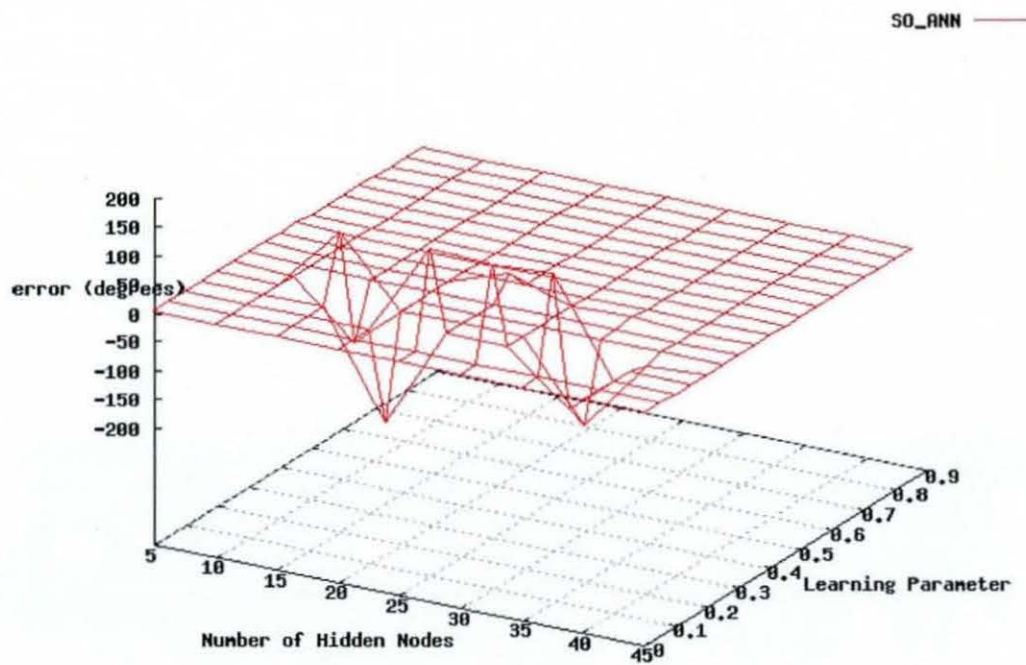


Figure 118 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 40 input nodes trained with backpropagation on synthetic data set 1

Pure synthetic 2

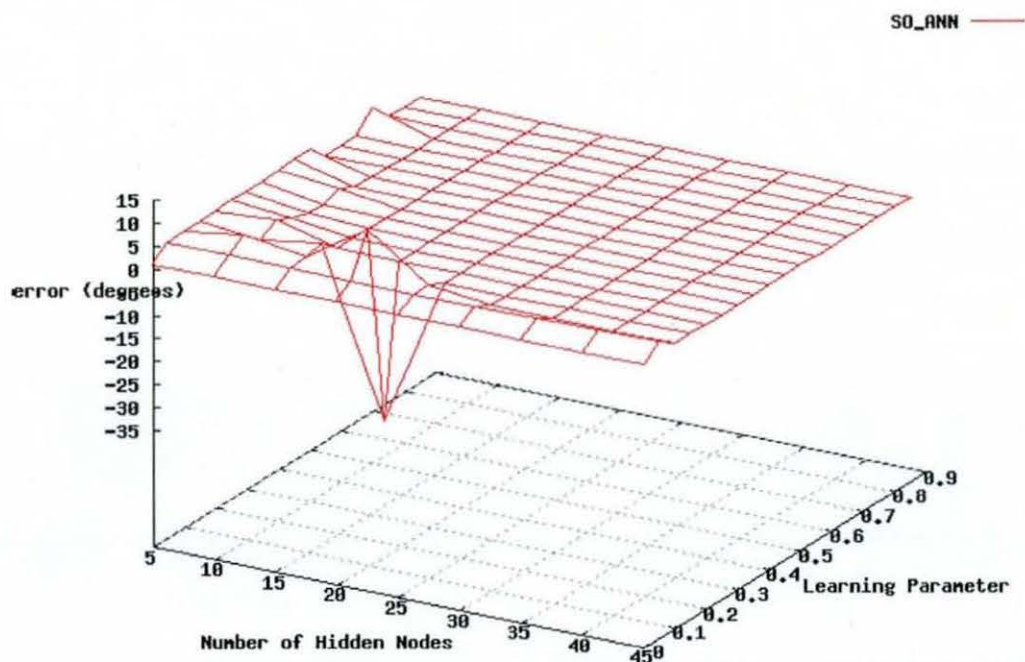


Figure 119 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 5 input nodes trained with backpropagation on synthetic data set 2

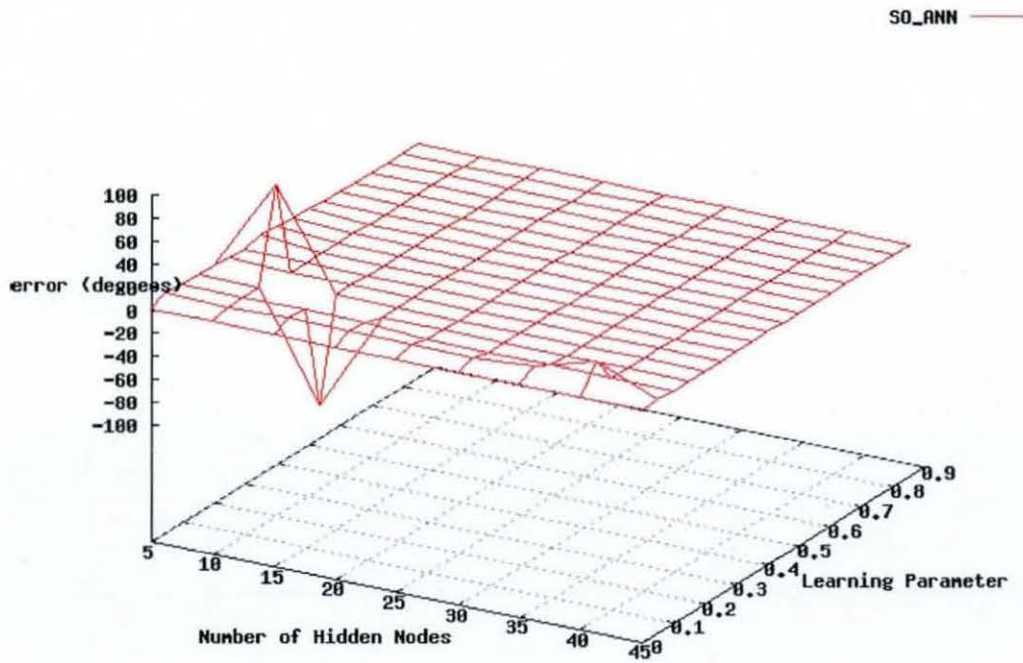


Figure 120 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 10 input nodes trained with backpropagation on synthetic data set 2

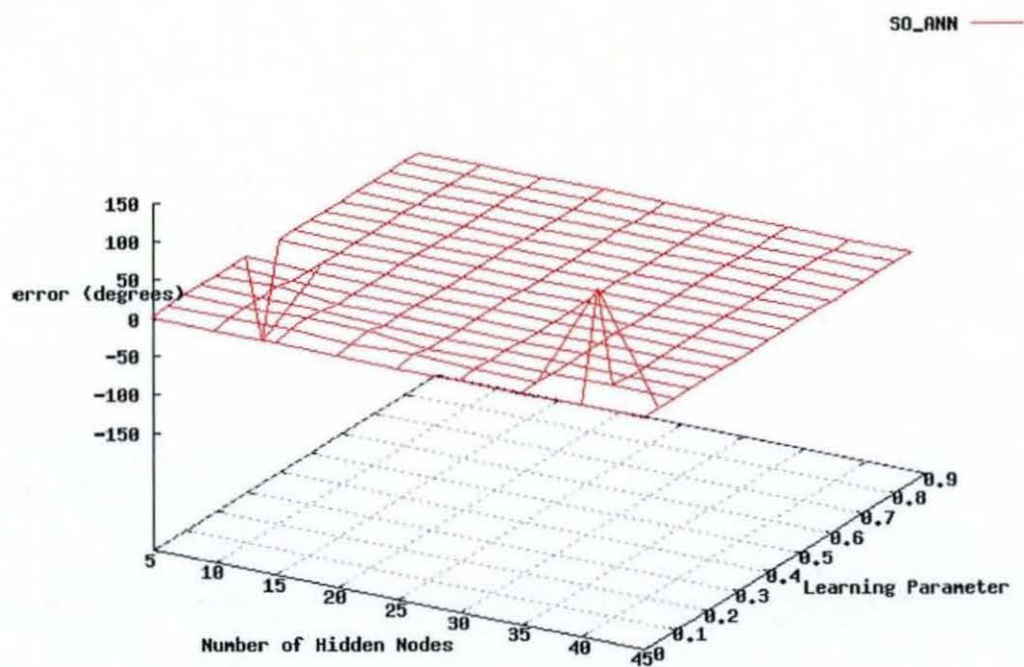


Figure 121 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 15 input nodes trained with backpropagation on synthetic data set 2

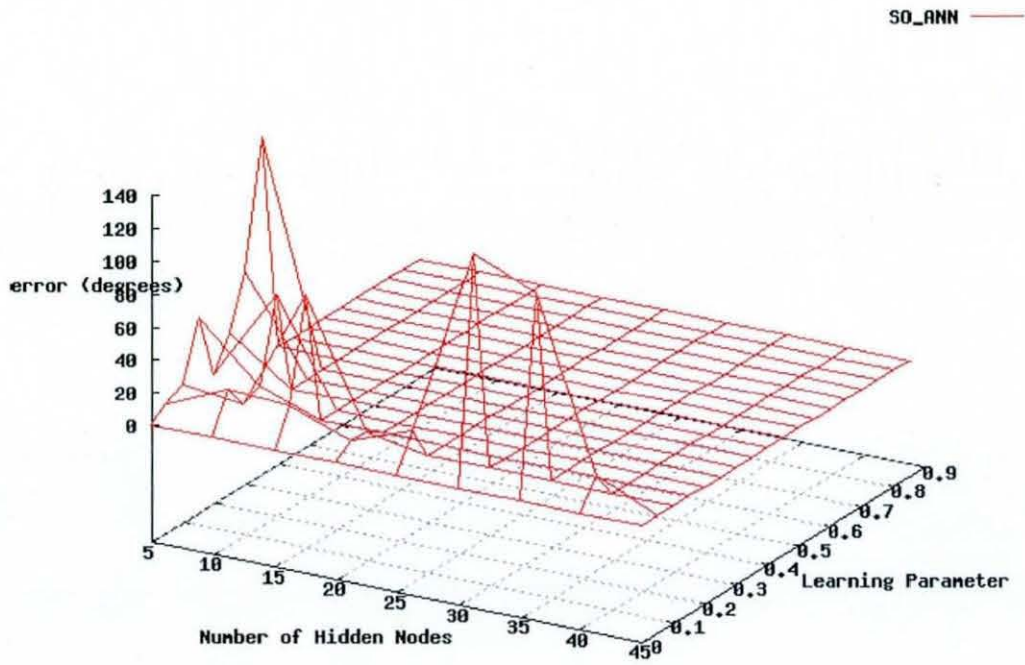


Figure 122 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 20 input nodes trained with backpropagation on synthetic data set 2

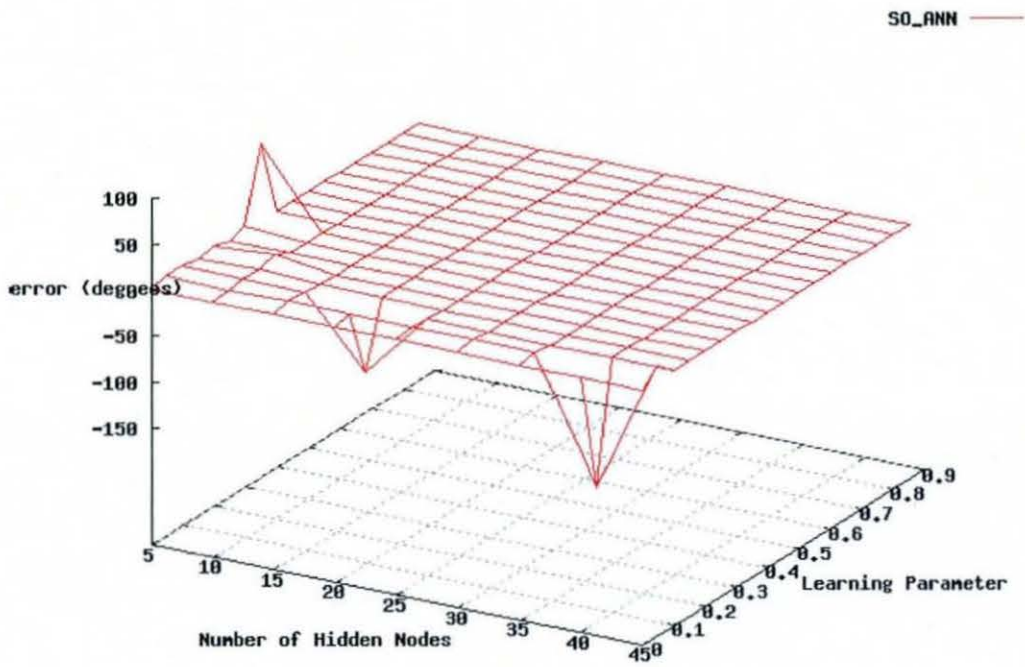


Figure 123 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 25 input nodes trained with backpropagation on synthetic data set 2

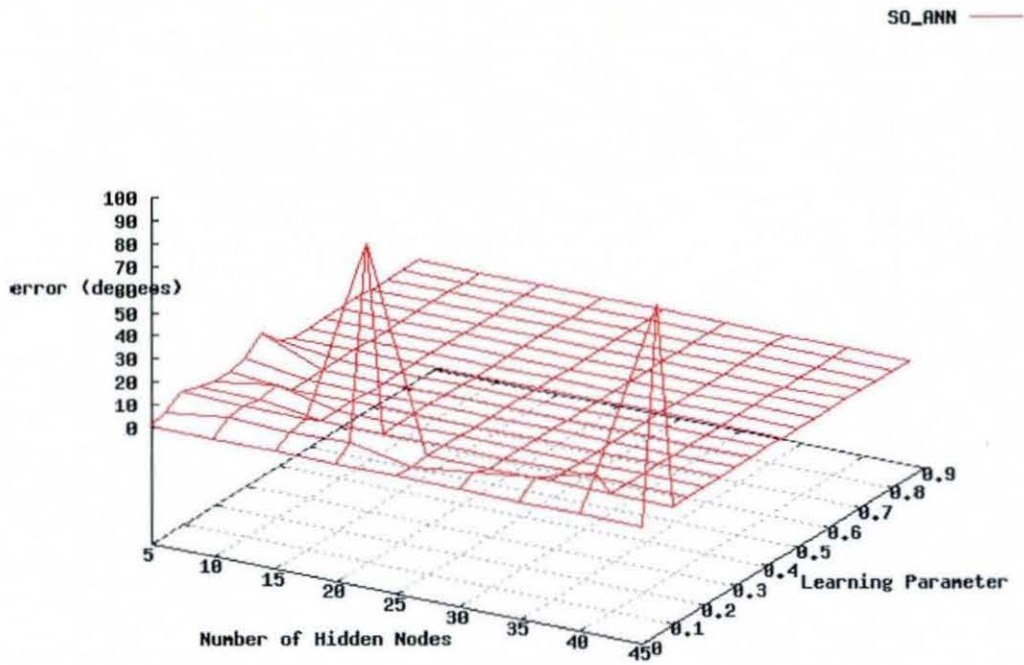


Figure 124 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 30 input nodes trained with backpropagation on synthetic data set 2

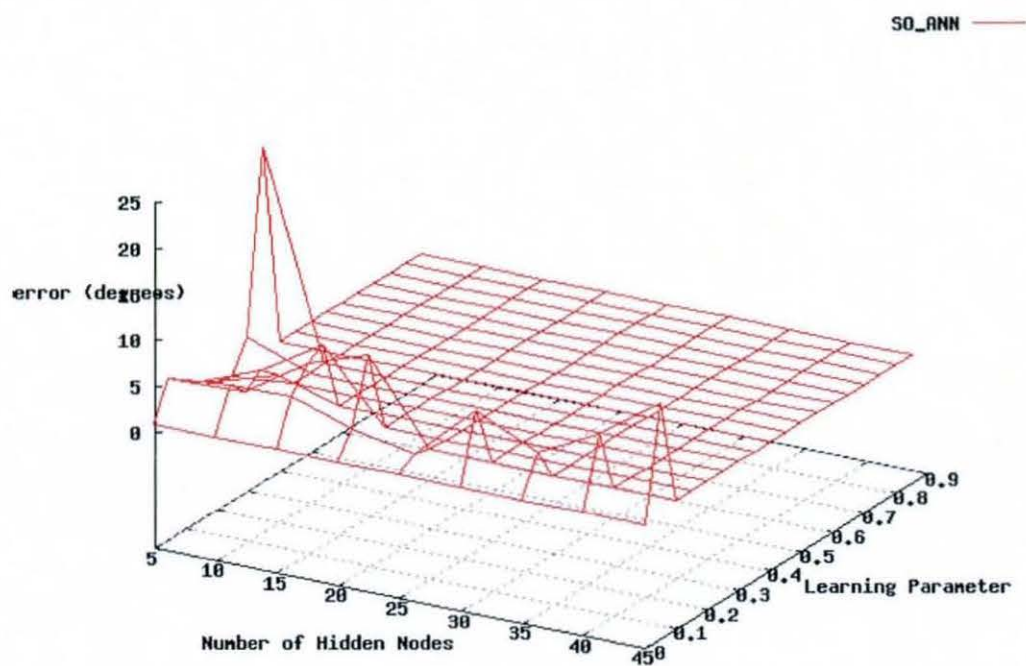


Figure 125 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 35 input nodes trained with backpropagation on synthetic data set 2

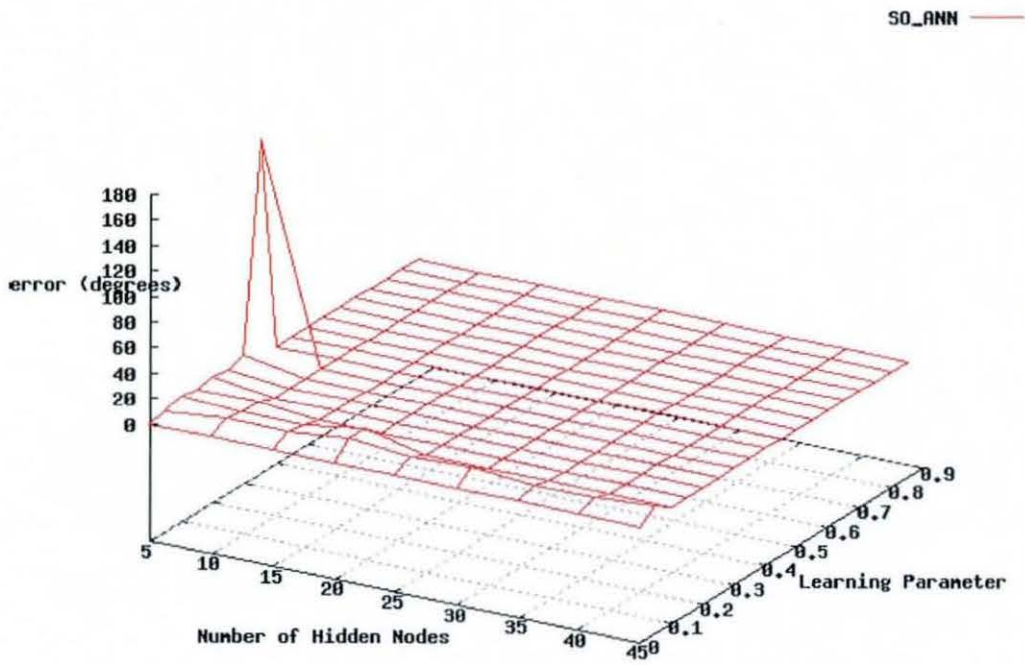


Figure 126 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 40 input nodes trained with backpropagation on synthetic data set 2

Pure synthetic 3

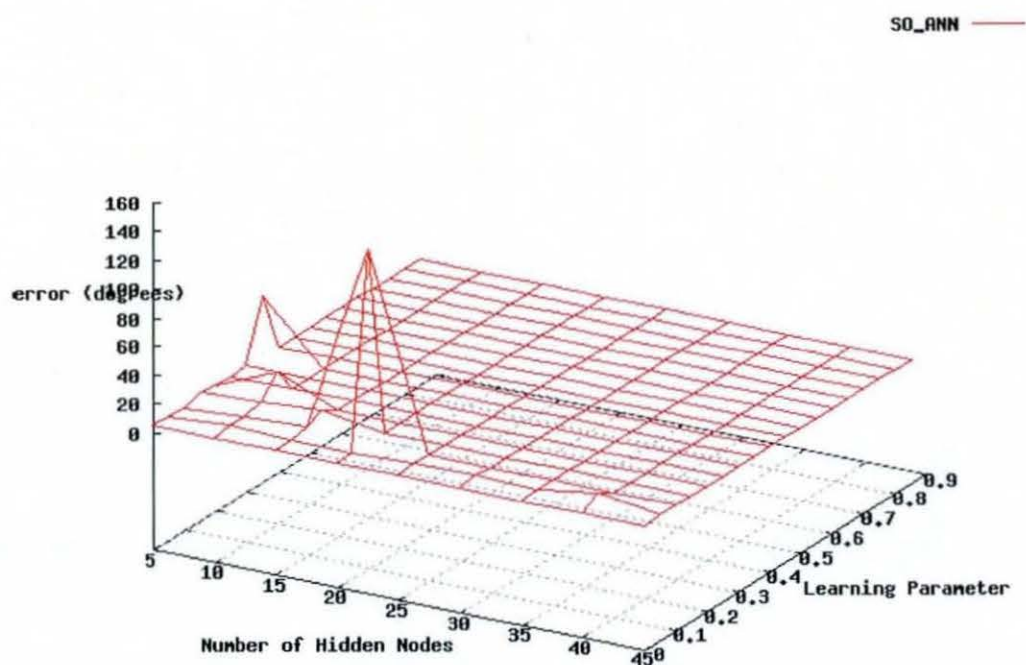


Figure 127 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 5 input nodes trained with backpropagation on synthetic data set 3

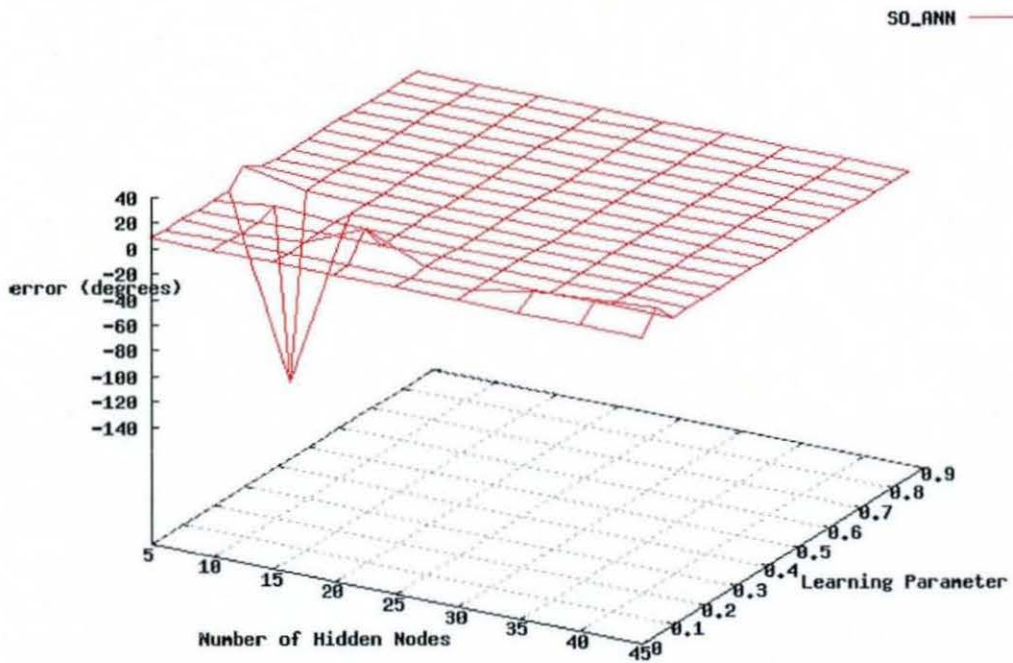


Figure 128 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 10 input nodes trained with backpropagation on synthetic data set 3

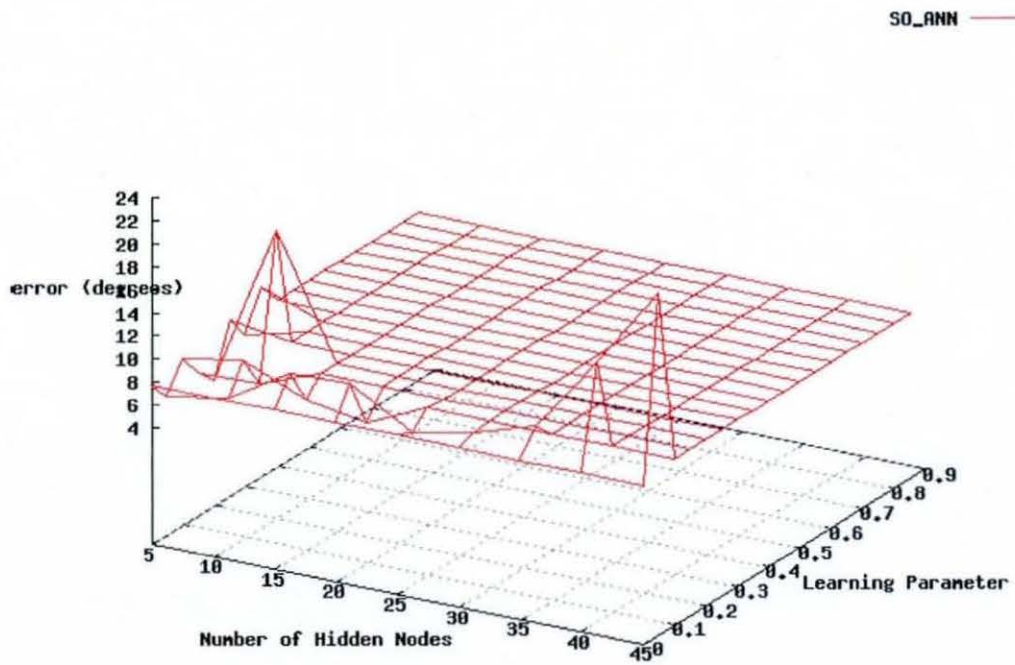


Figure 129 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 15 input nodes trained with backpropagation on synthetic data set 3

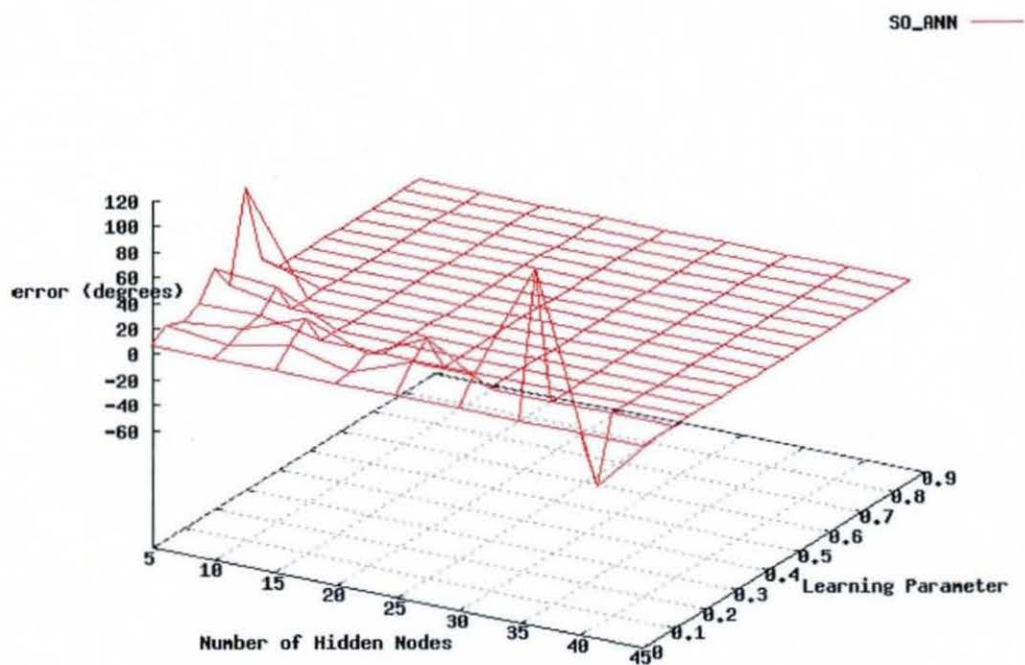


Figure 130 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 20 input nodes trained with backpropagation on synthetic data set 3

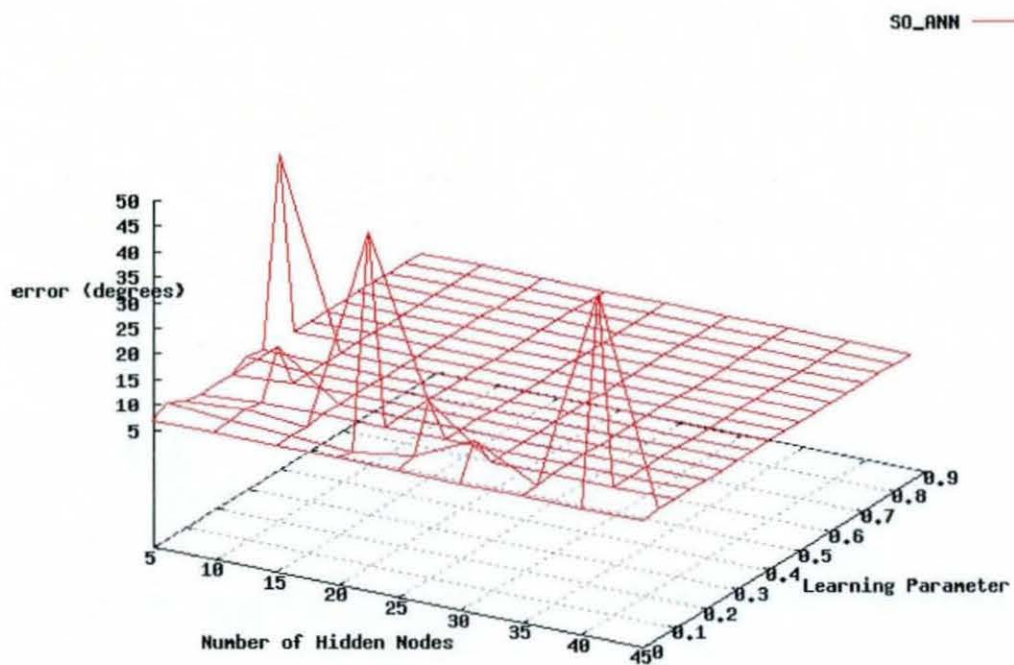


Figure 131 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 25 input nodes trained with backpropagation on synthetic data set 3

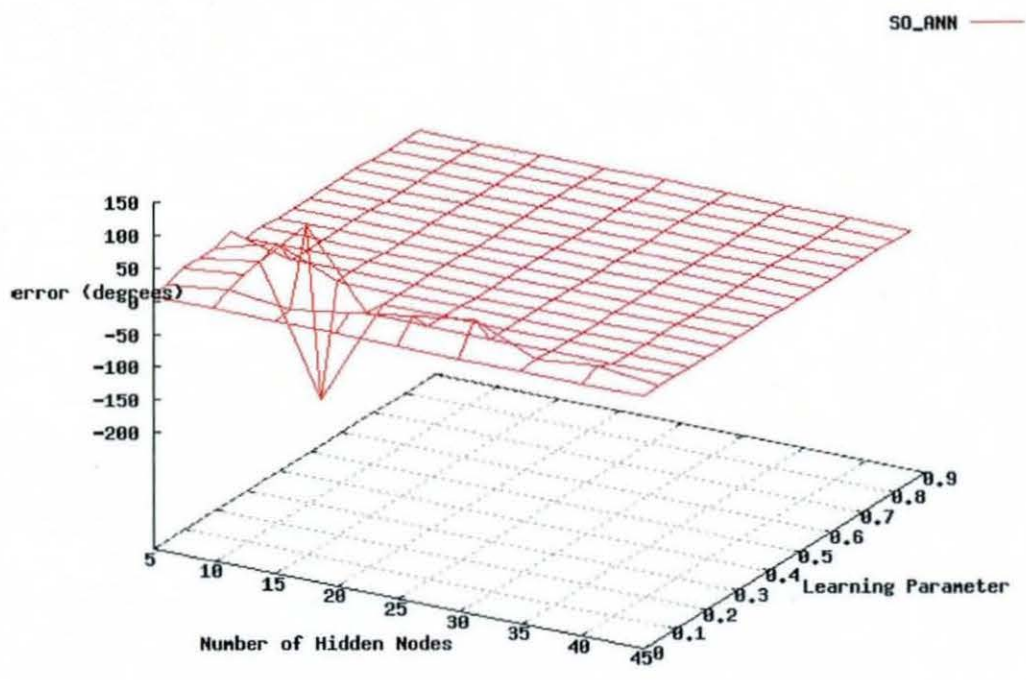


Figure 132 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 30 input nodes trained with backpropagation on synthetic data set 3

SO_ANN

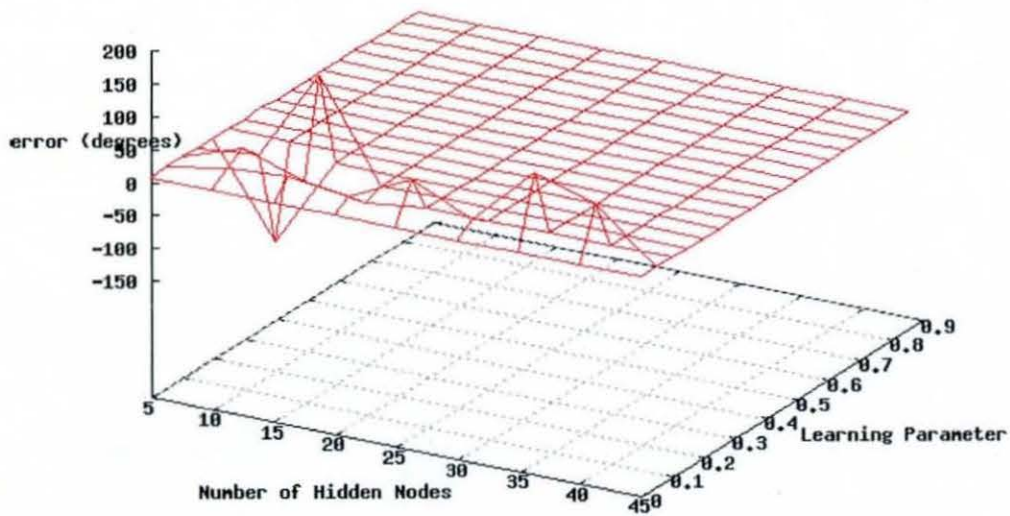


Figure 133 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 35 input nodes trained with backpropagation on synthetic data set 3

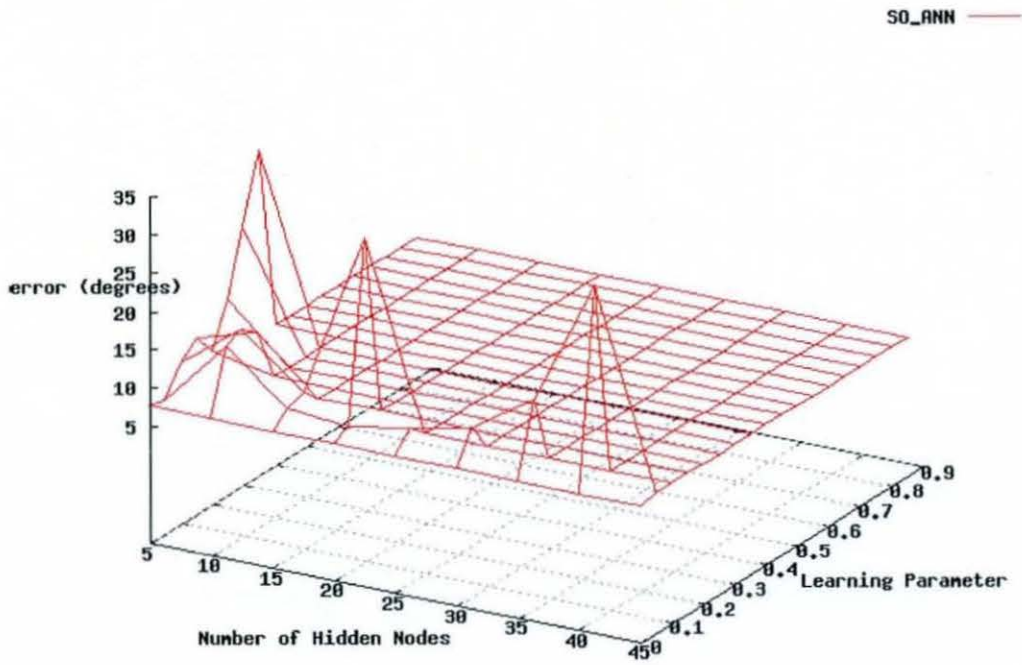


Figure 134 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 40 input nodes trained with backpropagation on synthetic data set 3

Pure synthetic 4

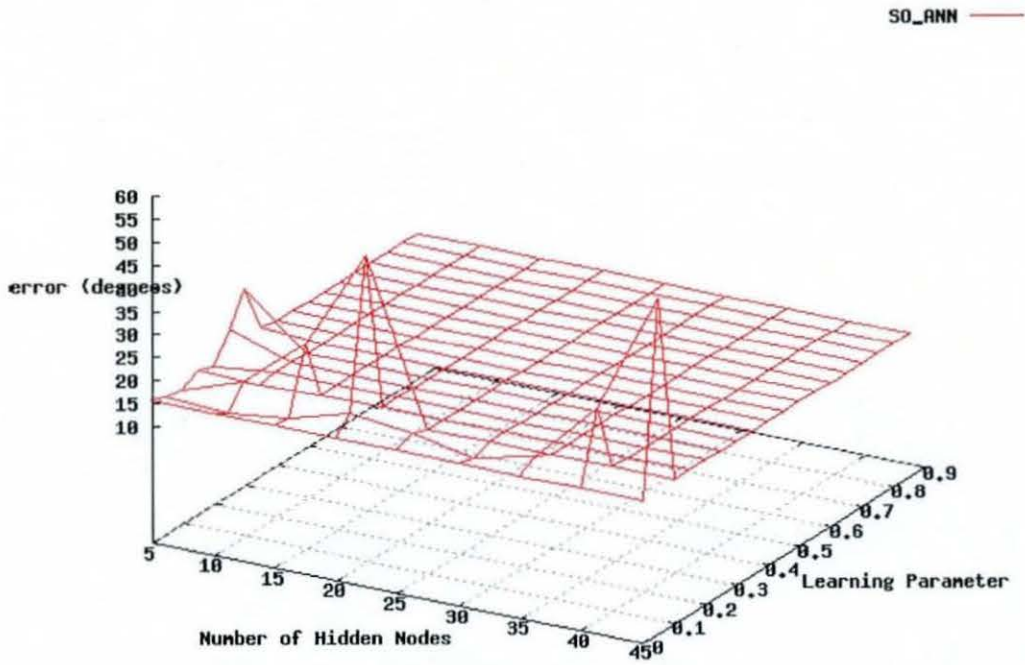


Figure 135 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 5 input nodes trained with backpropagation on synthetic data set 4

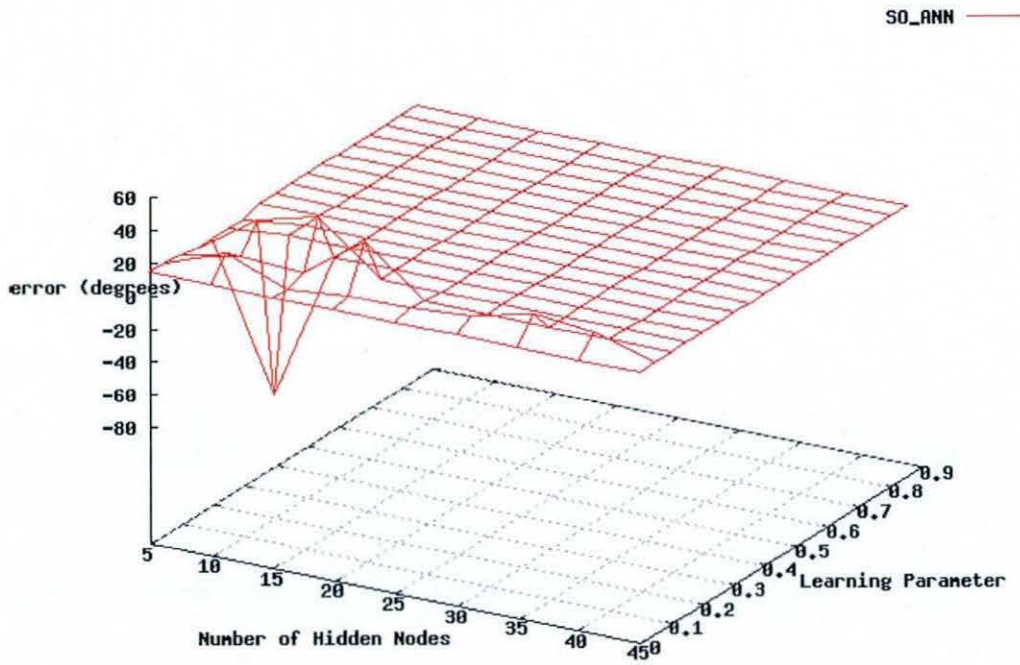


Figure 136 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 10 input nodes trained with backpropagation on synthetic data set 4

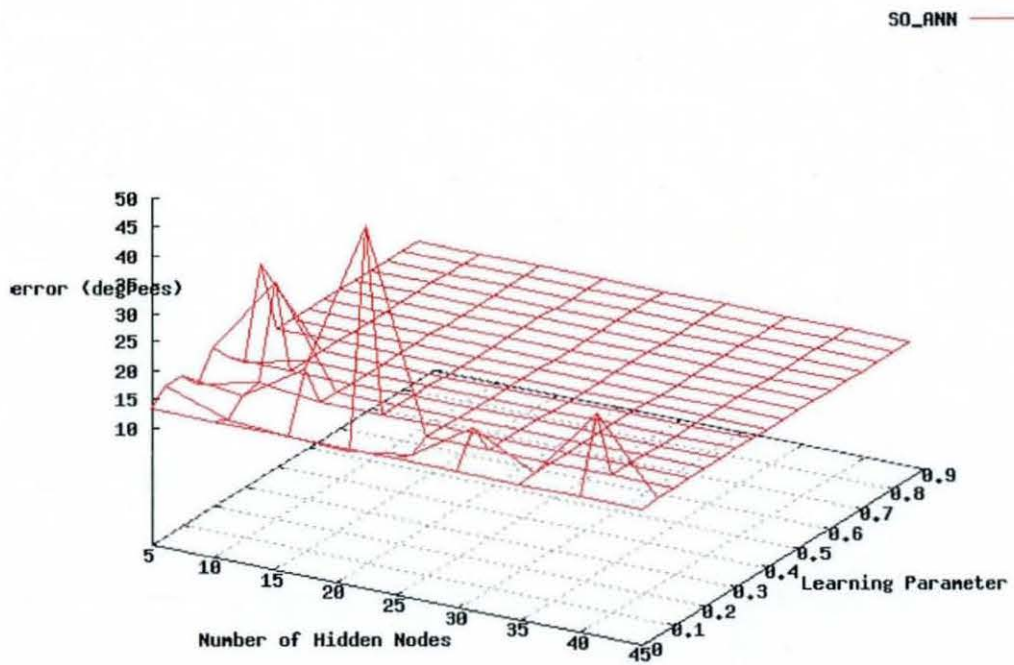


Figure 137 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 15 input nodes trained with backpropagation on synthetic data set 4

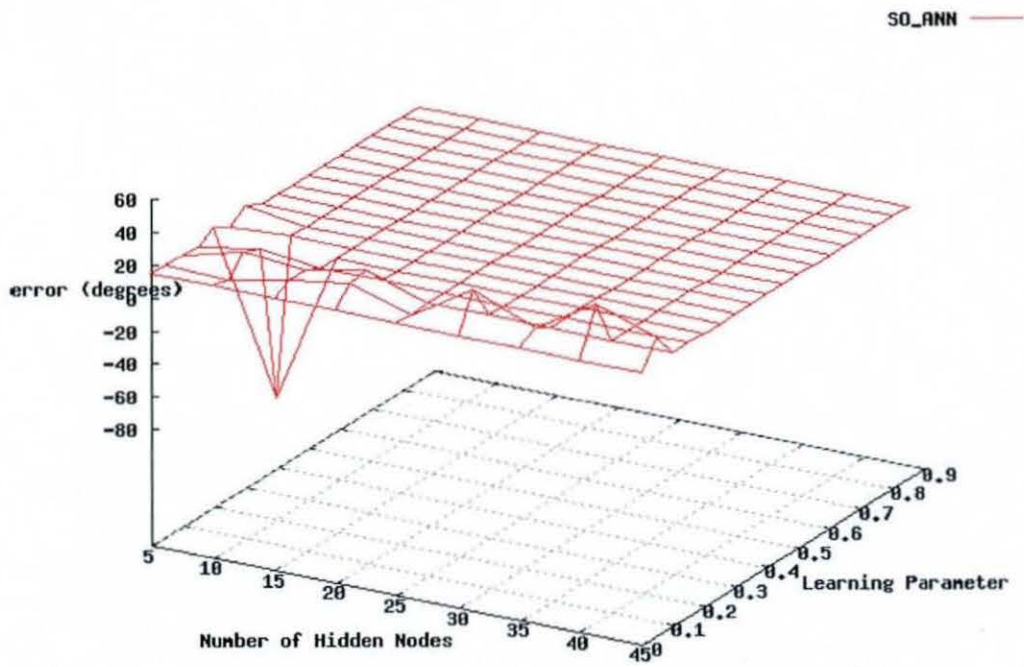


Figure 138 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 20 input nodes trained with backpropagation on synthetic data set 4

SO_ANN

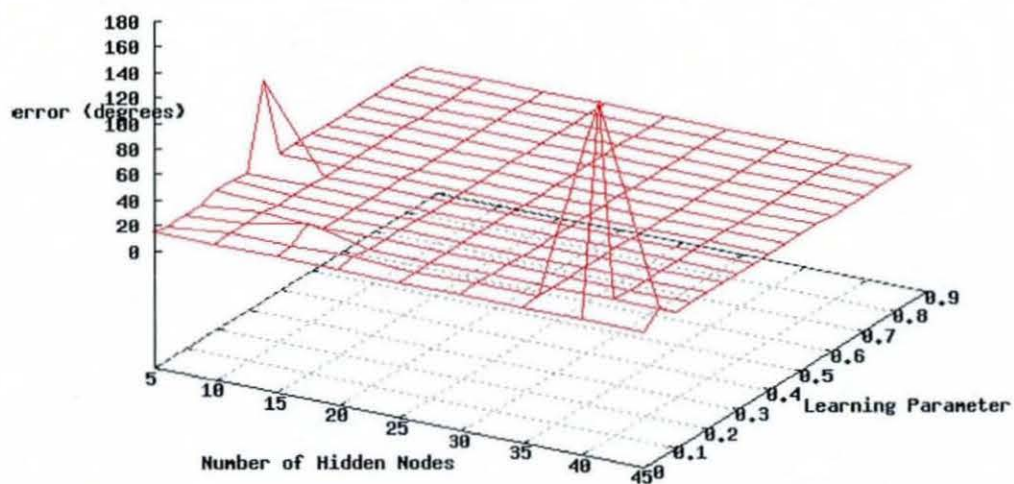


Figure 139 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 25 input nodes trained with backpropagation on synthetic data set 4

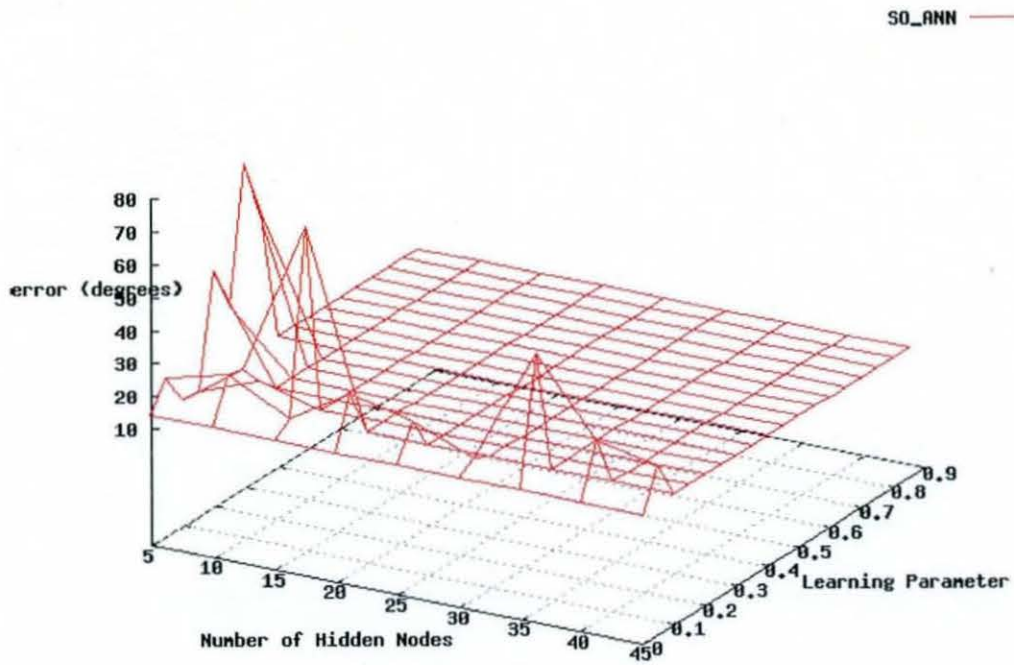


Figure 140 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 30 input nodes trained with backpropagation on synthetic data set 4

SO_ANN —

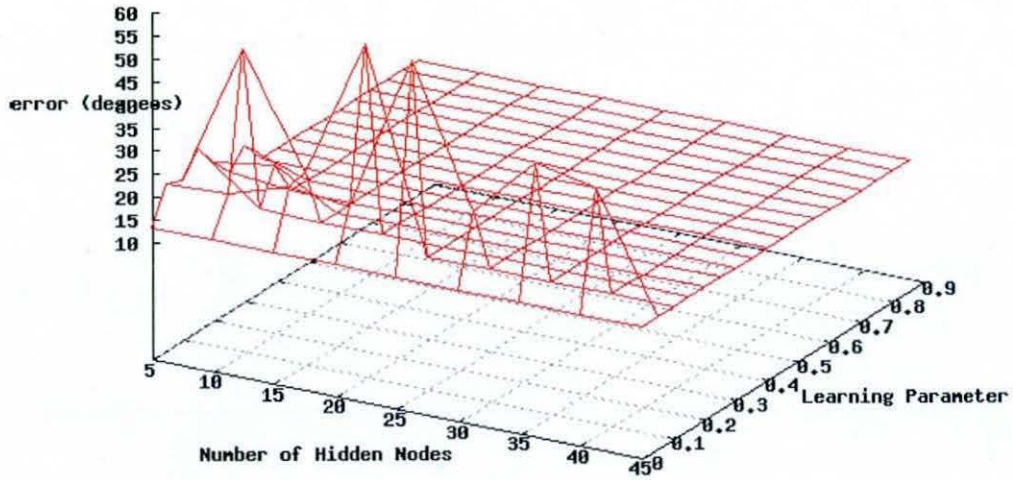


Figure 141 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 35 input nodes trained with backpropagation on synthetic data set 4

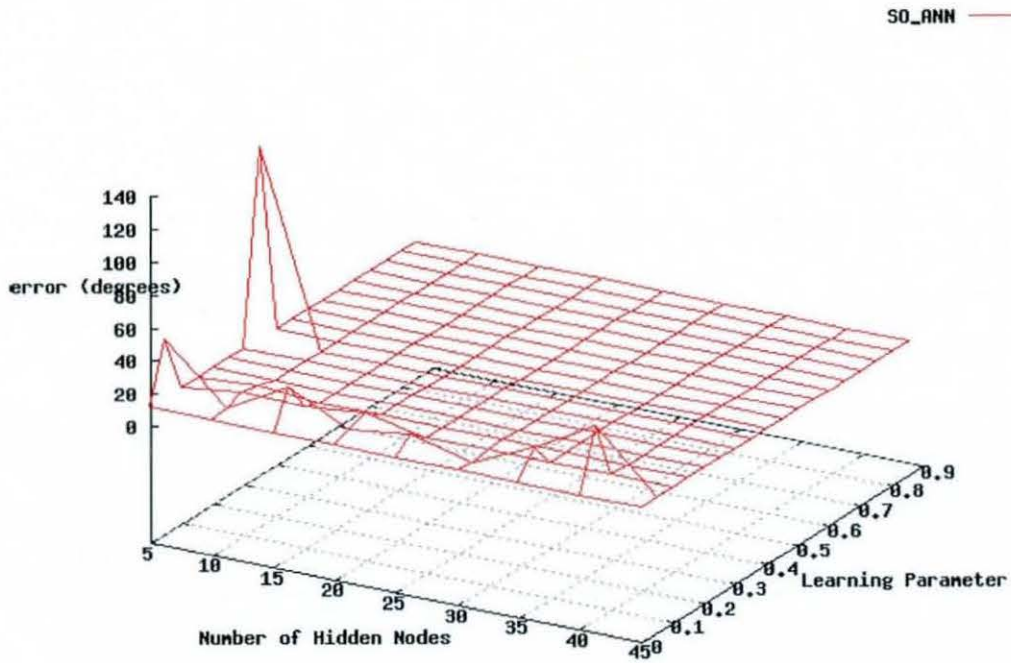


Figure 142 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 40 input nodes trained with backpropagation on synthetic data set 4

Semi-synthetic

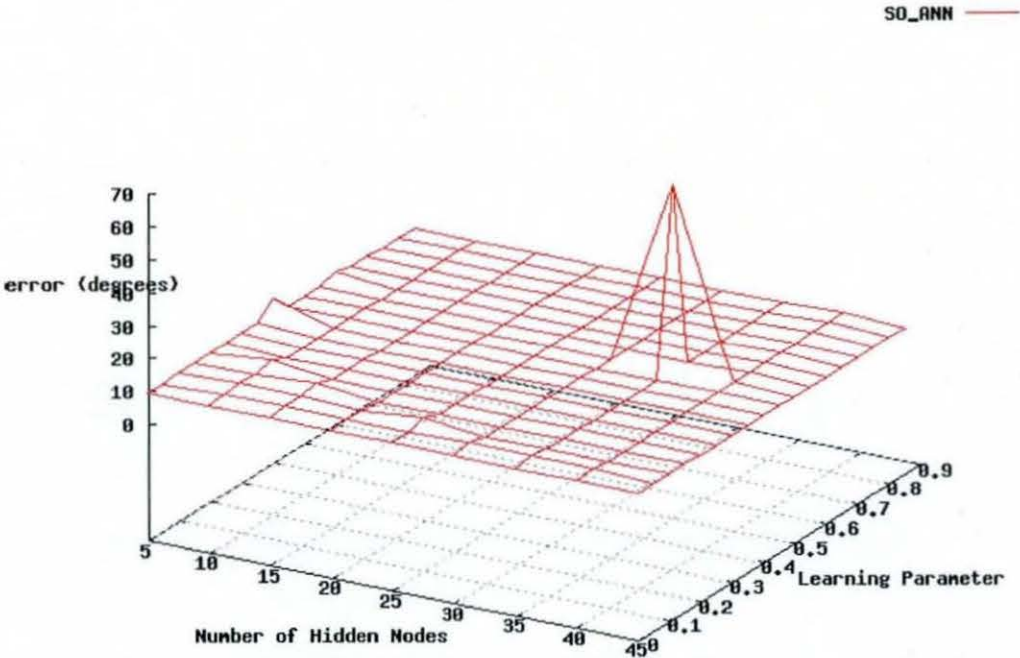


Figure 143 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 5 input nodes trained with backpropagation on semi-synthetic data set

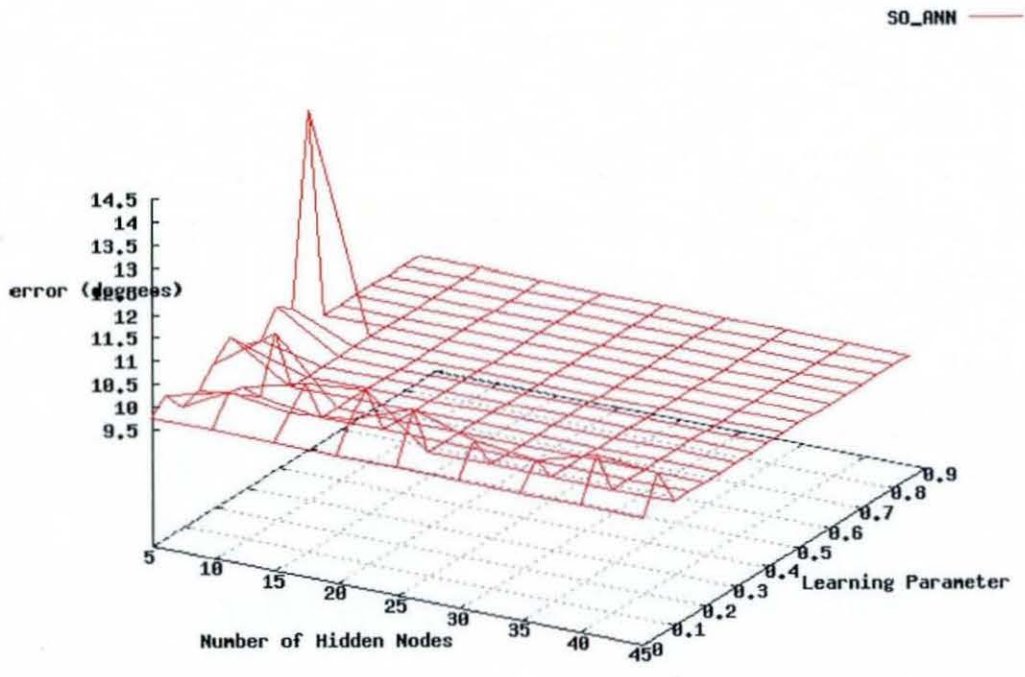


Figure 144 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 10 input nodes trained with backpropagation on semi-synthetic data set

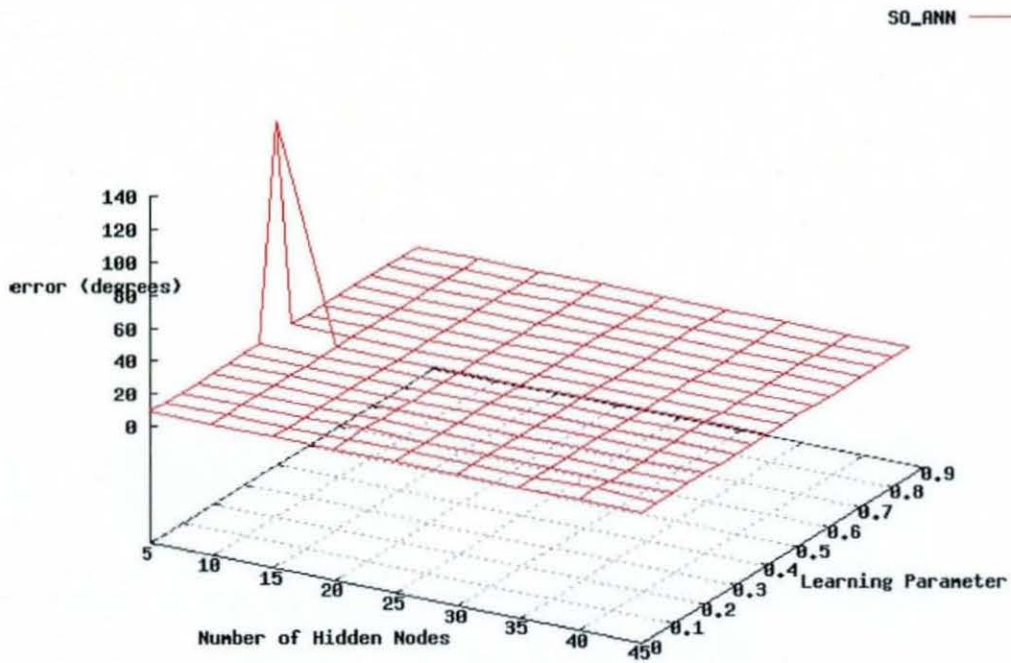


Figure 145 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 20 input nodes trained with backpropagation on semi-synthetic data set

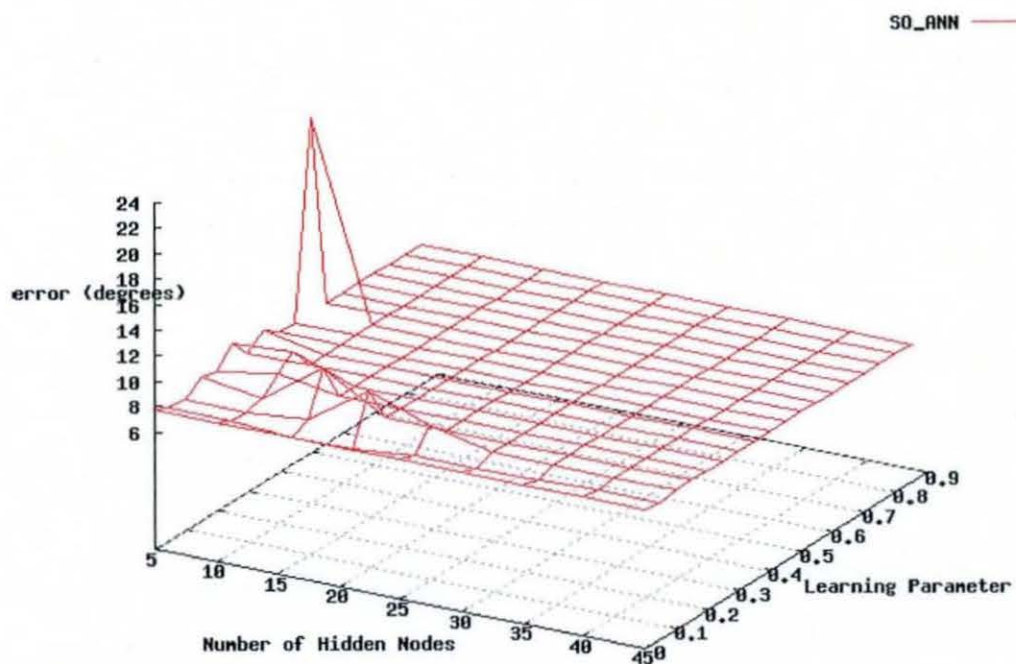


Figure 146 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 25 input nodes trained with backpropagation on semi-synthetic data set

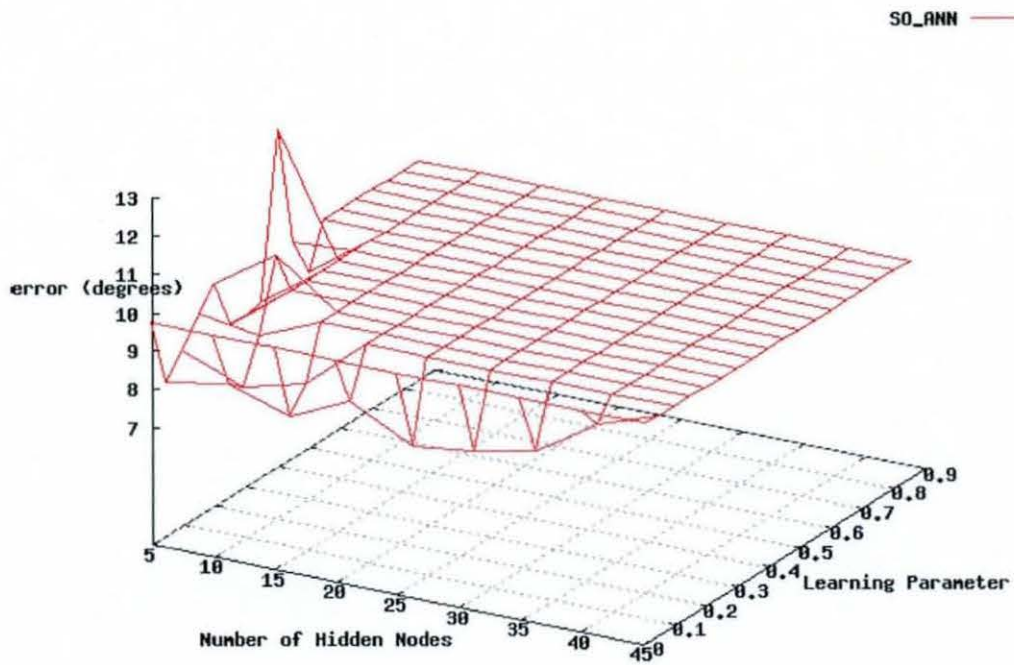


Figure 147 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 25 input nodes trained with backpropagation on semi-synthetic data set

SO_ANN —

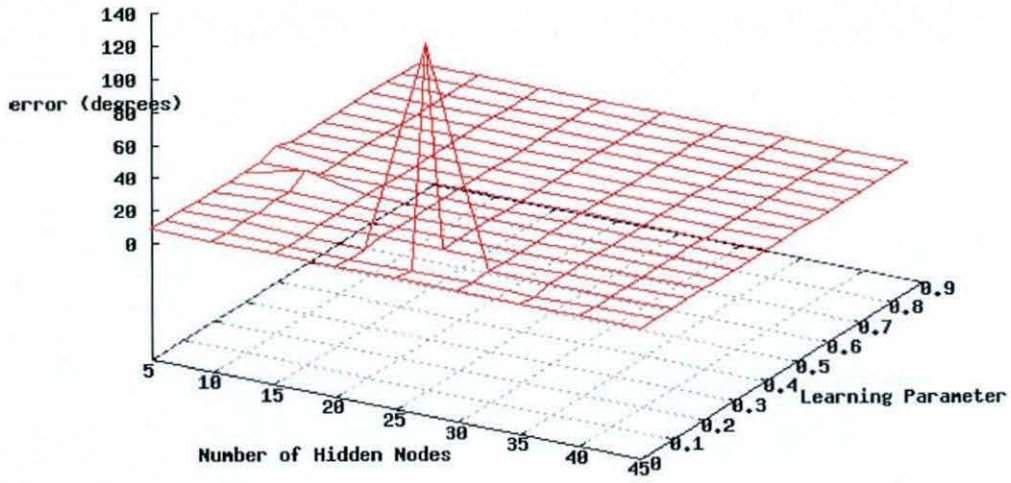


Figure 148 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 30 input nodes trained with backpropagation on semi-synthetic data set

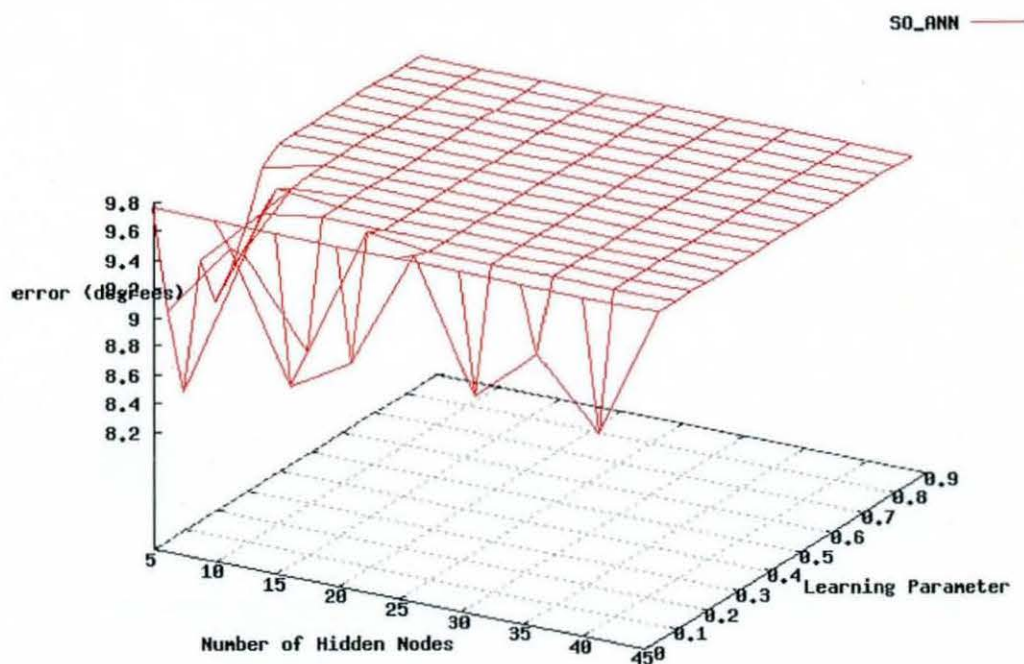


Figure 149 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 35 input nodes trained with backpropagation on semi-synthetic data set

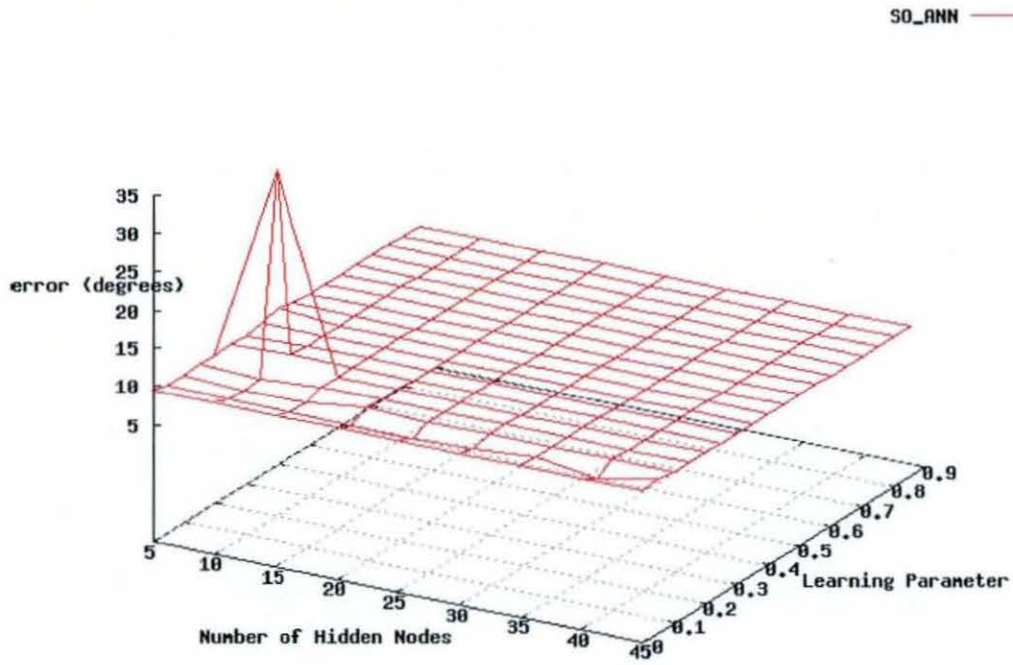


Figure 150 RMS bearing residual over number of hidden nodes between 5 and 45 and learning parameter between 0.01 and 0.9 for a single output feedforward neural network with 40 input nodes trained with backpropagation on semi-synthetic data set

Appendix E - Standard definitions

This appendix comprises of definitions of the standard techniques used within this thesis. These are common and well known enough to be found in most textbooks on the subject, or on the World Wide Web, therefore are excluded from the body of the thesis, however they are described here partly for completeness, and to ensure that the reader understands the author's definition where there is any ambiguity.

Artificial Neural Networks

A standard ANN is made up of discrete processing units known as artificial neurons or nodes. These are highly stylised and simplified imitations of the neurons found in the brain. A node accepts input from one or more sources and provides a single output, each input and output having a separately specified weight. The output of the node is calculated by summing the weighted inputs to the node, passing the result through an activation function, and multiplying by the output weight.

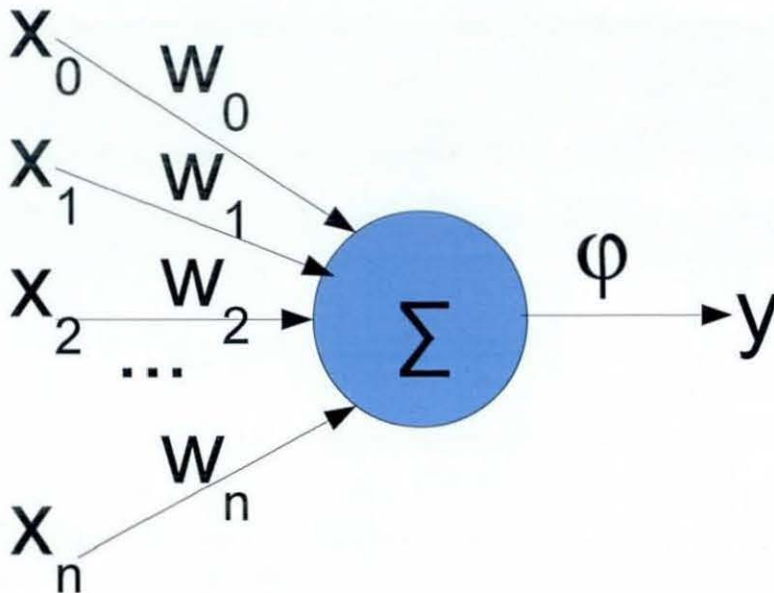


Figure 151 Artificial neuron or node

The function of the node is therefore $y = \phi \sum_{i=0}^n x_i w_i$.

When several of these nodes are connected together, complex behaviour can be obtained. In a typical ANN, a layer of input nodes will accept input. There will be one

or more hidden layers, and finally an output layer of nodes. However unless a non-linear activation function is used, there is little point in using several hidden layers as for every set of multiple hidden layers with linear activation functions there is an equivalent single layer which would give the same result while taking less time to train and test.

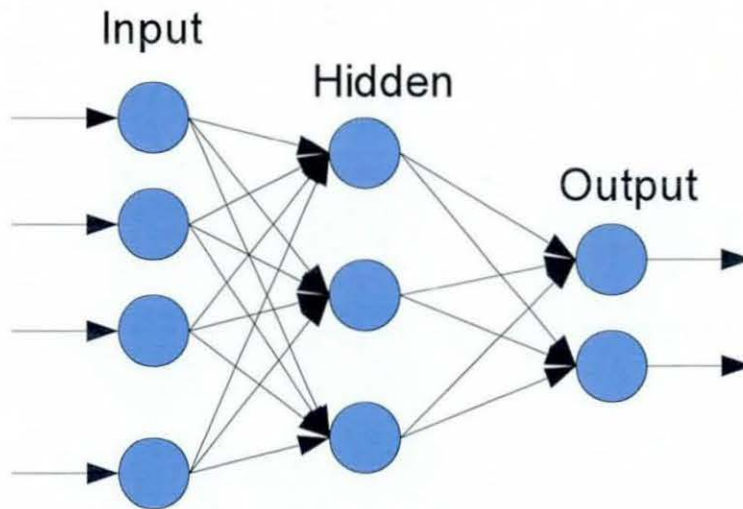


Figure 152 Artificial neural network

A training program must be run to establish the optimal values for the various weights, and a set of examples, known as the training data must be provided, testing the network on each set of inputs provided, and making small changes to the weights to make the result more like the set of outputs in the training data.

Training algorithms

There are two broad categories of ANN training; unsupervised and supervised learning. In unsupervised learning, a program is developed which allows the ANN to make generalisations about the data without being provided with any examples. This is useful in situations in which the truth is either unknown, or prohibitively difficult to calculate. Only input training data is required.

For the second category, supervised learning, both input and output data is required. Here the ANN is gradually tuned to give approximations to the outputs provided. Care must be taken to stop the ANN from overtraining, that is learning the training set so well that it also learns the noise, reducing both its generality and accuracy on the testing data set.

One common supervised learning technique is backpropagation. Backpropagation is a gradient decent technique for incrementally improving the performance of the network. First a forward pass is performed to calculate the output at each node for a given input, followed by a backward pass to calculate the ideal output at each node. The amount the weights between the hidden layer and the output layer must change is calculated, then the same is calculated for the weights between the input layer and the hidden layer. Finally the weights are adjusted towards the ideal values, the delta calculated for each node is multiplied by the learning parameter before use, to control the rate of learning. The learning parameter is normally set to between zero and one. At zero no change to the network is made, while if set to one the weights are adjusted to make the network perfectly tuned to the current training example. In most practical applications a low value is used, and several iterations of the training procedure are performed across as many examples as possible. The backpropagation training algorithm is good at finding local minima quickly.

Genetic algorithms

When searching for solutions to a problem, one technique which is capable of simultaneously searching for a collection of solutions is a Genetic Algorithm (GA). The GA in its simplest form has a predefined fitness function, which measures the relative performance of competing binary strings. This fitness function could, for example, describe the input parameters for an algorithm, in which case the fitness function would be a measure of how well the algorithm performed with the parameters parsed from the string.

A number of these binary strings are randomly generated, and then the performance of each is measured. Using this performance individuals are selected randomly, but with the probability of selection being proportional to their measured fitness. This can be imagined as being like a casino roulette wheel in which each individual has a slot on the wheel, however rather than having the standard, equal width divisions, the size of each slot is proportional to the fitness of the individual represented.

Solutions selected by the roulette wheel are 'mated' and a second generation of individuals which are combinations of their parents is created. The simplest approach for this is to select a random point along the string and swap the two parents over after that point, resulting in two children, although more complicated crossover schemes exist such as two point crossover.

Mutation is also performed on these children in which random bits in the binary string are inverted at a pre-defined rate. After several generations the resultant classifier should be able to outperform significantly any of the initial random individuals. The expectation is that this will lead to the population becoming increasingly fit over time.

Multi-objective genetic algorithms

Many situations exist in which there is not just one objective for success, but multiple non-complimentary competing objectives. An example of this would be to design a car, in which top speed and fuel efficiency are competing objectives. In these situations a genetic algorithm can still be used and there are many different ways of utilising the results for each objective;

1. Separate sub-populations

Effectively a separate GA is run for each objective, and the population is recombined at the end.

2. Combination function

A function $f(x,y)$ is created which combines results for each objective into a single number, allowing a standard GA to be used.

3. Pareto optimality and dominance ranking

See next section.

A full description of the state of the art in MOEAs is outside the scope of this thesis, however [4] provided a comprehensive survey.

Pareto optimality

Pareto optimality gives a formal way of establishing the set of individuals in a population which represent the best trade-off between two or more competing objectives. To find the Pareto set, the non-dominated individuals are found. An individual in the population is said to be dominated if another member of the population exists which is better for every one of the objectives. The non-dominated set of individuals forms what is known as the Pareto-Front. An example of this is shown in Figure 153, where the objective is to minimise the values of the two competing objectives, the line in green represents the Pareto Front.

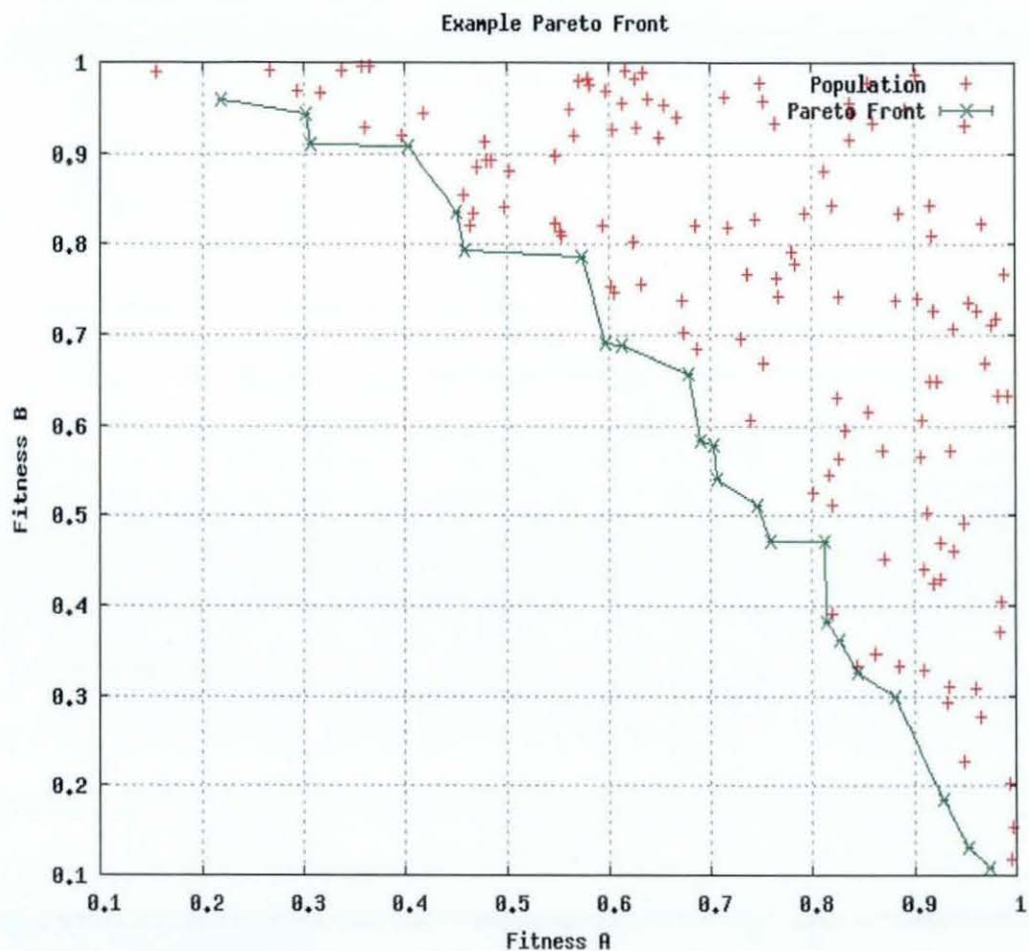


Figure 153 An example Pareto Front

It is possible to use the idea of dominance to form a ranking system. Figure 154 gives one way of calculating this (based on NSGA-II [77]), which assigns a fitness ranking to all individuals of a population which is inversely proportional to the distance of the individual to the Pareto Front. This is the system used in the MOGA experiments of this thesis to select the individuals for use in the following generation of the GA, the closer the individual is to the Pareto Front, the more likely it is to be selected for use in the following generation.

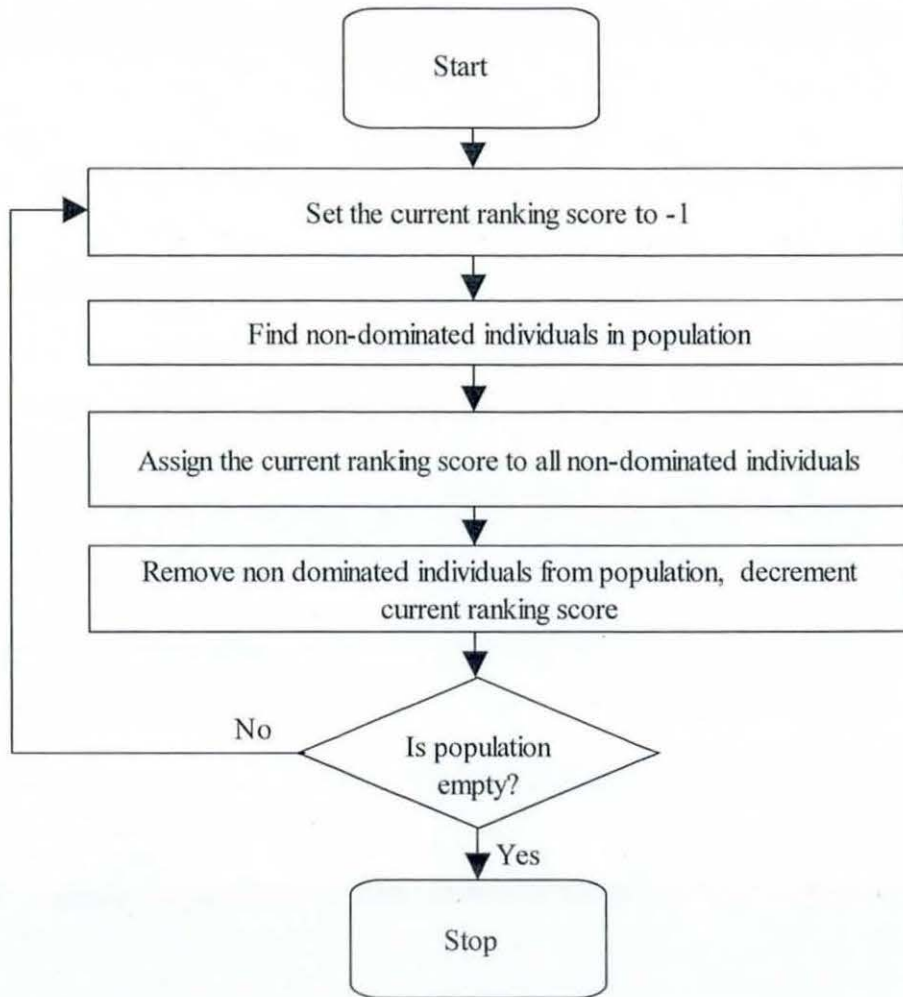


Figure 154 Calculating dominance ranking

Measuring diversity

[168] experimented with several measures of classifier diversity for classifiers and found that the pattern of the relationship between ensemble accuracy and diversity was not substantially different between the nine measures of diversity used. As there is little difference between their performance, the choice of which to use was arbitrary. From the nine, four have been selected for use in this thesis;

1. Entropy

$$E = \frac{1}{N} \frac{2}{L-1} \sum_{j=1}^N \min \left\{ \left(\sum_{i=1}^L y_{i,j} \right), \left(\sum_{i=1}^L L - y_{i,j} \right) \right\}$$

where L is the number of individual classifiers,

N is the number of data sets

and y_{ij} is the output of the i^{th} classifier on the j^{th} data set (where 1 denotes a correct classification and 0 an incorrect one.)

2. Kohavi-Wolpert distance [163]

$$Y(z_j) = \sum_{i=1}^L y_{i,j}$$

$$KW = \frac{1}{NL^2} \sum_{j=1}^N Y(z_j)(L - Y(z_j))$$

3. Generalised diversity

Generalised diversity is calculated by taking a subset of the test data and using it to calculate the probabilities of i members of the ensemble will fail on any given data set.

$$p(1) = \sum_{i=1}^L \frac{i}{L} p_i$$

$$p(2) = \sum_{i=1}^L \frac{i(i-1)}{L(L-1)} p_i$$

$$GD = 1 - \frac{p(1)}{p(2)}$$

where p_i is the probability that exactly i classifiers in the ensemble will fail, calculated empirically from the test data.

4. Hamming distance

Pairwise comparisons are made between each binary gene string in the ensemble, and the hamming distance is calculated between each. The mean of these values is used as the value for the ensemble.

Measuring accuracy

Four metrics have been chosen; RMS error due to its widespread use, and ease of understanding, Geometric Mean of Relative Absolute Error (GMRAE), Median RAE (MdRAE) and Median Absolute Percentage Error (MdAPE), the latter three being the statistics recommended by [10] as the best measures of accuracy in this type of problem. RMS error is too well known to outline here, however the other three are given in the following sections.

Geometric Mean of Relative Absolute Error (GMRAE)

In order to calculate the Relative Absolute Error (RAE) of each measurement, a random walk must be generated. The predictions are then compared to the random walk to

produce the RAE. RAE is defined as $error_{RAE} = \left| \frac{x_{predicted} - x_{observed}}{x_{random_walk} - x_{observed}} \right|$.

The values of RAE are then Winsorized; very low and very high values are removed and replaced with values on the boundaries, in order to eliminate outliers.

$$0.01 \text{ if } RAE < 0.01$$

$$WRAE = RAE \text{ if } 0.01 \leq RAE \leq 10$$

$$10 \text{ if } RAE > 10$$

Finally GMRAE is calculated as the mean of the Winsorized RAEs.

Median RAE (MdRAE)

The Winsorized RAE values are calculated, and then the MdRAE is defined as the median of the Winsorized RAEs.

Median Absolute Percentage Error (MdAPE)

Absolute Percentage Error (APE) is defined as $error_{APE} = \left| \frac{x_{predicted} - x_{observed}}{x_{observed}} \right|$.

This error value is calculated for every predicted value in the time series. MdAPE is the median value of APE for the whole time series.

