A critical comparison of feature selection algorithms for improved classification accuracy

W Snyman

orcid.org/0000-0002-8649-1052

Dissertation submitted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor: Prof PA van Vuuren

Graduation ceremony: May 2019

Student number: 29927978

Abstract

Feature selection is crucial for increasing the performance of predictive models in both classification accuracy and model training time. For high-dimensional data, feature selection becomes ever more necessary to select adequate features which complement the predictive model of choice. Filter, wrapper and embedded feature selection techniques are among the most popular algorithms to solve the feature selection conundrum of irrelevant and redundant features reducing the performance of classification models. It is also common to find hybrid techniques which combines filter, wrapper and embedded techniques to construct more robust feature selection algorithms. This study is dedicated to reveal the ongoing improvement in the field of feature selection and to dissect six different feature selection algorithms for a detailed insight into their success for high-dimensional data, specifically gene expression microarrays. The six algorithms for this study are: (i) three filter methods mRMR (min-Redundancy Max-Relevance), FCBF# (Fast Correlation Based Filter) and ORFS (Orthogonal Relevance Feature Selection), (ii) two wrapper methods FRBPSO (Fuzzy Rule Based Particle Swarm Optimisation) and SVM-RFE (Support Vector Machine-Recursive Feature Elimination), and (iii) one embedded method SBMLR (Sparse Multinomial Logistic Regression via Bayesian L_1 regularisation). The three filter methods are adapted into suitable hybrid techniques and multiple associative measures are explored to determine the best performance per algorithm. All algorithms include the pre-processing techniques MDL discretisation and SIS to explore their improvements and shortcomings. The performance per algorithm is based on their ability to improve classification accuracy with the least amount of features possible and compared to one another. After comparison, the algorithms best suited for classification improvement, computation speed advantage and feature removal capability are revealed. Thereafter, a case study involving plant foliage features where the amount of features greatly outnumber the number of samples, denoted by $p >> n$, is used to compliment the findings. The use of pre-processing techniques proved to be

crucial regarding improved classification accuracy and reduced computation time. Out of all six algorithms, mRMR, and SVM-RFE proved the most promising.

Keywords: feature selection, classification, support vector machines, particle swarm optimisation, mutual information

Acknowledgements

I would like to thank Prof. PA van Vuuren for his great mentorship and insight towards this dissertation.

Contents

1 Introduction **1**

1.1 Background . . . 1

1.2 Literature review . . . 2

1.2.1 Main approaches . . . 2

1.2.2 Recent focus . . . 3

1.2.3 Exploring the common methods . . . 4

1.2.4 Introducing hybrids . . . 5

1.2.5 Metaheuristic optimisation . . . 7

1.2.6 Feature selection stability . . . 8

1.2.7 SVM feature selection and hybridisation . . . 8

1.2.8 Sparsity and regression . . . 11

1.3 Research question . . . 12

1.4 Method . . . 12

1.5 Subsequent structure . . . 14

2 Formulation of main feature selection algorithms **15**

2.1 mRMR . . . 16

2.1.1 Alternative associative measures . . . 19

2.1.2 Implementation of mRMR . . . 20

2.1.3 A possible improvement using semi-mRMR . . . 21

2.1.4 Limitations of mRMR . . . 23

2.2 ORFS . . . 24

2.2.1 Orthogonolisation . . . 25

2.2.2 ORFS implementation . . . 28
2.3 FCBF . . . 29
2.3.1 Correlation measure . . . 30
2.3.2 FCBF methodology . . . 31
2.3.3 FCBF algorithm . . . 31
2.3.4 Implementing an alternate search strategy (FCBF#) . . . 33
2.4 PSO . . . 35
2.4.1 Defining PSO . . . 35
2.4.2 Binary PSO . . . 37
2.4.3 Including fuzzy logic . . . 38
2.4.4 Implementing FRBPSO . . . 40
2.5 SVM . . . 42
2.5.1 Formulation of a linearly separable Hard Margin SVM . . . 42
2.5.2 Formulation of a linearly separable Soft Margin SVM . . . 48
2.5.3 Solving a SVM using the dual formulation . . . 51
2.5.4 Implementing SVM feature selection . . . 57
2.6 Regression . . . 60
2.6.1 OLS regression . . . 60
2.6.2 Ridge regression . . . 63
2.6.3 Lasso regression . . . 65
2.6.4 Elastic Net regression . . . 67
2.6.5 Logistic regression . . . 71
2.7 Brief chapter summary . . . 75

3 Numerical experiments **77**

3.1 Filter methods . . . 78
3.1.1 mRMR . . . 79
3.1.2 FCBF# . . . 88
3.1.3 ORFS . . . 93
3.2 Wrapper and embedded methods . . . 102
3.2.1 FRBPSO . . . 102
3.2.2 SVM-RFE . . . 107

3.2.3 SBMLR 111
3.3 Summary of simulation results 116

4 Case study **121**
4.1 Context 121
4.2 Simulation results 123

5 Conclusion **125**

References **127**

List of Figures

1.1 Feature selection frameworks 4

1.2 Trade-offs between norms 10

2.1 Mutual Information between variables A and B 18

2.2 Relation between two variables A and B with class variable C 23

2.3 Vector diagram of feature f_1 and candidate feature f_2 26

2.4 Graphical illustration of GSO extremes 28

2.5 SVM optimisation problem 43

2.6 Distance between the margins 45

2.7 Not linearly separable 48

2.8 Non-linearly separable data example 55

2.9 Transformed data which is now linearly separable 55

2.10 Ordinary Least Squares regression on two dimensions 61

2.11 L_2-norm constraint on $\boldsymbol{\beta}$ coefficients in two dimensions 64

2.12 L_2-norm constraint and its effect on the optimal solution 65

2.13 Differences between L_2 and L_1-norms 66

2.14 L_1-norm constraint and its effect on the optimal solution 66

2.15 Instability of L_1-norm 67

2.16 A combination of L_1 and L_2-norms on the solution space in two dimensions 67

3.1 Best mRMR results for 11 Tumors 81

3.2 Best mRMR results for 9 Tumors 81

3.3 Best mRMR results for Brain Tumor 1 81
3.4 Best mRMR results for Brain Tumor 2 82
3.5 Best mRMR results for DLBCL 82
3.6 Best mRMR results for GLI-85 82
3.7 Best mRMR results for Leukemia 1 83
3.8 Best mRMR results for Leukemia 2 83
3.9 Best mRMR results for Lung Cancer 83
3.10 Best mRMR results for Prostate 84
3.11 Best mRMR results for SMK-CAN-187 84
3.12 Best mRMR results for SRBCT 84
3.13 Best unsupervised FCBF# for 9 Tumors 90
3.14 Best unsupervised FCBF# for Leukemia 1 90
3.15 Best unsupervised FCBF# for Lung Cancer 90
3.16 Best unsupervised FCBF# for Prostate 90
3.17 Best unsupervised FCBF# for SMK-CAN-187 90
3.18 Best supervised FCBF# for SRBCT 90
3.19 Best ORFS results for 11 Tumors 95
3.20 Best ORFS results for 9 Tumors 95
3.21 Best ORFS results for Brain Tumor 1 95
3.22 Best ORFS results for Brain Tumor 2 96
3.23 Best ORFS results for DLBCL 96
3.24 Best ORFS results for GLI-85 96
3.25 Best ORFS results for Leukemia 1 97
3.26 Best ORFS results for Leukemia 2 97
3.27 Best ORFS results for Lung Cancer 97
3.28 Best ORFS results for Prostate 98
3.29 Best ORFS results for SMK-CAN-187 98
3.30 Best ORFS results for SRBCT 98
3.31 Best FRBPSO results for 11 Tumors 103

3.32 Best FRBPSO results for 9 Tumors . . . 103
3.33 Best FRBPSO results for Brain Tumor 1 . . . 104
3.34 Best FRBPSO results for Brain Tumor 2 . . . 104
3.35 Best FRBPSO results for DLBCL . . . 104
3.36 Best FRBPSO results for 11 GLI-85 . . . 104
3.37 Best FRBPSO results for Leukemia 1 . . . 104
3.38 Best FRBPSO results for Leukemia 2 . . . 104
3.39 Best FRBPSO results for Lung Cancer . . . 105
3.40 Best FRBPSO results for Prostate Tumor . . . 105
3.41 Best FRBPSO results for SMK-CAN-187 . . . 105
3.42 Best FRBPSO results for SRBCT . . . 105
3.43 Best SVM-RFE results for 11 Tumors . . . 108
3.44 Best SVM-RFE results for 9 Tumors . . . 108
3.45 Best SVM-RFE results for Brain Tumor 1 . . . 109
3.46 Best SVM-RFE results for Brain Tumor 2 . . . 109
3.47 Best SVM-RFE results for DLBCL . . . 109
3.48 Best SVM-RFE results for GLI-85 . . . 109
3.49 Best SVM-RFE results for Leukemia 1 . . . 109
3.50 Best SVM-RFE results for Leukemia 2 . . . 109
3.51 Best SVM-RFE results for Lung Cancer . . . 110
3.52 Best SVM-RFE results for Prostate . . . 110
3.53 Best SVM-RFE results for SMK-CAN-187 . . . 110
3.54 Best SVM-RFE results for SRBCT . . . 110
3.55 SBMLR results for 11 Tumors . . . 113
3.56 SBMLR results for 9 Tumors . . . 113
3.57 SBMLR results for Brain Tumor 1 . . . 113
3.58 SBMLR results for Brain Tumor 2 . . . 113
3.59 SBMLR results for DLBCL . . . 113
3.60 SBMLR results for GLI-85 . . . 113

3.61 SBMLR results for Leukemia 1 114
3.62 SBMLR results for Leukemia 2 114
3.63 SBMLR results for Lung Cancer 114
3.64 SBMLR results for Prostate 114
3.65 SBMLR results for SMK-CAN-187 114
3.66 SBMLR results for SRBCT 114

4.1 Feature extraction process 122
4.2 Best results excluding FRBPSO 123
4.3 Best convergence run for FRBPSO 123

List of Tables

1.1 Importance of synergy 2

2.1 Gaussian membership function parameters 39
2.2 Fuzzy Rules 39
2.3 Fuzzy methods 40
2.4 Two common kernels widely used 57

3.1 Datasets used in this study 78
3.2 Original 1-NN LOOCV accuracies 78
3.3 Summary of the best mRMR results 85
3.4 Summary of the preferred associative measure 85
3.5 Summary of the preferred filter methods 86
3.6 Summary of computation time for the best results 86
3.7 Summary of the best FCBF# results 91
3.8 Summary of the preferred associative measure for each best result 91
3.9 Summary of the preferred filter methods for each best result 92
3.10 Summary of the best ORFS results 99
3.11 Summary of the preferred associative measure 99
3.12 Summary of the preferred filter methods 100
3.13 Summary of computation time for the best results 100
3.14 Summary of the best FRBPSO results 105
3.15 Summary of the preferred filter methods per best result 106

3.16 Average results over all 10 runs for the best pre-processing combination . . 106
3.17 Summary of the best SVM-RFE results 110
3.18 Summary of the preferred filter methods 111
3.19 Summary of the best SBMLR results 115
3.20 Summary of the preferred filter methods 115
3.21 Summary of best results for each algorithm 116
3.22 Summary of improvement for each algorithm 116
3.23 Summary of best combination per filter algorithm 119
3.24 Summary of best combination per wrapper and embedded algorithm . . . 119
3.25 Summary of advantages for each algorithm 120

4.1 Samples for each ailment 122
4.2 Summary of case study results for each algorithm 123
4.3 Preference for the filter algorithms 124
4.4 Preference for the wrapper and embedded algorithms 124

LIST OF COMMON ABBREVIATIONS

B-BPSO	Boolean operation Binary Particle Swarm Optimisation
CFS	Correlation based Feature Selection
DCD	Dual Coordinate Descent
dCorr	Distance Correlation
FCBF	Fast Correlation Based Filter
FIS	Fuzzy Inference System
FRBPSO	Fuzzy Rule Based Particle Swarm Optimisation
IG	Information Gain
SGD	Stochastic Gradient Descent
GSO	Gram-Schmidt Orthogonolisation
k-NN	k-Nearest-Neighbour
LDIW	Linearly Decreasing Inertia Weights
LOOCV	Leave One Out Cross Validation
LR	Logistic Regression
MDL	Minimum Descriptive Length
MI	Mutual Information
MIC	Maximal Information Coefficient
mRMR	Minimum Redundancy Maximum Relevance
OLS	Ordinary Least Squares
OMICFS	Orthogonal Maximal Information Coefficient Feature Selection
ORFS	Orthogonal Relevance Feature Selection
PC	Pearson correlation Coefficient
PSO	Particle Swarm Optimisation
RRPC	max-Relevance and max-Redundancy based on the Pearson correlation
SBMLR	Sparse Multinomial Logistic Regression via Bayesian L_1 Regularisation
SIS	Sure Independence Screening
SMLR	Sparse Multinomial Logistic Regression
SU	Symmetrical Uncertainty
SVM	Support Vector Machine
SVM-RFE	Support Vector Machine Recursive Feature Selection

Chapter 1

Introduction

This section describes the motivation to the study of feature selection algorithms for large datasets. A brief background is given followed by the problem statement. Relevant studies are then explored in the literature review and finally the proposed study is defined.

1.1 Background

A classifier can only perform as well as the data it is given. In the ideal case, the features given to a model are non-redundant, have perfect synergy among each other and show a clear relationship or correlation to the class variable, be it linear or non-linear. In many large datasets such as biological gene expression microarrays [1], these perfect features are unknown or difficult to detect by hand even with domain knowledge. Instead, all possible features regardless of redundancy or correlation strength to the class are extracted to be used by the predictive model. Even though the perfect features lie among the irrelevant mess of the rest, the irrelevant features throw off the predictive accuracy of the classification model. This effect is only further exaggerated in cases where the amount of features are far greater than the amount of samples, denoted as p»n, and is commonly referred to as the curse of dimensionality [1]. Intuitively, a handful of strong features are easier to evaluate by predictive models due to their strong correlations, but the individual strength of a feature to the class, albeit non-redundant, does not always prove useful to a predictive model [2].

[1] The difficulty to generalise high-dimensional data with an increase of features

Synergy (the compliment which two or more features have to the class variable) is also important. Refer to Table 1.1 below. Features A and B alone do not provide a clear class variable distinction as the feature to class variable relation is not one-to-one. The joint combination of A and B, however, does give a one-to-one synergy to the class variable. Another advantage of better feature selection for a predictive model is that it reduces the amount of features required by a model which in turn reduces computational complexity and memory usage for applications with limited resources.

Table 1.1: Importance of synergy

A	B	Class
1	0	1
1	0	1
1	1	2
0	0	2

1.2 Literature review

1.2.1 Main approaches

There are three general models of feature selection algorithms: filter, wrapper and embedded [3]. Filter methods are unsupervised methods which search for high correlations between features and the class variable while keeping the redundancy among selected features low. The correlations use statistical measures to score the features into a ranking. Depending on the search strategy, features are either discarded or kept according to the statistical score. Filter methods are advantageous for the following reasons: they are computationally efficient and have no problem with high dimensional data; they remove both irrelevant and redundant features and they tolerate inconsistencies in training data [4]. The main disadvantage of filter methods being the need to specify the number of features k to keep beforehand. The imprecise selection of k can lead to inconsistent performance because it is unknown whether the current best subset of selected features has a greater information gain to the class variable with or without the new candidate feature x_c.

Wrapper methods are supervised and use search strategies where a predictive model is used to evaluate the model accuracy of the currently selected features. The predictive model should be kept simple to avoid computational burden. The selection or search strategy of features can include a stochastic approach, the use of heuristics, a greedy

best-first search or just a simple forward addition or backwards elimination of features. The main disadvantage of wrapper methods is the bias introduced by the predictive model used to evaluate the model accuracy.

Embedded methods select features which best fit a classification or regressive model while the model is being created. A good example of this is the use of regression algorithms which deliberately optimises a model by reducing its complexity, in this case, unwanted features. Again, embedded methods are also effected by model biases such as the penalisation parameter in regression.

1.2.2 Recent focus

Recently there has been a focus in feature selection for high dimensional data, specifically in the biomedical field due to large datasets comprised of gene expression data regarding cancers. The review study for high dimensional data proposed by [5] shows the broad solutions available for feature selection with high dimensional datasets for all types of feature selection algorithms, even hybrid methods which incorporate the advantages of multiple types of feature selection techniques. The paper also reveals the constant improvements and novel algorithms from researchers which overcome challenges introduced by the work of others.

The paper proposed by [6] mentions the recent advances of feature selection. The work classified the filter and wrapper feature selection models into two different types of frameworks for, namely correlation and search-based feature selection. The search-based feature selection framework is a supervised framework and generates subsets of selected features which are evaluated and compared to one another with the goal to generate the best subset. If the next generation's subset is better than the previous, the new subset will become the best. The best subset continues to evolve until a stopping criteria is met.

The correlation-based framework is an unsupervised framework which considers inter-feature correlations as well as feature-class correlations. The goal is to reduce redundancy and eradicate weak relevance features, while improving maintaining strong relevant features to the class. This framework is more attractive over the search-based framework due to its effectiveness and efficiency. When working with large datasets, the filter methods pose the most attractive solutions due to their lack of predictive model bias, their simplicity and their fast computation speeds for large datasets [6].

Another framework worth mentioning is predictive model optimisation. Embedded feature selection falls under this framework. Embedded methods automatically remove unwanted features whilst optimising a predictive classifier during training. Embedded strategies

are useful as no search strategy needs to be employed to find the most relevant features.

Feature selection models can also be used in unison to create hybrid methods. The fast computation of filter methods can make for a good pre-processor to more complex feature selection methods. Different search strategies and evaluation measures from each method can also be combined or used interchangeably. Figure 1.1 below shows a summary of all three frameworks.

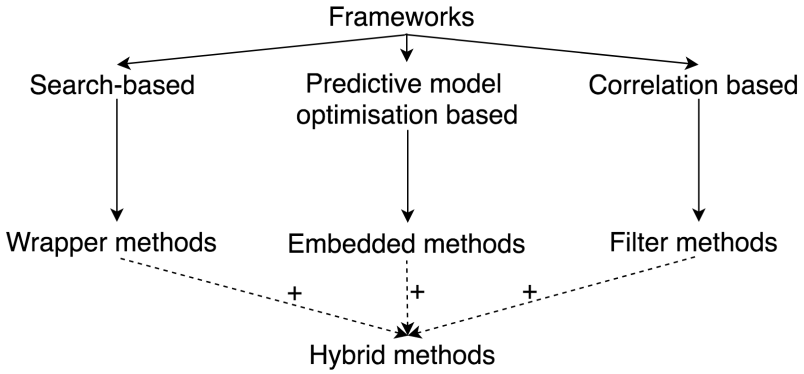


Figure 1.1: Feature selection frameworks

Besides the feature selection models, the search strategy used by filter and wrapper methods also plays an important role. The search strategies comprises of complete, sequential and random models. The complete search strategy guarantees an optimal solution based on the evaluation criteria due to the evaluation of each feature. The sequential strategy either removes or adds a feature to the best subset depending on the evaluation criteria. Lastly, the random search strategy introduces randomness when selecting the next subset of features.

1.2.3 Exploring the common methods

The most common methods of the correlation-based framework includes mRMR (min-Redundancy max-relevance), CFS (Correlation based Feature selection), FCBF (Fast Correlation Based Filter) and Mitra's [7]. The FCBF method is among the fastest and was first introduced in [8] and tested on gene expression datasets. The FCBF algorithm is ideal as it returns a subset of best features without selecting a pre-defined amount, but it can easily be modified to accommodate the need to select a pre-defined amount of features. FCBF uses Symmetrical Uncertainty (SU) as a normalised MI (Mutual Information) measure to select predominant features. If the SU of a feature to the class variable is above a pre-defined threshold then the feature is accepted as possibly predominant. The feature is only accepted as predominant if all of its redundant peers are

removed. A comparison between FCBF and three other common filter methods showed FCBF to select fewer features and perform similarly or better regarding classification accuracy. The work in [9] revised FCBF with a different search strategy. Their extension to the algorithm provides greater stability and performs better than FCBF only when a specific number of features are requested.

Even though FCBF is among the fastest filter methods, mRMR has gained more interest in feature selection. mRMR was first introduced in [10, 11] and applied to gene expression data with feature counts in the thousands. The mRMR framework is based on MI to obtain relevance and redundancy measures between feature-feature and feature-class relationships. The work in [12] improved the original mRMR algorithm by introducing a different associative measure to replace MI, namely distance correlation ($dCorr$) which proved to outperform the MI measure. The change in associative measure for mRMR is not uncommon and has also been proposed in [13] which used the Pearson Correlation (PC) coefficient as an associative measure and introduced a semi-supervised technique where not all samples are used to calculate feature-class dependencies with the aim to improve computation speed and maintain performance. The worked proposed in [14, 15] addresses the problem with mRMR which is unable to distinguish between relevant and irrelevant redundancy. The basic mRMR optimisation criteria will punish features which show high relevance to one another even though they might contribute different information towards the class variable. This is addressed using Kernel Canonical Correlation Analysis (KCCA) which maps the features into a higher dimensional space to determine nonlinear correlations, also known as the "kernel trick". The work in [14] proved that the KCCA transformation of features can avoid irrelevant redundancy when using mRMR. Their proposed method is called KCCAmRMR.

Another method proposed by [15] approaches the problem from a different perspective. The use of Gram-Schmidt Orthogonolisation (GSO) between features creates a new orthogonolised variable to avoid irrelevant redundancies. The orthogonolised variable is then compared the class variable using a fairly new statistical measure Maximal Information Coefficient (MIC). Their proposed method is called Orthogonolised Maximal Information Coefficient Feature Selection (OMICFS).

1.2.4 Introducing hybrids

Many improvements or alterations to the mRMR algorithm have been explored to find suitable or better performing algorithms using different approaches. One such example of a suitable replacement is the algorithm proposed in [16] which proved to be a valid and feasible new approach approach to mRMR where fuzzy entropy is used to calculate

feature-feature and feature-class dependencies. The mRMR algorithm has also been hybridised to include techniques from wrapper feature selection algorithms to deal with large gene expression datasets.

The paper presented in [17] proposed a two-stage algorithm which combines mRMR with a Genetic algorithm (GA). The mRMR algorithm is simply used as a pre-processing filter to remove noisy and redundant features. Highly discriminant features are then selected by a genetic algorithm which uses Leave One Out Cross Validation (LOOCV) predictive model accuracies as a fitness function. The GA contains a number of different subsets of selected features (chromosomes). The population of chromosomes are initially randomly created. The chromosomes with the highest fitness are kept and a new better population is created through mutation and crossover between the best chromosomes. The proposed method is named mRMR-GA and proved to be a combination which outperforms mRMR and GA when used individually. The paper in [18] did a similar hybridisation to mRMR for gene selection. The addition of another pre-processing filter is added, namely ReliefF (an effective filter for feature subset selection) to obtain a candidate gene set, thereafter mRMR is used to filter out redundancy and then a GA is used in the same manner to obtain the best subset of genes. The name of the method goes by R-m-GA.

The paper presented in [19] takes it one step further and compares three different metaheuristic approaches to aid mRMR, namely Particle Swarm optimisation (PSO), Cuckoo Search (CS) and Artificial Bee Colony (ABC). The performance for each is evaluated using k-nearest-Neighbour (k-NN) and Support Vector Machine (SVM) classifiers. Once again, the mRMR method is just used as a pre-processing filter to reduce noise and redundancy. The three different metaheuristic algorithms are then used to obtain the best subset of features and compared to one another. The mRMR filter makes metaheuristic approaches much more efficient and the combination of mRMR-CS showed the most promising results.

The use of the ABC algorithm was also used in [20] alongside mRMR. The ABC algorithm suffers in computational speed with high dimensional gene data therefore mRMR is an appropriate filter pre-processor. The mRMR filter once more reduces the amount of features by removing noisy and redundant features. An SVM is used to to provide a classification accuracy as a fitness value for the gene selection phase of the ABC algorithm. when compared to mRMR-GA or mRMR-PSO, the ABC algorithm performed the best when using similar initialisation parameters on gene expression data.

1.2.5 Metaheuristic optimisation

Metaheuristic algorithms and GAs have also been individually explored in feature selection application. The work presented in [21] proposed hybrid feature selection using a correlation measure in association with PSO for gene expression data. CFS based on the Pearson Product Moment Correlation (PPMC) measure is used as the fitness evaluator for a PSO algorithm. When using PSO as a feature selection algorithm for large datasets, the PSO algorithm has the possibility of falling in a local minima. This problem is addressed in [22] where a PSO algorithm was developed to reset the personal and global best solutions during the PSO search strategy, denoted as PSO with local searching and a global best particle resetting mechanism (PSO-LSRG). For the local search, each particle has the chance to update its personal best fitness during the PSO optimisation. The local search involves randomly selecting/deselecting features within the particle for numerous attempts to improve its personal best fitness. The global best particle is also reset when no new global best solution occurs for k iterations or generations. PSO-LSRG outperforms traditional PSO and shows promising results when selecting appropriate feature for gene expression datasets.

PSO has also gained attention in [23]. A Fuzzy Rule based Binary PSO (FRBPSO) algorithm is proposed where a Fuzzy Inference System (FIS) is used to update particle positions. In most PSO feature selection algorithm, the real value numbers within a particle refers to the probability of that feature being included. If the probability is above a predefined value, then the feature is accepted. The same is applicable for BPSO, except that the particles always take on discrete values. For FRBPSO, the fitness value per particle is calculated using LOOCV on a 1-NN predictive model. The particle position updates, however, is controlled via a FIS which has particle velocities as an input. The FRBPSO algorithm was tested on gene expression datasets and proves to outperform other Binary PSO (BPSO) methods. Another interesting PSO-based feature selection method is proposed in [24]. Their work improves the performance for PSO in multi-label feature selection by using multiple objectives in the fitness evaluation of PSO. The proposed method is named Multi-objective PSO Feature Selection (MPSOFS). The fitness of a particle is determined not only by the classification error rate defined by Hamming Loss (Hloss), it is also defined by the amount of features selected by the particle. Some particles will show similar Hloss values, but the particle with the least amount of features takes preference. Besides the dual fitness function, the explorative behaviour of the swarm is improved through a mutation of particles throughout optimisation. Furthermore, a local learning strategy based on differential learning is employed to explore areas of sparse solutions within the PSO search space. The MPSOFS algorithm is compared to PSO

without the mutations and local search strategy and proved to be beneficial in both classification accuracy and convergence time.

1.2.6 Feature selection stability

Depending on application, stability in a feature selection algorithm might also be required. In this context stability can be explained as follows: when studying features which are crucial for explaining behaviours or mechanisms to a system, the subset of selected features must show low variance when variations are introduced to the training data. If stability is of importance, ensemble method feature selection is suggested [6] which uses a split and aggregate approach. An example of an ensemble feature selection approach is proposed in [25] which uses linear Support Vector Machine Recursive Feature Elimination (SVM-RFE) combined with bootstrap sub-sampling. RFE is applied to each bootstrap sample to obtain a diversity in ranked features. The different rankings per bootstrap sample are then aggregated comparing Complete Linear aggregation (CLA) and Complete Weighted linear Aggregation (CWA) to the baseline. The ensemble method was testing on gene expression data with large feature sizes with few samples. Results show the tested ensemble method to outperform traditional linear SVM-RFE.

1.2.7 SVM feature selection and hybridisation

The use of SVM's has been used widely in feature selection applications. The work in [26] compares their own Backward Feature Elimination (BFE) SVM ranking algorithms to other well known SVM feature ranking algorithms, including linear and non-linear SVM-RFE, L_0-norm approximated SVM and L_1-norm SVM variants. The performance for each method was tested on gene expression datasets and compared to one another. The proposed BFE-SVM methods simply trains the SVM on all training data. Thereafter, subsequent features are removed and ranked according to a loss function. The worst 10 % of ranked features are then removed and the SVM is retrained. The loss functions used includes the standard 0-1 loss function and a balanced loss function. Further modifications were made to the BFE algorithm to includes a holdout strategy and is denoted HO-BFE. The strategy involves splitting the training data into a training and validation set. The loss functions are then computed from the validation set only. The HO-BFE strategies proposed outperforms other feature ranking methods and is also flexible with regards to using different kernel functions. The work in [27] also seeked to improve SVM-RFE by introducing a new method called multiple SVM-RFE (MSVM-RFE). The approach involves a similar technique to bootstrapping. Multiple linear SVMs are trained on smaller

sub-samples of the dataset. For each trained SVM, the weight vectors are normalised and a ranking score, c_i, is computed for each feature. The ranking score is computed by calculating the column mean of each weight vector and then dividing by their respective standard deviations. The feature which shows the lowest score is then removed and the process is repeated until all features are ranked. The MSVM-RFE algorithm was tested on gene expression datasets and showed a minor improvement of traditional SVM-RFE. Similarly, the work in [28] also focuses on SVM-RFE for gene expression datasets. The method employs Fuzzy C-means (FCM) clustering and is named FCM-SVM-RFE. FCM is used to cluster the dataset into small functional groups. A linear SVM is then modelled for each functional group and the features are ranked according to their respective squared weights. The higher ranking genes in each group are classified as the surviving features which are then combined to form the new dataset. This process is repeated until the new dataset consists of a feature count less or equal to that of a pre-defined value. Finally, the entire new dataset undergoes SVM-RFE for a final ranking. The FCM allows lower ranked features in each cluster to be removed before the final SVM-RFE selects the most informative features. FCM-SVM-RFE is more balanced when selecting positive and negative-related features than SVM-RFE and proved to be more accurate when predicting new samples in gene expression datasets. The traditional SVM-RFE algorithm simply uses the weight vector of the trained SVM to rank features.

Some studies using SVM-RFE prefer using pre-processing filters before RFE, and others simply use SVM-RFE as a comparison to new work as in [29]. Their novel perturbation feature selection algorithm is comparable to that of SVM-RFE performance. The motivation for feature perturbation is that the least important feature will have the least influence to the class variable when subjected to noise. Initially, a SVM classifier is trained on the entire training set and its 10-fold cross validated accuracy is stored. For each feature within the training set, uniform noise is added to all samples and the new accuracy to the training set is computed. The difference in accuracy before and after the feature is permuted determines the ranking score. The feature with the lowest ranking score is then removed and the process is repeated until no more features remain. The perturbation method was compared to traditional linear SVM-RFE. Diverse noise levels were applied to the new method which resulted in an improvement over the linear SVM-RFE method, however, SVM-RFE performs better when the amount of features selected are less than 14. As for the addition of pre-processors, the work in [30] modifies SVM-RFE to incorporate the mRMR filter metrics. When the SVM is trained, the weights for each feature are calculated. The ranking score is then computed as a combination of the absolute weight values per feature and the ratio of relevance to redundancy of the feature obtained by the mRMR metrics. If one would like either the SVM weight metric

or mRMR metric to be more dominant, a trade-off parameter can also be set. The feature showing the lowest ranking score is then eliminated and the process repeated. The new method outperformed traditional mRMR and SVM-RFE on most datasets with classification accuracy, and was consistently able to select fewer features. The disadvantage of the new method, however, is its increased computation time. A different hybrid approach is proposed in [31]. The FCBF filter method is modified to create groups of predominant features with their respective complimentary features to the class variable. Predominant features are obtained through the normalised MI metric SU. Candidate features are added to each predominant feature as a complimentary feature if the Conditional MI (CMI) between then predominant feature and candidate feature with respect to the class label is greater than zero. These groups of predominant and complimentary features are then aggregated into a single feature subset to be ranked using SVM-RFE with a polynomial kernel. The hybrid method showed improved performance when compared to four other state-of-the-art feature selection methods.

Besides the implementation of hybrid methods using SVM-RFE, some studies try to directly optimise SVMs to incorporate embedded feature selection as in [32]. The goal is to improve classification accuracy through direct penalisation of the kernel function. The Gaussian kernel is modified so that its shape is determined by numerous width updates per feature. The importance of a feature is directly linked to the contribution is has to the Gaussian kernel function. Besides the penalised kernel, a L_0-norm approximation penalisation is added for each feature in the dual representation of SVM. The method is known as Kernel Penalised SVM (KP-SVM).

The different combinations of feature selection techniques using SVMs is astounding. The work in [33] compares and formulates different SVM based approaches to feature selection, including variations of different L-norms, different kernels and also direct penalisation of features even in the transformed kernel space. Each method is compared to one another in experiments with various artificial and real-world datasets.

For further insight, Figure 1.2 below illustrates the trade-offs between different L-norms.

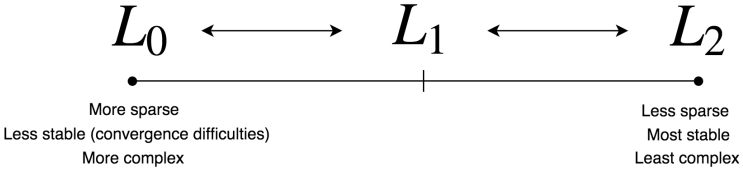


Figure 1.2: Trade-offs between norms

1.2.8 Sparsity and regression

For large datasets, a sparse feature selection solution is well suited to select discriminate features and reduce the amount of selected features. The L_1-norm penalty is a good implementation to incorporate sparsity. In SVMs, however, the L_2-norm provides better classification accuracy, but variations of L_1 and L_0 approximation norms select fewer features [33]. Sparse feature selection solutions are extensively used in embedded regressive feature selection models. In [34], the mRMR algorithm was used to filter irrelevant features from gene expression datasets. After the mRMR filter, an improved Lasso regressive model is used to remove redundant features. Before Lasso regression is applied, features are grouped into subsets. Each subset is merged with the current best subset of features (initially empty). The Lasso method is then used on the merged subset to determine the best features which are added to the list of best of features. This process is repeated for each subset of features to determine the final list of best features. The algorithm is knows as Feature Selection based on Mutual Information and Lasso (FSMIL). The use of grouped subsets was tested on six binary class gene expression datasets and showed improved results over the traditional Lasso method.

Modified regressive models can also be applied to feature selection. The work in [35] proposed a Linear Regression (LR) feature selection algorithm for microarray datasets. As a pre-processor, two different subtypes of the training dataset are compared to one another. The mean and differences between the features in the subtypes are compared and redundant features are removed. Thereafter, LR is applied on the dataset where each feature is considered as an output variable with all other features as inputs, therefore each value within a feature vector has an assigned weight to be optimised by the regressive model. Regression is computed on all features to produce a matrix of feature weights. The divergences of weight values are then calculated and sorted in descending order, allowing the top n features to be selected. When compared to other filter methods, the proposed LR model selected fewer features with better performance.

When working with datasets with nominal classes and high feature counts, LR is usually adopted as a sparse regressive model. The paper in [36] introduced a novel ensemble logistic feature selection technique for high dimensional gene expression microarray datasets. The proposed method relies on LR models built on small subsets of features sampled randomly from the full dataset. Initially, all features are ranked through the t-test. These initial ranks of features are refined and improved in an iterative fashion through the performance of the regularised LR models. Iterations stop until classification performance converges. The L_2-norm LR model is used instead of the L_1-norm sparse model. The reasoning behind this choice is that features are limited in subsets for each model. From

the t-test rankings, a subset of features are selected where 80 % is used as training data and the remainder for validation. The training data for the selected subset is evaluated through the L_2-norm LR model and its performance evaluated on the validation set. The fitness of the model is then compared to previously trained models and the rankings are updated accordingly. This process continues until no significant improvement occurs. Results show improvement over LR using Elastic Net regularisation.

In [37], Sparse Multinomial LR (SMLR) is proposed which is multiclass compatible, produces sparse solutions, is scalable in both number of samples and features, and was the first work to perform multinomial regression which enforces sparsity. The sparsity is introduced with L_1-norm Laplacian prior as with Lasso regression, except that a reasonable regularisation parameter is obtained through simple line search. SMLR was compared to L_2-norm Ridge Multinomial LR (RMLR) and showed better performance with better sparsity. In [38], SMLR was revisited by introducing Bayesian regularisation using the L_1-norm Laplacian prior and was given the name Sparse Multinomial LR via L_1 Bayesian regularisation (SBMLR). When compared to SMLR on gene expression datasets, both models show equal performance with regards to classification results. SBMLR does however produce slightly higher degrees of sparsity and dominates with regards to computation time and is up to 100 times faster than SMLR. [39] noticed a growing interest in L_1-norm Lasso regression applied to gene expression data. An improvement over Lasso was introduced with an L_q-norm LR model where $0.5 < q < 1$. The selectable q parameter is set to 0.5 which lead to more sparse solutions than traditional L_1-norm Lasso and Elastic Net regressive models. The motivation for $L_{0.5}$-norm was derived from the study in [39] where different q values were tested and compared to one another. Values for $q \geq 0.5$ has good convergence, provides more sparse solutions than L_1-norm and is also simpler than L_0-norm approximation.

1.3 Research question

With datasets becoming larger and more complex, are feature selection techniques becoming a necessity to improve predictive model accuracies and save computational resources?

1.4 Method

A total of six algorithms are explored:

- mRMR

- FCBF#
- OMICFS
- FRBPSO
- SVM-RFE
- SBMLR.

Each algorithm is decomposed into its fundamentals and reformulated to grasp the understanding as to why they work well for large feature selection applications, specifically gene expression datasets. The algorithms will also incorporate their own set of optimisation parameters, hybridisation and pre-processing where applicable.

The mRMR, FCBF and OMICFS algorithms are filter techniques which will be explored. The mRMR method has been used extensively alone and in conjunction with many other feature selection techniques. The individual use of mRMR is studied to view its performance. A supervised addition to mRMR is also explored to form a hybrid feature selection algorithm. FCBF will also be studied. It has not received as much attention as mRMR as a pre-processor, and therefore its performance is questioned. As with mRMR, a supervised hybrid method is also explored. OMICFS is explored as it tackles the shortcomings of mRMR and uses a recently new statistical measure for feature selection. Its effectiveness will be evaluated and a supervised hybrid approach is also implemented.

For the wrapper and embedded techniques, FRBPSO, SVM-RFE and SBMLR are the algorithms of interest. FRBPSO will be evaluated using numerous PSO improvement techniques to determine its effectiveness and stability. SVM-RFE will include a fast converging algorithm applicable for large datasets and three different kernels are explored. Finally, SBMLR is the most promising fast and stable multiclass regressive model to be evaluated and is also the only algorithm which does not incorporate any parameter optimisation in this study.

The tested performance on numerous large datasets of each algorithm, based on classification accuracy and feature removal capability, is explored and compared to one another and scenarios are derived where each algorithm will be more applicable. The performance of the algorithms is subject to datasets assumed to have no prior domain knowledge. This approach reduces the dependence on domain experts and focuses on the classification improvement of datasets without prior domain knowledge. This study will therefore test the ability of the feature selection algorithms to obtain a rich feature set irrespective of how the features were obtained. Lastly, a case study is tested for a dataset containing plant foliage information to assist the findings. The dataset is not excessively large, but does adhere to $p >> n$.

1.5 Subsequent structure

The remainder of this dissertation contains the following chapters:

2 Formulation of main feature selection algorithms

This chapter describes in detail a total of six different feature selection algorithms appropriate for datasets with $p >> n$. The first three algorithms are filter methods which are explained separately from the wrapper and embedded methods to follow.

3 Numerical experiments

This chapter deals with the simulations of each feature selection algorithm explained in Chapter 2. A total of 12 medically related datasets are used for testing. All datasets adhere to $p >> n$. After each simulation, a brief discussion on the performance of the algorithm is presented. Thereafter, the performance of all algorithm are compared to one another, and scenarios best suited for each algorithm discussed.

4 Case study

This chapter applies the six reviewed feature selection algorithms on potato plant foliage data where $p >> n$. An explanation to obtaining the data is given and thereafter the results from each feature selection algorithm is recorded and used to reinforce the findings in Chapter 3.

5 Conclusion

This chapter concludes the findings in Chapters 3 and 4 for each algorithm. A recommendation per algorithm is provided regarding classification accuracy, feature removal capability and computation time.

Chapter 2

Formulation of main feature selection algorithms

This chapter describes in detail a total of six different feature selection algorithms appropriate for datasets with $p >> n$. The first three algorithms are filter methods which are explained separately from the wrapper and embedded methods to follow. The shortcomings and possible improvements for each algorithm are also discussed.

Part I - Relevance and redundancy filter methods

When selecting suitable features in a dataset, the goal is to eliminate irrelevant features which do not contribute to classification accuracy of the labelled data. Features which are redundant also need to be removed in order to reduce the dimensions of the dataset. Three filter methods are explored, namely mRMR, FCBF and OMICFS. These methods are among the old and new which uses the trade-offs between relevance and redundancy in order to select the best subset of features. As discussed in 1.2, the main disadvantage of the filter methods, namely the lack of synergy, is later overcome by adding the advantage of supervision from wrapper methods. The bias of wrapper methods is reduced by using a 1-NN classifier as the predictive model. The use of 1-NN optimises on only 1-nearest point, making the model sensitive to noise in the data, but it comes at the necessary advantage of a low bias [40]. This attractive property of using a 1-NN classifier motivates the choice for its use to determine dataset classification accuracy for all algorithms in

this study.

2.1 Minimum Redundancy Maximum Relevance

An intuitive manner to approach a feature selection problem is to determine the relevance and redundancy of each feature using an associative measure such as MI. In this context, an associative measure is a statistical measure which can be used to determine feature-class and feature-feature dependencies. The paper in [10] defines the relevance of a subset of features as

$$Rel = \frac{1}{|S|} \sum_{i \in S} I(x_i; Y) \tag{2.1.1}$$

where S is a subset of features, $|\cdot|$ denotes cardinality, x_i is feature i in subset S, Y is the class label vector and $I(\cdot)$ is the associative measure for MI[1]. Refer to (2.1.1), the mean value of all MI values for all features in subset S are calculated. The subset with the maximum relevance is the best subset of features, but the features could still have rich redundancy. The redundancy of a set of features is given as

$$Red = \frac{1}{|S|^2} \sum_{i,j \in S} I(x_i; x_j) \tag{2.1.2}$$

If two features are highly dependent (redundant), removing only one redundant feature would not change the class-discriminative power by much, if at all. Combining (2.1.1) and (2.1.2) is called mRMR. An optimal subset of features can be obtained by combining the relevance and redundancy, defined by operator Φ, given as

$$\Phi = Rel - Red \tag{2.1.3}$$

An incremental search method is used to find the best subset of features of $\Phi(\cdot)$. Suppose dataset X contains n features, then the goal is to traverse through all n features in dataset X and append the best feature which maximises Φ to subset S, therefore $S = [S\ x_{best}]$ after each iteration. The m^{th} best feature is selected from the dataset $\{X = X - S_{m-1}\}$,

[1] The mutual dependence between two variable

which excludes the previous best feature already appended to S. The iterative selection process for selecting the m^{th} best feature is given as

$$\max_{x_i \in (X - S_{m-1})} \left(I(x_i; Y) - \frac{1}{m-1} \sum_{x_j \in S_{m-1}} I(x_i; x_j) \right) \tag{2.1.4}$$

Recall that $I(\cdot)$ is the MI association measure which measures the similarity or mutual dependency between two variables based on their joint probabilities $p(x_1, x_2)$, as well as their marginal probabilities $p(x_1)$ and $p(x_2)$. MI can be directly linked to a measure of entropy (amount of information) which two variables quantify. Shannon's entropy is a common measure of entropy given as

$$H(A) = -\sum_{a \in A} p(a) log_2 p(a), \tag{2.1.5}$$

where $p(a)$ is the probability of instance a occurring in set A. Entropy can be used to define MI. Figure 2.1 below illustrates the relationship of MI between two variables A and B using their entropy values. The discrete MI of A and B is denoted by the purple area $I(A; B)$. Note that there are multiple ways to define $I(A; B)$ so that

$$\begin{aligned} I(A; B) &= H(A) - H(A|B) && (2.1.6)\\ &= H(B) - H(B|A) && (2.1.7)\\ &= H(A) + H(B) - H(A, B) && (2.1.8)\\ &= H(A, B) - H(A|B) - H(B|A), && (2.1.9) \end{aligned}$$

where $H(A|B)$ and $H(B|A)$ are conditional entropies given by

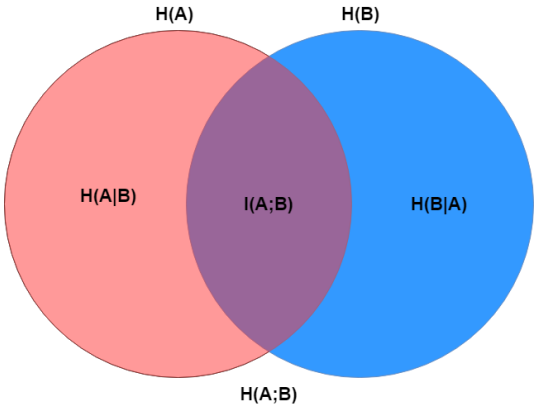


Figure 2.1: Mutual Information between variables A and B

$$H(A|B) = \sum_{b \in B} p(b) H(A|B=b) \tag{2.1.10}$$

$$= -\sum_{b \in B} p(b) \sum_{a \in A} p(a|b) log_2 p(a|b) \tag{2.1.11}$$

$$\text{Recall } p(a|b) = \frac{p(a,b)}{p(b)} \tag{2.1.12}$$

$$= -\sum_{b \in B} \sum_{a \in A} p(a,b) log_2 p(a|b) \tag{2.1.13}$$

$$= -\sum_{a \in A, b \in B} p(a,b) log_2 \frac{p(a,b)}{p(b)} \tag{2.1.14}$$

$$= \sum_{a \in A, b \in B} p(a,b) log_2 \frac{p(b)}{p(a,b)} \tag{2.1.15}$$

Note that if $I(A,B) = 0$, then the variables A and B are completely independent, but if $I(A,B) = 1$ then variables A and B are said to be completely dependent. Substituting (2.1.5) and (2.1.15) into (2.1.6) gives the MI of two variables A and B as

$$MI(A,B) = \sum_{a \in A} \sum_{b \in B} log \frac{p(a,b)}{p(a)p(b)} p(a,b) \tag{2.1.16}$$

The continuous case for MI is not suitable for feature selection as the features are not continuous functions, therefore the features need to be discretised ($X -> \hat{X}$) by grouping the values into spaced intervals (the size of the intervals are not necessarily equi-spaced, but are determined by the discretisation algorithm used). The discrete version therefore takes the form

$$\widehat{MI}(\hat{A},\hat{B}) = \sum_{\hat{a}\in\hat{A}}\sum_{\hat{b}\in\hat{B}} log\frac{p(\hat{a},\hat{b})}{p(\hat{a})p(\hat{b})}p(\hat{a},\hat{b}) \tag{2.1.17}$$

2.1.1 Alternative associative measures

MI is the original association measure for mRMR, but there are several other methods which can be used as an alternative. The work presented in [12] compares MI to ordinary correlation, Fisher-Correlation (FC), Distance Covariance ($dCov$) and Distance Correlation ($dCorr$) methods as alternative associative measures. Using $dCorr$ proved to show better results than MI and will be explored in experiments to determine its potential as a competing measure[2].

Suppose there are two vectors X and Y with n observations. In order to calculate $dCorr(X,Y)$ [41], the sample distance covariance of samples X and Y need to be calculated as follows

$$dCov(X,Y) = \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{j,k}B_{j,k}, \tag{2.1.18}$$

where $A_{j,k}$ and $B_{j,k}$ are doubly centered distance matrices of vectors X and Y respectively. $A_{j,k}$ and $B_{j,k}$ is denoted by

$$A_{j,k} = a_{j,k} - \bar{a}_j - \bar{a}_k + \bar{a} \tag{2.1.19}$$

$$B_{j,k} = b_{j,k} - \bar{b}_j - \bar{b}_k + \bar{b}, \tag{2.1.20}$$

where $a_{j,k}$ is a $n \times n$ distance matrix[3] of vector X, $\bar{a}_j$ is j^{th} row mean of $a_{j,k}$, $\bar{a}_k$ is k^{th} column mean $a_{j,k}$, and $\bar{a}$ is the grand mean of $a_{j,k}$. The same notation is followed for $B_{j,k}$. Next, the distance variance of X and Y are calculated using

$$dVar(X) = dCov(X,X) \tag{2.1.21}$$

$$= \frac{1}{n^2}\sum_{j,k} A_{j,k}^2 \tag{2.1.22}$$

[2] One measure will not necessarily always be better than another

[3] A square matrix containing pairwise distances between vector elements

Finally, the distance correlation between X and Y is calculated by dividing the distance covariance of X and Y by the product of their distance standard deviations as shown below.

$$dCorr(X,Y) = \frac{dCov(X,Y)}{\sqrt{dVar(X)dVar(Y)}} \tag{2.1.23}$$

Some attractive properties of dCorr [41] include

1. $0 \leq dCorr(X,Y) \leq 1$
2. If $dCorr(X,Y) = 0$ then X and Y are independent. Similarly, if $dCorr(X,Y) = 1$ then X and Y are almost surely similar.
3. $dCorr(X,Y)$ shows high statistical power[4] if X and Y have a non-linear relationship.
4. X and Y can be any arbitrary dimensions.
5. $dCorr$ has properties of a true dependence measure and is considered a good empirical measure of dependence.
6. $dCorr$ is easy to compute.

2.1.2 Implementation of mRMR

Suppose dataset X consists of i rows and j columns. Each row, X_i, represents a data sample with j features. Let Y represent the class label vector for each sample x_i. Recall that $I(\cdot)$ represents the associative measure. Algorithm 1 below shows the pseudo code for mRMR.

[4] In [42], statistical power between variables X and Y refers to "the probability that a statistic, when evaluated on data exhibiting a true dependence between X and Y, will yield a value that is significantly different from that for data in which X and Y are independent"

Algorithm 1 mRMR

- Initialise:

Size of best feature subset $MaxIte$
$F_s = \emptyset$ - Subset of best features
$F_a = X(:,:)$ - Subset of remaining features

1: Precompute the relevance of each feature $Rel(i) = I(x_i; Y)\ \forall_i$
2: Find the first best feature $F_{best} = \max\ Rel(i)$
3: **for** `k = 2,3,..,MaxIte` **do**
4: $\quad F_s = [F_s\ F_{best}]$
5: $\quad F_a = F_a - \{F_{best}\}$
6: $\quad$ **if** `isempty(Fa)` **then**
7: $\quad\quad$ `break`
8: $\quad$ **end**
9: $\quad F_{best} = \max_{x_i \in F_a} \left(I(x_i; Y) - \frac{1}{length(F_s)} \sum_{x_j \in F_s} I(x_i; x_j) \right)$
10: **end**

The mRMR algorithm selects the first best feature which shows the highest $Rel(\cdot)$ value in lines 1 - 2. The remaining best features are selected in an iterative fashion in lines 3 - 10. The previously selected feature is appended to the best subset of features F_s and removed from the remaining features in F_a. The next best feature is then determined in line 9. The algorithm terminates when the target feature size is reached, or if no more features in F_a remain.

2.1.3 A possible improvement using semi-mRMR

The mRMR algorithm is a filter method for feature selection which shows a time complexity [13] of

$$O(n(m+1)^2), \tag{2.1.24}$$

where n is the amount of samples, and m is the amount of features. The expression $(m+1)$ stems from the fact that the class label vector $\bar{Y}$ acts as an additional feature during computation. Taking a closer look at Algorithm 1, the pre-computation of $I(\bar{X}_i; \bar{Y})$ removes $\bar{Y}$ as an additional feature, reducing the time complexity of the mRMR algorithm to

$$O(nm^2) \tag{2.1.25}$$

A paper presented by [13] suggests a revision of mRMR into a semi-supervised filter algorithm for an improved time complexity with similar or better performance. The method was given the name 'max-relevance min-redundancy criterion based on the Pearson Correlation (RRPC) coefficient'. Instead of using all labelled data of size n, the datasets were split into labelled data X_L of size l, and unlabelled data X_U of size u, where $l + u = n$. The relevance criteria from (2.1.1) only needs to be computed l times. If pre-computation of $I(x_i; Y_L)$ is not considered, then the new time complexity of RRPC is given by

$$O(l(m+1)^2 + um^2) \tag{2.1.26}$$

Besides the reduced time complexity, the results of their RRPC algorithm boasts comparable and better performance than mRMR, even though the associative measures are different. At the time of comparison, mRMR used MI as its original associative measure [10]. RRPC makes use of PC as its associative measure, PC is a good measure for linear data, but lacks the ability to capture non-linear correlations which is common in real world data. The Pearson Correlation coefficient r between variable pair (X, Y) is given by

$$r = \frac{\sum_i (x_i - \bar{x_i})(y_i - \bar{y_i})}{\sqrt{\sum_i (x_i - \bar{x_i})^2}\sqrt{\sum_i (y_i - \bar{y_i})^2}}, \tag{2.1.27}$$

where $\bar{x_i}$ and $\bar{y_i}$ is the mean of X and Y respectively. PC is a symmetrical measure which lies in [-1, 1]. Taking the absolute of r changes the range of r to [0, 1], where 0 suggests X and Y are completely linearly independent and a value of 1 suggests the latter. As for the RRPC algorithm, it shows no difference in the mRMR algorithm except that the relevance of features are computed from less samples. RRPC is recreated by simply changing line 9 in Algorithm 1 to

$$F_{best} = \max_{x_i \in F_a} \left(I(x_i; Y_L) - \frac{1}{length(F_s)} \sum_{x_j \in F_s} I(x_i; x_j) \right), \tag{2.1.28}$$

where Y_L suggests that only the labelled data is used, and $I(\cdot)$ refers to the PC associative measure. To date, a more suitable name for RRPC would be 'semi-mRMR using PC as an associative measure'. The remark of better performance than mRMR cannot be

justified with their comparison, but the use of PC as the associative measure is where the claimed improvement over traditional MI as an associative measure lies.

2.1.4 Limitations of mRMR

The subtraction of redundancy terms in the objective function of mRMR in (2.1.3) is known to include irrelevant redundancy when selecting features [14, 15, 43]. Ideally, the irrelevant redundancy should not be taken into account as it is completely independent from the class variable. Figure 2.2 below illustrates the relation between three variables namely, the class variable C, a previously selected variable A, and the next candidate feature B. The class variable C in this context is a vector containing class labels for each instance in a dataset. The regions r_i denotes the different information regions.

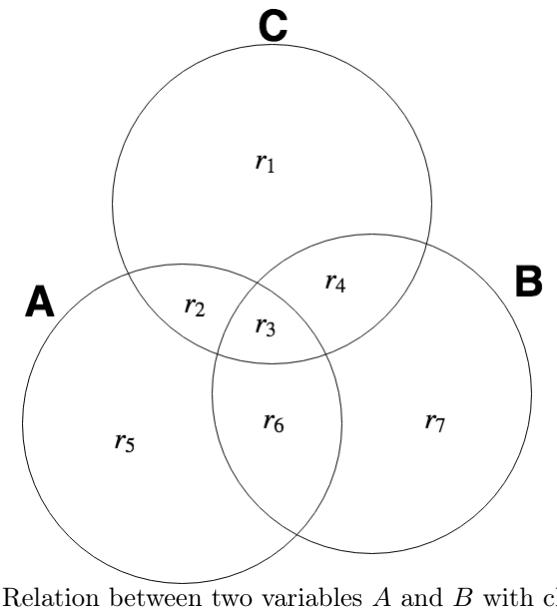


Figure 2.2: Relation between two variables A and B with class variable C

According to (2.1.3), the score of candidate feature x_2 with regards to feature x_1 is given by

$$\Phi = I(x_2, C) - I(x_2, x_1) \tag{2.1.29}$$
$$= (r_3 + r_4) - (r_3 + r_6) \tag{2.1.30}$$
$$= r_4 - r_6 \tag{2.1.31}$$

As mentioned in [15], the relevant and irrelevant redundancy is denoted by r_3 and r_6 respectively. The irrelevant redundancy, r_6, is completely independent from the class, and should not be considered in the mRMR score as in (2.1.29). Ideally, r_6 should be

removed, leaving only r_4 which carries only the information between candidate feature B and class C.

2.2 Orthogonal Relevance Feature Selection

The removal of features containing irrelevant redundancy has been thoroughly studied by [14, 43], of which the latest to date being [15], which presents a novel filter feature selection method called Orthogonal Maximal Information Coefficient Feature Selection (OMICFS). The algorithm is based on the Maximal Information Coefficient (MIC) and Gram-Schmidt Orthogonolisation (GSO) to deal with the aforementioned irrelevant redundancy problem. OMICFS shows promise as it makes use of a relatively new statistical measure, the MIC statistic, which is used as the associative measure instead of MI. GSO is a variation of Principle Component Analysis (PCA) and is used to obtain an orthogonolised variable between the candidate feature and all previously selected features in order to remove the redundancy between them. MIC is used on the variable obtained by GSO and the class variable in order to indirectly optimise (2.1.3). The score given by OMICFS between A and B in Figure 2.2 is given as

$$OMIC_{B,A} = MIC(GSO(B, A), C) \tag{2.2.1}$$
$$= (r_4 + r_7) - r_7 \tag{2.2.2}$$
$$= r_4, \tag{2.2.3}$$

where $GSO(B, A)$ is the orthogonolised variable between the candidate feature B and the previously selected feature A, and $MIC(GSO(B, A), C)$ represents the relevance measure between the orthogonal variable and the class variable C. This new score eliminates the irrelevant redundancy seen in the original mRMR strategy. To support the OMIC score visually, $GSO(B, A)$ is denoted by $r_4 + r_7$ in Figure 2.2. The GSO measure is the essential metric which eliminates the irrelevant redundancy in r_6. The MIC score determines the degree of relevance between a variable x_i and the the class variable C, which will be denoted as $MIC(x_i, C)$ for the remainder of this chapter. The value of $MIC(x_i, C)$ is given by

$$MIC(x_i, C) = \frac{MI(x_i, C)}{Z_{MIC}}, \tag{2.2.4}$$

where $Z_{MIC} = log_2(min(n_{x_i}, n_C))$ and $MI(x_i, C)$ is the binning schemed mutual information between x_i and C calculated using (2.1.17). The parameters n_{x_i} and n_C represents the amount of bins imposed on x_i and C respectively.

The binning scheme is an iterative process where the best number of bins are determined so that (i) $n_{x_i} n_C$ do not exceed a user defined value B and (ii) the value of $MIC(x_i, C)$ is maximised. The value of B is typically between $N^{0.55}$ and $N^{0.6}$ where N is the sample size. The MIC statistic is essentially MI calculated by selecting an appropriate grid search binning scheme and normalised by Z_{MIC}. As with MI, MIC also falls in range [0, 1]. It is interesting to note that MIC is identical to MI if two bins are assigned to either x_i or C, as $Z_{MIC} = 1$. MIC, being a relatively new statistic, received scrutiny in [42] which shows that MIC lacks mathematical reinforcement and has a very low statistical power in linear, parabolic, circular and checkerboard relationship data types with added noise. Comparing MIC to *dCorr*, the work in [42] mentioned *dCorr* to be more powerful than the MIC statistic. In addition, a MI estimator proposed by [44], which uses a k-NN approach to estimate MI based on the distance between k neighbour data points, was also analysed in [42] which found competing results to *dCorr* and an improvement over MIC with regards to statistical power. When using the k-NN estimator, the value of k is essentially a smoothing parameter. Although there is no guideline for selecting the appropriate value for k, smaller values of k provides high variance but low bias estimations, whereas high values of k provides the latter.

In conclusion to [42], MIC is computationally expensive due to the exhaustive search for an appropriate binning scheme, it shows a bias in estimation when increasing the number of features, and traditional MI is shown to be computationally faster and have a higher statistical power than its MIC counterpart. These findings motivate the need to explore associative measures *dCorr*, MIC, and MI to the feature selection algorithm OMICFS.

GSO is where the strength in OMICFS lies, and the estimators simply act as an associative measure between the output of GSO and the class label. For the purpose of naming convention, it seems fit to revise the name of OMICFS to Orthogonal Relevance Feature Selection (ORFS), where the term *Relevance* refers to the associative measure used to calculate the relevance between GSO and the class label.

2.2.1 Orthogonolisation

The ORFS score between candidate feature x_i and all previously selected features $S = \{x_1, x_2 ... x_m\}$ is given as

$$ORFS(x_i; S) = R(GSO(x_i, S)), \tag{2.2.5}$$

where R is the relevance associative measure and $GSO(x_i, S)$ is the orthogonalised variable between x_i and all the previously selected features within S. GSO allows a candidate feature vector to be represented as an orthogonal contribution vector of other feature vectors through projections of one vector onto another.

Assume the basis $F = \{\boldsymbol{f_1}, \boldsymbol{f_2}, ..., \boldsymbol{f_k}\}$, where F is not orthonormal and contains all feature vectors. GSO constructs an orthonormal basis for F to benefit from the elegant mathematics to calculate the projection of previously selected features $\boldsymbol{f_1}...\boldsymbol{f_i}$ onto candidate feature $\boldsymbol{f_c}$. An orthonormal basis contains basis vectors which are orthogonal to one another (dot product between the vectors are 0), and each vector is also normalised to a unit vector. For simplicity, a one dimensional subspace $F_1 = span(\boldsymbol{f_1})$ contains only one vector $\boldsymbol{f_1}$, which is orthogonal because it is the only member in the set therefore also making $\boldsymbol{f_1}$ the only basis vector, denoted as $\boldsymbol{u_1}$. To create the orthonormal basis, all that is needed is to normalise $\boldsymbol{u_1}$ to a unit vector $\boldsymbol{q_1} = \frac{\boldsymbol{u_1}}{||\boldsymbol{u_1}||}$. The vector $\boldsymbol{q_1}$ is said to be an orthonormal basis for subspace F_1.

This orthogonalisation process can be expanded to two dimensions with subspace $F_2 = span(\boldsymbol{f_1}, \boldsymbol{f_2})$. Because $\boldsymbol{f_1}$ is a linear combination of $\boldsymbol{u_1}$, subspace F_2 can be rewritten as $F_2 = span(\boldsymbol{q_1}, \boldsymbol{f_2})$. Currently, $\boldsymbol{f_2}$ cannot be represented as a linear combination of $\boldsymbol{q_1}$ or $\boldsymbol{f_1}$. The subspace F_2 needs to be an orthonormal basis, which requires $\boldsymbol{f_2}$ to be replaced by a new vector which is orthogonal to $\boldsymbol{f_1}$ and still be able to reconstruct $\boldsymbol{f_2}$. Refer to Figure 2.3 below. It can be seen that vector $\boldsymbol{f_2}$ can be reconstructed through the addition of some multiple of x to $\boldsymbol{u_2}$, therefore $\boldsymbol{u_2}$ can be calculated by subtracting x from $\boldsymbol{f_2}$.

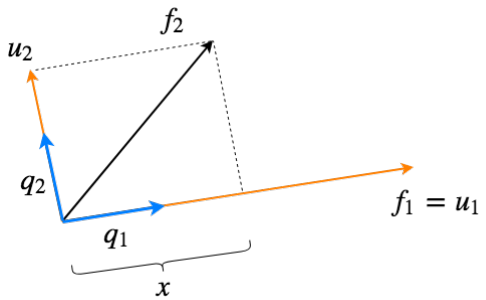


Figure 2.3: Vector diagram of feature f_1 and candidate feature f_2

By definition, x is the projection of $\boldsymbol{f_2}$ onto subspace F_1, therefore $\boldsymbol{u_2}$ given as

$$\boldsymbol{u_2} = \boldsymbol{f_2} - proj_{F_1}^{\boldsymbol{f_2}} \tag{2.2.6}$$

The orthonormal vector $\boldsymbol{q_2}$ is thus $\frac{\boldsymbol{u_2}}{||\boldsymbol{u_2}||}$. The benefit of having an orthonormal basis is the simplicity to calculate the projection of any vector $\boldsymbol{f}$ onto any subspace F_R by using the basis vectors of F_R as shown in (2.2.7) below.

$$proj_{F_R}^{\boldsymbol{f}} = \sum_i (\boldsymbol{f} \cdot \boldsymbol{q_i})\boldsymbol{q_i}, \tag{2.2.7}$$

where $\boldsymbol{q_i}$ are the basis vectors within subspace F_R. In this example, the projection of $\boldsymbol{f_2}$ onto subspace F_1 will therefore be

$$proj_{F_1}^{\boldsymbol{f_2}} = (\boldsymbol{f_2} \cdot \boldsymbol{q_1})\boldsymbol{q_1} \tag{2.2.8}$$

Notice that there is only one basis vector $\boldsymbol{u_1}$ as F_1 is only a one dimensional subspace. The orthonormal basis vector $\boldsymbol{q_2}$ in Figure 2.3 above is therefore

$$\boldsymbol{q_2} = \frac{\boldsymbol{u_2}}{||\boldsymbol{u_2}||}, \boldsymbol{u_2} = \boldsymbol{f_2} - (\boldsymbol{f_2} \cdot \boldsymbol{q_1})\boldsymbol{q_1}, \tag{2.2.9}$$

Subspace F_2 can now be rewritten as $F_2 = span(\boldsymbol{q_1}, \boldsymbol{q_2})$. The above process describes the Gram-Schmidt process for orthonormalising a subspace to calculate its basis vectors. The general term for calculating the basis vector for any subspace F_R is given as

$$\boldsymbol{q_k} = \frac{\boldsymbol{u_k}}{||\boldsymbol{u_k}||}, \boldsymbol{u_k} = \boldsymbol{f_k} - \sum_{i=1}^{k-1} (\boldsymbol{f_k} \cdot \boldsymbol{q_i})\boldsymbol{q_i}, \tag{2.2.10}$$

where $\boldsymbol{u_k}$ is the orthogonal basis vector and $\boldsymbol{q_k}$ is the orthonormal basis vector.

ORFS uses GSO to determine the basis vector $\boldsymbol{q_k}$ according to all previously selected features. Refer to Figure 2.3 once again. On one extreme, if both vectors $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ lie on top of each other then $\boldsymbol{q_2}$ would be a vector of zeros indicating that the vector $\boldsymbol{f_2}$ is completely relevant to $\boldsymbol{f_1}$. A good way to visualise this occurrence is to think of features

$\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ in a Venn diagram where $\boldsymbol{f_1}$ encloses $\boldsymbol{f_2}$ completely as shown in Figure 2.4a below. As for the other extreme, if the vectors are completely orthonormal to one another, $\boldsymbol{q_2}$ would simply be $\boldsymbol{f_2}$ normalised, which can be represented in a Venn diagram as $\boldsymbol{f_1}$ being mutually exclusive from $\boldsymbol{f_2}$ as shown in Figure 2.4b below. Values inbetween these extremes justifies the reasoning as to why $GSO(B, A)$ is denoted by $r_4 + r_7$ in Figure 2.2.

(a) Complete relevance **(b)** Mutually exclusive

Figure 2.4: Graphical illustration of GSO extremes

2.2.2 ORFS implementation

In ORFS, the candidate feature x_i with the greatest ORFS score is the most promising. Let $S = \{s_1, s_2..., s_m\}$ denote the best subset of features already selected from dataset X. When any promising feature s_i is selected to form part of the best subset of features, it is appended to S and removed from X. Initially, the best feature to start with is the feature which shows the highest relevance to the class labels calculated by the associative measure. The best feature is selected using

$$s_1 = \arg\max_{x_i \in X} R(x_i, C), \tag{2.2.11}$$

where s_1 is the first promising feature, C is the class label and R is the chosen associative measure. The next promising feature, s_m, is selected by maximising (2.2.5):

$$s_m = \arg\max_{x_i \in X-S} R(GSO(x_i, S), C), \tag{2.2.12}$$

The iterative process of finding the next promising feature is terminated until a total of T features are selected, defined by the user. In [15], the theory of Sure Independence Screening (SIS) is used to create a speedup by reducing the amount of features in high

dimensional datasets. In the event of $p >> n$, candidate features can be ranked in descending order using a correlative relevance measure between normalised features and the class label [45]. Only the top $\frac{\lambda N}{logN}, \lambda \geq 1$ features which show the highest relevance to the class labels are screened for further processing. The parameter λ is user defined.

The pseudo code implementation of ORFS is given in Algorithm 2 below.

Algorithm 2 ORFS

- Initialise:

$S = \emptyset$ - Subset of best features
C - Class vector
T - Target number of features
$R(\cdot)$ - Associative measure
$X = \{x_1, x_2, ..., x_n\}$ - Feature dataset with n features

1: $s_1 = \arg\max_{x_i \in X} R(x_i, C)$
2: $S = [S\ s_1]$
3: **for** `k = 2,3,..,T` **do**
4: $\quad s_m = \arg\max_{x_i \in X-S} R(GSO(x_i, S), C)$
5: $\quad S = [S\ s_m]$
6: $\quad X = X$ - $\{s_m\}$
7: **end**
The features remaining in S are the best features

The ORFS algorithm starts by selecting the first best feature in line 1. The iterative process of selecting the next best feature is given in lines 3 - 7. Once the next best feature is chosen, it is appended into the best subset of features S and removed from the remaining features in dataset X. The algorithm terminates once the target variable T is reached.

2.3 Fast Correlation Based Filter

Recall that filter models for feature selection algorithms has the advantage of computational efficiency, which makes them suitable for high dimensional datasets. The paper presented by [8] proposed a filter method feature selection algorithm (FCBF) for high dimensional data. This section will touch on how FCBF achieves fast and effective feature selection, while remaining scalable for large datasets. Furthermore, an improved search strategy for FCBF will be discussed.

2.3.1 Correlation measure

As the name of FCBF suggests, a feature is considered suitable based on a correlation measure. The use of a correlation measure is used to determine the relevance and redundancy of a feature according to its class labels and other relevant features respectively. For a feature to be suitable, it requires a high relevance (predictability) to its class labels while ensuring that no other relevant feature has a better or equivalent capability of doing the same prediction. FCBF makes use of entropy and MI as its correlation measure. Recall that the MI between discrete variable pairs (X, Y) is defined in (2.1.17) as

$$\widehat{MI}(\hat{A}, \hat{B}) = \sum_{\hat{a}\in\hat{A}} \sum_{\hat{b}\in\hat{B}} log \frac{p(\hat{a}, \hat{b})}{p(\hat{a})p(\hat{b})} p(\hat{a}, \hat{b})$$

MI alone can be used as a suitable correlation, but MI is biased in favour of variable pair (X,Y) with more values. To avoid this bias, Symmetrical Uncertainty (SU) is used, which normalises MI by the sum of entropy for each vector such that

$$SU(X, Y) = \frac{2 \times MI(X, Y)}{H(X) + H(Y)}, \tag{2.3.1}$$

where SU is the Symmetrical Uncertainty measure, $H(X)$ is the entropy of variable X and $H(Y)$ is the entropy of variable Y. The values of SU are in the range [0,1], where 0 indicates that X and Y are completely independent and 1 indicates that X and Y can completely predict one another.

SU makes use of MI which requires nominal[5] features to be predicted, therefore real valued data needs to be discretised into nominal values. FCBF uses the discretisation method proposed by [46] which employs the Minimum Description Length (MDL) principle to determine optimal cut points in real valued data. The cut points, if any, are then used to bin the real valued data into their nominal values.

Many different discretisation algorithms exist, each with their own advantage. The paper presented in [47] suggests if the outcome is to simply discretise the data, then MDL should be the first choice. There are, however, different methods to consider when dealing with feature selection algorithms. The Chi2 merging discretisation method uses the χ^2 statistic and is able to remove redundant and irrelevant features during discretisation and stands as a feature removal algorithm on its own. Chi2 could prove to be a suitable

[5]Discrete, multi-valued data. Also commonly referred to as categorical data

pre-processing filter to remove additional irrelevant/redundant features before applying the FCBF algorithm but will not be considered as it is already a standalone feature removal technique. Furthermore, the removal of irrelevant/redundant features can also be partially applied to the method proposed by [46]. After dicretisation, a feature may be removed if no cut point exists, or if the nominal values assigned to a feature are repeated in other features.

2.3.2 FCBF methodology

Suppose a dataset S consists of feature vectors such that $S = [f_1, f_2...f_n]$. SU is used as a 'goodness' measure. $SU_{i,c}$ denotes the correlation between f_i and class label vector C, whereas $SU_{i,j}$ denotes the correlation between features f_i and f_j. A subset of relevant features S' can be obtained when $SU_{i,c}$ of any feature exceeds a certain threshold δ such that $\forall f_i \in S'$, $1 \leq i \leq n$, $SU_{i,c} \geq \delta$.

A correlation is said to be predominant if $SU_{i,c} \geq \delta$ and $\forall f_j \in S'$ $(j \neq i)$, such that no f_j exists where $SU_{i,j} \geq SU_{i,c}$. If such a feature f_j exists, it is a redundant peer to f_i and appended to the redundant peer set P_i. If $P_i \neq \emptyset$, then it is split into two subsets, namely P_i^+ and P_i^-, where $P_i^+ = \{f_j | f_j \in P_i, SU_{j,c} > SU_{i,c}\}$ and $P_i^- = \{f_j | f_j \in P_i, SU_{j,c} \leq SU_{i,c}\}$. A feature is said to be predominant if it adheres to the predominant correlation before or after removing redundant peers, if any.

Predominant features are identified using three separate heuristics which also removes redundant features without the need to identify all redundant peers for $\forall f_i \in S'$. The three heuristics are as follows

1. If $P_i^+ = \emptyset$, then f_i is a predominant feature. Keep f_i and remove all features from P_i^-.

2. If $P_i^+ \neq \emptyset$, process all features in P_i^+ first. If any of the features within P_i^+ become predominant, follow Heuristic 1, else remove f_i and remove P_i^- based on the remaining features in S'.

3. The starting point is the feature with the greatest $SU_{i,c}$ value and is considered as a predominant feature.

2.3.3 FCBF algorithm

The algorithm for FCBF is given below in algorithm 3. A brief explanation of the algorithm follows.

Algorithm 3 FCBF

- Initialise:

S = $[f_1, f_2 ... f_n]$ - Data set
C - Class vector
Threshold $\delta = 0.3$
$F_s = \forall f_i, SU(i,c) \geq \delta$ - Subset of best features in descending order

```
1:  Find the first best feature f_p = firstElement(F_s)
2:  while (f_p != NULL) do
3:      f_q = nextElement(F_s − {f_p})
4:      if (f_q != NULL) then
5:          while (f_q != NULL) do
6:              SU_{p,q} = SU(f_p, f_q)
7:              SU_{q,c} = SU(f_q, C)
8:              if (SU_{p,q} ≥ SU_{q,c}) then
9:                  F_s = F_s - {f_q}
10:                 f_q = nextElement(F_s)
11:             else
12:                 f_q = nextElement(F_s)
13:             end
14:         end
15:         f_p = nextElement(F_s)
16:     end
17: end
```

The features remaining in F_s are the best features

All features with a $SU_{i,c}$ value greater than δ are selected as possible candidates for the final subset of features (F_s). The candidates are sorted in descending order, starting with the most promising feature first (line 1 - Heuristic 3). In an iterative fashion (line 2), each candidate feature (f_p in F_s) is compared to all other possible candidates f_q in F_s for redundancy according to Heuristics 1 and 2 (lines 5 - 14). The search for redundancy starts from the index of the feature adjacent from f_p in F_s, i.e the index at $\{f_p\} + 1$. Once all redundant features, if any, within F_s are removed (line 9) then the next feature in F_s is selected as f_p (line 15) and once again tested for redundancies with the remaining features in F_s (lines 5 - 14). The algorithm terminates when all features within F_s have been tested.

The calculation of F_s in the beginning has a linear time complexity of N, where N is the number of features. During the elimination process, the best case results in all features being eliminated, while the worst case being no features are eliminated. The assumption is made that at least half of the features are eliminated each iteration [8], which results in a time complexity of log$N$. The calculation of SU is linear with regards to the amount of features M, therefore the overall combined time complexity is $O(MN\log N)$.

2.3.4 Implementing an alternate search strategy (FCBF#)

A method proposed by [9] considered FCBF with a different search strategy. Their motivation stems from the fact that FCBF may eliminate too many features when inputs are highly correlated. The new search algorithm (FCBF#) involves a different feature elimination strategy to perform better when selecting a specific amount of features k, instead of the best subset. In each iteration of FCBF, all features which show a higher correlation to the predominant feature than the class label is removed. This effect is desirable to determine the best subset, but may cause highly correlated features to the class to be eliminated very early in the algorithm.

This early elimination of correlated features to the class is not ideal when choosing k best features, but instead these features need to be eliminated last. FCBF# achieves this by assigning a temporary predominance to each feature in the elimination process which is used to eliminate features with the least correlation to the class. Unlike FCBF, FCBF# allows each feature to eliminate only one feature in each step of iteration, creating a more balanced elimination process. Once each feature had a chance to remove possible redundancies, the elimination process repeats for the remaining features and continues until k has been reached, or if no further eliminations are possible. The FCBF# algorithm is given below in Algorithm 4. A brief explanation to the algorithm follows.

As with FCBF, F_s starts off as a descending list of best candidate features. Once again in an iterative fashion (line 1), each feature f_p in F_s is compared to all other possible candidates f_q in F_s for redundancy according to Heuristics 1 and 2 (lines 5 - 14), except that the new search strategy for redundancy requires f_q to start at the end of the list F_s and make its way towards f_p. If f_q reaches f_p in the search strategy, the search is terminated (lines 8 - 10). If a feature is removed, flags $pass$ and rem are set to '1' (lines 13 - 20). Setting $pass = 1$ terminates the search for redundancy at line 7, which prevents further elimination of features (therefore only the first redundant feature is removed) and f_p is assigned to the next feature in F_s (line 24). Once all f_p in F_s are tested (lines 4 - 25), rem is either remains '0' or is set to '1', where '1' indicates that features were removed and allows the algorithm to re-search through F_s to eliminate the possibility more new redundant features. Notice f_p once again starts at the beginning of F_s (line 2). If rem remains 0 for the entirety of the removal procedure, then no more features can be removed and the algorithm terminates at line 1. If the number of remaining features amounts to k, then a termination flag is set (line 18) which terminates the algorithm at lines 4 and 1. After the algorithm terminates, the best subset of features remain in the list F_s.

Algorithm 4 FCBF#

- Initialise:

S = $[f_1, f_2 ... f_n]$ - Data set
C - Class vector
Threshold $\delta = 0.3$
$F_s = \forall f_i, SU(i,c) \geq \delta$ - Subset of best features in descending order
k - Target size
$rem = 1$ - Variable indicating a feature was removed
$flag = 0$ - Stop criteria when k is met
$pass = 0$ - Variable preventing multiple features from being deleted at once

```
1:  while (rem == 1 AND flag != 1) do
2:      f_p = firstElement(F_s)
3:      rem = 0
4:      while (flag != 1 AND f_p != NULL) do
5:          f_q = lastElement(F_s)
6:          pass = 0
7:          while (f_q != NULL AND pass != 1) do
8:              if (f_q == f_p) then
9:                  BREAK
10:             end
11:             SU_{p,q} = SU(f_p, f_q)
12:             SU_{q,c} = SU(f_q, C)
13:             if (SU_{p,q} ≥ SU_{q,c}) then
14:                 F_s = F_s - {f_q}
15:                 pass = 1
16:                 rem = 1
17:                 if (|F_s| == k) then
18:                     flag = 1
19:                 end
20:             else
21:                 f_q = prevElement(F_s)
22:             end
23:         end
24:         f_p = nextElement(F_s)
25:     end
26: end
```

The features remaining in F_s are the best features

A comparison between FCBF and FCBF# in [9] shows FCBF to be slightly computationally quicker. When the best subset of features are determined (without k), FCBF# selects fewer features than FCBF, but FCBF outperforms FCBF# with regards to classification accuracy. Selecting the best subset of features without choosing a value for k favours the traditional FCBF search strategy, but FCBF# shows dominance over the traditional search strategy with improved accuracy when a feature subset of size k is selected. Only FCBF# will be the considered in experiments.

For an in depth analysis, different associative measures, search strategies, pre-processing, SIS manipulation and supervised additions to mRMR, FCBF and ORFS will be tested and compared to one another where applicable. The addition of supervision[6] for each algorithm is an attempt to include a test for feature synergy by comparing the predictive model's accuracy before and after the addition of the next best feature.

Part II - Wrapper and embedded techniques

The next three algorithms include two wrapper methods and one embedded method. The Particle Swarm Optimisation (PSO) and SVM (Support Vector Machine) algorithms are the wrapper methods, while logistic regression is used for the embedded method. The predictive model used for the wrapper methods is once again a 1-NN predictor to reduce predictor bias. The background for all three aforementioned methods are explained in detail and their possible improvement and optimisation discussed.

2.4 Particle Swarm Optimisation

The paper presented in [23] proposed the FRBPSO algorithm suitable for high-dimensional data. FRBPSO is combined with techniques from B-BPSO as well as Linearly-Decreasing Inertia Weights (LDIW) to find an effective feature selection method using PSO. The Fuzzy component adds an additional layer of intelligence in order to avoid the PSO algorithm getting stuck in a local optimum. The behaviour of PSO, FRBPSO, B-BPSO and LDIW is explained in detail, thereafter an implementation of the FRBPSO feature selection algorithm is provided. For this section, note that vectors in text and equations are specified in bold unless otherwise stated.

2.4.1 Defining PSO

PSO is a meta-heuristic algorithm which optimises a problem according to an objective function (usually a minimisation or maximisation) by swarming the solution space with candidate solutions (particles). Each particle is assigned a fitness value through the objective function, which indicates the quality of that particle's current solution. Each particle traverses the solution space in an iterative fashion. PSO is an attractive optimising algorithm as it can search through very large spaces of candidate solutions without needing to calculate the gradient of the function which needs optimisation. This allows

[6] A predictive model is incorporated to test classification accuracy during the feature selection process

PSO to be applied to extremely complex or non-differentiable functions which would otherwise be very difficult to solve using methods such as Stochastic Gradient Descent (SGD).

The PSO algorithm starts with a random initialisation of particles. All particles, X, are assigned a position where $\boldsymbol{x_i} = [x_{i1}, ...x_{id}]$ represents the position of particle i with d dimensions. All particles are also assigned a velocity V, where $\boldsymbol{v_i} = [v_{i1}, ...v_{id}]$ is associated with each particle. The position of a particle represents a possible solution, whereas the velocity indicates the movement of the particle. The new velocity of a particle v_{id}^{new} is determined by the difference between the particle's positions at its previous best solution pb_{id} and its current position x_{id}^{old}, denoted as vp_{id}, as well as the difference between the position of the current particle and global best particle gb_{id}, denoted as vg_{id}. Mathematically, the updated velocity of a particle proposed by [48] is given by

$$v_{id}^{new} = v_{id}^{old} + vp_{id} + vg_{id}, \tag{2.4.1}$$

where v_{id}^{old} is the previous velocity of particle i, feature d. The velocities vp_{id} and vg_{id} are defined as

$$vp_{id} = c_1 r_1 (pb_{id} - x_{id}^{old}) \tag{2.4.2}$$

$$vg_{id} = c_2 r_2 (gb_d - x_{id}^{old}), \tag{2.4.3}$$

where pb_{id} is particle i's personal best position, gb_d is the particle which holds the global best position, c_1 and c_2 are constants, and r_1 and r_2 are random numbers $\in [0, 1]$. Choosing the correct values for $c1$ and c_2 is an optimisation problem in itself. The velocity is then used to update the position of the particle such that

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new} \tag{2.4.4}$$

The optimisation is terminated when the fitness value of a particle falls within a user defined tolerance, or when the maximum iterations allowed are met. To balance between exploration and exploitation of possible solutions, an inertial weight ω is multiplied with v_{id}^{old}. Equation (2.4.1) becomes

$$v_{id}^{new} = \omega * v_{id}^{old} + vp_{id} + vg_{id} \tag{2.4.5}$$

Selecting the appropriate ω is also an optimisation problem in itself. The study in [49, 50] describes the formulation and advantages of LDIW as a method to vary the value of ω in each iteration. Using LDIW is a simple approach which updates ω with

$$\omega_{new} = (\omega_{max} - \omega_{min})\frac{Maxite - t}{Maxite} + \omega_{min}, \tag{2.4.6}$$

where $Maxite$ is the maximum allowable iterations for the PSO algorithm, t is the current iteration number, ω_{max} is the maximum allowable inertia, and ω_{min} is the minimum allowable inertia. The value of ω will vary between ω_{max} and ω_{min} depending on the iteration. The inertial weight will start at ω_{max}, which improves the initial global search for the optimal solution. As the iterations continue, ω will tend to ω_{min}, which improves the local search for the optimal solution at later iterations of the algorithm. The paper presented in [49] shows a value of 0.9 and 0.4 for ω_{max} and ω_{min} respectively provides the most promising convergence rates.

2.4.2 Binary PSO

The particles for PSO can be a vector of any length d, but the values within the vector are not limited to real numbers. A vector of binary bits can also be used to define particles, as used in BPSO [48] which can be used to solve discrete problems and is especially useful in the case for feature selection. Because the particles are defined by a binary vector, the new position of a particle x_{id}^{new} cannot be updated using (2.4.4), instead, an if-else statement is used to determine x_{id}^{new} such that

$$x_{id}^{new} = \begin{cases} 1 & \text{if } F < S(v_{id}^{new}) \\ 0 & \text{otherwise,} \end{cases} \tag{2.4.7}$$

where F is the choice between a random number generator or a constant of 0.5, and S is a sigmoid function to map the values of v_{id}^{new} between 0-1 which acts as a probability value. The paper presented in [51] improved BPSO by avoiding local minima by means of a Boolean $and(\cdot)$ operator to '*and*' each bit of $\boldsymbol{pb}$ for all particles in order to create a new global best particle $\boldsymbol{gb}$. The creation of a new global best particle only occurs if

the fitness value for ***gb*** is identical after three generations or iterations. An iteration is complete once all particles have been fully updated, including global and personal best positions.

A further improvement to PSO was proposed in the study of FRBPSO [23] which re-evaluated equations (2.4.2) and (2.4.3). They argue that assigning the difference of position to a velocity requires some re-thinking. To create a true sense of velocity, equations (2.4.2) and (2.4.3) are divided by a unit time steps $p_{timestep}$ and $g_{timestep}$ respectively, which are the time (iteration) differences between obtaining the previous personal best and global best fitness values and the current time (iteration) respectively. The new equations for vp_{id} and vg_{id} are given by

$$vp_{id} = c_1 r_1 (pb_{id} - x_{id}^{old})/p_{timestep} \tag{2.4.8}$$

$$vg_{id} = c_2 r_2 (gb_d - x_{id}^{old})/g_{timestep} \tag{2.4.9}$$

This change for vp_{id} and vg_{id} is more intuitive and claims to provide faster convergence.

2.4.3 Including fuzzy logic

PSO, being a meta-heuristic algorithm, suffers from the disadvantage that the solution might be suboptimal if any particle fails to converge to the optimal solution, in which case the PSO algorithm is repeated multiple times in order to find the optimal solution. It is possible for a particle to miss the optimal solution if the particle traverses past the optimal point due to an excessively large particle update, or if the particle gets stuck in a local minima or maxima. Having a look at (2.4.2), the randomness of parameter r also adds uncertainty as part of the solution, which contributes to the possibility of PSO finding suboptimal solutions. To avoid suboptimal solutions, the work in [23] proposed an extra layer of intelligence using fuzzy logic to help avoid suboptimal solutions by updating each particle based on their velocities.

Suppose a car has an automated system which starts applying the brakes fully when the car is closer than 'X' m from an object. The system therefore has a binary output of '0' (do not apply brakes) or '1' (apply the brakes) depending on the distance to an object. The system will assist the driver to avoid collision if the car is at a low speed, but what if the car is travelling at a higher speed, it would make sense for the brakes to be applied earlier. The system lacks the intelligence to make better predictions when and how hard to apply the brakes in such situations. This is where a Fuzzy Inference System (FIS) can

be useful. Using the speed of the car, as well as the distance to the object in front of the car, a FIS will use these inputs to determine when and how hard the brakes should be applied according to a set of predefined rules. This effectively changes the discrete output system into an continuous analog output system.

The work in [23] uses the fuzzy nature of FIS in conjunction with BPSO in order to update x_{id}^{new}. The inputs to the FIS are v_{id}^{old} (memory of velocity), vp_{id} (cognitive component) and vg_{id} (social component) in the ranges [-6,6], [-2,2] and [-2,2] respectively. The reason of choice for these ranges were not specified and therefore the performance influence to the FIS induced by changing these ranges can be studied, but is not within scope of this study. Each input is mapped to two Gaussian membership functions (High(H) and Low(L)), therefore there are eight possible input combinations which determine the output of the Fuzzy system by predefined rules. The output of the Fuzzy system is x_{id}^{new}, which takes on singleton values of 0.2 (L - reject the feature) and 0.8 (H - accept the feature) depending on the input. The parameters for each Gaussian membership function and all eight Fuzzy rules are in given below in Tables 2.1 and 2.2 respectively.

Table 2.1: Gaussian membership function parameters

	Mean	Variance
vp_{id}(L)	-2	1.5
vg_{id}(L)	-2	1.5
v_{id}^{old}(L)	-6	-2.5
vp_{id}(H)	2	-1.5
vg_{id}(H)	2	-1.5
v_{id}^{old}(H)	6	2.5

Table 2.2: Fuzzy Rules

vp_{id}	vg_{id}	v_{id}^{old}	x_{id}^{new}
L	L	L	L
L	L	H	L
L	H	L	L
L	H	H	H
H	L	L	L
H	L	H	H
H	H	L	H
H	H	H	H

Refer to Table 2.2 above. A feature is accepted (x_{id}^{new} = H) when both the social and cognitive component are high, but a feature is rejected (x_{id}^{new} = L) when both the social

and cognitive component are low. If the social and cognitive components are opposite (one is high and the other is low), then the output is determined by the previous velocity of the feature. The fuzzification and defuzzification specifications are given in Table 2.3 below. The new feature vector $\boldsymbol{x_i}^{new}$ is obtained by rounding the output of the FIS to the nearest integer.

Table 2.3: Fuzzy methods

Fuzzification		Defuzzification
'And' method	Product	Centroid
Implication	Minimum	
Aggregation	Summation	

2.4.4 Implementing FRBPSO

The feature selection algorithm FRBPSO is combined with the methods LDIW and B-BPSO in order to evaluate the effectiveness to select appropriate features in large datasets. Algorithm 5 below describes the algorithm in detail. Recall that the fitness function used is a 1-NN classifier represented as

$$\text{k-NN model} = \text{1-NN}(\boldsymbol{x}, D, \boldsymbol{y}), \tag{2.4.10}$$

where D is a dataset with t samples and m features, $\boldsymbol{x}$ is a binary vector of size m which determines whether a feature in dataset D is kept ('1') or discarded ('0'), and $\boldsymbol{y}$ is the class label vector of size t. The fitness value of a particle is calculated using Leave One Out Cross Validation (LOOCV) as follows

$$f_i = LOOCV(\text{1-NN}(\boldsymbol{x_i}, D, \boldsymbol{y})), \tag{2.4.11}$$

where f_i is the fitness value for particle $\boldsymbol{x_i}$. In feature selection, it is also important to try keep the minimum amount of necessary features, therefore if the fitness value of two particles are identical, the particle with the least amount of features is chosen as the best particle. The feature size is applicable when updating $\boldsymbol{pb_i}$ and $\boldsymbol{gb}$. The feature size of a particle can also be considered within the fitness function by contributing to the fitness value, for example

$$f_i = LOOCV(\text{1-NN}(\boldsymbol{x_i}, D, \boldsymbol{y})) + \lambda \frac{sum(\boldsymbol{x_i})}{M}, \tag{2.4.12}$$

where λ is a loss factor variable, and M is defined by the user as the maximum allowable features. For testing purposes, (2.4.11) is used in Algorithm 5 below which shows the

pseudo code implementation of FRBPSO. In this implementation, a new global best particle $\boldsymbol{gb}$ is created if there is no change in $\boldsymbol{gb}$ within five iterations. One must be careful in choosing the amount of iterations, as a value too low can cause no features to be retained due to the initial inconsistency of selected features in all particles.

Algorithm 5 Boolean FRBPSO with LDIW

- Initialise:

Maximum allowable iterations $MaxIte$
$n = \#$particles, $m = \#$features
$c_1 = c_2 = 2$, $\omega_{max} = 0.9$, $\omega_{min} = 0.4$, $v_{max} = C$
$v_{id}^{initial}$ in $rand(\in [-v_{max}, v_{max}])$ $\forall_{i,d}$
$x_{id}^{initial}$ in $rand(\in \{0, 1\})$ $\forall_{i,d}$
$pb_{id} = x_{id}$
D is the dataset with t rows and m columns
$\boldsymbol{y}$ is the class label vector of size t

- Find initial fitness values $f_i = LOOCV(\text{1-NN}(\boldsymbol{x_i}, D, \boldsymbol{y}))$ $\forall_i$
- Find the global best particle $\boldsymbol{gb} = \boldsymbol{x_i}$ where $f_g = \min_i(\boldsymbol{f})$

while ($f_g >$ tolerance) ***and*** (iterations $\leq MaxIte$) **do**
  $\omega = (\omega_{max} - \omega_{min})\frac{Maxite - iterations}{Maxite} + \omega_{min}$
  for `i = 1,2,3,..,n` **do**
    $\boldsymbol{vp_i} = c_1\boldsymbol{r_1}(\boldsymbol{pb_i} - \boldsymbol{x_i}^{old})/p_{timestep}$
    $\boldsymbol{vg_i} = c_2\boldsymbol{r_2}(\boldsymbol{gb} - \boldsymbol{x_i}^{old})/g_{timestep}$
    $(\boldsymbol{v_i}^{old} > v_{max}) = v_{max}$
    $(\boldsymbol{v_i}^{old} < -v_{max}) = -v_{max}$
    $\boldsymbol{x_i}^{new} = round(FIS(\boldsymbol{vp_i}, \boldsymbol{vg_i}, \boldsymbol{v_i}^{old}))$
    $v_{id}^{new} = \omega * v_{id}^{old} + vp_{id} + vg_{id}$
  end
  Evaluate new fitness values $\boldsymbol{f}$ =1-NN($\boldsymbol{x_i}^{new}, \boldsymbol{y}$) $\forall_i$
  for `i = 1,2,3,..,n` **do**
    if ($f_i < f_i^{pb}$) ***or*** ($f_i = f_i^{pb}$ ***and*** $sum(\boldsymbol{x_i}^{new}) < sum(\boldsymbol{pb_i})$) **then**
      $\boldsymbol{pb_i} = \boldsymbol{x_i}^{new}$
    end
  end
  if ($min_i(\boldsymbol{f}) < f_g$) ***or*** ($min_i(\boldsymbol{f}) = f_g$ ***and*** $sum(\boldsymbol{x_i}) < sum(\boldsymbol{gb})$) **then**
    $\boldsymbol{gb} = \boldsymbol{x_i}^{new}$
    $f_g = f_i$
  else if ($\boldsymbol{gb}$ unchanged for past 5 iterations) **then**
    Create new $\boldsymbol{gb} = \boldsymbol{pb_1}$
    for k = 2,...,n **do**
      $\boldsymbol{gb} = \boldsymbol{AND}(\boldsymbol{gb}, \boldsymbol{pb_k})$
    end
    f_g =1-NN($\boldsymbol{gb}, \boldsymbol{y}$)
  end
  iteration = iteration + 1
end

The FRBPSO algorithm only terminates if the tolerance of the fitness function is met or

if the maximum allowable iterations met (line 1). The LDIW update takes place at the beginning of each iteration (line 2). Each particle is then updated in lines 3 - 10. For each particle, their new velocities $\boldsymbol{v_i}^{new}$ and positions $\boldsymbol{x_i}^{new}$ are updated in lines 4 - 9. Recall that the velocities are bound within the values of v_{max}. Notice the calculation for $\boldsymbol{x_i}^{new}$ is based on a Fuzzy Inference System. The fitness values for all updated particles are then calculated and the new personal best positions and global best particle is then evaluated (lines 11 - 19). If the new fitness value outperforms its previous best fitness then the particle's personal best position is updated. Likewise, if the new fitness outperforms the global best particle, the current particle becomes the new global best. Notice that the amount of features within each particle is also compared to its previous best only if the fitness values are identical. This test for feature length is necessary to award particles which show similar fitness values but contain less features. If a new global best particle has not been assigned in the last five iterations (line 20), a new global best particle is created in lines 21 - 25 and its new fitness value calculated.

2.5 Support Vector Machines

SVMs are common predictive models used for classification. The fundamentals behind SVMs is to solve an optimisation problem which involves finding an optimal hyperplane to separate data which belongs to different class labels. In order to find the optimal hyperplane, a loss function is optimised, which can be further applied to feature selection. To accommodate large datasets, it is recommended to use a fast converging algorithm such as SGD for hyperplane optimisation. In the following section, all formulations were achieved with the help of [52]. Subsections 2.5.1 to 2.5.3 explains how an SVM functions as a classifier. The use of SVMs for feature selection is then discussed in subsection 2.5.4.

2.5.1 Formulation of a linearly separable Hard Margin SVM

For clarity, the naming convention used for this section is as follows: given a dataset with sample-label pairs $(\boldsymbol{x_i}, y_i)$, $i = 1, .., n$, $\boldsymbol{x_i}$ represents a sample such that $\boldsymbol{x_i} \in \mathbb{R}^m$, y_i is the class label for sample $\boldsymbol{x_i}$ such that $y \in \{-1, 1\}$. The optimisation problem of an SVM is to determine the optimal separating hyperplane (also known as a decision boundary) given a set of instance-label pairs. Figure 2.5 below illustrates a binary example of an optimal linear separating hyperplane. The 'X's represent positive examples, and 'O's the latter. The solution is optimal when the distance between the hyperplane (blue dotted line) and features of each class are maximum, defined by a margin (red lines).

Refer to Figure 2.5 below. It is crucial to determine whether the decision boundary is

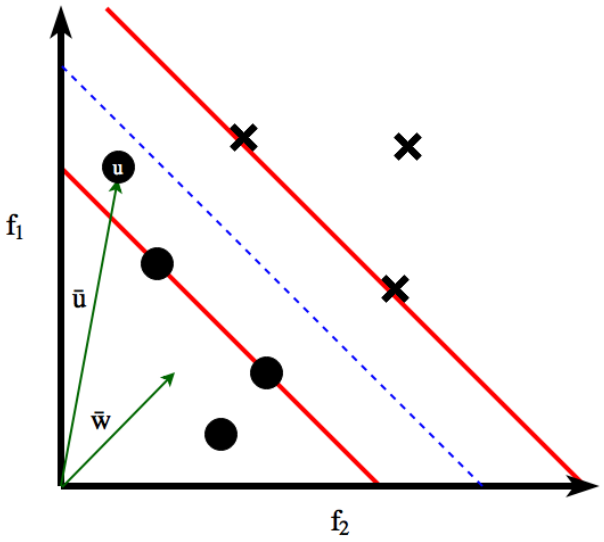


Figure 2.5: SVM optimisation problem

optimal. A decision rule needs to be formulated to use the decision boundary. Let vector $\boldsymbol{w}$ represent a vector of arbitrary length constrained to be perpendicular to the decision boundary. Let $\boldsymbol{u}$ represent an unknown vector pointing to an unknown point u in the feature space. The unknown point belongs to an unknown class, and can be classified by checking which side of the boundary decision it lies on. If vector $\boldsymbol{u}$ is projected onto vector $\boldsymbol{w}$, the proportional distance in the direction of $\boldsymbol{w}$ is known, and if this projected value is greater than some constant C, the side which point u lies on can be determined. A positive example is illustrated by (2.5.1) below.

$$\boldsymbol{w} \cdot \boldsymbol{u} > C \tag{2.5.1}$$

Rearranging (2.5.1) gives

$$\boldsymbol{w} \cdot \boldsymbol{u} + b > 0, \tag{2.5.2}$$

where $b = -C$. The decision rule is now formed, but $\boldsymbol{w}$ and b are still unknown, therefore there is a need for more constraints in order to solve for $\boldsymbol{w}$ and b. The decision rule given by (2.5.2) insists that an unknown point belongs to a positive sample if it lies on the right hand side of the decision boundary. Thus far, the margins do not determine whether an unknown belongs to a specific class. To enforce this, let

$$\boldsymbol{w} \cdot \boldsymbol{x}_+ + b \geq 1, \tag{2.5.3}$$

where $\boldsymbol{x}_+$ is a known positive sample and $\boldsymbol{w}$ and b are chosen to provide a margin of 1 relative to the size of b. This constraint forces the distance between the decision boundary and any margin to be $|1|$. Likewise,

$$\boldsymbol{w} \cdot \boldsymbol{x}_- + b \leq -1, \tag{2.5.4}$$

where $\boldsymbol{x}_-$ is a known positive sample. Introducing a new variable y_i merges (2.5.3) and (2.5.4) for mathematical convenience, where

$$y_i = \begin{cases} 1, & \text{for + samples} \\ -1, & \text{for - samples} \end{cases}$$

The decision rule can then be written as

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1, \tag{2.5.5}$$

where $\boldsymbol{x_i}$ is any sample. Rearranging (2.5.5) gives

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) - 1 \geq 0 \tag{2.5.6}$$

Because this constraint is based on the margins, we know that any point that satisfies

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) - 1 = 0 \tag{2.5.7}$$

will lie on the margin. Recall that the optimisation problem is to find the optimal decision boundary. This can be achieved by finding the largest margin. An expression is therefore required to express the distance between the two margins. Figure 2.6 illustrates how the distance can be calculated.

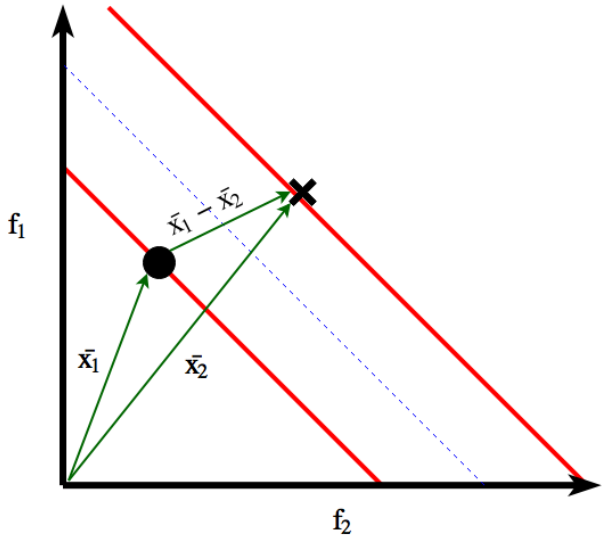


Figure 2.6: Distance between the margins

Refer to Figure 2.6 above. Using basic vector algebra, the distance between the two margins can be calculated by as follows

$$Width = (\boldsymbol{x_1} - \boldsymbol{x_2}) \cdot \boldsymbol{n}, \tag{2.5.8}$$

where $\boldsymbol{n}$ is a normal vector to the margins. The normal vector $\boldsymbol{n}$ can be determined through $\boldsymbol{w}$ as it is already perpendicular to the margins. Equation (2.5.8) can be simplified to

$$\begin{aligned} Width &= (\boldsymbol{x_1} - \boldsymbol{x_2}) \cdot \frac{\boldsymbol{w}}{\| \boldsymbol{w} \|} \\ Width &= \boldsymbol{x_1} \cdot \frac{\boldsymbol{w}}{\| \boldsymbol{w} \|} - \boldsymbol{x_2} \cdot \frac{\boldsymbol{w}}{\| \boldsymbol{w} \|} \\ Width &= \frac{1}{\| \boldsymbol{w} \|}(\boldsymbol{x_1} \cdot \boldsymbol{w} - \boldsymbol{x_2} \cdot \boldsymbol{w}) \end{aligned} \tag{2.5.9}$$

Substituting (2.5.7) into (2.5.9), given y_i, gives

$$\begin{aligned} Width &= \frac{1}{\| \boldsymbol{w} \|}((1 - b) - (b - 1)) \\ Width &= \frac{2}{\| \boldsymbol{w} \|} \end{aligned} \tag{2.5.10}$$

The optimisation is to maximise the width of the margins in (2.5.10), shown in (2.5.11) below.

$$\begin{aligned} &\max \frac{2}{\| \boldsymbol{w} \|} \\ \approx &\max \frac{1}{\| \boldsymbol{w} \|} \\ \approx &\min \| \boldsymbol{w} \| \\ \approx &\min \frac{1}{2} \| \boldsymbol{w} \|^2 \end{aligned} \tag{2.5.11}$$

The format for (2.5.11) is for mathematical convenience, and also illustrates the primal form of the optimisation problem. In order to solve an optimisation problem with a set number of constraints, Lagrange mathematics is applied. The Lagrange equation with the SVM optimisation problem described above is given as follows

$$L = \frac{1}{2} \| \boldsymbol{w} \|^2 - \sum_i \alpha_i [y_i (\boldsymbol{w} \cdot \boldsymbol{x_i} + b) - 1], \tag{2.5.12}$$

where $\alpha_i \geq 0 \in \mathbb{R}$. Notice the first half of (2.5.12) represents the parameter to optimise, and the second half of the equation represents the constraint to the optimisation parameter. To find the extrema of (2.5.12), the partial derivative of each parameter that has to be solved needs to be taken and set to 0. The partial derivative with respect to vector $\boldsymbol{w}$ is given in (2.5.13) below. Keep in mind that the derivative of a vector follows the same convention as taking the derivative of a scalar.

$$\begin{aligned} \frac{\delta L}{\delta \boldsymbol{w}} &= \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x_i} \\ 0 &= \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x_i} \\ \boldsymbol{w} &= \sum_i \alpha_i y_i \boldsymbol{x_i} \end{aligned} \tag{2.5.13}$$

Notice that $\boldsymbol{w}$ is a linear sum of scalars. The partial derivative with respect to b is given in (2.5.14) below.

$$\begin{aligned}\frac{\delta L}{\delta b} &= -\sum_i \alpha_i y_i \\ 0 &= -\sum_i \alpha_i y_i \\ 0 &= \sum_i \alpha_i y_i \end{aligned} \tag{2.5.14}$$

Substitution of (2.5.13) into (2.5.12) gives

$$\begin{aligned} L =& \frac{1}{2}(\sum_i \alpha_i y_i \boldsymbol{x_i}) \cdot (\sum_j \alpha_j y_j \boldsymbol{x_j}) - (\sum_i \alpha_i y_i \boldsymbol{x_i}) \cdot (\sum_j \alpha_j y_j \boldsymbol{x_j}) \\ & - b\sum_{\boldsymbol{i}} \boldsymbol{\alpha_i y_i} + \sum_i \alpha_i \end{aligned} \tag{2.5.15}$$

Notice the similarity between (2.5.14) and the bold section in (2.5.15). Simplifying (2.5.15) yields

$$\begin{aligned} L &= \sum_i \alpha_i + \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j} - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j} \\ L &= \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j} \end{aligned} \tag{2.5.16}$$

Equation (2.5.16) is also referred to as the dual representation of the optimisation problem which needs to be maximised. Notice the optimisation depends only on the dot product of the pairs of samples. Substituting (2.5.13) into the original decision rule in (2.5) which yields the new decision rule for a positive sample as

$$\sum_i \alpha_i y_i \boldsymbol{x_i} \cdot \boldsymbol{u} + b \geq 0 \tag{2.5.17}$$

In summary, the primal form for the the Hard margin SVM is as follows

$$\begin{aligned} &\min_w \frac{1}{2}||\boldsymbol{w}||^2 \\ &\text{s.t } y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b) \geq 1 \end{aligned}$$

The classification test for a new x : $\boldsymbol{w}^T\boldsymbol{x_i} + b > 0$. The dual form for the the Hard margin SVM is as follows

$$\max_{\alpha_1 \ldots \alpha_M} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{j=1}^{M} \sum_{k=1}^{M} \alpha_j \alpha_k y_j y_k \langle \boldsymbol{x_j}, \boldsymbol{x_k} \rangle$$

$$\text{s.t } \alpha_i \geq 0$$

$$\sum_{i=1}^{M} \alpha_i y_i = 0$$

The classification test for a new $\boldsymbol{x}$: $\sum_i \alpha_i y_i \langle \boldsymbol{x}, \boldsymbol{x_i} \rangle + b > 0$.

2.5.2 Formulation of a linearly separable Soft Margin SVM

Suppose the data is not entirely linearly separable, but an error margin in the classification is acceptable, therefore some points are allowed to violate the decision rule. Figure 2.7 below shows an example of such a case.

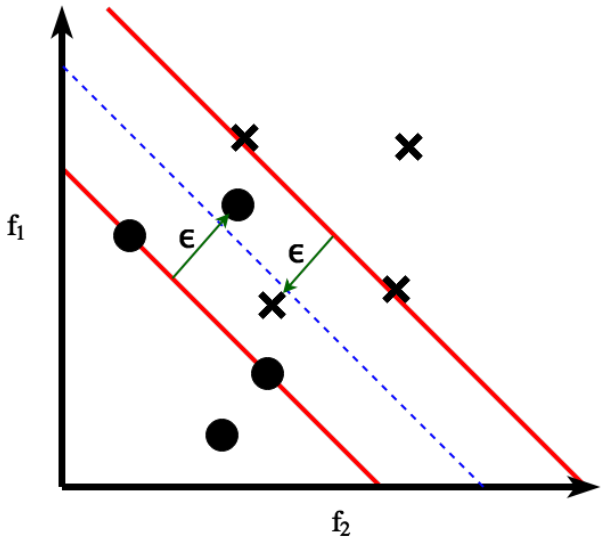


Figure 2.7: Not linearly separable

In Figure 2.7 above, there are two points which violate the decision rule given in (2.5.5) by the amount of ϵ (the distance which the point violates the margin). If the decision rule changed to

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1 - \epsilon_i, \tag{2.5.18}$$

then points which violate the original decision rule by $\boldsymbol{\epsilon}$ will still be chosen as support vectors[7]. The addition of $\boldsymbol{\epsilon}$ are known as slack variables. The new optimisation problem is to still maximise the margin, but with the addition of minimising the error margin ($\boldsymbol{\epsilon}$) allowed for all data points, scaled by some factor C. The primal form of the optimisation can therefore be written as

$$\begin{aligned} &\min_{\boldsymbol{w},\epsilon} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_i \epsilon_i \\ &\text{s.t } y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 - \epsilon_i \\ &\epsilon_i \geq 0 \end{aligned} \tag{2.5.19}$$

Refer to (2.5.19) above. It is intuitive for $\epsilon_i \geq 0$, as a data point either violates the decision rule, or it does not. Note that if C is large, then the error contribution is very large, and therefore a very small error margin is allowed. If C tends to ∞, then the optimisation simply becomes the Hard margin case. Similarly, if C is very small, then the optimisation allows for large error margins, allowing many points to violate the decision boundary. It is important to notice that the addition of $\boldsymbol{\epsilon}$ is a loss term, and in this specific case, an L_1-norm loss case. To formulate the new dual problem, the new Lagrange multipliers needs to be accounted for. Equation (2.5.19) is re-written in vector notation for ease of use.

$$\begin{aligned} &\min_{\boldsymbol{w},\boldsymbol{\epsilon}} \frac{\boldsymbol{w}^T\boldsymbol{w}}{2} + C^T\boldsymbol{\epsilon} \\ &\text{s.t } A\boldsymbol{w} - \boldsymbol{y}b + \boldsymbol{\epsilon} - 1 \geq 0 \\ &\boldsymbol{\epsilon} \geq 0, \end{aligned} \tag{2.5.20}$$

where $A = (\boldsymbol{y}X)^T$ and X is a the dataset matrix with all samples $\boldsymbol{x_i}$. Due to the extra variable $\boldsymbol{\epsilon}$, another Lagrange multiplier ($\boldsymbol{t}$) is needed. The Lagrange equation is as follows

$$\begin{aligned} &L = \frac{\boldsymbol{w}^T\boldsymbol{w}}{2} + C^T\boldsymbol{\epsilon} - \boldsymbol{\alpha}^T(A\boldsymbol{w} - \boldsymbol{y}b + \boldsymbol{\epsilon} - 1) - \boldsymbol{t}^T\boldsymbol{\epsilon} \\ &\boldsymbol{\alpha} \geq 0 \\ &\boldsymbol{t} \geq 0 \end{aligned} \tag{2.5.21}$$

[7]The vectors of samples chosen to maximise the margin

The objective is to minimise (2.5.21) with respect to α, b and $\boldsymbol{\epsilon}$. The partial derivatives for each variable need to be taken and set to 0. The partial derivative for w is given below.

$$\begin{aligned}\frac{\delta L}{\delta \boldsymbol{w}} &= \boldsymbol{w} - A\boldsymbol{\alpha} \\ 0 &= \boldsymbol{w} - A\boldsymbol{\alpha} \\ \boldsymbol{w} &= A\alpha \end{aligned} \tag{2.5.22}$$

The partial derivative for b is given below.

$$\begin{aligned}\frac{\delta L}{\delta b} &= \boldsymbol{\alpha}^T \boldsymbol{y} \\ 0 &= \boldsymbol{\alpha}^T \boldsymbol{y} \end{aligned} \tag{2.5.23}$$

The partial derivative for $\boldsymbol{\epsilon}$ is given below.

$$\begin{aligned}\frac{\delta L}{\delta \boldsymbol{\epsilon}} &= C - \boldsymbol{\alpha} - \boldsymbol{t} \\ 0 &= C - \boldsymbol{\alpha} - \boldsymbol{t} \end{aligned} \tag{2.5.24}$$

Substitution of (2.5.22) to (2.5.24) yields

$$\begin{aligned} L &= \frac{\boldsymbol{\alpha}^T A A^T \boldsymbol{\alpha}}{2} - \boldsymbol{\alpha}^T A A^T \boldsymbol{\alpha} + \boldsymbol{\epsilon}(C - \boldsymbol{\alpha} - \boldsymbol{t}) + \boldsymbol{\alpha}^T \boldsymbol{y} b + \boldsymbol{\alpha}^T \\ L &= -\frac{1}{2}\boldsymbol{\alpha}^T A A^T \boldsymbol{\alpha} + 1^T \boldsymbol{\alpha} \\ L &= 1^T \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha}, \end{aligned} \tag{2.5.25}$$

where $Q = AA^T$ and 1^T is a diagonal matrix consisting of the value '1' to ensure matrix form compatibility. Equation (2.5.25) only needs to optimise $\boldsymbol{\alpha}$, which now has a new constraint. From (2.5.24),

$$
\begin{aligned}
0 &= C - \boldsymbol{\alpha} - \boldsymbol{t} \\
\boldsymbol{\alpha} &= C - \boldsymbol{t}, \qquad (2.5.26)
\end{aligned}
$$

therefore it will always hold that $\boldsymbol{\alpha} \leq C$. The complete dual for the Soft margin L_1-norm SVM in vector form is given below.

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} L &= 1^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \\
\min_{\boldsymbol{\alpha}} L &= \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - 1^T \boldsymbol{\alpha} \qquad (2.5.27) \\
\text{s.t } 0 &\leq \boldsymbol{\alpha} \leq C \\
\boldsymbol{\alpha}^T \boldsymbol{y} &= 0
\end{aligned}
$$

The formulation for L_2-norm SVM can be found at [53]. L_2-norm SVM is similar to L_1-norm SVM, except that $\boldsymbol{\alpha}$ is not bounded, and a factor of $\frac{1}{2C}$ is added to the diagonal of matrix Q.

2.5.3 Solving a SVM using the dual formulation

Suppose a dataset X consists of n samples, with m features per sample : $x_i = \{1, ..., m\} \in \mathbb{R}^m$. With regards to the primal form and solution of the optimisation problem, $m + 1$ variables required for $\boldsymbol{w}$ and b respectively, and there also exists n constraints, one for each variable. With respect to the dual formulation, only n variables are needed, and only one equality exists. Also, the variable Q can be pre-computed. The dual formulation is therefore more suited for datasets with large feature sizes, as it is independent of the amount of features. A fast, efficient method for solving the dual formulation is to use a SGD algorithm. The work presented in [54, 55] solves the dual using the dual Coordinate Descent (DCD) method. The approach to this solution is described below. Refer to (2.5.27). After optimising $\boldsymbol{\alpha}$, the weights can be calculated using (2.5.13). To accommodate the bias parameter, an extra sample is appended to each $\boldsymbol{x_i}$ so that

$$
\begin{aligned}
\boldsymbol{x_i} &= [\boldsymbol{x_i}, 1] \qquad (2.5.28) \\
\text{s.t } \boldsymbol{w} &= [\boldsymbol{w}, b] \qquad (2.5.29)
\end{aligned}
$$

Initially, the $\boldsymbol{\alpha}$ vector is set to 0. The procedure consists of an outer iteration where $\boldsymbol{\alpha}^k$ is updated to $\boldsymbol{\alpha}^{k+1}$ in each outer iteration, and therefore generates a sequence of vectors $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$. The coordinate descent algorithm therefore needs to individually update α_i^k to α_{i+1}^k. The one-variable sub-problem to be solved for each outer iteration k is then:

$$\begin{aligned} &\min_{\delta} f(\boldsymbol{\alpha} + \delta e_i) \\ &\text{s.t } 0 \leq \alpha_i + \delta \leq C, \end{aligned} \tag{2.5.30}$$

where e_i, $i = 1, ..., m$ is a zeros column vector, but element i in $e = 1$. As an example, $e_2 = [0, 1, 0, ..., 0]^T$.

The coordinate descent algorithm solves for $\min_{0 \leq \alpha \leq C} f(\boldsymbol{\alpha})$, which requires a one variable sub problem update. The basic principle of the algorithm [56] is shown Algorithm 6 below.

Algorithm 6 DCD algorithm

```
1: procedure CoordDescent(α)
2:     for k = random(1,2,3...) do
3:         for α_i...α_m do
4:             min_δ f(α_i + δe_i)          ▷ such that 0 ≤ α_i + δ ≤ C
5:             α_i = α_i + δ
6:         end
7:     end
8: end
```

Expanding $f(\boldsymbol{\alpha}^k + \delta e_i)$ for each outer loop k into (2.5.27) yields:

$$\begin{aligned} f(\boldsymbol{\alpha}^k + \delta e_i) &= \frac{1}{2}(\boldsymbol{\alpha}^k + \delta e_i)^T Q(\boldsymbol{\alpha}^k + \delta e_i) - \sum_i (\boldsymbol{\alpha}^k) - \delta \\ &= \frac{1}{2}(\boldsymbol{\alpha}^k)^T Q\boldsymbol{\alpha}^k + \mathbf{\frac{1}{2}(\alpha^k)^T Q\delta e_i + (\frac{1}{2}\delta e_i)^T Q\alpha^k} \\ &\quad + (\frac{1}{2}\delta e_i)^T Q\delta e_i - \sum_i (\boldsymbol{\alpha}^k) - \delta \\ &= \frac{1}{2}(\boldsymbol{\alpha}^k)^T Q + (\boldsymbol{\alpha}^k)^T Q\delta e_i + \frac{Q_{ii}}{2}\delta^2 - \sum_i (\boldsymbol{\alpha}^k) - \delta \\ &= \frac{1}{2}(\boldsymbol{\alpha}^k)^T Q + \frac{Q_{ii}}{2}\delta^2 + \delta((\boldsymbol{\alpha}^k)^T Qe_i - 1) - \sum_i (\boldsymbol{\alpha}^k) \\ &= \frac{1}{2}(\boldsymbol{\alpha}^k)^T Q + \frac{Q_{ii}}{2}\delta^2 + \delta\left(((\boldsymbol{\alpha}^k)^T Q)_i - 1)\right) - \sum_i (\boldsymbol{\alpha}^k) \end{aligned} \tag{2.5.31}$$

$$= \frac{1}{2}(\boldsymbol{\alpha}^k)^T Q + \frac{Q_{ii}}{2}\delta^2 + \nabla_i f(\boldsymbol{\alpha}^k)\delta - \sum_i (\boldsymbol{\alpha}^k) \tag{2.5.32}$$

Recall that matrix Q is a symmetric matrix, therefore the bold red parameters in (2.5.31) are equivalent and can therefore be added. In order to minimise (2.5.32), the derivative with regards to δ is calculated and set to 0.

$$\begin{aligned} 0 &= \frac{\Delta f(\boldsymbol{\alpha}^k + \delta e_i)}{\Delta \delta} \\ 0 &= Q_{ii}\delta + \nabla_i f(\boldsymbol{\alpha}^k) \\ \delta &= \frac{-\nabla_i f(\boldsymbol{\alpha}^k)}{Q_{ii}} \end{aligned} \tag{2.5.33}$$

However, it is required that $0 \leq \alpha_i^k + \delta_i \leq C$, therefore

$$0 \leq \alpha_i^k - \frac{\nabla f(\alpha_i^k)}{Q_{ii}} \leq C, \tag{2.5.34}$$

thus the update for α_i^k is

$$\alpha_{i*}^k = \min\left(\max\left(\alpha_i^k - \frac{\nabla f(\alpha_i^k)}{Q_{ii}}, 0\right), C\right) \tag{2.5.35}$$

One last constraint is to project the gradient into a feasible region. If α_i^k is very small or close to zero, then there is no need to update α_i^k. On the other hand if α_i^k is close to or lies on the upper bound, there is only the need to improve α_i^k by minimising its value, eliminating the need to do an update on α_i^k if the calculated gradient is negative. The projected gradient rule is given as follows:

$$\nabla^P f(\alpha_i^k) = \begin{cases} \nabla f(\alpha_i^k), & \text{if } 0 < \alpha_i^k < C \\ \min(0, \nabla f(\alpha_i^k)), & \text{if } \alpha_i^k \approx 0 \\ \max(0, \nabla f(\alpha_i^k)), & \text{if } \alpha_i^k \approx C, \end{cases} \tag{2.5.36}$$

where $\nabla_i^P f(\alpha_i^k)$ is the projected gradient. An update will therefore only occur on α_i if the $\nabla_i^P f(\alpha_i^k) \neq 0$. Finally, the weights for the support vectors can be updated using

$$w_i^{k+1} = w_i^k + (\alpha_{i*}^k - \alpha_i^k) y_i x_i \tag{2.5.37}$$

The full algorithm for DCD for a linear SVM is given below [54].

Algorithm 7 DCD Linear SVM

- Initially $\boldsymbol{\alpha} = \bar{0}$, $\boldsymbol{w} = \bar{0}$,
- G = Gradient, PG = Projected Gradient

while $\boldsymbol{\alpha}$ not optimised **do**
  for `i = random(1,2,3,..,m)` **do**
    $G = (\boldsymbol{\alpha}^T Q)_i - 1$
    $$PG = \begin{cases} G, & \text{if } 0 < \alpha_i < C \\ \min(0, G), & \text{if } \alpha_i \approx 0 \\ \max(0, G), & \text{if } \alpha_i \approx C \end{cases}$$
    if $|PG| \neq 0$ **then**
      $\alpha_i^* = \alpha_i$
      $\alpha_i = \min\left(\max\left(\alpha_i^k - \frac{PG}{Q_{ii}}, 0\right), C\right)$
      $\boldsymbol{w} = \boldsymbol{w} + (\alpha_i - \alpha_i^*) y_i x_i$
    end
  end
end

Algorithm 7 has an iteration cost ($\boldsymbol{\alpha^k}$ to $\boldsymbol{\alpha^{k+1}}$) of $O(m\bar{n})$, where $\bar{n}$ is the average amount of non zero elements per instance. The convergence of Algorithm 7 is also proved in [54].

The training speed of the DCD method creates a suitable environment for fast recursive feature selection, namely SVM-RFE. SVM-RFE is a backwards feature elimination method which is able to remove irrelevant features. Furthermore, the DCD algorithm can be adapted to accommodate non linear data as well. Thus far only linearly separable datasets have been considered, but what if the data cannot be separated linearly. To overcome this problem, the kernel trick [57] is introduced. Because the dual formulation only depends on the dot product of the sample space, it is possible to map the sample space into a higher dimensional space, and apply the same mathematics. Consider the following non-linearly separable data shown in Figure 2.8 below.

It is impossible to separate the data in Figure 2.8 in a linear fashion. Suppose the data is transformed into a non-linear feature space with a higher dimension. Consider the transform from a 2-dimensional input space to a 3-dimensional feature space given by

$$\Phi(\boldsymbol{x}) = \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}, \tag{2.5.38}$$

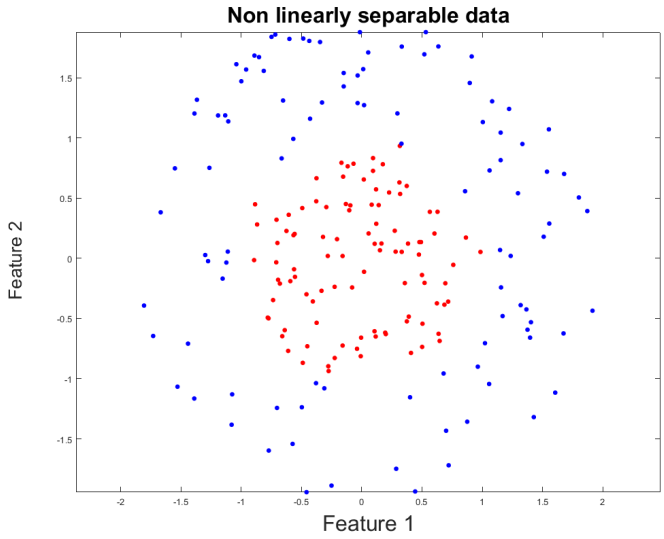


Figure 2.8: Non-linearly separable data example

where $\Phi(\boldsymbol{x})$ is the transformed feature space. Applying (2.5.38) to the non-linearly separable data shown in Figure 2.8 creates a scenario where the data is once again linearly separable through a plane, as shown in Figure 2.9 below.

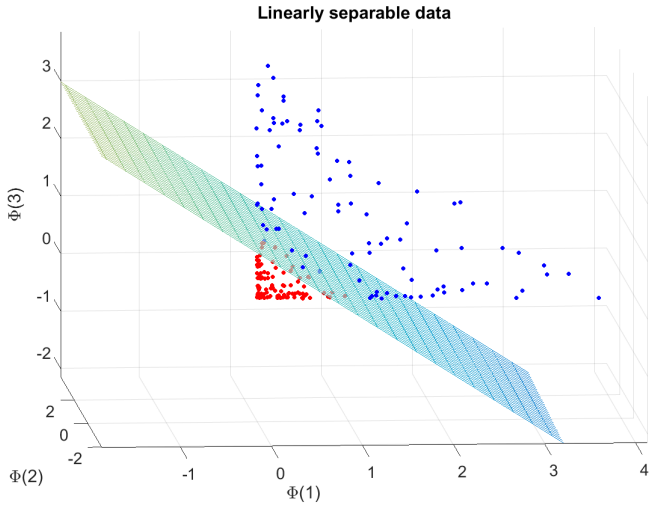


Figure 2.9: Transformed data which is now linearly separable

Recall that the Lagrangian dual in (2.5.27) is given by

$$\min_{\boldsymbol{\alpha}} L = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - 1^T \boldsymbol{\alpha} \tag{2.5.39}$$

Expanding (2.5.27) gives

$$\begin{aligned}\min_{\boldsymbol{\alpha}} L &= \frac{1}{2}\boldsymbol{\alpha}^T AA^T\boldsymbol{\alpha} - 1^T\boldsymbol{\alpha} \\ \min_{\boldsymbol{\alpha}} L &= \frac{1}{2}\boldsymbol{\alpha}^T(\boldsymbol{y}X)^T\boldsymbol{y}X\boldsymbol{\alpha} - 1^T\boldsymbol{\alpha} \\ \min_{\boldsymbol{\alpha}} L &= \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{y}^T(X^TX)\boldsymbol{y}\boldsymbol{\alpha} - 1^T\boldsymbol{\alpha}\end{aligned} \tag{2.5.40}$$

If the input space is mapped to a higher dimension (2.5.40) becomes

$$\min_{\boldsymbol{\alpha}} L = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{y}^T(\Phi(X)^T\Phi(X))\boldsymbol{y}\boldsymbol{\alpha} - 1^T\boldsymbol{\alpha}, \tag{2.5.41}$$

where $\Phi(\boldsymbol{X})$ is the transformation of dataset X containing all samples $\boldsymbol{x_i}$ into a higher feature space. Assume X is a dataset with 2-dimensional features, therefore $X = \{\boldsymbol{x_1}, \boldsymbol{x_2}\}$, then $\phi(X) = \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$. Assume Z is also a dataset with 2-dimensional features, therefore $\phi(Z) = \{z_1^2, \sqrt{2}z_1z_2, z_2^2\}$. Refer to (2.5.41), notice that the optimisation depends solely on the dot product in the feature space. Taking the dot product between X and Z gives

$$\begin{aligned}\Phi(X)^T\Phi(Z) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T(z_1^2, \sqrt{2}z_1z_2, z_2^2) \\ &= (x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2) \\ &= (x_1z_1 + x_2z_2)^2 \\ &= (X^TZ)^2\end{aligned} \tag{2.5.42}$$

Using (2.5.42) above,

$$\Phi(X)^T\Phi(X) = (X^TX)^2, \tag{2.5.43}$$

and formally,

$$K(X, X^T) = (X^TX)^2, \tag{2.5.44}$$

where K is referred to as the kernel. Equation (2.5.41) can be formally written as

$$\min_{\boldsymbol{\alpha}} L = \frac{1}{2}\boldsymbol{\alpha}^T Y^T K(X, X^T) Y\boldsymbol{\alpha} - 1^T\boldsymbol{\alpha}, \tag{2.5.45}$$

Notice the optimisation problem can be solved using a transformed feature space without having to ever calculate, or know how to calculate the transformation $\Phi(X)$. Only the kernel needs to be calculated, which simply involves the dot product of the original feature space X. This trick is referred to as the kernel trick. Equation (2.5.44) is an example of a homogeneous polynomial kernel. Table 2.4 shows other popular pre-defined kernels.

Table 2.4: Two common kernels widely used

kernel type	Inner product
polynomial	$K(X, X^T) = (X^T X + c)^d$
Gaussian (RBF)	$K(X, X^T) = e^{-\frac{1}{2\sigma^2}\|\|X - X^T\|\|^2}$

Choosing the correct kernel in practice is mostly done via experimentation with different kernels, adjusting their parameters until the classification error of the test set is optimal. A kernel is a similarity metric between the inputs, and therefore has to adhere to some properties in order to be a kernel function. A kernel function requires the following properties

- The kernel $K(\boldsymbol{x_1}, \boldsymbol{x_2})$ must be symmetrical.
- The kernel must satisfy the Cauchy-Schwarz inequality.
- The kernel must satisfy Mercer's Theorem.

For a short summary on the explanation of the above properties, refer to [57].

2.5.4 Implementing SVM feature selection

Returning to the feature selection problem, refer back to Algorithm 7, considering linearly separable data once again. The weight vector $\boldsymbol{w}$ is a linear combination of $\boldsymbol{x_i}$ and α_i as shown in (2.5.13). After $\boldsymbol{\alpha}$ has been solved, a ranking score [58] for feature p can be defined as:

$$C_p = w_p^2 \tag{2.5.46}$$

Equation (2.5.46) is intuitive, as the weight with the lowest value will contribute the least towards a classification. The feature corresponding to the lowest ranking point is then removed and appended to a feature ranking list. The SVM is retrained with the remaining features, and once again the feature with the lowest ranking score is removed and appended to a feature ranking list. This is repeated until no more features remain. Algorithm 8 shows how SVM-RFE ranks features for a linearly separable dataset.

Algorithm 8 Linear SVM-RFE

- Initially the selected feature subset $\boldsymbol{S} = \{1, ..., m\}$
- The ranked feature list $\boldsymbol{R} = \emptyset$
- Dataset $X = \{\boldsymbol{x_1}, ...\boldsymbol{x_m}\}$
- Class labels $y = \{y_1, ...y_m\}$

1: **while** $\boldsymbol{S} \neq \emptyset$ **do**
2: $X = X(:, S)$
3: $\boldsymbol{\alpha}$ =TrainSVM(X, y)
4: $\boldsymbol{w} = \sum_i^m \alpha_i y_i \boldsymbol{x_i}$
5: $C_i = w_i^2 \ \forall \ i = 1, ..., m$
6: $f = \arg\min \boldsymbol{C}$
7: $\boldsymbol{R} = [S(f), \boldsymbol{R}]$
8: $\boldsymbol{S} = \boldsymbol{S} - \{S(f)\}$
9: **end**

Refer to Algorithm 8 above. All features are ranked individually until no more features remain to be ranked (line 1). To rank the next feature, an SVM is trained on the remaining features in X (lines 2 - 3). The weights for each α_i is then calculated to create a score for each feature, of which the lowest scoring feature is ranked as the weakest and removed from the list of features which still require ranking (lines 3 - 8).

SVM-RFE can also be used for non-linear datasets using general kernels. The work presented in [59] used SVM-RFE for fault diagnosis, but instead of using a Linear kernel, the Gaussian kernel was used to train the SVM. The σ parameter for the Gaussian distribution was selected using the median method proposed in [60], which showed the most promising results between a 5-fold cross validation iterative search and the novel method proposed in [61]. Using the median method, the optimal σ is selected using

$$\sigma = median(\sqrt{||\boldsymbol{x_i} - \bar{\boldsymbol{x}}||^2}), \ i = 1, ..., n, \tag{2.5.47}$$

where $\bar{\boldsymbol{x}}$ is the mean of all training samples.

Given the cost function as in (2.5.27), the change in L caused by removing a feature i can be computed as

$$C_i = L - L(-i)$$
$$C_i = \frac{1}{2}\boldsymbol{\alpha} Q \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha} Q(-i)\boldsymbol{\alpha}, \tag{2.5.48}$$

The cost C_i requires a SVM classifier to retrain for each candidate feature to be removed. The work in [62] shows that this can be avoided by assuming no change in alpha, and therefore a re-computation of only matrix Q is necessary to obtain the rank of the least relevant feature. SVM-RFE for the general kernel case is shown in Algorithm 9.

Algorithm 9 General SVM-RFE

- Initially the selected feature subset $\boldsymbol{S} = \{1, ..., m\}$
- The ranked feature list $\boldsymbol{R} = \emptyset$
- Dataset $X = \{\boldsymbol{x_1}, ...\boldsymbol{x_m}\}$
- Class labels $y = \{y_1, ...y_m\}$

1: **while** $\boldsymbol{S} \neq \emptyset$ **do**
2: ...
3: ...
4: $C_i = \frac{1}{2}\boldsymbol{\alpha} Q \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha} Q(-i)\boldsymbol{\alpha}\ \forall\ i = 1, ..., m$
5: $f = \arg\min \boldsymbol{C}$
6: ...
7: ...
8: ...
9: **end**

Lines 4 - 5 in Algorithm 9 replaces lines 4 - 6 in Algorithm 8. Notice that there is no need to calculate $\boldsymbol{w}$ anymore, which is especially useful for the kernel space as $\Phi(x_i)$ is not required. For further speed-up considerations, matrix $Q\ \forall\ i$ does not have to be recomputed for each iteration, instead, the new Q matrix ($Q(-i)$) can be calculated by subtracting Q_i from Q. This means that the full matrix Q can be calculated beforehand and saved in memory. Furthermore, the previously calculated α_i can be the new initial values when solving for α_{i+1} in the next iteration.

Thus far, all SVM algorithms implemented are only applied to binary class datasets. For multi-class datasets, a variety of methods can be considered. The most popular being the one-against-one and one-against-all methods. There are, however, methods which involve solving a multi-class SVM in one step, called all-together methods. The work is [63] shows that the one-against-one outperforms the other methods in computation time and classification accuracy for large problems. The one-against-one method constructs $\frac{k(k-1)}{2}$ classifiers, where k is the number of classes. Each classifier is trained on the the data with two classes. A simple voting strategy can be used to determine which class a

sample belongs to:

$$sign((w^{ij})^T \boldsymbol{x} + b^{ij}), \tag{2.5.49}$$

where $\boldsymbol{x}$ is the sample. If $\boldsymbol{x}$ belongs to the i^{th} class, then a vote for class i is incremented by one, and vice versa for class j. The class belonging to x is predicted by the class with the majority of the votes. The strategy is called "Max Wins" and can be used to determine the importance of features with a new ranking score of

$$C_i = \sum_k^n (\frac{1}{2}\boldsymbol{\alpha_k} Q_k \boldsymbol{\alpha_k} - \frac{1}{2}\boldsymbol{\alpha_k} Q_k(-i)\boldsymbol{\alpha_k}), \tag{2.5.50}$$

where n is the amount of classifiers required.

For feature selection, a Gaussian RFE-SVM with L_2-norm is proposed using the DCD algorithm to solve the SVM. The median method is used to calculate σ for the Gaussian kernel, and the "one-vs-one" method is used for datasets with more than two classes.

2.6 Regression

When a dataset contains many noisy samples of data points with complex relationships in $\mathbb{R}^n$ dimensions, it is useful to approximate a model for the given data in order to predict new or unseen data. The approximation is essentially the line of best fit which is an estimate over the real data. The line will approximate the data points by assigning weight values to each dimension so that the total error with regards to the original data points are a minimum.

2.6.1 OLS regression

Ordinary Least Squares (OLS) regression is among the most basic types of regression which is only suitable for linearly dependent data. The fundamentals of OLS need to be understood as it is the foundation to other, more complicated regression techniques used for data with higher dimensions and non-linear dependencies. Assume dataset D consists of two variable ω_1 and ω_2. Figure 2.10 below shows an example of random data within X, which shows a linear dependency between the variables x and y. The green line is a candidate line of best fit. The error ϵ_i between a single data point d_i and the

line is the difference in the y axis. Extending this error onto the x axis creates a square shape where each side is equivalent to the error ϵ_i, as shown in Figure 2.10.

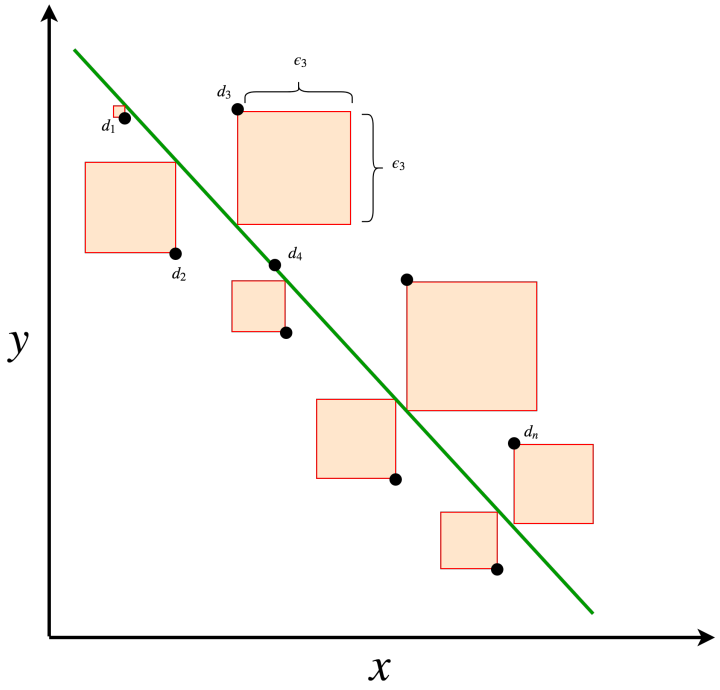


Figure 2.10: Ordinary Least Squares regression on two dimensions

The area of the square is known as the square error, hence the name of OLS regression. The sum of all square errors need to be minimised in order to find the line of best fit. The estimation of a data point using the line of best fit is given as

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \tag{2.6.1}$$

where y_i is simply a straight line equation and ϵ_i is the square error between the real data point and the estimation of y_i. The parameters β_0 and β_1 are the y intercept and gradient parameters respectively, which need to be determined in order to create the line of best fit. Rearranging (2.6.1) to make ϵ_i the subject yields

$$\epsilon_i = y_i - \beta_0 - \beta_1 x_i. \tag{2.6.2}$$

To get the sum of all square errors, the L_2-norm of all ϵ_i needs to be squared, therefore (2.6.2) becomes

$$||\epsilon||_2^2 = \sum_i (y_i - \beta_0 - \beta_1 x_i)^2 \tag{2.6.3}$$

The parameter $||\epsilon||_2^2$ will be denoted by E for simplicity. Recall that the sum of all square errors need to be minimised to obtain the line of best fit, therefore the optimisation problem is given as

$$\min_{\beta_0,\beta_1} E = \min_{\beta_0,\beta_1} \sum_i (y_i - \beta_0 - \beta_1 x_i)^2 \tag{2.6.4}$$

The vector notation of (2.6.4) for any given dimension is given as

$$\min_{\beta} E = \min_{\beta} || \begin{bmatrix} y_1 \\ . \\ . \\ y_n \end{bmatrix} - \begin{bmatrix} 1 & x_1 \\ . & . \\ . & . \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ . \\ . \\ \beta_n \end{bmatrix} ||_2^2 \tag{2.6.5}$$

$$\min_{\beta} E = \min_{\beta} ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2, \tag{2.6.6}$$

where $\boldsymbol{y}$ is the output, X is the data input matrix and $\boldsymbol{\beta}$ is the coefficients matrix. To minimise (2.6.4), the partial derivatives of β_0 and β_1 are obtained and set to 0 in (2.6.7) and (2.6.8) below.

$$\frac{\delta E}{\delta \beta_0} = -2 \sum_i (y_i - \beta_0 - \beta_1 x_i) = 0 \tag{2.6.7}$$

$$\frac{\delta E}{\delta \beta_1} = -2 \sum_i (y_i - \beta_0 - \beta_1 x_i)(x_i) = 0 \tag{2.6.8}$$

Expanding (2.6.7) and simplifying yields

$$-2(\sum_i y_i - \sum_i \beta_0 - \sum_i \beta_1 x_i)) = 0 \tag{2.6.9}$$

$$\sum_i y_i - n\beta_0 - \sum_i \beta_1 x_i = 0 \tag{2.6.10}$$

$$n\boldsymbol{y} - n\beta_0 - n\beta_1\boldsymbol{x} = 0 \tag{2.6.11}$$

$$\boldsymbol{y} - \beta_0 - \beta_1\boldsymbol{x} = 0 \tag{2.6.12}$$

$$\beta_0 = \boldsymbol{y} - \beta_1\boldsymbol{x} \tag{2.6.13}$$

Expanding (2.6.8) and simplifying yields

$$-2(\sum_i y_i x_i - \sum_i \beta_0 x_i - \sum_i \beta_1 x_i^2) = 0 \tag{2.6.14}$$

$$\sum_i y_i x_i - \beta_0 n\boldsymbol{x} - \beta_1 \sum_i x_i^2 = 0 \tag{2.6.15}$$

Substitute (2.6.13) in (2.6.15) and re-arranging:

$$0 = \sum_i y_i x_i - (\boldsymbol{y} - \beta_1\boldsymbol{x})n\boldsymbol{x} - \beta_1 \sum_i x_i^2 \tag{2.6.16}$$

$$\beta_1(\sum_i x_i^2 - n\boldsymbol{x}^2) = \sum_i y_i x_i - n\boldsymbol{y}\boldsymbol{x} \tag{2.6.17}$$

$$\beta_1 = \frac{\sum_i y_i x_i - n\boldsymbol{y}\boldsymbol{x}}{\sum_i x_i^2 - n\boldsymbol{x}^2} \tag{2.6.18}$$

The beta coefficients can now easily be calculated to give the line of best fit.

2.6.2 Ridge regression

According to the Gauss-Markov Theorem, OLS is a Best Linear Unbiased Estimator (BLUE) which means that it shows low variance, and does not show bias in the calculated values for $\boldsymbol{\beta}$ coefficients when repeating a solution multiple times. When more dimensions are added to a problem, say three or more $\boldsymbol{\beta}$ coefficients which are highly correlated, OLS regression tends to give large values to its $\boldsymbol{\beta}$ coefficients and also shows a high variance in coefficient values. Ideally, a low variance is wanted so that results can be replicated and a stable answer is returned for the same problem. To overcome this high variance, a low variance is enforced by constraining the $\boldsymbol{\beta}$ coefficients to bounded values subject to some constant C so that

$$||\boldsymbol{\beta}||_2^2 \leq C^2 \tag{2.6.19}$$

Recall that the L_2-norm of a coordinate vector is the distance from the origin to the point defined by the coordinate vector such that

$$||\boldsymbol{\beta}||_2 = \sqrt{\beta_0^2 + \beta_1^2 + ... + \beta_n^2} \tag{2.6.20}$$

$$\therefore ||\boldsymbol{\beta}||_2^2 = \beta_0^2 + \beta_1^2 + ... + \beta_n^2 \tag{2.6.21}$$

Figure 2.11 below illustrates the bound by C graphically using a two dimensional coordinate system, where C is essentially the allowable length of the L_2-norm vector. It is clear to see that the bounded values for $\boldsymbol{\beta}$ lie within a circle, whereby any solution outside the circle is unacceptable.

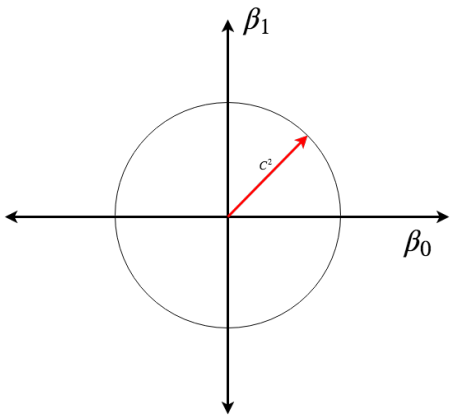


Figure 2.11: L_2-norm constraint on $\boldsymbol{\beta}$ coefficients in two dimensions

The acceptable coefficient values to the circle in Figure 2.11 is given as

$$\beta_0^2 + \beta_1^2 \leq C^2 \tag{2.6.22}$$

The addition of the L_2-norm bound on the $\boldsymbol{\beta}$ coefficients is called Ridge regression, and is defined by

$$\min_{\boldsymbol{\beta}} E = \min_{\boldsymbol{\beta}} ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2 \tag{2.6.23}$$

$$\text{s.t } ||\boldsymbol{\beta}||_2^2 \leq C^2$$

Figure 2.12 below illustrates how ridge regression restricts the solution of the $\boldsymbol{\beta}$ coefficients

according to an elliptic level curve solution space, where the center of the ellipse is the optimal solution for the $\boldsymbol{\beta}$ coefficients. The optimal solution cannot be reached due to the bounding circle of C^2, therefore the best solution lies on the level curve closest to the optimal solution. In this case, the optimal solution is where the level curve meets the ridge of the bounding circle denoted by the red dot. The bounding circle avoids large differences in $\boldsymbol{\beta}$ and also allows for solutions with low variance. It can be seen that coefficient β_0 contributes more to the solution than coefficient β_1, which emphasises β_0 is of greater importance. This interpretation of coefficient weights can be used for feature selection, but a sparse solution is more suitable which drives a coefficient towards zero instead of a real value.

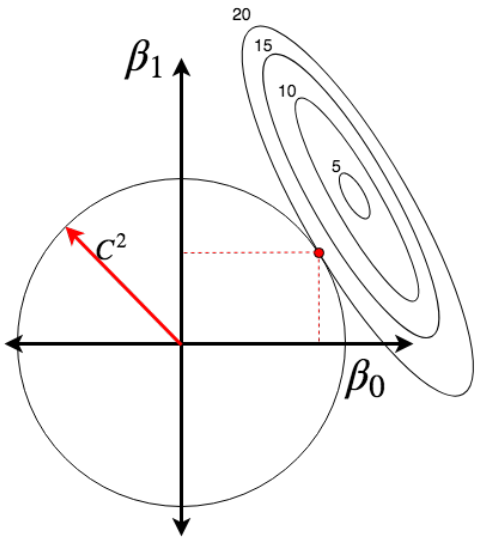


Figure 2.12: L_2-norm constraint and its effect on the optimal solution

2.6.3 Lasso regression

To create a sparse solution, L_1-norm can be used instead of L_2-norm. L_1-norm works by summing the the absolute value of the coefficients such that

$$||\boldsymbol{\beta}||_1 = |\beta_0| + |\beta_1| + ... + |\beta_n| \tag{2.6.24}$$

For clarity, the two dimensional representation of L_1-norm bounded by C is given by

$$|\beta_0| + |\beta_1| \leq C \tag{2.6.25}$$

Figure 2.13 below illustrates the difference between norms L_1 and L_2 both bounded by C.

(a) L_2-norm **(b)** L_1-norm

Figure 2.13: Differences between L_2 and L_1-norms

The use of L_1-norm instead of L_2-norm is called Lasso regression and is defined by

$$\min_{\boldsymbol{\beta}} E = \min_{\boldsymbol{\beta}} ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2 \quad (2.6.26)$$
$$\text{s.t } ||\boldsymbol{\beta}||_1 \leq C$$

Figure 2.14 below shows how a sparse solution can be determined using Lasso regression. The sharp corners of the diamond shape are most likely to cross the solution space first, driving the other coefficient to 0, which is ideal for feature selection because all coefficients with a value of 0 can be removed without further consideration.

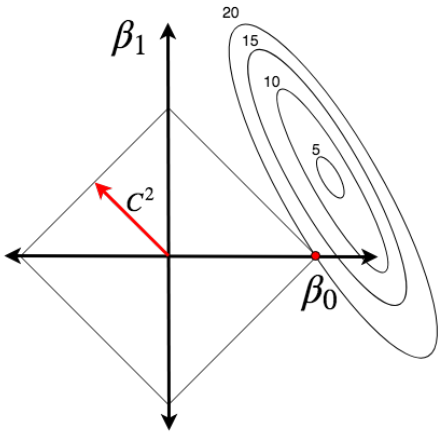
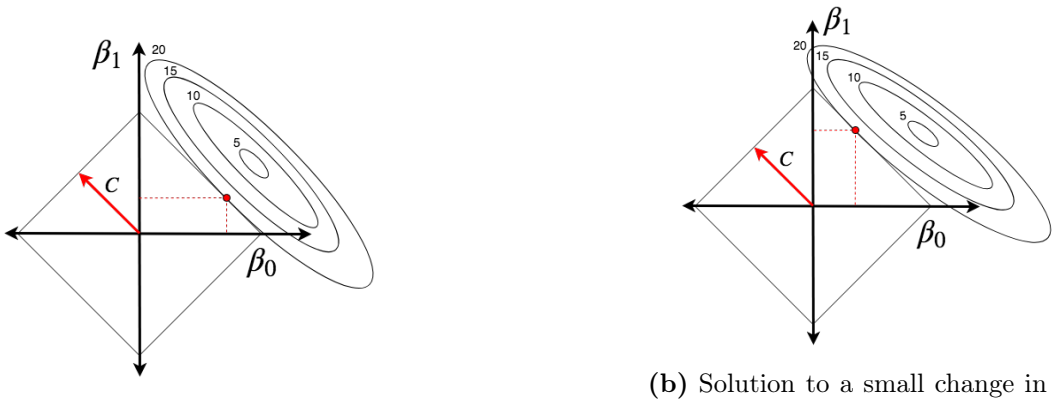


Figure 2.14: L_1-norm constraint and its effect on the optimal solution

The sparse contribution of L_1-norm is ideal, but it comes at a cost of stability. Figure 2.15 below illustrates this instability by comparing two scenarios. Small perturbations in

data can have a drastic effect on the coefficient weights and ultimately lead to a different solution.



(a) Original solution

(b) Solution to a small change in the original data

Figure 2.15: Instability of L_1-norm

2.6.4 Elastic Net regression

To overcome this instability, a combination of L_1-norm and L_2-norm is necessary. The addition of both norms is called Elastic Net regression. Figure 2.16 below shows a graphical illustration of the solution space when combining both norms.

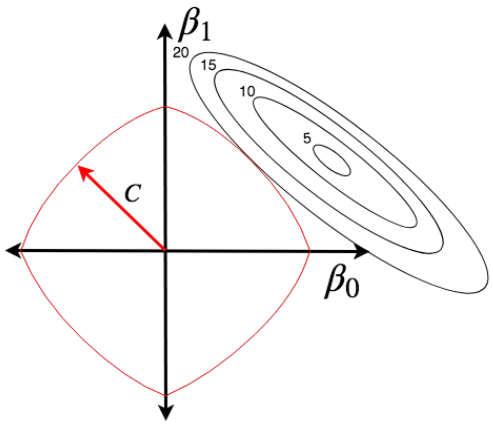


Figure 2.16: A combination of L_1 and L_2-norms on the solution space in two dimensions

Notice that there is still the possibility for a sparse solution, and that small perturbations in data now gives a more stable result due to the bulges included by L_2-norm.The equation for Elastic Net regression is therefore given as

$$\min_{\boldsymbol{\beta}} E = \min_{\boldsymbol{\beta}} ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2 \tag{2.6.27}$$

$$\text{s.t } ||\boldsymbol{\beta}||_1 \leq C$$
$$\text{s.t } ||\boldsymbol{\beta}||_2^2 \leq C^2$$

Whenever dealing with a minimisation problem subject to a constraint, the Lagrangian expansion is always helpful. The Lagrangian expansion of (2.6.27) is given as

$$L(\boldsymbol{\beta}, \lambda_1, \lambda_2) = ||\boldsymbol{y} - A\boldsymbol{\beta}||_2^2 + \lambda_1(||\boldsymbol{\beta}||_1 - C) + \lambda_2(||\boldsymbol{\beta}||_2^2 - C^2) \tag{2.6.28}$$

Equation (2.6.28) needs to be minimised in order to find the best $\boldsymbol{\beta}$ values, as well as the values for lambda. Notice that the minimisation of L does not include parameter C. The current convention for solving this equation is to try a random λ value to find the best $\boldsymbol{\beta}$ coefficients, therefore the minimisation problem can be re-written as

$$\min_{\boldsymbol{\beta}} E = \min_{\boldsymbol{\beta}} ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2 + \lambda_1||\boldsymbol{\beta}||_1 + \lambda_2||\boldsymbol{\beta}||_2^2 \tag{2.6.29}$$

Typically $\lambda_1 + \lambda_2 = 1$ so that the contribution of the L_1 and L_2-norms can be specified, i.e if $\lambda_1 = 1$ then $\lambda_2 = 0$ which is the same as Lasso regression. Similarly, if $\lambda_1 = 0.5$ then $\lambda_2 = 0.5$ which suggests that both L_1 and L_2-norms are considered equally. The simplified Elastic Net equation can therefore be written as

$$\min_{\boldsymbol{\beta}} E = \min_{\boldsymbol{\beta}} ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 + (1-\lambda)||\boldsymbol{\beta}||_2^2 \tag{2.6.30}$$

Thus far, all regression techniques discussed provide the best *linear* solution. Fortunately there is a way to perform non-linear regression by introducing the kernel trick once more. Before applying the kernel trick, Ridge regression can be reformulated into its dual representation [64]. The minimisation for Ridge regression follows (2.6.30) but without the L_1-norm term, so that

$$\min_{\boldsymbol{\beta}} E = \min_{\boldsymbol{\beta}} ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2,$$

which can be re-expressed as

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{k} (y_i - \boldsymbol{\beta}\boldsymbol{x_i})^2 + \lambda ||\boldsymbol{\beta}||_2^2 \tag{2.6.31}$$

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{k} \xi_i^2 + \lambda ||\boldsymbol{\beta}||_2^2, \tag{2.6.32}$$

such that $\xi_i = y_i - \boldsymbol{\beta}\boldsymbol{x_i}$. Treating ξ_i as a constraint, the Lagrange equation to (2.6.32) is given as

$$L(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \sum_{i=1}^{k} \xi_i^2 + \lambda ||\boldsymbol{\beta}||_2^2 + \sum_{i=1}^{k} \alpha_i (y_i - \boldsymbol{\beta}\boldsymbol{x_i} - \xi_i) \tag{2.6.33}$$

$$L(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \sum_{i=1}^{k} \xi_i^2 + \lambda ||\boldsymbol{\beta}||_2^2 + \sum_{i=1}^{k} \alpha_i y_i - \sum_{i=1}^{k} \alpha_i \boldsymbol{\beta}\boldsymbol{x_i} - \sum_{i=1}^{k} \alpha_i \xi_i, \tag{2.6.34}$$

where the $\boldsymbol{\alpha}$ coefficients are the Lagrange multipliers. Because the problem is a minimisation problem, the partial derivatives of $\boldsymbol{\beta}$, $\boldsymbol{\xi}$ and $\boldsymbol{\alpha}$ need to be calculated and set to 0. The partial derivative of $\boldsymbol{\beta}$ is given as

$$\frac{\delta \boldsymbol{\beta}}{\delta L} = 2\lambda\boldsymbol{\beta} - \sum_{i}^{k} \alpha_i \boldsymbol{x_i} \tag{2.6.35}$$

$$0 = 2\lambda\boldsymbol{\beta} - \sum_{i}^{k} \alpha_i \boldsymbol{x_i} \tag{2.6.36}$$

$$\boldsymbol{\beta} = \frac{1}{2\lambda} \sum_{i}^{k} \alpha_i \boldsymbol{x_i} \tag{2.6.37}$$

Substituting $\boldsymbol{\beta}$ into (2.6.34) gives

$$L(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \sum_{i=1}^{k} \xi_i^2 + \lambda || \frac{1}{2\lambda} \sum_{i}^{k} \alpha_i \boldsymbol{x_i} ||_2^2 + \sum_{i=1}^{k} \alpha_i y_i - \sum_{i=1}^{k} \alpha_i (\frac{1}{2\lambda} \sum_{i}^{k} \alpha_i \boldsymbol{x_i}) \boldsymbol{x_i} - \sum_{i=1}^{k} \alpha_i \xi_i \tag{2.6.38}$$

$$= \sum_{i=1}^{k} \xi_i^2 + \frac{1}{4\lambda} \sum_{i,j}^{k} \alpha_i \alpha_j (\boldsymbol{x_i} \cdot \boldsymbol{x_j}) + \sum_{i=1}^{k} \alpha_i y_i - \frac{1}{2\lambda} (\sum_{i=1}^{k} \alpha_i \boldsymbol{x_i}) \cdot (\sum_{i}^{k} \alpha_i \boldsymbol{x_i}) - \sum_{i=1}^{k} \alpha_i \xi_i \tag{2.6.39}$$

$$= -\frac{1}{4\lambda}\sum_{i,j}^{k}\alpha_i\alpha_j(\boldsymbol{x_i}\cdot\boldsymbol{x_j}) + \sum_{i=1}^{k}\xi_i^2 + \sum_{i=1}^{k}\alpha_i y_i - \sum_{i=1}^{k}\alpha_i\xi_i \tag{2.6.40}$$

Taking the partial derivative of ξ_i in (2.6.40) yields

$$\frac{\delta\xi}{\delta L} = 2\sum_{i}^{k}\xi_i - \sum_{i}^{k}\alpha_i \tag{2.6.41}$$

$$0 = 2\sum_{i}^{k}\xi_i - \sum_{i}^{k}\alpha_i \tag{2.6.42}$$

$$\sum_{i}^{k}\xi_i = \frac{1}{2}\sum_{i}^{k}\alpha_i \tag{2.6.43}$$

Notice that α_i and ξ_i are proportional, therefore

$$\xi_i = \frac{1}{2}\alpha_i \tag{2.6.44}$$

Substituting ξ_i into (2.6.40) gives

$$L(\boldsymbol{\beta},\boldsymbol{\xi},\boldsymbol{\alpha}) = -\frac{1}{4\lambda}\sum_{i,j}^{k}\alpha_i\alpha_j(\boldsymbol{x_i}\cdot\boldsymbol{x_j}) + \frac{1}{4}\sum_{i=1}^{k}\alpha_i^2 + \sum_{i=1}^{k}\alpha_i y_i - \frac{1}{2}\sum_{i=1}^{k}\alpha_i^2 \tag{2.6.45}$$

$$= -\frac{1}{4\lambda}\sum_{i,j}^{k}\alpha_i\alpha_j(\boldsymbol{x_i}\cdot\boldsymbol{x_j}) + \sum_{i=1}^{k}\alpha_i y_i - \frac{1}{4}\sum_{i=1}^{k}\alpha_i^2 \tag{2.6.46}$$

Finally, differentiating with respect to $\boldsymbol{\alpha}$ yields

$$\begin{aligned}
\frac{\delta\alpha}{\delta L} &= -\frac{1}{2\lambda}(\boldsymbol{x_i}\cdot\boldsymbol{x_j})\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha} + \boldsymbol{y} \\
0 &= -\frac{1}{2\lambda}(\boldsymbol{x_i}\cdot\boldsymbol{x_j})\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha} + \boldsymbol{y} \\
-\boldsymbol{y} &= -\frac{1}{2\lambda}(\boldsymbol{x_i}\cdot\boldsymbol{x_j})\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha} \\
\boldsymbol{y} &= \boldsymbol{\alpha}(\frac{1}{2\lambda}(\boldsymbol{x_i}\cdot\boldsymbol{x_j}) + \frac{1}{2}) \\
2\lambda\boldsymbol{y} &= \boldsymbol{\alpha}((\boldsymbol{x_i}\cdot\boldsymbol{x_j}) + \lambda) \\
\boldsymbol{\alpha} &= 2\lambda((\boldsymbol{x_i}\cdot\boldsymbol{x_j}) + \lambda I)^{-1}\boldsymbol{y}
\end{aligned} \tag{2.6.47}$$

$$\boldsymbol{\alpha} = 2\lambda(K(\boldsymbol{x_i}, \boldsymbol{x_j}) + \lambda I)^{-1}\boldsymbol{y}, \tag{2.6.48}$$

where $K(\boldsymbol{x_i}, \boldsymbol{x_j})$ is a kernel function which maps X to a higher dimensional feature space. The RBF or polynomial kernels can be used to do kernelised Ridge regression for non-linear datasets. Recall that $\boldsymbol{\beta}$ can now be easily computed by using (2.6.37) defined by

$$\boldsymbol{\beta} = \frac{1}{2\lambda}\sum_{i}^{k}\alpha_i \boldsymbol{x_i}$$

Kernelised regression can also be expanded to Lasso and Elastic Net [65] for the advantage of a sparse solution, though the mathematics are not as elegant due to the addition of L_1-norm, therefore closed-form expressions such as (2.6.48) are not as simple.

2.6.5 Logistic regression

Thus far, the regression techniques discussed assumes real-valued data. For feature selection to be effective for categorical classes, a different regression technique is necessary, namely Logistic regression. Logistic regression calculates the probability of an outcome based on specific inputs. Suppose dataset X consists of n features and m samples, where subscripts denote the sample and superscripts denote the feature such that $X = \{x_i^1, x_i^2, x_i^3 ... x_m^n\}$. Given X with predictor $Y = \{0; 1\}$, predicting the probability of the outcome of sample $\boldsymbol{x_i}$ is

$$p(\boldsymbol{x_i}) = P(Y = 1|X = \boldsymbol{x_i}) \tag{2.6.49}$$

The probability can be re-written as a linear combination of $\boldsymbol{\beta} \cdot \boldsymbol{x_i}$ such that

$$p(\boldsymbol{x_i}) = \beta_0 + \beta_1 x_i^1 + \beta_2 x_i^2 + ... + \beta_m x_i^m \tag{2.6.50}$$

Notice that the linear combination is unbounded, therefore values outside the range of [0, 1] does not agree with using the bounded probability of $p(\boldsymbol{x_i})$, i.e the linear sum of $\boldsymbol{\beta} \cdot \boldsymbol{x_i}$ values which tend to $[-\infty, \infty]$ have no meaning. In order to make $p(\boldsymbol{x_i})$ a bounded probability model, some manipulation with $ln(\cdot)$ is used to create the model

$$ln[\frac{p(\boldsymbol{x_i})}{1-p(\boldsymbol{x_i})}] = \beta_0 + \beta_1 x_i^1 + \beta_2 x_i^2 + ... + \beta_m x_i^m \tag{2.6.51}$$

It can be seen that if $p(\boldsymbol{x_i}) = 0$, then the model on the left tends towards $-\infty$ and when $p(\boldsymbol{x_i}) = 1$ then it tends towards ∞, therefore the model on the left is now also unbounded and can be used in conjunction with the linear combination of $\boldsymbol{\beta} \cdot \boldsymbol{x_i}$. Solving for $p(\boldsymbol{x_i})$ yields

$$p(\boldsymbol{x_i}) = \frac{1}{e^{-(\boldsymbol{\beta} \cdot \boldsymbol{x_i})} + 1} \tag{2.6.52}$$

Notice that the form of $p(\boldsymbol{x_i})$ has the shape of a sigmoid function, which is bounded between (0, 1). Recall that the above equation is a prediction of (2.6.49) where $Y = 1$. In order to determine the optimal $\boldsymbol{\beta}$ coefficients, the probability for $Y = 1$ to occur needs to be maximised and the probability for $Y = 0$ to occur needs to be minimised. A cost function according to the probabilities given a specific $\boldsymbol{\beta}$ is used to adhere to the above criteria and is given as

$$Cost(p_{\boldsymbol{\beta}}(\boldsymbol{x_i}), Y) = \begin{cases} -ln(p_{\boldsymbol{\beta}}(\boldsymbol{x_i})), & \text{if } Y = 1 \\ -ln(1 - p_{\boldsymbol{\beta}}(\boldsymbol{x_i})), & \text{if } Y = 0 \end{cases}$$

The summation of both cases provides the cost function for a single sample $\boldsymbol{x_i}$. Just like ordinary regression, the cost function needs to be minimised for all samples of $\boldsymbol{x_i}$, therefore the complete cost function including a L_2-norm regularisation parameter is

$$C(\boldsymbol{\beta}) = \frac{1}{m} \sum_i^m \Big(-(y_i) ln(p_{\boldsymbol{\beta}}(\boldsymbol{x_i})) - (1 - y_i) ln(1 - p_{\boldsymbol{\beta}}(\boldsymbol{x_i})) \Big) + \frac{\lambda}{2m} \sum_i^m \beta_i^2 \tag{2.6.53}$$

$$= -\frac{1}{m} \sum_i^m (y_i) ln(p_{\boldsymbol{\beta}}(\boldsymbol{x_i})) + (1 - y_i) ln(1 - p_{\boldsymbol{\beta}}(\boldsymbol{x_i})) + \frac{\lambda}{2m} ||\boldsymbol{\beta}||_2^2 \tag{2.6.54}$$

The optimal $\boldsymbol{\beta}$ coefficients can be calculated by using a SGD algorithm. Other methods such as Newton-Raphson can also be used, but for datasets where $p >> n$, the Hessian matrix becomes too large and computationally expensive, therefore SGD is the method of choice. The $\boldsymbol{\beta}$ coefficient updates requires the gradient of $\boldsymbol{\beta}$ with regards to the cost function to be calculated. The derivative of β_j according to the sigmoid function is

$$\frac{\delta ln(p(\boldsymbol{x_i}))}{\delta \beta_j} = \frac{\delta[ln(1) - ln(1 + e^{-\boldsymbol{\beta x_i}})]}{\delta \boldsymbol{\beta}} \tag{2.6.55}$$

$$= \frac{x_i^j e^{-\boldsymbol{\beta x_i}}}{1 + e^{-\boldsymbol{\beta x_i}}} \tag{2.6.56}$$

$$= x_i^j (1 - p(\boldsymbol{x_i})) \tag{2.6.57}$$

The derivative to $ln(1 - p(\boldsymbol{x_i}))$ can be obtained through the answer of (2.6.57), therefore

$$\frac{\delta ln(1 - p(\boldsymbol{x_i}))}{\delta \beta_j} = \frac{\delta[-\boldsymbol{\beta x_i} - ln(1 + e^{-\boldsymbol{\beta x_i}})]}{\delta \boldsymbol{\beta}} \tag{2.6.58}$$

$$= -x_i^j + x_i^j (1 - p(\boldsymbol{x_i})) \tag{2.6.59}$$

$$= -p(\boldsymbol{x_i}) x_i^j \tag{2.6.60}$$

The derivative to β_j for the entire cost function C is then derived as

$$\frac{\delta C}{\delta \beta_j} = -\frac{1}{m} \sum_i^m y_i x_i^j (1 - p(\boldsymbol{x_i})) - (1 - y_i) x_i^j p(\boldsymbol{x_i}) + \frac{\lambda}{m} \beta_j \tag{2.6.61}$$

$$= \frac{1}{m} \sum_i^m (p(\boldsymbol{x_i}) - y_i) x_i^j + \frac{\lambda}{m} \beta_j \tag{2.6.62}$$

$$= \frac{1}{m} \Big(\sum_i^m (p(\boldsymbol{x_i}) - y_i) x_i^j + \lambda \beta_j \Big) \tag{2.6.63}$$

$$A = \begin{bmatrix} x_1^1 & . & . & x_1^n \\ . & . & . & . \\ . & . & . & . \\ x_m^1 & . & . & x_m^n \end{bmatrix} \tag{2.6.64}$$

The simplified vector notation of (2.6.62) which calculates the gradient $\forall\ \boldsymbol{\beta}$ is given as

$$\frac{\delta C}{\delta \boldsymbol{\beta}} = -\frac{1}{m} A^T (p(X) - \boldsymbol{y}) \tag{2.6.65}$$

$$\nabla_{\boldsymbol{\beta}} C = A^T (p(X) - \boldsymbol{y}), \tag{2.6.66}$$

where $\nabla_{\boldsymbol{\beta}} C$ is the gradient of the cost function according to $\boldsymbol{\beta}$. When using SGD, the learning rate α captures $-\frac{1}{m}$ when using the update equation of

$$\boldsymbol{\beta}_{new} = \boldsymbol{\beta}_{old} - \alpha \nabla_{\boldsymbol{\beta}} C \tag{2.6.67}$$

The SGD algorithm to solve for binary linear logistic regression is given below in Algorithm 10.

Algorithm 10 Binary logistic regression

- Initialise:

X - Dataset with m samples and n features
Y - Class label vector
$MaxIte$ - Maximum number of gradient updates
α - Learning rate
λ - L_2-norm regression coefficient
tol - Early termination tolerance
$\boldsymbol{\beta} = \{\beta_0, \beta_1 ..., \beta_n\}$ - Randomly initialised at first
$preCost = 0$

1: **for** `k = 1,2,..,MaxIte` **do**
2: | **for each** $\beta_j \in \boldsymbol{\beta}$ **do**
3: | | $\nabla \beta_j = \sum_i^m (p(\boldsymbol{x_i}) - y_i) x_i^j + \lambda \beta_j$
4: | **end**
5: | $\nabla \boldsymbol{\beta} = \frac{\nabla \boldsymbol{\beta}}{m}$
6: | $\boldsymbol{\beta} = \boldsymbol{\beta} - \alpha \nabla \boldsymbol{\beta}$
7: | **if** $\frac{abs(Cost - preCost)}{Cost} \leq tol$ **then**
8: | | `break`
9: | **end**
10: | $preCost = Cost$
11: **end**

Algorithm 10 terminates if the the cost function tolerance is met, or if the maximum iterations for SGD is met (line 1). For each iteration, the gradient for each $\boldsymbol{\beta}$ coefficient is calculated in lines 2 - 5 and the coefficients are then updated in line 6. The cost for the updated $\boldsymbol{\beta}$ coefficients are then calculated and compared to the previous cost. If the difference in cost percentage is below a user defined threshold then the optimisation is complete and SGD is terminated (lines 7 - 9). If the difference in cost is unsuitable, the previous cost is assigned to the current cost and the next iteration for SGD starts (line 10 - 11).

The SGD algorithm for binary logistic regression is computationally efficient for large datasets. The disadvantage of the above algorithm is that it solves the regression for

linear relationships. To avoid this, the features in X can be extended to higher dimensions using kernel functions, but then the advantage of using regression is forfeited as the new $\boldsymbol{\beta}$ coefficients calculated for the new higher dimension features are no longer linked towards the original features. This forces a recursive search for the best subset of features instead of just examining the $\boldsymbol{\beta}$ coefficients.

Furthermore, if the linear relationship is kept then a sparse solution is ideal, but the above implementation is not based on the L_1-norm. Lastly, the above implementation is binary and can therefore only be applied on binary datasets. A multinomial sparse logistic regression solution is therefore required for an ideal feature selection technique.

For completeness, the method Sparse Multinomial Logistic Regression via Bayesian L_1 Regularisation (SBMLR) proposed by [38] will be used for feature selection. To justify the choice of SBMLR, ordinary Sparse Multinomial Logistic Regression (SMLR) optimises the L_1 regularisation parameter with a simple line search method [37] while SBMLR uses Bayesian regularisation based on Laplacian prior. SBMLR performs just as well as conventional SMLR, except that it provides a slightly higher degree of sparsity and is orders of magnitude faster regarding computation speed. SBMLR is well suited for datasets where $p >> n$ and is the embedded feature selection technique of choice for this study.

2.7 Brief chapter summary

Three filter methods mRMR, FCBF and OMICS were reformulated and explored. All three methods require associative measures to determine feature-class and feature-feature relevance.

For mRMR, *dCorr* seems the most promising. mRMR simply extracts features which exhibits the most relevance to the class label but least redundancy to the features already selected. The main disadvantage of mRMR is its inability to remove irrelevant redundancy.

FCBF tries to isolate predominant features using SU as its the main associative measure. A new search strategy is adopted to find more suitable predominant features when a predefined amount of features are wanted, and is called FCBF#.

OMICFS makes use of orthogonalisation to remove features which exhibit irrelevant redundancy. The associative measure used is a more recent statistical measure called MIC, but its performance is questioned. OMICFS is renamed to ORFS to incorporate additional associative measures for testing.

Two wrapper methods FRBPSO and SVM-RFE are also explored. FRBPSO is essentially BPSO feature selection with an additional layer of fuzzy intelligence to avoid local minima. Further improvements such as convergence speed, LDIW and global particle resetting is also implemented.

SVM-RFE is a recursive feature elimination method which trains multiple L_2-norm SVMs using a fast SGD algorithm to converge to a solution. The use of different kernels is explored and the ability to accompany multi-class datasets is accomplished through a "one-vs-one" approach.

For a logistic regressive feature selection algorithm, SBMLR is the embedded method of choice due to its sparse properties and ability to accompany multi-class datasets. SBMLR is however limited to a Linear kernel.

Chapter 3

Numerical experiments

This chapter deals with the simulations of each feature selection algorithm explained in Chapter 2. A total of 12 medically related datasets selected while exploring [1, 66] are used for testing. All datasets adhere to $p >> n$ and are assumed to have no prior knowledge. After each simulation, a brief discussion on the performance of the algorithm is presented. Thereafter, the performance of all algorithms are compared to one another, and scenarios best suited for each algorithm discussed.

Table 3.1 below summarises all 12 medically related gene expression datasets. All experiments are done on a workbench running on Windows 10 with 8 GB of RAM using an Intel® Core™ i7-3610QM CPU @2.3 GHz. The program of choice is Matlab™ 2016b.

Table 3.1: Datasets used in this study

Dataset name	Number of features	Number of samples	Classes
9_Tumors	5727	60	4
11_Tumors	12534	174	5
Brain_Tumor1	5921	90	9
Brain_Tumor2	10368	50	11
DLBCL	5470	77	4
GLI-85	22283	85	2
Leukemia1	5328	72	3
Leukemia2	11226	72	3
Lung_Cancer	12601	203	3
Prostate_Tumor	10510	102	2
SMK-CAN-187	19993	187	2
SRBCT	2309	83	2

The original classification accuracy per dataset using 1-NN LOOCV is given in Table 3.2 below. Each feature is pre-processed to their respective z-scores[1] before classification. The overall average accuracy and amount of features are compared with the results for each feature selection algorithm to determine their improvement.

Table 3.2: Original 1-NN LOOCV accuracies

Dataset	Accuracy (%)	# features
9_Tumors	41.67	5727
11_Tumors	74.71	12534
Brain_Tumor1	85.56	5921
Brain_Tumor2	60	10368
DLBCL	84.42	5470
GLI-85	88.24	22283
Leukemia1	83.33	5328
Leukemia2	86.11	11226
Lung_Cancer	87.68	12601
Prostate_Tumor	82.35	10510
SMK-CAN-187	63.64	19993
SRBCT	85.54	2309
Average	**76.94**	**10356**

3.1 Filter methods

The mRMR, FCBF and ORFS methods each have their own specialised experiments but share a few experiments in common. Each algorithm has the following choices:

[1] Conversion to a common scale with a mean of 0 and standard deviation of 1

- Incorporate an appropriate use of supervision
- Pre-processing data using SIS
- Pre-processing data using a discretisation filter

The SIS pre-processing method reduces the amount of features to the top 20 % according to *dCorr* scores obtained between each feature and the class variable. Calculating MI and using the discretisation filter makes use of Irani's MDL discretisation method. Each dataset adheres to $p >> n$, therefore there is a good chance that the discretisation result of any feature is repeated in other features, and all repeated features can be removed. There is also a chance that an entire feature is discretised into one single interval, and can thus be removed as well. To discretise data, WEKA™ v3.8 was used in conjunction with Matlab™. As for the supervision method, a 1-NN classifier using LOOCV from the Matlab™ Machine Learning toolbox is used to obtain classification accuracy. The use of supervision for each algorithm is discussed individually. Recall that the use of supervision is an attempt to include synergy with selected features.

From the aforementioned choices, a total of eight combinational choices exist and each combination is applied to 12 datasets which generates a minimum of 96 experiments per algorithm. Some algorithms include more experiments due to the use of additional associative measures. Due to the large number of experiments, each algorithm is optimised to make use of parallel processing on all four cores of the CPU. In each experiment, the computation time, best classification accuracy and accuracy gain per ranked feature is recorded. All features are normalised to their respective z-scores before further processing and only the top 100 ranked features are considered after the feature selection process. After a brief explanation of the results for each respective algorithm, their results are compared to one another and discussed.

3.1.1 mRMR

The common associative measure of MI will be compared to *dCorr*. Each associative measure contains its own 96 experiments, giving a total of 192 experiments. The supervised method of mRMR includes an additional test before accepting a feature. Once the next best feature is selected by the algorithm, the classification accuracy of all previously selected features including the currently selected feature is tested. If there is no improvement in classification accuracy, the feature is discarded from the list of best features, which also influences the selection of the next best feature in the mRMR algorithm. The pseudo code to mRMR including supervision and pre-processing is summarised in Algorithm 11 below.

Algorithm 11 Supervised mRMR

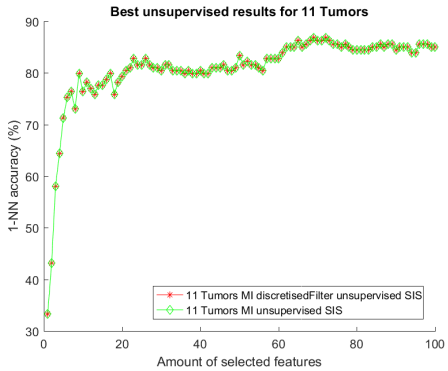
- Initialise:

Size of best feature subset $MaxIte$
$F_s = \emptyset$ - Subset of best features
$F_a = X(:,:)$ - Subset of remaining features
$s_v = \{0, 1\}$ - Enable or disable supervision assistance
$acc = 0$ - Classifier accuracy when using supervision
Y - Class label vector

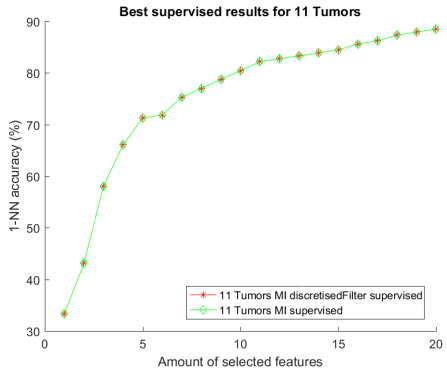
Pre-process F_a according to SIS and discretisation choices
Select relevance method
Pre-compute the relevance of each feature $Rel(i) = I(X(:,i);Y)\ \forall_i$
Find the first best feature $F_{best} = \max\ Rel(i)$

$maxAcc = 0$
$F_s = F_{best}$
for `k = 2,3,..,MaxIte` **do**
    $F_a = F_a - \{F_{best}\}$
    if `isempty(Fa)` **then**
        `break`
    end
    $F_{best} = \max_{x_i \in F_a} \left(I(x_i;Y) - \frac{1}{length(F_s)} \sum_{x_j \in F_s} I(x_i;x_j) \right)$
    if $s_v == 1$ **then**
        $acc =$ 1-NN$([F_s\ F_{best}], Y)$
        if acc `>` $maxAcc$ **then**
            $maxAcc = acc$
            $F_s = [F_s\ F_{best}]$
        end
    else
        $F_s = [F_s\ F_{best}]$
    end
end

The results for mRMR on all 12 datasets are given below. Figures 3.1 to 3.12 show the best performance for associative measures MI and $dCorr$ for each dataset. For readability, the results for unsupervised and supervised mRMR are given in the figures to the left and right of the page respectively. The figures are followed by Tables 3.3 to 3.6 which summarise the best outcome for each dataset, preferred associative measure, preferred filter method, overall average accuracy across all datasets and computation time for each best scenario.

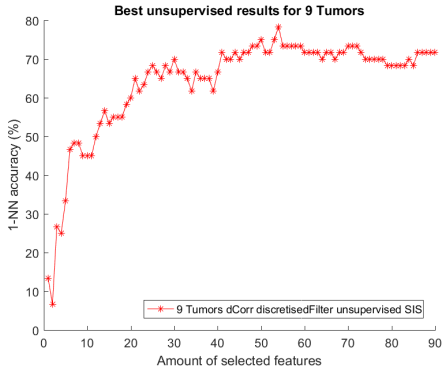


(a) Best unsupervised results

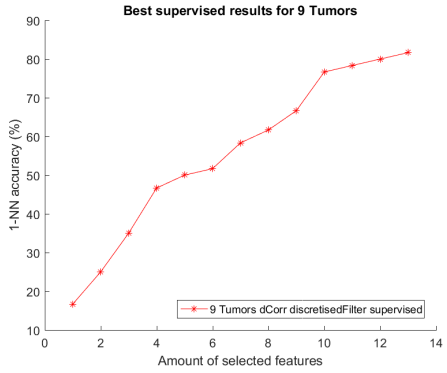


(b) Best supervised results

Figure 3.1: Best mRMR results for 11 Tumors

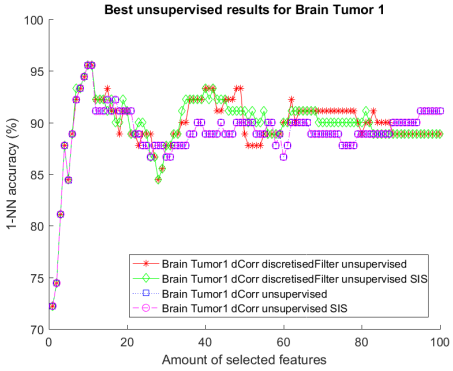


(a) Best unsupervised results

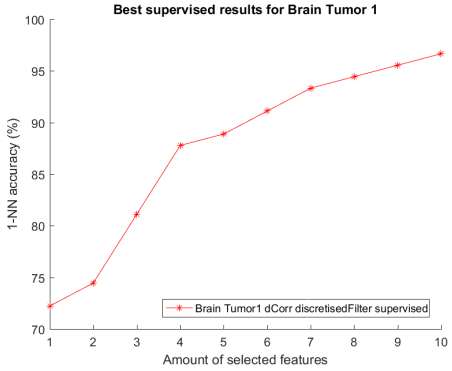


(b) Best supervised results

Figure 3.2: Best mRMR results for 9 Tumors

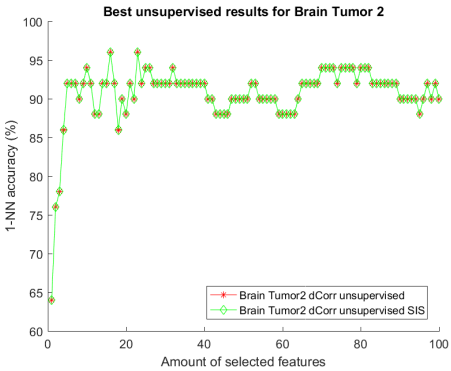


(a) Best unsupervised results

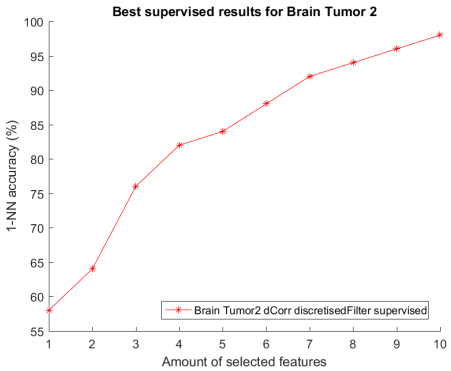


(b) Best supervised results

Figure 3.3: Best mRMR results for Brain Tumor 1

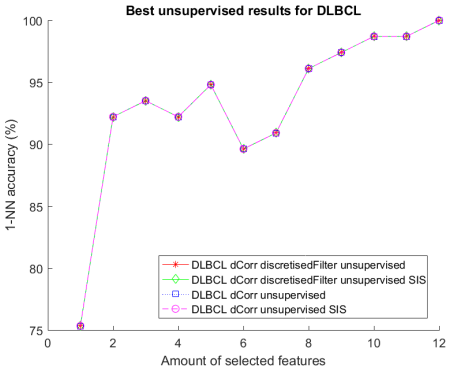


(a) Best unsupervised results

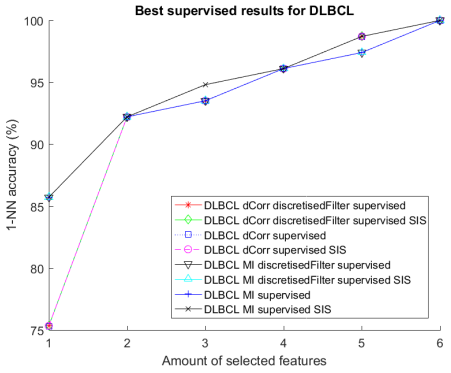


(b) Best supervised results

Figure 3.4: Best mRMR results for Brain Tumor 2

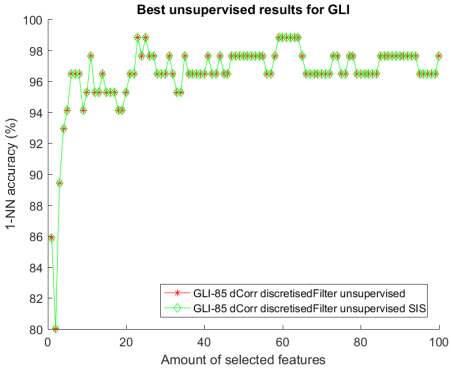


(a) Best unsupervised results

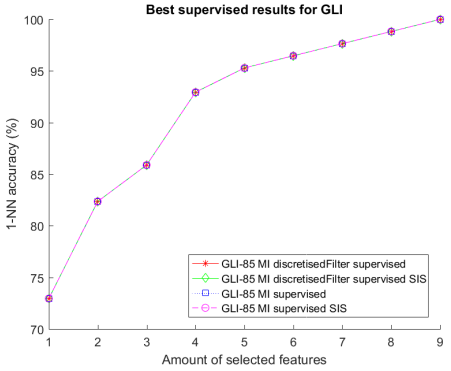


(b) Best supervised results

Figure 3.5: Best mRMR results for DLBCL

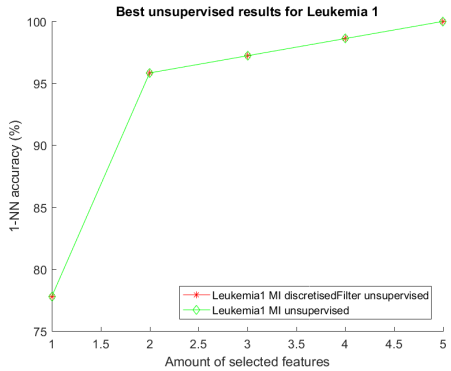


(a) Best unsupervised results

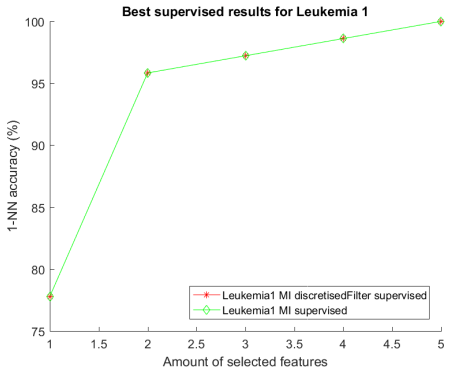


(b) Best supervised results

Figure 3.6: Best mRMR results for GLI-85

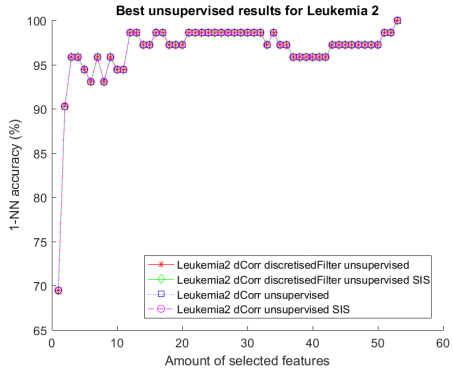


(a) Best unsupervised results

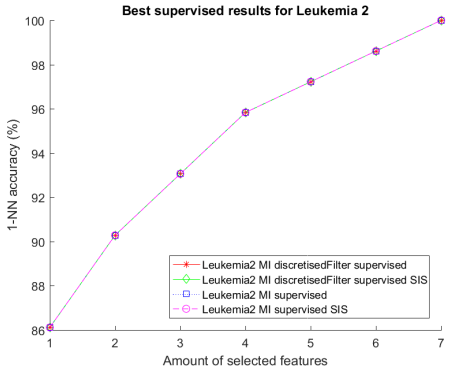


(b) Best supervised results

Figure 3.7: Best mRMR results for Leukemia 1

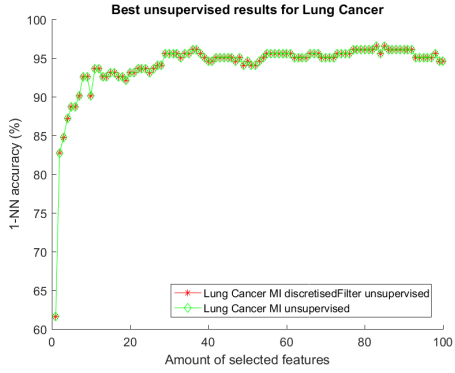


(a) Best unsupervised results

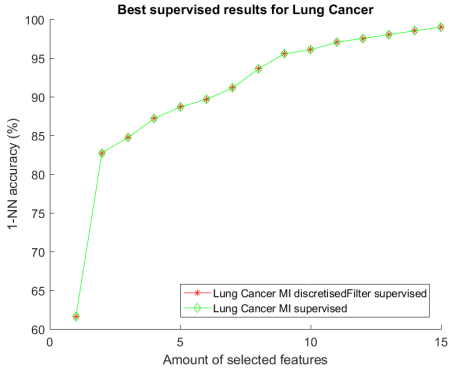


(b) Best supervised results

Figure 3.8: Best mRMR results for Leukemia 2

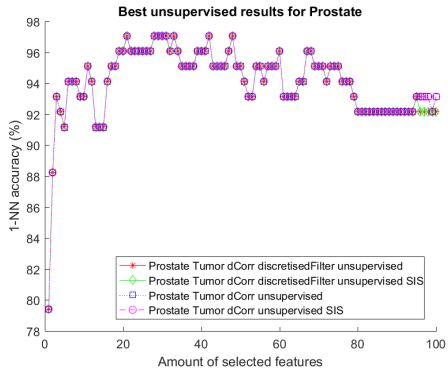


(a) Best unsupervised results

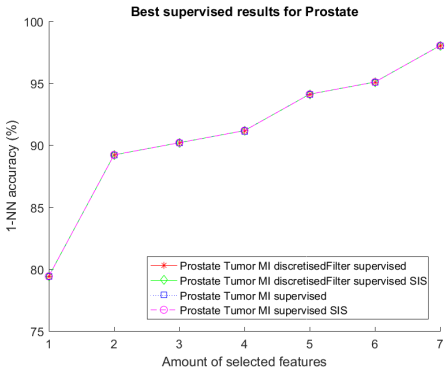


(b) Best supervised results

Figure 3.9: Best mRMR results for Lung Cancer

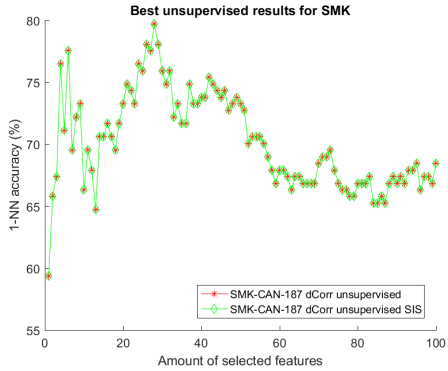


(a) Best unsupervised results

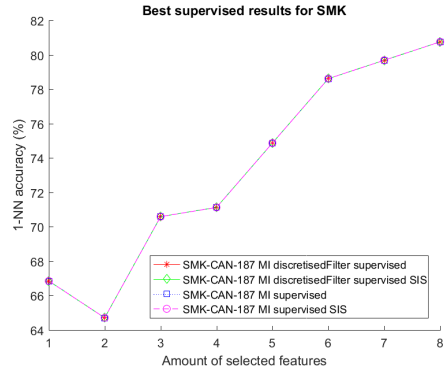


(b) Best supervised results

Figure 3.10: Best mRMR results for Prostate

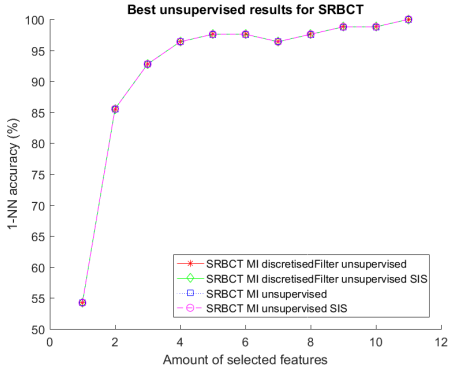


(a) Best unsupervised results

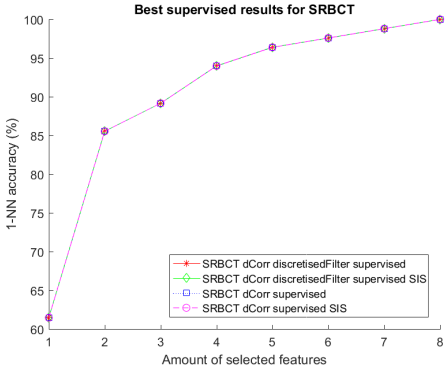


(b) Best supervised results

Figure 3.11: Best mRMR results for SMK-CAN-187



(a) Best unsupervised results



(b) Best supervised results

Figure 3.12: Best mRMR results for SRBCT

Table 3.3: Summary of the best mRMR results

Dataset	Supervised accuracy (%)	Unsupervised accuracy (%)	# Supervised features	# Unsupervised features
11 Tumors	**88.51**	86.78	**20**	69
9 Tumors	**81.67**	78.33	**13**	54
Brain Tumor 1	**96.67**	95.56	**10**	**10**
Brain Tumor 2	**98**	96	**10**	16
DLBCL	**100**	**100**	**6**	12
GLI-85	**100**	98.82	**9**	23
Leukemia 1	**100**	**100**	**5**	**5**
Leukemia 2	**100**	**100**	**7**	53
Lung Cancer	**99.01**	96.55	**15**	83
Prostate Tumor	**98.04**	97.06	**7**	21
SMK-CAN-187	**80.75**	79.68	**8**	28
SRBCT	**100**	**100**	**8**	11
Average	**95.22**	94.07	**9.83**	32.08

Table 3.4: Summary of the preferred[2] associative measure

Dataset	Best supervised associative measure	Best unsupervised associative measure
11 Tumors	MI	MI
9 Tumors	dCorr	dCorr
Brain Tumor 1	dCorr	dCorr
Brain Tumor 2	dCorr	dCorr
DLBCL	MI/dCorr	dCorr
GLI-85	MI	dCorr
Leukemia 1	MI	MI
Leukemia 2	MI	dCorr
Lung Cancer	MI	MI
Prostate Tumor	MI	dCorr
SMK-CAN-187	MI	dCorr
SRBCT	dCorr	MI

[2]The preferred associative measure is chosen according to highest classification accuracy attainable

Table 3.5: Summary of the preferred[3] filter methods

Dataset	Best supervised filter combinations	Best unsupervised filter combinations
11 Tumors	Discretised	SIS SIS + Discretised
9 Tumors	Discretised	SIS + Discretised
Brain Tumor 1	Discretised	Any combination
Brain Tumor 2	Discretised	SIS
DLBCL	Any combination	Any combination
GLI-85	Any combination	SIS SIS + Discretised
Leukemia 1	Discretised	Discretised
Leukemia 2	Any combination	Any combination
Lung Cancer	Discretised	Discretised
Prostate Tumor	Any combination	Any combination
SMK-CAN-187	Any combination	SIS
SRBCT	Any combination	Any combination

Table 3.6: Summary of computation time for the best results

Dataset	Time (s)
11 Tumors	273.60
9 Tumors	103.25
Brain Tumor 1	358.21
Brain Tumor 2	407.40
DLBCL	23.40
GLI-85	94.32
Leukemia 1	4.23
Leukemia 2	53.44
Lung Cancer	391.27
Prostate Tumor	148.81
SMK-CAN-187	259.83
SRBCT	71.40
Average	182.43

Recall that the unsupervised approach is simply the standard mRMR algorithm. Table 3.4 shows *dCorr* to be the more promising associative measure with regards to the

[3]The preferred pre-processing filter method is chosen according to highest classification accuracy attainable

unsupervised technique, however, MI shows to perform better in the supervised case. Notice that the supervised Figures 3.1b to 3.12b are ever increasing due to the approval strategy of supervision which only accepts a feature upon an improvement to classification accuracy. Not only did the supervised method outperform the traditional unsupervised mRMR algorithm by more than 1 % as shown in Table 3.3, it was able to select fewer features as well with an average of 9.93 as opposed to 32.08 with the unsupervised technique. The supervised technique in essence will perform better than the traditional unsupervised technique.

Filter and wrapper methods will find the possible best m features, but those features are not necessarily the m best features to be used by a classifier as shown in the results given in Table 3.3. The supervised technique overcomes this issue by only searching for improvement and therefore features which provide the same or lower classification accuracy are also deemed redundant and will influence the selection of the next feature. The amount of selected features obtained by the supervised technique are limited and do not adhere to the target feature size. Instead, the target feature size indicates the amount of iterations in order to find all the suitable features which increase classification accuracy.

The supervised strategy also provides a computational advantage because the list of best features F_s will not increase in size unless the the next best feature improves classification accuracy. The reduced size of F_s reduces the iterations needed to compute the relevance between the remaining features and the features within F_s. The time saved by reducing the size of F_s is greater than the time it takes to recompute classification accuracy. The supervised mRMR algorithm performs very well with an overall average classification improvement of 18.28 % from the baseline of 76.94 %.

Table 3.5 shows the advantage of using a pre-processing filter to decrease the dimensions of features for improved computation speed and accuracy. Using pre-processing filters provided equivalent or better results as opposed to using all features. For the supervised case, the discretisation filter is recommended as it remained the only filter to provide the optimal results for each dataset provided discretisation is the combinational choice for “any combination”. As for the unsupervised case, the use of SIS alone or alongside discretisation is recommended which is still computationally advantageous with less features. Notice that there is no advantage in trying to use all features for any dataset.

3.1.2 FCBF#

As with mRMR, the combinational pre-processing options remain the same. The associative measures of interest includes SU and *dCorr* which yields 192 experiments once again. The addition of supervision to FCBF# takes a different approach. The FCBF# algorithm is capable of selecting the best subset of features without having to select a predefined amount of features to remain. In the current implementation of FCBF#, each remaining feature is tested for predominance or made predominant by removing all irrelevant features regarding the feature in question. When a feature is, or has become, predominant then it remains in the list of best features. Introducing supervision first tests the newly selected predominant feature and checks to see if it has a positive influence on classification accuracy, and if so it will remain in the list of best features, else it is also removed along with all its redundant peers. The use of supervision allows predominant features to be removed as well, which further reduces the size of the best subset of features. Once again the target feature size is set to 100 and the best subset of features along with its accuracy is also calculated. The pseudo code to FCBF# including supervision and pre-processing is summarised in Algorithm 12 below.

Due to lack of space, the code within the 3^{rd} while loop is omitted, but can be found in Algorithm 4 in Section 2.3.4. Recall that each algorithm is optimised for parallel processing, but FCBF# is an exception due to the lack of for loops. Fortunately FCBF# is a filter method and does not require excessive computational power. The addition of supervision is given in lines 14 - 20. Notice that there is the possibility for predominant features to be removed as well. To determine the best subset of features, target size k must be set to 0. As for the threshold value δ, it is not used in this study, instead the first 200 features with the highest relevance scores are chosen for further processing for consistency.

Algorithm 12 Supervised FCBF#

- Initialise:

S = $[f_1, f_2 ... f_n]$ - Data set
C - Class vector
k - Target size
$rem = 1$ - Variable indicating a feature was removed
$flag = 0$ - Stop criteria when k is met
$pass = 0$ - Variable preventing multiple features from being deleted at once
$s_v = \{0, 1\}$ - Enable or disable supervision assistance

Pre-process S according to SIS and discretisation choices
$Rel(\cdot)$ - Selected relevance method
Threshold δ
$F_s = \forall f_i, Rel(i, c) \geq \delta$ - Subset of best features in descending order
$maxAcc = 0$

while (rem == 1 AND $flag$!= 1) **do**
    $f_p = firstElement(F_s)$
    $rem = 0$
    while ($flag$!= 1 AND f_p != NULL) **do**
        $f_q = lastElement(F_s)$
        $pass = 0$
        while (f_q != NULL AND $pass$!= 1) **do**
            Remove redundant peers
        end
        if $s_v == 1$ **then**
            accWithFp = 1-NN(F_s, Y)
            accWithoutFp = 1-NN($F_s - \{f_p\}, Y$)
            if accWithoutFp >= accWithFp **then**
                $F_s = F_s - \{f_p\}$
            end
        end
        $f_p = nextElement(F_s)$
    end
end
The features remaining in F_s are the best features

The results for FCBF# on all 12 datasets are given below. Figures 3.13 to 3.18 show the best performance when $k = 100$ for associative measures SU and $dCorr$ for each dataset. The omission of figures related to specific datasets indicates that the result for the best subset of features outperformed the selection process and is summarised in Table 3.9 below the figures. Thereafter, a summary of the best results, preferred associative measure and pre-processing combinations are given in Tables 3.7, 3.8 and 3.9 respectively.

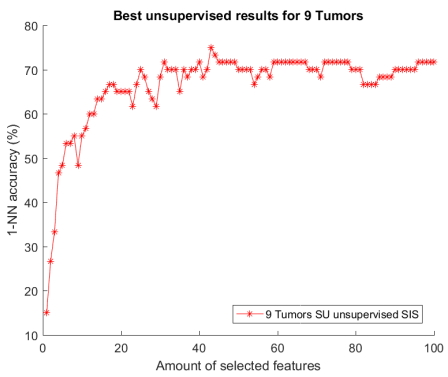


Figure 3.13: Best unsupervised FCBF# for 9 Tumors

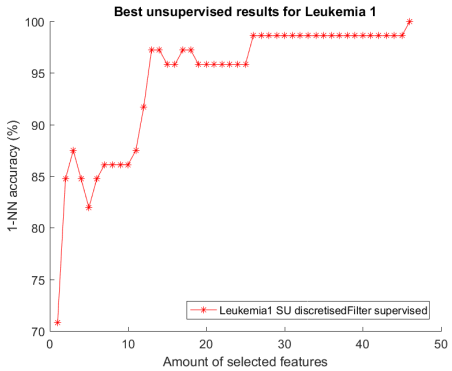


Figure 3.14: Best unsupervised FCBF# for Leukemia 1

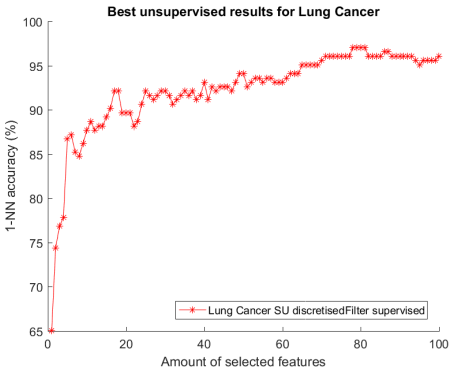


Figure 3.15: Best unsupervised FCBF# for Lung Cancer

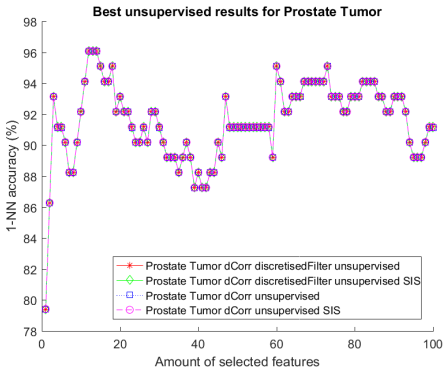


Figure 3.16: Best unsupervised FCBF# for Prostate

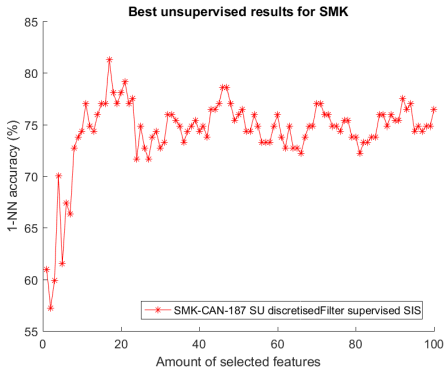


Figure 3.17: Best unsupervised FCBF# for SMK-CAN-187

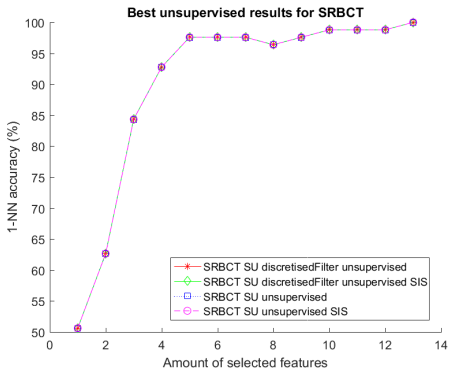


Figure 3.18: Best supervised FCBF# for SRBCT

Table 3.7: Summary of the best FCBF# results

Dataset	Best strategy	Accuracy (%)	# features	Time (s)
11 Tumors	Best supervised subset	91.95	35	554.585
9 Tumors	Unsupervised	75	43	0.032
Brain Tumor 1	Best supervised subset	93.33	23	303.366
Brain Tumor 2	Best supervised subset	92	26	136.633
DLBCL	Best supervised subset	100	48	163.061
GLI-85	Best supervised subset	97.65	22	309.158
Leukemia 1	Unsupervised	100	46	0.187
Leukemia 2	Best supervised subset	98.61	14	216.389
Lung Cancer	Unsupervised	97.04	78	0.432
Prostate Tumor	Unsupervised	96.08	12	0.096
SMK-CAN-187	Unsupervised	81.28	17	0.094
SRBCT	Supervised	100	12	2.367
Average	-	93.58	31.33	140.533

Table 3.8: Summary of the preferred associative measure for each best result

Dataset	Best associative measure
11 Tumors	SU
9 Tumors	SU
Brain Tumor 1	SU
Brain Tumor 2	SU
DLBCL	SU
GLI-85	SU
Leukemia 1	SU
Leukemia 2	SU
Lung Cancer	SU
Prostate Tumor	dCorr
SMK-CAN-187	SU
SRBCT	SU

Table 3.9: Summary of the preferred filter methods for each best result

Dataset	Best filter combinations
11 Tumors	SIS + Discretised
9 Tumors	SIS
Brain Tumor 1	None
Brain Tumor 2	SIS + Discretised
DLBCL	None
GLI-85	Discretised
Leukemia 1	Discretised
Leukemia 2	Discretised
Lung Cancer	Discretised
Prostate Tumor	Any combination
SMK-CAN-187	SIS + Discretised
SRBCT	Any combination

Refer to Table 3.7 which shows the summary of the best results. The majority of best results are determined by a supervised strategy, mainly the best subset of features obtained by the supervised strategy. Some datasets do however prefer the use of the ordinary unsupervised FCBF# algorithm but only with the addition of a pre-processing filter as seen in Table 3.9. Table 3.9 also shows the dominant pre-processing techniques to be discretisation or a combination between discretisation and SIS. As for the associative measure, the originally proposed SU measure outperforms *dCorr* regardless of any specific technique used as shown in Table 3.8. Overall, the FCBF# algorithm perform adequately and shows an overall average classification improvement of 16.64 % from the baseline of 76.94 %, with its main advantage being its computation speed without the use or need for parallel processing. Table 3.7 shows the computation time in seconds for each best result. The supervised methods take substantially longer than the unsupervised method due to the addition of the 1-NN classifier in each iteration. The unsupervised method is able to complete in under one second, while the supervised method can take up to ten minutes, which is still fast with regards to feature selection as it is only required to run once.

Recall that the threshold value is not used in these experiments, instead the top 200 features are used for further processing. In future, it might be meaningful to determine whether each dataset has a preferred threshold according to the amount of features instead of using a predefined amount.

3.1.3 ORFS

The ORFS method compares three different associative measures:

- MIC
- MI
- $dCorr$.

The investigation of Kraskov's MI estimator (KMI) is not in the scope of this study but instead provides the doubt necessary to compare the boasted power of MIC to its MI roots as well as $dCorr$. In future, KMI along with other MI estimators can also be tested.

The MIC measure is calculated using the code provided in the paper of [15] with default parameters. The same combinational choices are also used for ORFS, which yields a total of 288 experiments. The addition of supervision resembles the same technique used in supervised mRMR, where the next best candidate feature is checked to see whether it increases classification accuracy. If there is no improvement, the feature is discarded from the list of best and remaining features. This supervised feature removal influences the choice for the next best feature due to the possible absence of features when calculating GSO on the set of previously selected best features. ORFS is attractive due to the removal of irrelevant redundancy even though the search strategy remains simple. The pseudo code to ORFS including supervision and pre-processing is summarised in Algorithm 13 below.

Algorithm 13 Supervised ORFS

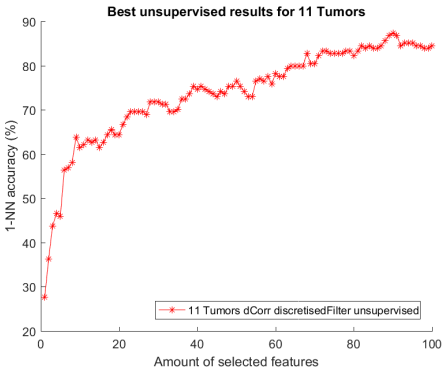
- Initialise:

$S = \emptyset$ - Subset of best features
C - Class vector
T - Target number of features
$X = \{x_1, x_2, ..., x_n\}$ - Feature dataset with n features
$s_v = \{0, 1\}$ - Enable or disable supervision assistance

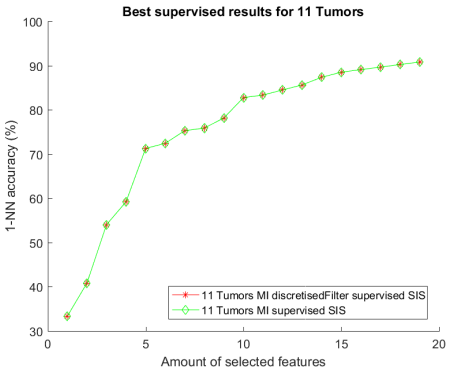
Pre-process X according to SIS and discretisation choices
$Rel(\cdot)$ - Selected relevance method

$maxAcc = 0$
$s_1 = \arg\max_{x_i \in X} Rel(x_i, C)$
$S = [S\ s_1]$
for `k = 2,3,..,T` **do**
    $s_m = \arg\max_{x_i \in X-S} Rel(GSO(x_i, S), C)$
    $S = [S\ s_m]$
    $X = X$ - $\{s_m\}$
    if $s_v == 1$ **then**
        $acc =$ 1-NN(S, C)
        if acc `>` $maxAcc$ **then**
            $maxAcc = acc$
        end
    else
        $S = S - \{s_m\}$
    end
end
The features remaining in S are the best features

Parallel processing is used to calculate the variable s_m. The amount of features to select T is set to 100. The results for ORFS on all 12 datasets are given below. Figures 3.19a to 3.30b show the best performance for all associative measures on all datasets. A summary of the best results, preferred associative measure, preferred pre-processing method and computation times follow the figures in Tables 3.10, 3.11, 3.12 and 3.13 respectively.

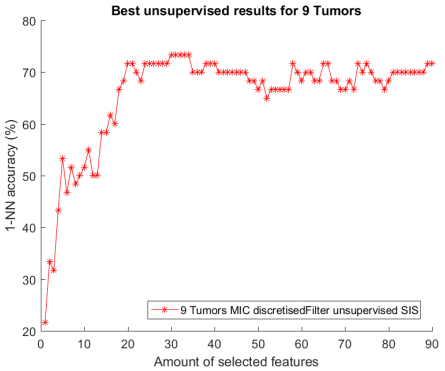


(a) Best unsupervised results

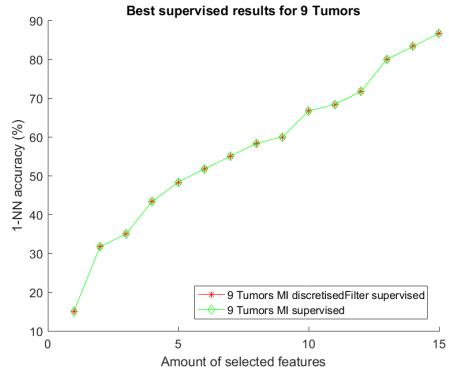


(b) Best supervised results

Figure 3.19: Best ORFS results for 11 Tumors

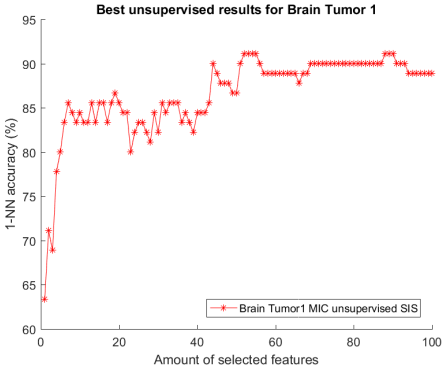


(a) Best unsupervised results

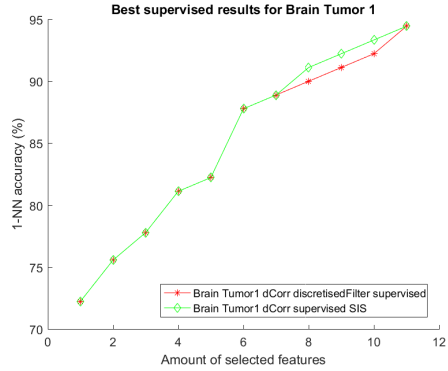


(b) Best supervised results

Figure 3.20: Best ORFS results for 9 Tumors

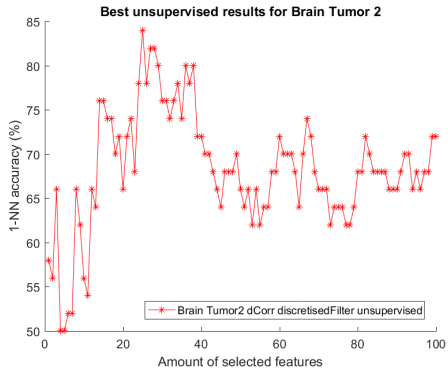


(a) Best unsupervised results

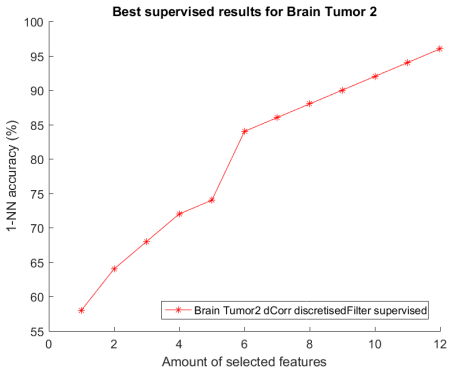


(b) Best supervised results

Figure 3.21: Best ORFS results for Brain Tumor 1

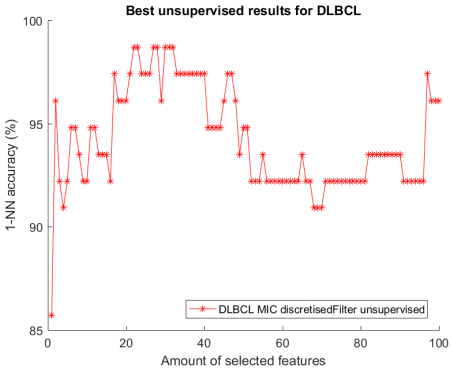


(a) Best unsupervised results

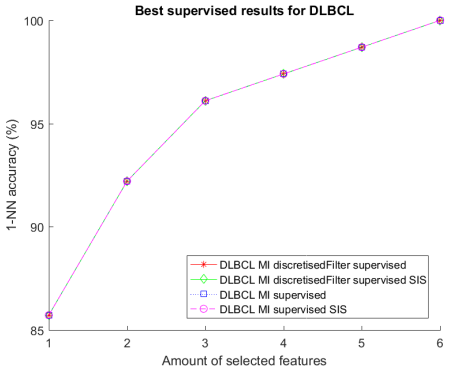


(b) Best supervised results

Figure 3.22: Best ORFS results for Brain Tumor 2

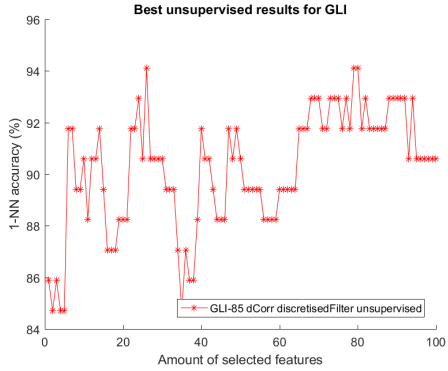


(a) Best unsupervised results

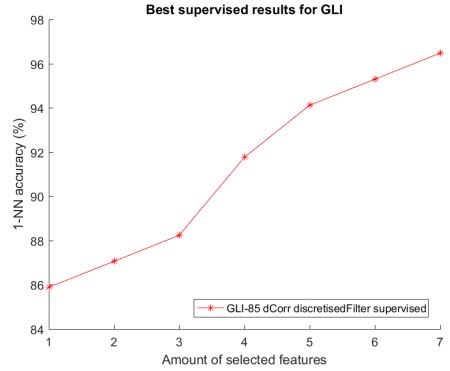


(b) Best supervised results

Figure 3.23: Best ORFS results for DLBCL

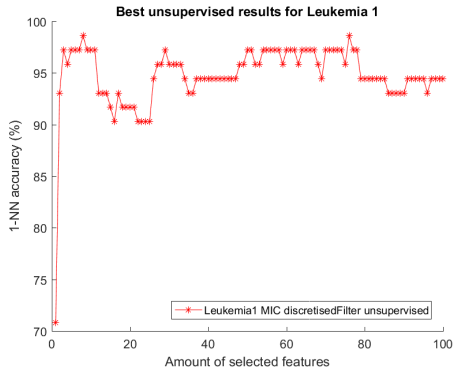


(a) Best unsupervised results

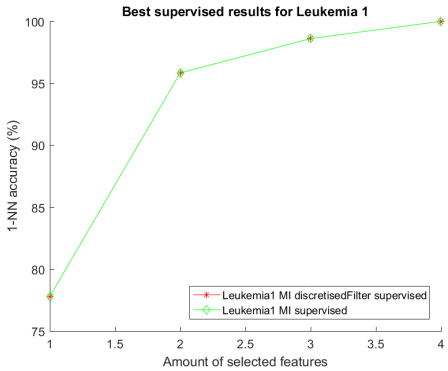


(b) Best supervised results

Figure 3.24: Best ORFS results for GLI-85

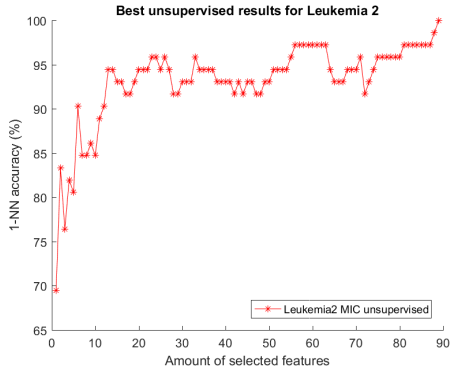


(a) Best unsupervised results

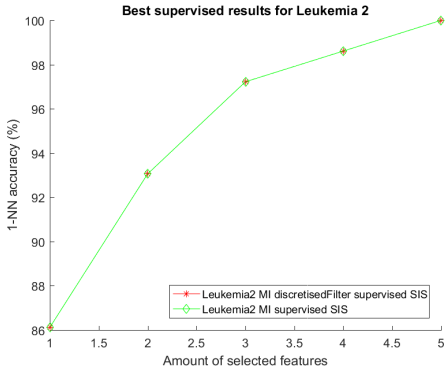


(b) Best supervised results

Figure 3.25: Best ORFS results for Leukemia 1

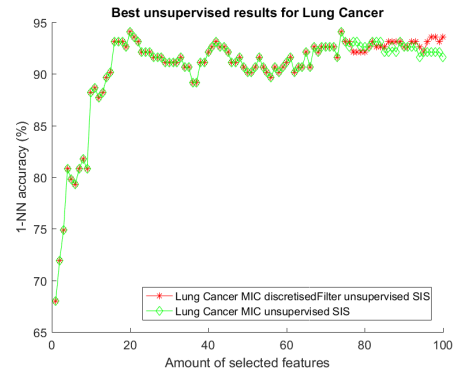


(a) Best unsupervised results

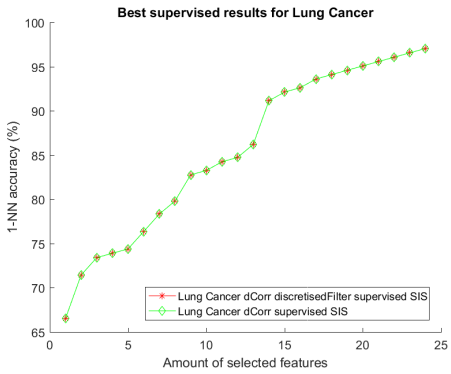


(b) Best supervised results

Figure 3.26: Best ORFS results for Leukemia 2

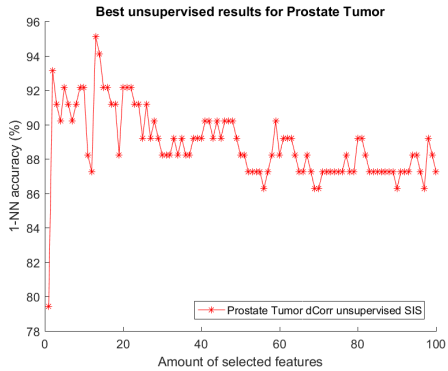


(a) Best unsupervised results

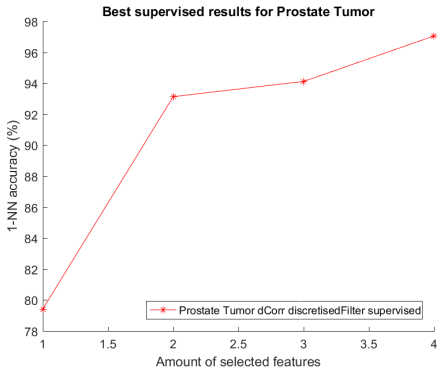


(b) Best supervised results

Figure 3.27: Best ORFS results for Lung Cancer

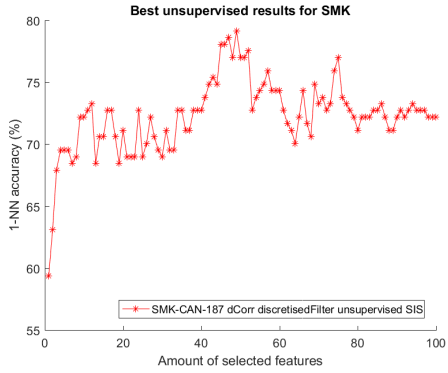


(a) Best unsupervised results

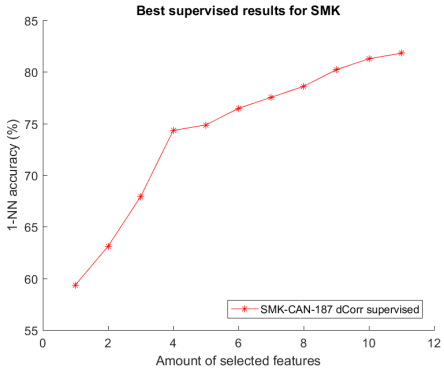


(b) Best supervised results

Figure 3.28: Best ORFS results for Prostate

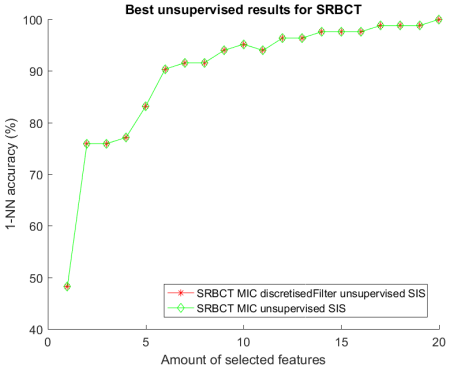


(a) Best unsupervised results

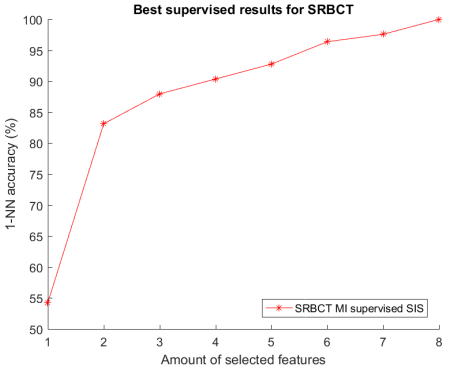


(b) Best supervised results

Figure 3.29: Best ORFS results for SMK-CAN-187



(a) Best unsupervised results



(b) Best supervised results

Figure 3.30: Best ORFS results for SRBCT

Table 3.10: Summary of the best ORFS results

Dataset	Supervised accuracy (%)	Unsupervised accuracy (%)	# Supervised features	# Unsupervised features
11 Tumors	**90.80**	87.36	**19**	90
9 Tumors	**86.67**	73.33	**15**	30
Brain Tumor 1	**94.44**	91.11	**11**	52
Brain Tumor 2	**96**	84	**12**	25
DLBCL	**100**	98.7	**6**	22
GLI-85	**96.47**	94.12	**7**	26
Leukemia 1	**100**	98.61	**4**	8
Leukemia 2	**100**	**100**	**5**	89
Lung Cancer	**97.04**	94.09	24	**20**
Prostate Tumor	**97.06**	95.10	**4**	13
SMK-CAN-187	**81.82**	79.14	**11**	49
SRBCT	**100**	**100**	**8**	20
Average	**95.03**	91.30	**10.5**	37

Table 3.11: Summary of the preferred associative measure

Dataset	Best supervised associative measure	Best unsupervised associative measure
11 Tumors	MI	dCorr
9 Tumors	MI	MIC
Brain Tumor 1	dCorr	MIC
Brain Tumor 2	dCorr	dCorr
DLBCL	MI	MIC
GLI-85	dCorr	dCorr
Leukemia 1	MI	MIC
Leukemia 2	MI	MIC
Lung Cancer	dCorr	MIC
Prostate Tumor	dCorr	dCorr
SMK-CAN-187	dCorr	dCorr
SRBCT	MI	MIC

Table 3.12: Summary of the preferred filter methods

Dataset	**Best supervised filter combinations**	**Best unsupervised filter combinations**
11 Tumors	SIS SIS + Discretised	Discretised
9 Tumors	Discretised	SIS + Discretised
Brain Tumor 1	SIS Discretised	SIS
Brain Tumor 2	Discretised	Discretised
DLBCL	Any combination	Discretised
GLI-85	Discretised	Discretised
Leukemia 1	Discretised	Discretised
Leukemia 2	SIS SIS + Discretised	None
Lung Cancer	SIS SIS + Discretised	SIS SIS + Discretised
Prostate Tumor	Discretised	Discretised
SMK-CAN-187	None	SIS + Discretised
SRBCT	SIS	SIS SIS + Discretised

Table 3.13: Summary of computation time for the best results

Dataset	Time (s)
11 Tumors	318.62
9 Tumors	82.80
Brain Tumor 1	172.29
Brain Tumor 2	221.41
DLBCL	66.63
GLI-85	245.42
Leukemia 1	11.40
Leukemia 2	42.66
Lung Cancer	831.62
Prostate Tumor	193.23
SMK-CAN-187	2477.45
SRBCT	17.40
Average	390.08

Refer to Table 3.10. As with mRMR, the supervised method outperforms the unsupervised method on average by 3.73 % accuracy and also yields a lower average of selected

features of 10.5 instead of 37. Table 3.11 shows that the MIC and MI associative measures seem to favour specific methods. For the supervised method, MI and *dCorr* are equally prevalent but MIC does not make an appearance. As for the unsupervised method, MIC and *dCorr* are preferred with MIC showing a slight advantage with seven out of the 12 datasets. Notice that MI does not make an appearance for the unsupervised case. As for the boasted performance of MIC, it does outperform MI in the unsupervised case and shows similar performance to *dCorr*, but does not show any significance with the introduction of supervision. From the aforementioned results, it can be said that MIC performs better than MI when having to select the most relevant feature to the class without any assistance in ORFS. Overall, *dCorr* remains the most stable associative measure to use for ORFS regardless of supervision.

As for the preferred pre-processing filters, Table 3.12 shows no sure pattern for using a specific pre-processing filter, it only shows that using a filter does improve performance with regards to using all available features. The same elegance cannot be said for the unsupervised case, where there is no sure pattern for selecting the appropriate filter. Thankfully the results from Table 3.10 and 3.12 determines the best approach for ORFS, being supervised with a discretisation preprocessor and using *dCorr* or MI as an associative measure. In future, MIC can be applied as an associative measure for other feature selection algorithms to determine its true strength, but unfortunately for ORFS it did not yield the best performance.

Refer to Table 3.13. The computation times are based on the best results. For those results which provide similar performance with different pre-processing filters, the lowest computation time is taken. The average computation time for ORFS is 6.50 minutes. The addition of supervision provides a computational advantage as well. Recall that GSO is calculated on the list of best features. The larger the list becomes, the more computationally expensive calculation GSO becomes. When using supervision, many features will not be added to the best list of features because they do not increase classification accuracy. GSO will therefore compute on a smaller list of features in each iteration, unlike the unsupervised case where GSO is calculated on an ever growing feature list with each iteration. Notice the large computation time for dataset SMK-CAN-187. This is the result of not using any pre-processing filter on the dataset to reduce the amount of features before further processing. It is also the only dataset which showed a better result when using all features. Overall, ORFS provides an average classification accuracy improvement of 18.09 % from the baseline of 76.94 %.

3.2 Wrapper and embedded methods

The methods FRBPSO, SVM-RFE and SBMLR each have their own separate experiments which explores their effectiveness. The pre-processing choices, however, remains the same throughout the experiments except for the choice of supervision. Previous experiments have shown that the use of all features in X is not advantageous, therefore only SIS and/or Irani MDL discretisation methods will be applied for the remainder of the feature selection algorithms. This gives a total of three combinations instead of eight for testing each algorithm. The supervised methods also do not require associative measures. Given 12 datasets, a minimum of 36 experiments per method exists. Some methods will have more experiments when exploring different optimisation techniques.

3.2.1 FRBPSO

The core PSO algorithm consists of the following parameters:

- Particle size n is set to 40
- Acceleration factors $c_1 = 2$ and $c_2 = 2$
- The velocities are within [-6;6]
- $w_{min} = 0.4$ and $w_{max} = 0.9$
- Maximum iterations $k = 100$
- The upper and lower bound values are determined by the FIS

Recall that the inertial weights w are updated using LDIW. The objective function consists of a 1-NN classifier using LOOCV as its objective score. If more than one particle shows the same objective score, the particle with the least amount of features wins. Local minima are avoided by choosing a new global best particle by using an 'AND' operation on the personal best particles, but only if no change to the previous global best particle is made within 10 iterations. The PSO algorithm also makes use of personal best and global best time steps for faster convergence. Each experiment is repeated 10 times for consistency and the average and best results are determined.

The optimisation of the FRBPSO algorithm lies within the initialisation of random particles and the objective function. The best objective score for the particle with the least amount of features needs to be determined. For initialisation, the particles are split into four groups of 10. Each pair is set to have an initial amount of selected features

of 10%, 20%, 30% and 40% respectively. This initialisation strategy is used to avoid the particles starting with the same amount of features, which encourages solutions with less features. The strategy is also more computationally feasible than testing individual initialisations of 10%, 20%, 30% and 40% for all particles. If all particles are randomly initialised using a normal distribution, then 50% of features on average will always be initialised.

In total, there are only three initialisation parameters, namely the three pre-processing options which are carried out on 12 datasets, therefore a total of 36 experiments are needed and repeated 10 times. recall that FRBPSO is a meta-heuristic algorithm which contains some randomness and therefore each experiment is repeated 10 times to obtain the best and average results.

Additional tuning parameters for particle size n, acceleration factors c_1 and c_2, length penalty factor λ and other initialisation techniques can also be implemented but is beyond the scope of this study. Currently the objective score is only determined by classification accuracy. If the particles get stuck in a local optima where a high accuracy is obtained but many features are still retained there is no certainty that a similar or better classification accuracy is achievable with less features. For future work the objective score can have an added feature length penalty, where the percentage of selected features to total features are taken into account and added to the objective score. The objective score for particle p_i and class variable C is given as follows:

$$ofun = LOOCV(\text{1-NN}(p_i, C)) + \lambda \frac{\#features(p_i)}{\#TotalFeatures}, \tag{3.2.1}$$

where λ is a tuneable parameter. The best convergence characteristic for each dataset is given in Figures3.31 to 3.42 below. A summary of the best and average results, as well as the preferred filter method is given below the figures in Tables 3.14 to 3.16. Note that the best method is defined by the highest accuracy with the least amount of features.

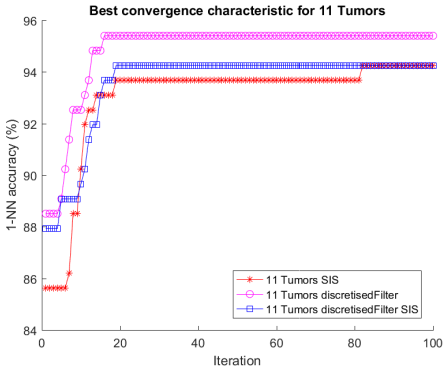


Figure 3.31: Best FRBPSO results for 11 Tumors

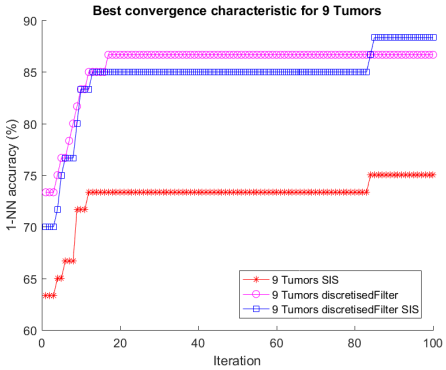


Figure 3.32: Best FRBPSO results for 9 Tumors

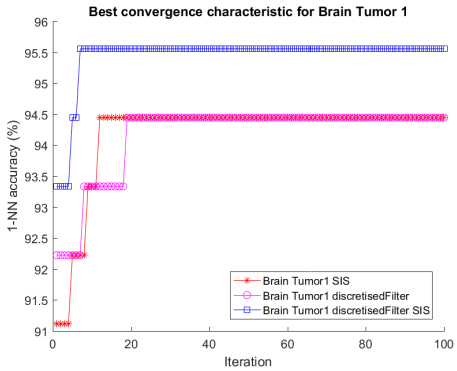


Figure 3.33: Best FRBPSO results for Brain Tumor 1

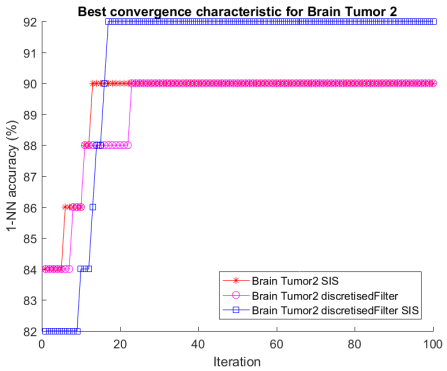


Figure 3.34: Best FRBPSO results for Brain Tumor 2

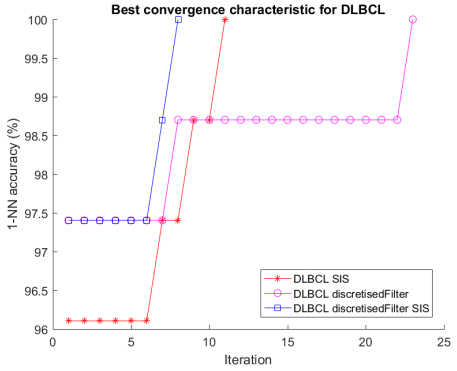


Figure 3.35: Best FRBPSO results for DLBCL

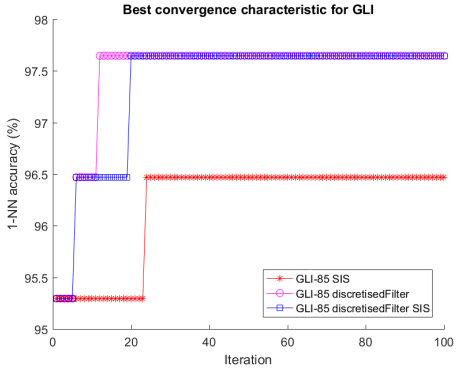


Figure 3.36: Best FRBPSO results for 11 GLI-85

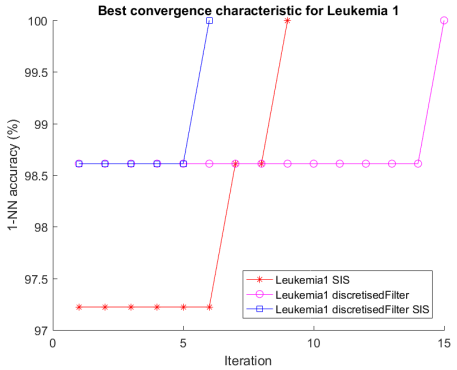


Figure 3.37: Best FRBPSO results for Leukemia 1

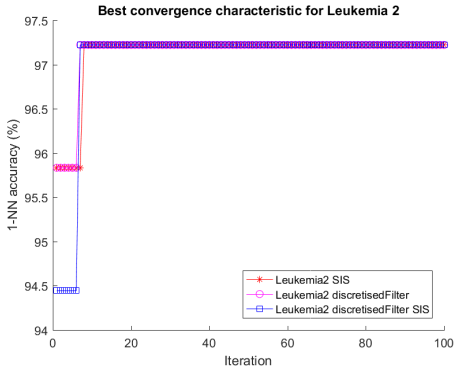


Figure 3.38: Best FRBPSO results for Leukemia 2

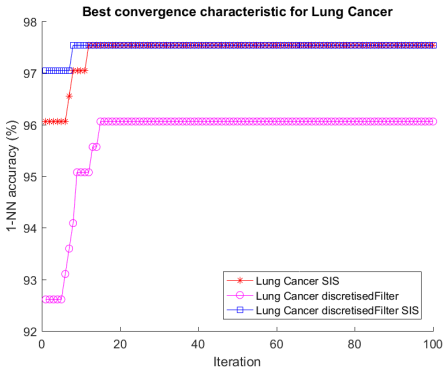


Figure 3.39: Best FRBPSO results for Lung Cancer

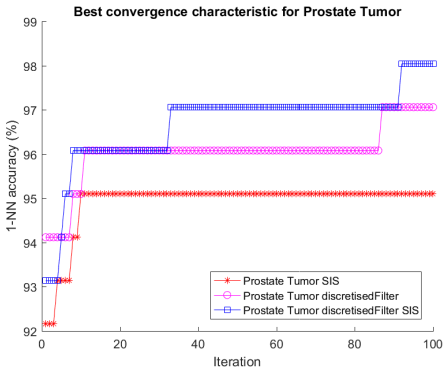


Figure 3.40: Best FRBPSO results for Prostate Tumor

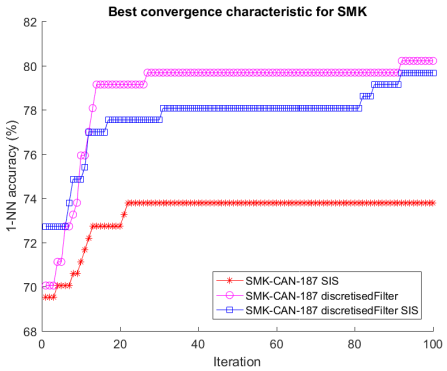


Figure 3.41: Best FRBPSO results for SMK-CAN-187

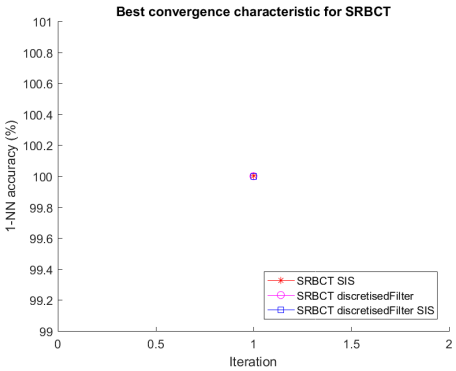


Figure 3.42: Best FRBPSO results for SRBCT

Table 3.14: Summary of the best FRBPSO results

Dataset	Accuracy (%)	# features	Time (h)
11 Tumors	95.40	801	30.45
9 Tumors	88.33	34	2.71
Brain Tumor 1	95.56	255	4.63
Brain Tumor 2	92	191	2.63
DLBCL	100	114	2.66
GLI-85	97.65	263	5.40
Leukemia 1	100	101	2.36
Leukemia 2	97.22	192	3.89
Lung Cancer	97.54	260	19.99
Prostate Tumor	98.04	106	5.28
SMK-CAN-187	80.21	409	19.11
SRBCT	100	96	0.08
Average	95.16	235.17	8.27

Table 3.15: Summary of the preferred filter methods per best result

Dataset	Best filter combinations
11 Tumors	Discretised
9 Tumors	Discretised + SIS
Brain Tumor 1	Discretised + SIS
Brain Tumor 2	Discretised + SIS
DLBCL	Discretised + SIS
GLI-85	Discretised + SIS
Leukemia 1	Discretised + SIS
Leukemia 2	SIS
Lung Cancer	Discretised + SIS
Prostate Tumor	Discretised + SIS
SMK-CAN-187	Discretised
SRBCT	Discretised + SIS

Table 3.16: Average results over all 10 runs for the best pre-processing combination

Dataset	Average accuracy (%)		Average # features	
11 Tumors	92.53	±0.98	1178.90	±200.25
9 Tumors	82.33	±2.26	35.80	±4.59
Brain Tumor 1	94.56	±0.67	240.40	±53.72
Brain Tumor 2	88	±3.77	440.70	±215.50
DLBCL	99.09	±0.88	191	±50.87
GLI-85	96.71	±0.74	534.50	±245.73
Leukemia 1	99.17	±0.72	157.10	±67.74
Leukemia 2	96.39	±0.97	383.40	±183.42
Lung Cancer	96.90	±0.47	798	±303.82
Prostate Tumor	96.37	±0.93	339.80	±135.81
SMK-CAN-187	77.54	±1.87	676.10	±177.66
SRBCT	100	±0	151.50	±20.23
Average	93.30		427.27	

Refer to Table 3.14 and 3.15. The average accuracy for all 10 runs are given in Table 3.16 for the pre-processing method which resulted in the best overall score. In the best case scenario FRBPSO provides an average classification accuracy improvement of 18.22 % from the baseline of 76.94 %. When taking the average of all runs, though, the improvement is only 16.36 % but standard deviation for all runs are particularly low with less than 1 % for most datasets. The computation time in Table 3.14 is calculated over all 10 runs with the help of parallel processing but still has an average computation time of over eight hours. Notice the difference in computation time for the datasets which used

a combination including SIS such as 11 Tumors and Lung Cancer which have a similar amount of features. SIS greatly reduces computation time for FRBPSO as it removes 80 % of all features prior to FRBPSO, reducing the computational burden of the 1-NN fitness function. Overall FRBPSO performs best when both SIS and discretisation filters are applied to a dataset.

FRBPSO has no problem in improving classification accuracy, but it lacks the ability to select a stable amount of features as shown by the large standard deviations in Table 3.16. The algorithm also lacks to ability to select the least amount of features when comparing the average amount of features selected between the best results and all 10 runs in Tables 3.14 and 3.16.

In future, the inclusion of a length penalty and better initialisation strategies can be explored to find a trade-off between feature size and classification accuracy. It is better understood when looking at Figure 3.42. For each run, the classification accuracy was already found to be 100% with the average amount of features being approximately 152 as shown in table 3.16. The random initialisation of particles found a sweet spot for dataset SRBCT. This is a disadvantage for the current FRBPSO algorithm. The fact that 100 % accuracy was achieved so early prevents the optimisation from finding a better solution with less amount of features.

3.2.2 SVM-RFE

The following parameters are set for the SVM:

- Maximum SGD iterations is set to 1000
- The early termination tolerance for the SGD algorithm is 10×10^{-6}.
- The L_2-norm SVM is used, therefore there is no upper bound to the alpha values
- The box constraint parameter C is set to 5

As for the kernels, the polynomial kernel is of degree 3 and has a polynomial constant of 1. The RBF kernel can be represented as an infinite sum over polynomial kernels by using the Taylor expansion. The proof can be found in [67]. For testing purposes and computational efficiency, the first 20 terms of the Taylor series is used. The Taylor series expansion of the RBF kernel for a single feature is given as

$$\Phi(x) = e^{-\gamma x^2}\left(1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \sqrt{\frac{(2\gamma)^3}{3!}}x^3 ...\right), \tag{3.2.2}$$

where $\gamma = \frac{1}{2\sigma^2}$. Using the first 20 terms for each feature expands the feature size of the dataset by a factor of 20 as well.

The RFE algorithm requires each and every feature to be ranked in a backwards fashion starting by ranking the least favourable feature. The "one-vs-one" method is used and the required amount of SVMs are created. The amount of SVMs requires for any amount of classes is given as

$$SVMs = \frac{k(k-1)}{2}, \tag{3.2.3}$$

where k is the amount of classes. Dataset 'Brain_Tumor2' contains the most amount of classes, namely 11, which results in the need for 55 different SVMs. Each dataset will undergo SVM-RFE a total of three times with Linear, Polynomial and RBF kernels as options. As before, the three pre-processing choices remain. In total, 108 experiments are conducted and the results for the top 100 features in each experiment is recorded. Figure 3.43 to 3.54 show the best results per dataset. Tables 3.17 and 3.18 summarises the results and shows the preferred pre-processing filters respectively.

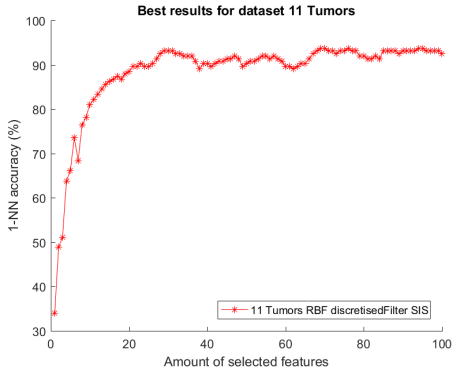


Figure 3.43: Best SVM-RFE results for 11 Tumors

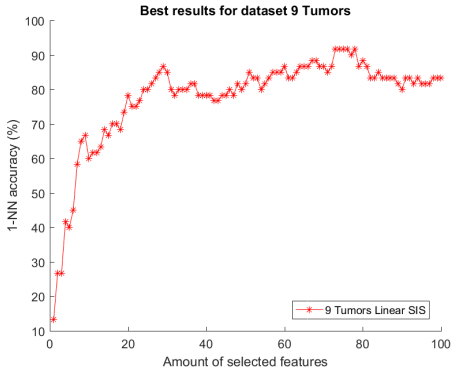


Figure 3.44: Best SVM-RFE results for 9 Tumors

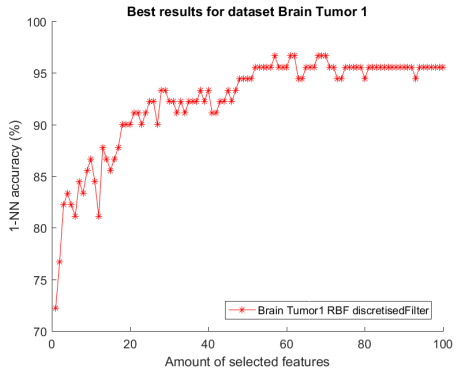


Figure 3.45: Best SVM-RFE results for Brain Tumor 1

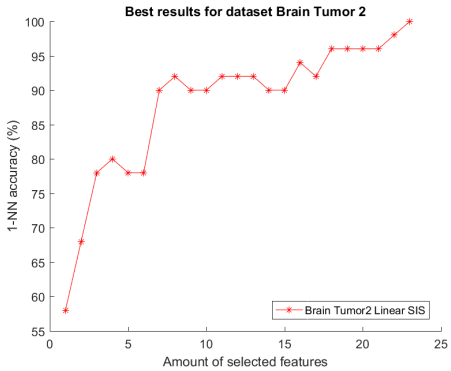


Figure 3.46: Best SVM-RFE results for Brain Tumor 2

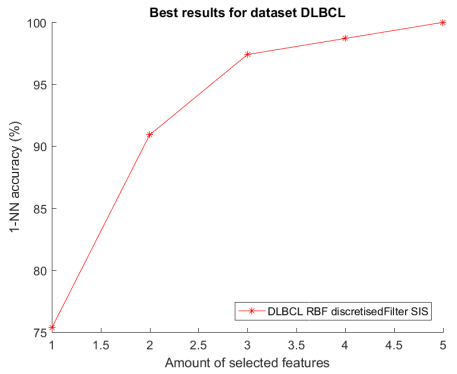


Figure 3.47: Best SVM-RFE results for DLBCL

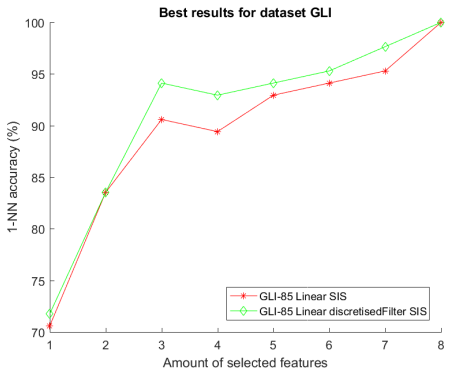


Figure 3.48: Best SVM-RFE results for GLI-85

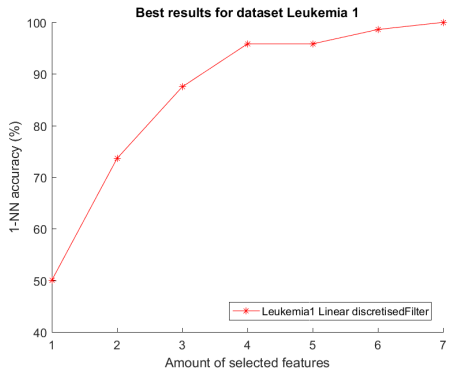


Figure 3.49: Best SVM-RFE results for Leukemia 1

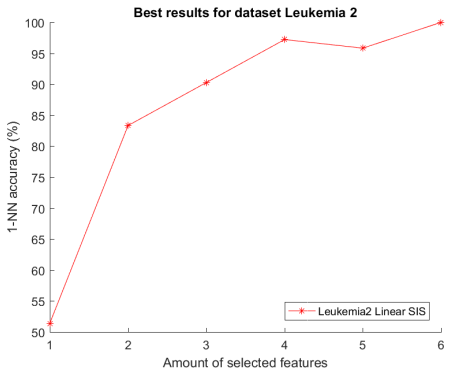


Figure 3.50: Best SVM-RFE results for Leukemia 2

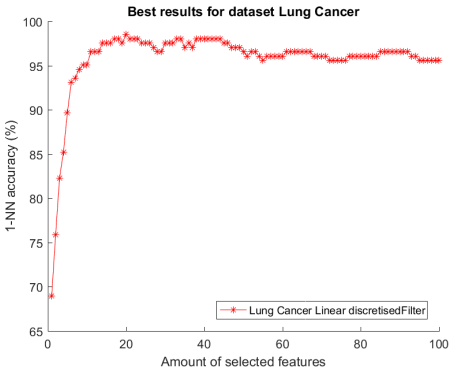


Figure 3.51: Best SVM-RFE results for Lung Cancer

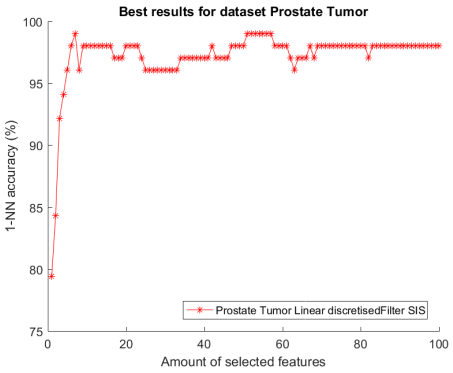


Figure 3.52: Best SVM-RFE results for Prostate

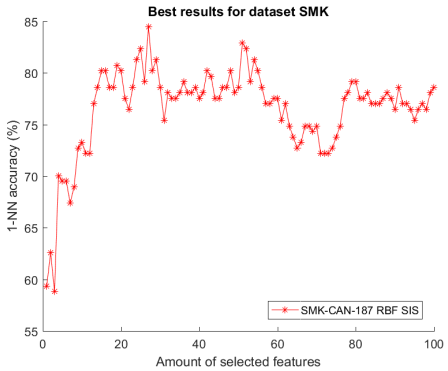


Figure 3.53: Best SVM-RFE results for SMK-CAN-187

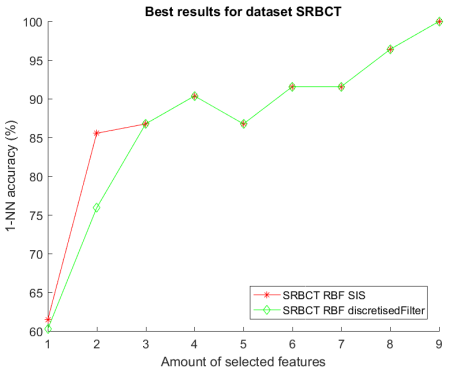


Figure 3.54: Best SVM-RFE results for SRBCT

Table 3.17: Summary of the best SVM-RFE results

Dataset	Accuracy (%)	# features	Time (h)	Kernel
11 Tumors	93.68	69	3.52	RBF
9 Tumors	91.67	73	0.19	Linear
Brain Tumor 1	96.67	57	0.85	RBF
Brain Tumor 2	100	23	0.20	Linear
DLBCL	100	5	0.09	RBF
GLI-85	100	8	0.29	Linear
Leukemia 1	100	7	0.05	Linear
Leukemia 2	100	6	0.22	Linear
Lung Cancer	98.52	20	12.08	Linear
Prostate Tumor	99.02	7	0.09	Linear
SMK-CAN-187	84.49	27	1.92	RBF
SRBCT	100	9	0.12	RBF
Average	97	25.92	1.64	-

Table 3.18: Summary of the preferred filter methods

Dataset	Best filter combinations
11 Tumors	Discretised + SIS
9 Tumors	SIS
Brain Tumor 1	Discretised
Brain Tumor 2	SIS
DLBCL	Discretised + SIS
GLI-85	SIS
	Discretised + SIS
Leukemia 1	Discretised
Leukemia 2	SIS
Lung Cancer	Discretised
Prostate Tumor	Discretised + SIS
SMK-CAN-187	SIS
SRBCT	Discretised
	SIS

Refer to Table 3.17. The "one-vs-one" approach performed very well for all datasets and an overall classification accuracy improvement of 20.06 % from the baseline of 76.94 % was achieved. Out of all three kernels tested, the 3^{rd} order polynomial did not manage to perform as well as the Linear or RBF kernels. Other polynomial orders can also be tested but is not within the scope of this study. When it comes to deciding whether a Linear or RBF kernel is recommended, they perform almost identically with the Linear kernel being preferred 58.33 % of the time. It is recommended to test a dataset with both Linear and RBF kernels to determine which is best suited for that specific dataset.

Notice the large computation time of over 12 hours for dataset Lung Cancer. This is because Lung Cancer is one of the largest datasets available and SIS was not the filter of choice. The dataset also showed little redundancy after discretisation which led to few features being removed through the discretisation filter. As for the preferred filter combinations shown in Table 3.18, SIS is the more popular choice, but there is no sure pattern as to which combination to use. It is still recommended that one be used to reduce the size of the datasets to improve computation speed, which is especially necessary for a feature elimination strategy.

3.2.3 SBMLR

For multinomial regression, only the linear solution is considered. Currently, there is no formal or mathematical convenient use of a Gram matrix $K(\cdot, \cdot)$ to avoid having to

work with $\Phi(\cdot)$ as with SVMs, therefore a higher dimensional feature map on datasets with $p >> n$ is computationally infeasible for the testing platform. To give insight, the dimensionality of $\Phi(x_i)$ for a polynomial kernel [68] is given as

$$S(n+d,d) = \frac{(n+d)(n+d-1)...(n+1)}{d!}, \tag{3.2.4}$$

where n is the original feature size, d is the polynomial degree and S is the dimensionality of the mapped feature. For this study, a polynomial of degree three is used for testing SVM-RFE. Suppose dataset $X = \{x_1^2, x_1^2...x_n^m\}$ consists of n samples and m features. If, for example, $n = 50$ and $m = 1000$, then each sample will have a mapped feature size of approximately 167 668 501, therefore $\Phi(X)$ is a dataset with over 8.38×10^9 entries. Removing a feature from X eliminates at most m features from $\Phi(X)$, which is a negligible amount at the start of feature elimination. Eliminating the first worst feature results in solving the SBMLR cost n times on a dataset $\Phi(X)$ with approximately 8.38×10^9 entries. For the RBF kernel, the first 30 Taylor coefficients increases the size of each feature 30 fold. Using the same example as before with $m = 1000$, the maximum dimensionality of the mapped feature is 30 000. Due to the nature of feature elimination each feature needs to be removed and tested to find the minimum cost associated with a specific removed feature. The total amount of tests for all features is given as

$$\#Tests = (n-1) + (n-2) + ... + 1 \tag{3.2.5}$$

$$\#Tests = n(n-1)/2, \tag{3.2.6}$$

where $\#Tests$ in this scenario with $m = 1000$ would amount to 499 500 tests. Recall that a new feature map will have to be created for each test, but for each feature removed, the amount of mapped features are reduced by 30. This still amounts to a total of 499 500*30 = 14 985 000 newly mapped features for the entirety of the elimination algorithm with a generous size for m. Furthermore, the advantage of regression on features is forfeited because the regression on the higher dimensional feature map does not pose any useful regression results on the original unmapped features.

The three pre-processing choices are the only choices for SBMLR on each dataset, which results in a total of 35 experiments. The best results for each dataset are given in Figures 3.55 to 3.66 below. A summary of the best results and preferred pre-processing filter is given in Tables 3.19 and 3.20 respectively.

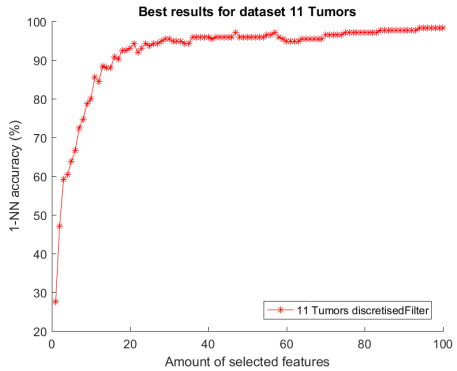


Figure 3.55: SBMLR results for 11 Tumors

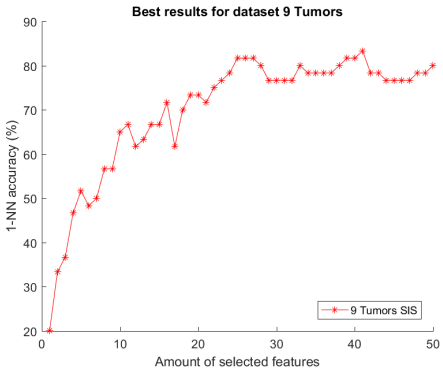


Figure 3.56: SBMLR results for 9 Tumors

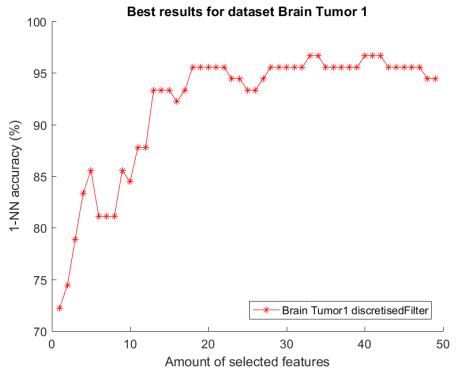


Figure 3.57: SBMLR results for Brain Tumor 1

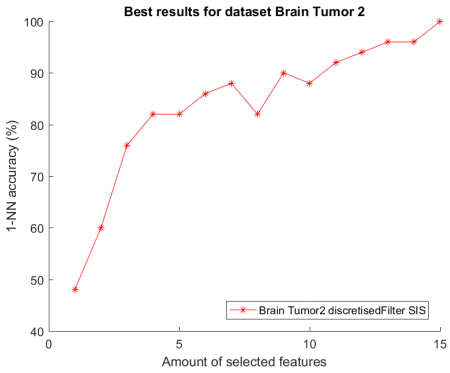


Figure 3.58: SBMLR results for Brain Tumor 2

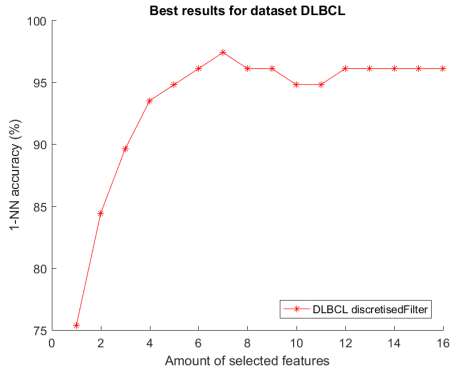


Figure 3.59: SBMLR results for DLBCL

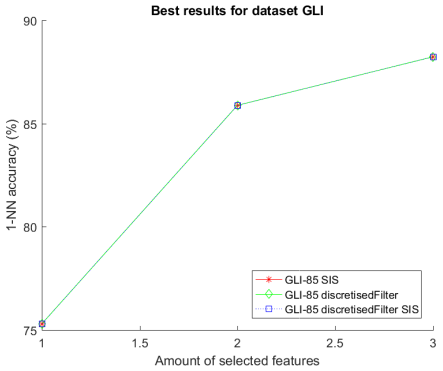


Figure 3.60: SBMLR results for GLI-85

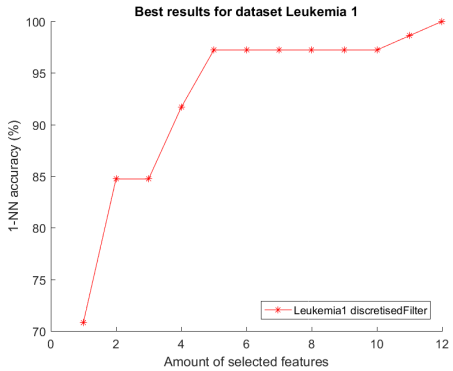


Figure 3.61: SBMLR results for Leukemia 1

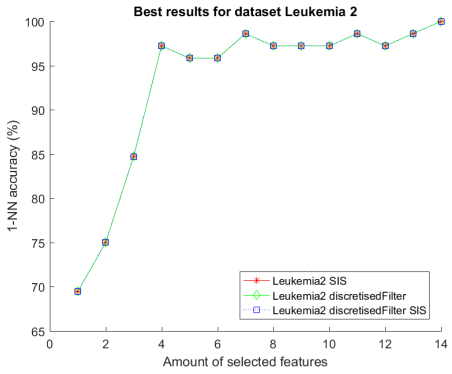


Figure 3.62: SBMLR results for Leukemia 2

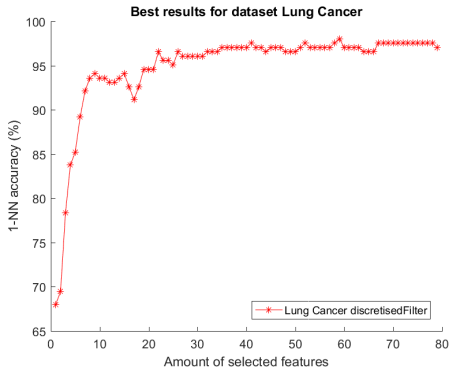


Figure 3.63: SBMLR results for Lung Cancer

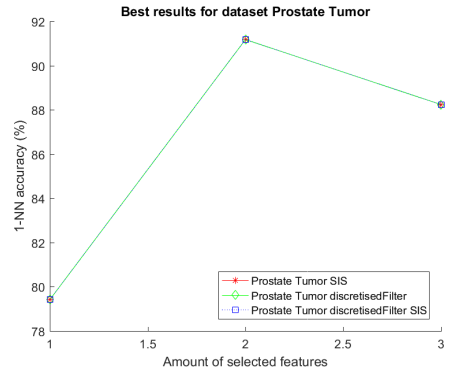


Figure 3.64: SBMLR results for Prostate

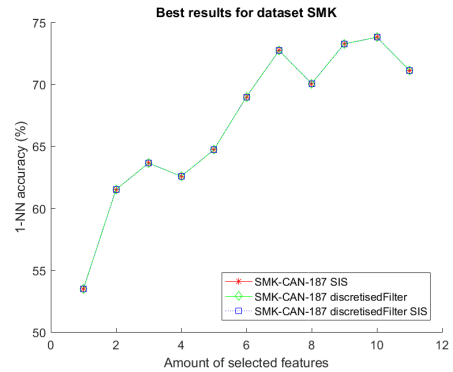


Figure 3.65: SBMLR results for SMK-CAN-187

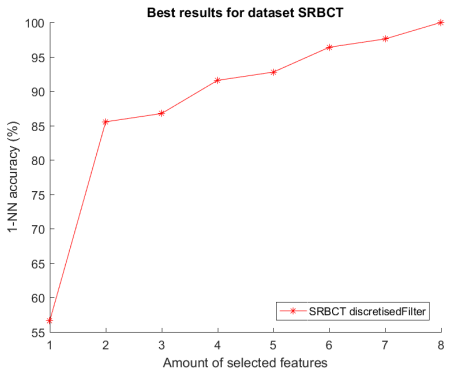


Figure 3.66: SBMLR results for SRBCT

Table 3.19: Summary of the best SBMLR results

Dataset	Accuracy (%)	# features	Time (s)
11 Tumors	98.28	94	4.30
9 Tumors	83.33	41	0.41
Brain Tumor 1	96.67	33	0.51
Brain Tumor 2	100	15	0.44
DLBCL	97.40	7	142.61
GLI-85	88.24	3	0.03
Leukemia 1	100	12	0.10
Leukemia 2	100	14	0.18
Lung Cancer	98.03	59	8.66
Prostate Tumor	91.18	2	0.03
SMK-CAN-187	73.80	10	0.08
SRBCT	100	8	0.07
Average	93.91	24.83	13.12

Table 3.20: Summary of the preferred filter methods

Dataset	Best filter combinations
11 Tumors	Discretised
9 Tumors	SIS
Brain Tumor 1	Discretised
Brain Tumor 2	Discretised + SIS
DLBCL	Discretised
GLI-85	Any combination
Leukemia 1	Discretised
Leukemia 2	Any combination
Lung Cancer	Discretised
Prostate Tumor	Any combination
SMK-CAN-187	Any combination
SRBCT	Discretised

Recall that SBMLR does not make use of any elimination process and is also limited to the use of linear relationships. Refer to Table 3.19. SBMLR can be used as an effective feature removal algorithm and provides an overall classification accuracy improvement of 16.97 % from the baseline of 76.94 %, and is also advantageous regarding computation speeds with the majority being under one second after applying the pre-processing filters. Notice the large computation time on dataset DLBCL. It is one of the smallest datasets yet it shows an excessively large computation time with respect to the other datasets. Because the SBMLR algorithm is implemented as in [38] (all copyright to the SBMLR

algorithm goes to Dr. Gavin Cawley), it is possible there exists an optimisation problem within the code. As for the preferred filter combinations, the discretisation filter is the most popular choice when considering it can also be selected in the "any combination" option.

3.3 Summary of simulation results

A summary of all the best results from each aforementioned algorithm are compared in Table 3.21 below. A discussion for the scenarios which best suit each algorithm is then discussed where applicable.

Table 3.21: Summary of best results for each algorithm

Algorithm	Average accuracy (%)	Average # features	Average Time (hh:mm:ss)
mRMR	95.22	**9.93**	00:03:02
FCBF#	93.58	31.33	00:02:19
ORFS	95.03	10.50	00:06:30
FRBPSO	95.16	235.17	08:16:12
SVM-RFE	**97**	25.92	01:38:23
SBMLR	93.91	24.83	**00:00:13**

Table 3.22 below summarises the overall average improvement over the original average results in Table 3.2.

Table 3.22: Summary of improvement for each algorithm

Algorithm	Average accuracy improvement (%)	Average # features improvement (%)[4]
mRMR	18.28	**99.99915**
FCBF#	16.64	99.99697
ORFS	18.09	99.99899
FRBPSO	18.22	99.97729
SVM-RFE	**20.06**	99.99749
SBMLR	16.97	99.99760

Refer to Tables 3.21 and 3.22 above. SVM-RFE provides the best results with an overall classification accuracy improvement of 20.06 % and was able to obtain accuracies of 100 % for half of the datasets. Unfortunately there is no recommended kernel or pre-processing

[4]Calculated as follows: $\frac{\#\text{Removed features}}{\#\text{Total features}} * 100$

filter which showed dominance, other than the polynomial kernel which failed to produce usable results. Even though the average computation time allows results in under two hours, it is necessary to test both Linear and RBF kernels with all combinations of pre-processing filters to find the optimal result for new datasets. This amounts to six experiments each scaling differently regarding computation time. If all experiments were to complete within the same amount of time, the worst case scenario computation time would be sixfold, which is worth the wait if optimal results are crucial.

SVM-RFE can also be changed to discard more than one of the lowest ranking features in each iteration, which will reduce computation time but will possibly have a negative influence on performance. Note that more to date workbenches will drastically reduce the computation time as well. SVM-RFE also reduces the amount of features to an average of 25.92, which is a 99.99749 % average reduction in features. If, however, the least amount of features obtainable is crucial, it is recommended that mRMR is used.

mRMR is able to reduce the average amount of features by an astounding 99.99915 % and is also the runner up for classification accuracy with an overall improvement of 18.28 %. Unlike SVM-RFE, the recommendation for mRMR is to use the supervised method with a discretisation pre-processing filter which will lead to results within an average of three minutes. The supervised case in mRMR does show that MI is the likely candidate for an associative measure, but testing both *dCorr* and MI is recommended to get the best results for new datasets, which also doubles computation time. If testing multiple new datasets of similar feature sizes are necessary where the lowest amount of features are required, mRMR should be the algorithm of choice.

Taking a look at the third contender regarding classification accuracy, FRBPSO. This algorithm provides similar improvements to that of mRMR, but requires an excessive amount of computation time. FRBPSO also has too many tunable parameters to ensure optimal results. The amount of maximum particle update iterations, runs, amount of particles, initialisation techniques, weight adjustment techniques, choices for the acceleration parameters c_1 and c_2, FIS optimisation and also being a metaheuristic algorithm with some randomness can create different results for each run. There is a good chance that a single run produces great classification accuracies, but at the expense of computation time, especially if more runs are necessary.

FRBPSO, though, lacks the ability to select the least possible amount of features per dataset. As mentioned earlier, the exploration of a length penalty parameter to penalise the objective score for high feature counts is the best place to start the optimisation for FRBPSO as it shows the worst feature reduction performance of 99.97729 %. As with mRMR, FRBPSO also shows preference to a specific pre-processing filter, namely the

combination of SIS and discretisation. FRBPSO also shows the longest average computation time of over eight hours for all 10 runs. If time and exploration for optimisation is not of interest to an application, it is best to prefer mRMR which will produce better classification improvements and the lowest possible amount of features.

The fourth contender with regards to classification improvement is ORFS with an improvement of 18.09 %. With the use of GSO one would imagine classification results which outperforms mRMR, but this is not the case in neither the supervised or unsupervised case. ORFS might not be the top competitor for classification accuracy but it is the runner up with regards to feature reduction with an average of 99.99899 %. Despite its best efforts, mRMR outperforms ORFS in every regard including computation speed. Nevertheless, if ORFS is to be used on new datasets, it is recommended to use the supervised method with either choice of MI or *dCorr* and avoid using MIC for the supervised case. Even though the discretisation filter is popular 58.33 % of the time for the supervised case, it will be necessary to test all combinations to determine the best results. Having to test both MI and *dCorr* as well as each pre-processing combination requires six experiments (the case for testing ORFS using all features is not valid), which will increase the computation time sixfold for the worst case scenario. Note that the actual computation time is most likely to be less due to each combination of associative measure and pre-processing filter having different computational complexity.

The fifth and second last contender is SBMLR which gives an average classification accuracy improvement of 16.97 %. Due to the nature of SBMLR explained in subsection 3.2.3, there are no comparative experiments which were made, instead the algorithm was only run once per pre-processing combination. Nonetheless it performed third best with regards to feature removal with an average of 99.976 % and also proved to be the fastest algorithm out of the six with an average computation time of 13 seconds. Note that most datasets completed in under 1 second but the average computation time for dataset DLBCL is high due to a possible optimisation issue.

SBMLR is ideal for a feature selection scenario with datasets with similar or larger feature sizes than was tested. SBMLR is especially useful if a feature removal solution is immediately necessary, or if multiple datasets with many pre-processing combinations need to be tested in quick succession. As for the suggested pre-processing filter, the discretisation filter is recommended.

The last contender is FCBF# which shows the second best computation time of approximately two minutes. FCBF# gives an average classification improvement of 16.64 % which is the worst out of all results. FCBF# shows that the supervised method is preferred an unconvincing 58.33 % of the time, with the supervised subset being the most

popular supervised strategy. As for the associative measure, the originally suggested SU measure is well suited for FCBF# in both the unsupervised and supervised case. The best pre-processing filter, though, remains an uncertainty. To get the best results for FCBF# in new datasets, all supervised and unsupervised cases with all pre-processing filters need to be tested with the only constant being the SU associative measure. Once again, a pre-processing filter is mandatory and testing FCBF# on all features is not applicable. This totals to 6 experiments (note that the best subset is calculated in unison with the supervised and unsupervised strategies to spare the amount of experiments needed). The additional experiments increases computation time sixfold for the worst case scenario. Once again actual computation time will differ from the average due to each combination of pre-processing filter and supervised/unsupervised methods having different computational complexities. The inclusion of six experiments forfeits FCBF#'s second place for computation time to mRMR which only has two, therefore making mRMR the most versatile algorithm.

Tables 3.23 and 3.24 below shows a summary of the recommended combinations to use per algorithm as previously discussed. The tests needed indicates the amount of combinations to test per algorithm to determine the best result.

Table 3.23: Summary of best combination per filter algorithm

Algorithm	Associative measure	Pre-processor	Supervised?	Tests needed
mRMR	MI/dCorr	Discretise	Yes	2
FCBF#	SU	Inconclusive	Inconclusive	6
ORFS	MI/dCorr	Inconclusive	Yes	6

Table 3.24: Summary of best combination per wrapper and embedded algorithm

Algorithm	Specific preference	Pre-processor	Tests needed
FRBPSO	N/A	SIS + discretise	1
SVM-RFE	Linear/RBF	Inconclusive	6
SBMLR	N/A	Discretise	1

In a nutshell, Table 3.25 below summarises the advantages of each algorithm in a rudimentary manner. Note that computation time is rated based on average computation time and the amount of tests needed to obtain the optimal results as discussed in the aforementioned paragraph.

Table 3.25: Summary of advantages for each algorithm

Algorithm	Average accuracy improvement	Average # features removed	Computation time
mRMR	++	+++	++
FCBF#	+	++	++
ORFS	++	+++	++
FRBPSO	++	+	+
SVM-RFE	+++	++	+
SBMLR	+	++	+++

From Table 3.25 above, the appropriate algorithm can be selected for different scenarios.

Chapter 4

Case study

This chapter applies the six reviewed feature selection algorithms on potato plant foliage data where $p >> n$. An explanation to obtaining the data is given and thereafter the results from each feature selection algorithm is recorded and used to reinforce the findings in Section 3.3.

4.1 Context

The diagnosis of ailments which show symptoms on the leaves of potato plants take part in this case study. Only four ailments are considered, namely Brown Spot, Verticillium Wilt, Potato Virus Y (PVY) and late Blight. The sample images were obtained by a simple Google search for the ailments in question and only a total of 29 samples are considered. For each sample, a total of 356 features are extracted and includes colour, texture, size, location and shape characteristics of the diseased leaf. In order to isolate the diseased leaf in an image, a watershed background subtraction algorithm is used. Table 4.1 below summarises the dataset of ailments.

Table 4.1: Samples for each ailment

Ailment	# samples
Brown spot	6
Late blight	5
PVY	11
Verticillium wilt	7
Total	29

Figure 4.1 below demonstrates the process of feature extraction and classification on an example image of Brown spot. The same process is followed to extract features for all samples. Only a brief explanation of the process is provided due to a confidentiality agreement.

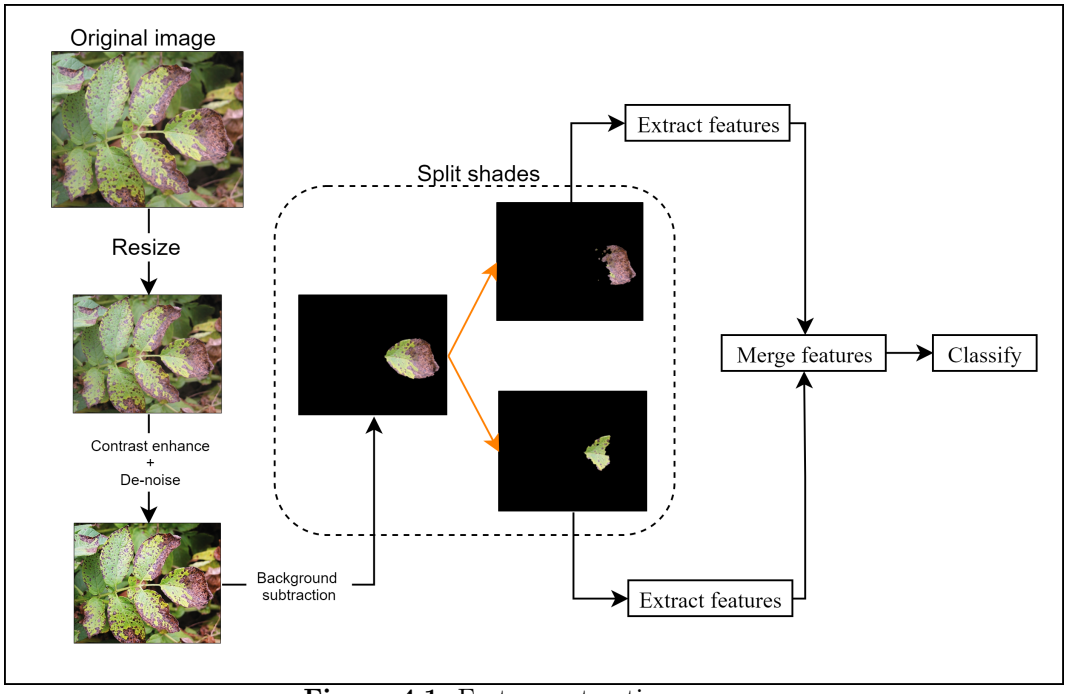


Figure 4.1: Feature extraction process

Refer to Figure 4.1 above. A sample image is first resized to a resolution of 640x480 to improve computational burden. The image is then contrast enhanced and a slight blur is added to reduce image noise. For background subtraction, a user interactive watershed algorithm is used to extract the required leaf. Numerous enhancement and noise reduction filters are used to assist with the subtraction process which are not mentioned. The leaf is then split between different shade clusters and each cluster undergoes feature extraction. These features are then merged to give a total of 356 features.

4.2 Simulation results

For each feature selection algorithm, the ideal combinations suggested in Table 3.23 and 3.24 are explored to determine the best subset of features within the ailment dataset. Note that computation time is calculated by adding the computation time of all necessary combinational tests per algorithm if applicable. Each feature is normalised to its respective z-score before further processing. The same LOOCV 1-NN classifier is used to verify classification accuracy and the initial accuracy with all features inclusive is 79.31 %. Figures 4.2 and 4.3 below show the best results for all algorithms. FRBPSO is given a separate figure to show its convergence.

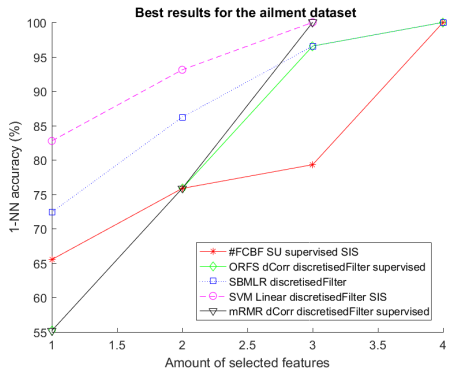


Figure 4.2: Best results excluding FRBPSO

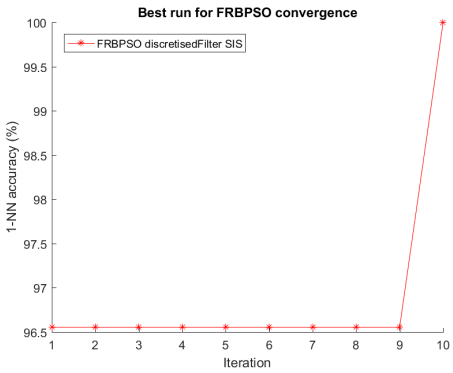


Figure 4.3: Best convergence run for FRBPSO

Table 4.2 below summarises the above results neatly. The FRBPSO results are taken from the best run, but the computation time considers all 10 runs. Recall that the computation time is calculated by adding the computation time of all necessary combinational tests per algorithm where applicable.

Table 4.2: Summary of case study results for each algorithm

Algorithm	Accuracy (%)	# features	Computation time (mm:ss)
mRMR	100	3	01:09
#FCBF	100	4	02:25
ORFS	100	4	02:39
FRBPSO	100	8	29:24
SVM-RFE	100	3	01:36
SBMLR	100	4	<00:01

Tables 4.3 and 4.4 below shows the combinations which provided the best results for each algorithm.

Table 4.3: Preference for the filter algorithms

Algorithm	Associative measure	Pre-processor	Supervised?
mRMR	dCorr	Discretise	Yes
#FCBF	SU	SIS	Yes
ORFS	dCorr	Discretise	Yes

Table 4.4: Preference for the wrapper and embedded algorithms

Algorithm	Specific preference	Pre-processor
FRBPSO	N/A	SIS + discretise
SVM-RFE	Linear	SIS + discretise
SBMLR	N/A	Discretise

Refer to Table 4.2. Each algorithm was able to give an accuracy of 100 % and therefore only a comparison to the amount of selected features and computation time can be made. As the results in section 3.3 shows, mRMR provides the least amount of features and FRBPSO provides the most amount of features. As with mRMR, SVM-RFE was also able to select the fewest amount of features. With regards to computation time, SBMLR is faster than any algorithm by many orders of magnitude as expected, and performs just as well as #FCBF and ORFS with regards to the amount of selected features. As discussed in section 3.3, #FCBF indeed took longer to compute than mRMR because of the extra experiments needed, therefore mRMR is the second fastest algorithm out of the lot. #FCBF also shows similar results to that of ORFS, being only slightly faster. FRBPSO, as expected, took much longer to compute and did the poorest when selecting the least amount of features.

Refer to Tables 4.3 and 4.4. MI did not perform as well as *dCorr* as an associative measure for neither mRMR nor ORFS for this dataset. With #FCBF, the supervised strategy (not supervised best subset) showed the most promise instead of the unsupervised approach. As for the inconclusive pre-processing filters, SIS was preferred for #FCBF, discretisation for ORFS and a combination of SIS and discretisation for SVM-RFE. Overall, mRMR or SVM-RFE is recommended for this dataset as they provide maximum classification accuracy with the least amount of features in under two minutes.

Chapter 5

Conclusion

This study decomposes and reviews six different types of feature selection algorithms, each with their own optimisation and pre-processing standards where applicable. It is observed that the three filter methods mRMR, FCBF# and ORFS perform equivalent or better when implemented as hybrid methods with the addition of supervision in both performance and computation time. In testing the filter methods with supervision, the associative measure best used for mRMR is MI or *dCorr*. For FCBF#, the SU measure is still preferred regardless of supervision and for ORFS, the MIC statistic is not recommended and either MI or *dCorr* should be used instead. As for the preferred pre-processing filter assuming supervision, mRMR prefers MDL discretisation, while FCBF# and ORFS does not favour any specific combination. Nevertheless, a pre-processing filter still provides better results than if it were not used. To obtain the optimal solution for mRMR, FCBF and ORFS, the amount of tests or individual experiments needed is two, six and six respectfully. This places mRMR in favour for the fastest algorithm to iterate through all experiments to obtain the best subset of features in the least amount of time. mRMR also shows the best classification accuracy performance over FCBF# and ORFS, making it the filter algorithm of choice.

For the two wrapper and one embedded methods SVM-RFE, FRBPSO and SBMLR, each except SBMLR has a variety of parameters for optimisation, and the parameters used in this study proved useful for gene expression microarray datasets. It is observed that FRBPSO favours a pre-processor including both of SIS and MDL discretisation, SVM-RFE does not show any preference and SBMLR favours MDL discretisation. When considering kernels for SVM-RFE and SBMLR, SVM-RFE swings in favour of Linear or RBF with neither being dominant and the use of kernels with SBMLR was not computationally feasible in this study. For the best results of SVM-RFE, six experiments are

needed to accompany all combinations of kernels and pre-processors, but is still much more computationally efficient than FRBPSO which shows the worst computation time of all the algorithms. As for SBMLR, it proved the fastest algorithm of them in orders of magnitude.

When comparing the algorithms to one another, SVM-RFE showed the best performance regarding classification accuracy with mRMR following close behind. FCBF# shows the worst performance, followed closely by SBMLR. The fastest algorithm as stated previously is SBMLR, with the slowest being FRBPSO. The algorithm which produced the best classification accuracy with the least amount of features is mRMR. ORFS showed similar performance to that of mRMR but was unable to improve upon it, while FRBPSO shows the worst performance with regards to removing the most features. The results from the case study also supports the results obtained from the gene expression datasets. Following the tests recommended for each algorithm will provide optimal results.

This study sheds light on the possible improvement over existing feature selection algorithms for high-dimensional data through pre-processing techniques, algorithm hybridisation and optimisation without prior domain knowledge of the datasets. The possibilities and combinations of such techniques are only hindered by the creativity of researchers in the field and further improvements in the field of feature selection can be expected in future. As a recommendation, using the hybrid mRMR as a standalone feature selection technique is feasible, but will greatly improve upon other more complicated feature selection methods as a filter pre-processor. If time is of importance and feature selection is required while training a predictive model, SBMLR will provide the fastest results. ORFS is not recommended over mRMR, but other feature selection algorithms which also avoid irrelevant redundancy might prove more successful. FRBPSO is not recommended due to its inability to select the fewest features and also the sheer amount of parameter optimisation possibilities. As for SVM-RFE, it proved the best classification accuracy in this study with fast computation time and high feature removal.

Although prior knowledge to the datasets are unknown, having some prior knowledge or indication of which features to extract is always a good point of departure. The feature selection algorithms can then be used to determine the most promising features within the initial prior knowledge. To elaborate, the case study focuses on ailments visible on plant foliage. A good point of departure would be to extract features such as colour and texture, of which the most important is unknown without expert domain knowledge. Therefore, even with a lack of expert domain knowledge regarding the best colour and texture features, all types of colour and texture features can be extracted, and the most important ones determined by the feature selection algorithms.

References

[1] A. Statnikov, I. Tsamardinos, Y. Dosbayev, and C. F. Aliferis, "GEMS: a system for automated cancer diagnosis and biomarker discovery from microarray gene expression data." *International journal of medical informatics*, vol. 74, no. 7-8, pp. 491–503, 8 2005.

[2] A. Shishkin, A. Bezzubtseva, and A. Drutsa, "Efficient High-Order Interaction-Aware Feature Selection Based on Conditional Mutual Information," *Nips*, no. Nips, pp. 1–9, 2016.

[3] Jason Brownlee, "An Introduction to Feature Selection," 2014. [Online]. Available: https://machinelearningmastery.com/an-introduction-to-feature-selection/

[4] P. Cassara, A. Rozza, and M. Nanni, "A Cross-Entropy-based Method to Perform Information-based Feature Selection," 2016. [Online]. Available: http://arxiv.org/abs/1607.07186

[5] D. Asir, S. Appavu, and E. Jebamalar, "Literature Review on Feature Selection Methods for High-Dimensional Data," *International Journal of Computer Applications*, vol. 136, no. 1, pp. 9–17, 2016. [Online]. Available: http://www.ijcaonline.org/research/volume136/number1/singh-2016-ijca-908317.pdf

[6] Y. Li, T. Li, and H. Liu, "Recent advances in feature selection and its applications," *Knowledge and Information Systems*, vol. 53, no. 3, pp. 551–577, 5 2017.

[7] P. Mitra, C. Murthy, and S. Pal, "Unsupervised feature selection using feature similarity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 301–312, 3 2002. [Online]. Available: http://ieeexplore.ieee.org/document/990133/

[8] L. Yu and H. Liu, "Feature Selection for High-Dimensional Data: A Fast

Correlation-Based Filter Solution," *International Conference on Machine Learning (ICML)*, pp. 1–8, 2003. [Online]. Available: http://www.aaai.org/Papers/ICML/2003/ICML03-111.pdf

[9] B. Senliol, G. Gulgezen, L. Yu, and Z. Cataltepe, "Fast Correlation Based Filter (FCBF) with a different search strategy," *2008 23rd International Symposium on Computer and Information Sciences, ISCIS 2008*, 2008.

[10] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[11] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of Bioinformatics and Computational Biology*, vol. 3, no. 2, pp. 185–205, 2005. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-17644384367&doi=10.1142%2FS0219720005001004&partnerID=40&md5=3b4794891c033270e74360b17be99cf5

[12] J. R. Berrendero, A. Cuevas, and J. L. Torrecilla, "The mRMR variable selection method: a comparative study for functional data," *Journal of Statistical Computation and Simulation*, vol. 86, no. 5, pp. 891–907, 2016. [Online]. Available: https://doi.org/10.1080/00949655.2015.1042378

[13] J. Xu, B. Tang, H. He, and H. Man, "Semisupervised Feature Selection Based on Relevance and Redundancy Criteria," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 9, pp. 1974–1984, 2017.

[14] C. O. Sakar, O. Kursun, and F. Gurgen, "A feature selection method based on kernel canonical correlation analysis and the minimum Redundancy-Maximum Relevance filter method," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3432–3437, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2011.09.031

[15] H. Lyu, M. Wan, J. Han, R. Liu, and C. Wang, "A filter feature selection method based on the Maximal Information Coefficient and Gram-Schmidt Orthogonalization for biomedical data mining," *Computers in Biology and Medicine*, vol. 89, pp. 264–274, 10 2017.

[16] S. An, Q. Hu, and D. Yu, "Fuzzy entropy based max-relevancy and min-redundancy feature selection," *2008 IEEE International Conference on Granular Computing, GRC 2008*, pp. 101–106, 2008.

[17] A. E. Akadi, A. Amine, A. E. Ouardighi, and D. Aboutajdine, "A two-stage gene selection scheme utilizing MRMR filter and GA wrapper," *Knowledge and Information Systems*, vol. 26, no. 3, pp. 487–500, 2011. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-79951514865&doi=10.1007%2Fs10115-010-0288-x&partnerID=40&md5=475d25335f1cc6c65c3f5d4d464e3842

[18] S. S. Shreem, S. Abdullah, M. Z. A. Nazri, and M. Alzaqebah, "Hybridizing relief, mRMR filters and GA wrapper approaches for gene selection," *Journal of Theoretical and Applied Information Technology*, vol. 47, no. 3, pp. 1338–1343, 2013. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84876059801&partnerID=40&md5=e21c14fabb7980e7cd0ea832414c1504

[19] N. S. Mohamed, S. Zainudin, and Z. Ali Othman, "Metaheuristic approach for an enhanced mRMR filter method for classification using drug response microarray data," *Expert Systems with Applications*, vol. 90, pp. 224–231, 2017. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85028024950&doi=10.1016%2Fj.eswa.2017.08.026&partnerID=40&md5=1a79033a4d6a8e7497f4308265a22bb8

[20] H. Alshamlan, G. Badr, and Y. Alohali, "MRMR-ABC: A hybrid gene selection algorithm for cancer classification using microarray gene expression profiling," *BioMed Research International*, vol. 2015, 2015. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84928809451&doi=10.1155%2F2015%2F604910&partnerID=40&md5=a6ae8312e40fa708f91bdf2e485c9fe1

[21] A. Chinnaswamy and R. Srinivasan, "Hybrid feature selection using correlation coefficient and particle swarm optimization on microarray gene expression data," Department of Computer Science and Engineering, Amrita School of Engineering, Amrita Nagar, Coimbatore, Tamilnadu, 641112, India, pp. 229–239, 2016. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84951962460&doi=10.1007%2F978-3-319-28031-8_20&partnerID=40&md5=2acc242ea1b76a9c8b2764497b14d02f

[22] B. Tran, B. Xue, M. Zhang, and S. Nguyen, "Investigation on particle swarm optimisation for feature selection on high-dimensional data: local search and selection bias," *Connection Science*, vol. 28, no. 3, pp. 270–294, 2016. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84969802864&doi=10.1080%2F09540091.2016.1185392&partnerID=40&md5=10e387dfe0f72d439b5d17253060ab95

[23] S. Agarwal, R. Rajesh, and P. Ranjan, "FRBPSO: A Fuzzy Rule Based Binary PSO for Feature Selection," *Proceedings of the National Academy of Sciences India Section A - Physical Sciences*, vol. 87, no. 2, pp. 221–233, 2017.

[24] Y. Zhang, D. W. Gong, X. Y. Sun, and Y. N. Guo, "A PSO-based multi-objective multi-label feature selection method in classification," *Scientific Reports*, vol. 7, no. 1, 12 2017.

[25] T. Abeel, T. Helleputte, Y. Van de Peer, P. Dupont, and Y. Saeys, "Robust biomarker identification for cancer diagnosis with ensemble feature selection methods," *Bioinformatics*, vol. 26, no. 3, pp. 392–398, 2 2010. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btp630

[26] S. Maldonado, R. Weber, and F. Famili, "Feature selection for high-dimensional class-imbalanced data sets using Support Vector Machines," *Information Sciences*, vol. 286, pp. 228–246, 2014.

[27] K.-B. Duan, J. C. Rajapakse, H. Wang, and F. Azuaje, "Multiple SVM-RFE for gene selection in cancer classification with expression data," *IEEE Transactions on Nanobioscience*, vol. 4, no. 3, pp. 228–233, 2005. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-26644466101&doi=10.1109%2FTNB.2005.853657&partnerID=40&md5=98284566c39f21be81cb8478be12ad55

[28] Y. Tang, Y.-Q. Zhang, and Z. Huang, "FCM-SVM-RFE gene feature selection algorithm for leukemia classification from microarray gene expression data," in *IEEE International Conference on Fuzzy Systems*, Department of Computer Science, Georgia State University, Atlanta, GA 30302-3994, United States, 2005, pp. 97–101. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-23944504100&partnerID=40&md5=66e693e4ea983c1793774088a9badb96

[29] J. Canul-Reich, L. O. Hall, D. Goldgof, and S. A. Eschrich, "Feature selection for microarray data by AUC analysis," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620, United States, 2008, pp. 768–773. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-69949144350&doi=10.1109%2FICSMC.2008.4811371&partnerID=40&md5=7875099d1abd6398a98a6cbad071e1c8

[30] P. A. Mundra and J. C. Rajapakse, "SVM-RFE with MRMR filter for gene selection," *IEEE Transactions on Nanobioscience*,

vol. 9, no. 1, pp. 31–37, 2010. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-77950538864&doi=10.1109%2FTNB.2009.2035284&partnerID=40&md5=bcdb879c0ce92deac8cf2a453fd5f876

[31] J. Zhang, X. Hu, P. Li, W. He, Y. Zhang, and H. Li, "A hybrid feature selection approach by correlation-based filters and SVM-RFE," in *Proceedings - International Conference on Pattern Recognition*, School of Computer and Information, Hefei University of Technology, Hefei, China, 2014, pp. 3684–3689. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84919918822&doi=10.1109%2FICPR.2014.633&partnerID=40&md5=11a637e78e75cf10d85745f3fa068b1b

[32] S. Maldonado, R. Weber, and J. Basak, "Simultaneous feature selection and classification using kernel-penalized support vector machines," *Information Sciences*, vol. 181, no. 1, pp. 115–128, 2011.

[33] J. Neumann, C. Schnörr, and G. Steidl, "Combined SVM-based feature selection and classification," *Machine Learning*, vol. 61, no. 1-3, pp. 129–150, 2005.

[34] W. Zhongxin, S. Gang, Z. Jing, and Z. Jia, "Feature selection algorithm based on mutual information and lasso for microarray data," *Open Biotechnology Journal*, vol. 10, pp. 278–286, 2016. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84995877236&doi=10.2174%2F1874070701610010278&partnerID=40&md5=db98471dc82e62e7407a78b72eacfc85

[35] M. Hasan, M. Hasan, and M. A. Mottalib, "Linear regression-based feature selection for microarray data classification," *International Journal of Data Mining and Bioinformatics*, vol. 11, no. 2, pp. 167–179, 2015. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84920864852&doi=10.1504%2FIJDMB.2015.066776&partnerID=40&md5=dfffb03696ced5172e4a123c18806553

[36] R. Zakharov and P. Dupont, "Ensemble logistic regression for feature selection," Machine Learning Group, ICTEAM Institute, Université Catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium, pp. 133–144, 2011. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-80455173733&doi=10.1007%2F978-3-642-24855-9_12&partnerID=40&md5=233326eaab513f77e20193ac01c1434c

[37] B. Krishnapuram, L. Carin, M. A. Figueiredo, and A. J. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 957–968, 2005.

[38] G. Cawley, N. Talbot, and M. Girolami, "Sparse multinomial logistic regression via bayesian l1 regularisation," *Neural Information Processing Systems*, vol. 19, p. 209, 2007. [Online]. Available: https://www.google.com/books?hl=tr&lr=lang_en&id=Tbn1l9P1220C&oi=fnd&pg=PA209&dq=Sparse+Multinomial+Logistic+Regression+via+Bayesian+L1+Regularisation&ots=V3meFmqq-Y&sig=5IddsZYFn8OtPN_0LPneoR_za0A

[39] Y. Liang, C. Liu, X.-Z. Luan, K.-S. Leung, T.-M. Chan, Z.-B. Xu, and H. Zhang, "Sparse logistic regression with a L1/2 penalty for gene selection in cancer classification," *BMC Bioinformatics*, vol. 14, no. 1, 2013. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84879049361&doi=10.1186%2F1471-2105-14-198&partnerID=40&md5=ef3a3c445551c2730d4e7822b68a0e13

[40] T. Hastie, R. Tibshirani, and J. Friedman, "Springer Series in Statistics The Elements of Statistical Learning Data Mining, Inference, and Prediction," in *The Elements of Statistical Learning*, 2nd ed. Stanford: Springer, 2017, p. 465. [Online]. Available: https://web.stanford.edu/~hastie/ElemStatLearn//printings/ESLII_print12.pdf

[41] G. J. Székely, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing dependence by correlation of distances," *Annals of Statistics*, vol. 35, no. 6, pp. 2769–2794, 2007.

[42] J. B. Kinney and G. S. Atwal, "Equitability, mutual information, and the maximal information coefficient," 2013.

[43] M. Radovic, M. Ghalwash, N. Filipovic, and Z. Obradovic, "Minimum redundancy maximum relevance feature selection approach for temporal gene expression data," *BMC Bioinformatics*, vol. 18, no. 1, pp. 1–14, 2017. [Online]. Available: http://dx.doi.org/10.1186/s12859-016-1423-9

[44] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 69, no. 6 2, 2004.

[45] J. Fan and J. Lv, "Sure independence screening for ultrahigh dimensional feature space," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 70, no. 5, pp. 849–911, 2008.

[46] U. M. Fayyad, "Multi- Interval Discretization of Continuous- Valued Attributes for Classification Learning - I C "..." *Machine Learning*, pp. 1022–1027, 1993.

[47] H. Liu, "Discretization: An Enabling Technique," *Data Mining and Knowledge*

Discovery, vol. 6, pp. 393–423, 2002. [Online]. Available: https://sci2s.ugr.es/keel/pdf/algorithm/articulo/liu1-2.pdf

[48] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, pp. 4104–4108, 1997. [Online]. Available: http://ieeexplore.ieee.org/document/637339/

[49] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3, pp. 1–1950, 1999.

[50] M. A. Arasomwan and A. O. Adewumi, "On the Performance of Linear Decreasing Inertia Weight Particle Swarm Optimization for Global Optimization," *The Scientific World Journal*, vol. 2013, p. 12, 2013. [Online]. Available: http://dx.

[51] C.-S. Yang, L.-Y. Chuang, C.-H. Ke, and C.-H. Yang, "Boolean Binary Particle Swarm Optimization for Feature Selection," *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 2093 – 2098, 2008.

[52] R. Baxter, N. Hastings, A. Law, and E. J. Glass, *An introduction to Support Vector Machines*, 1st ed. Cambridge University Press, 2008, vol. 39, no. 5.

[53] Y. Koshiba and S. Abe, "Comparison of L1 and L2 support vector machines," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3, 2003, pp. 2054–2059. [Online]. Available: http://ieeexplore.ieee.org/document/1223724/

[54] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear SVM," *Proceedings of the 25th international conference on Machine learning - ICML '08*, no. 2, pp. 408–415, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1390156.1390208

[55] C. ALLAUZEN, C. CORTES, and M. MOHRI, "A DUAL COORDINATE DESCENT ALGORITHM FOR SVMs COMBINED WITH RATIONAL KERNELS," *International Journal of Foundations of Computer Science*, vol. 22, no. 08, pp. 1761–1779, 2011. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0129054111009021#.VA2pIvW_EI4.mendeley

[56] C.-J. Hsieh, "ECS289 : Scalable Machine Learning," 2015. [Online]. Available: http://web.cs.ucdavis.edu/~chohsieh/scalable_ML_lectures/lecture1.pdf

[57] M. Hofmann, "Support Vector Machines—Kernels and the Kernel Trick," *Universität Bamberg*, pp. 1–16, 2006. [Online]. Available: http://www.cogsys.wiai.uni-bamberg.de/teachingarchive/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf

[58] E. S. Youn, "Feature Selection in Support Vector Machines," Ph.D. dissertation, University of Florida, 2002. [Online]. Available: http://etd.fcla.edu/UF/UFE1000171/youn_e.pdf

[59] Y. Xue, L. Zhang, B. Wang, Z. Zhang, and F. Li, "Nonlinear feature selection using Gaussian kernel SVM-RFE for fault diagnosis," *Applied Intelligence*, 2018.

[60] L. Zhang, W.-D. Zhou, and F.-Z. Li, "Kernel sparse representation-based classifier ensemble for face recognition," *Multimedia Tools and Applications*, vol. 74, no. 1, pp. 123–137, 1 2015. [Online]. Available: http://link.springer.com/10.1007/s11042-013-1457-1

[61] Zongben Xu, Mingwei Dai, and Deyu Meng, "Fast and Efficient Strategies for Model Selection of Gaussian Support Vector Machine," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 5, pp. 1292–1307, 10 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4808205

[62] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene Selection for Cancer Classification using Support Vector Machines," *Machine Learning*, vol. 46, no. 1/3, pp. 389–422, 2002. [Online]. Available: http://link.springer.com/10.1023/A:1012487302797

[63] C. W. Hsu and C. J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[64] C. Saunders, A. Gammerman, and V. Vovk, "Ridge Regression Learning Algorithm in Dual Variables," *Proceedings of the 15th International Conference on Machine Learning*, pp. 515–521, 1998. [Online]. Available: http://eprints.ecs.soton.ac.uk/8942/1/Dualrr_ICML98.pdf%5Cnhttp://eprints.soton.ac.uk/258942/

[65] Y. Feng, Y. Yang, Y. Zhao, S. Lv, and J. A. Suykens, "Learning with Kernelized Elastic Net Regularization," Tech. Rep., 2016. [Online]. Available: ftp://ftp.esat.kuleuven.be/pub/sista/yfeng/KENReg2014.pdf

[66] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu,

"Feature Selection: A Data Perspective," *ACM Computing Surveys*, vol. 50, no. 6, 1 2018. [Online]. Available: http://dx.doi.org/10.1145/3136625

[67] A. Cotter, J. Keshet, and N. Srebro, "Explicit Approximations of the Gaussian Kernel," pp. 1–11, 2011. [Online]. Available: http://arxiv.org/abs/1109.4603

[68] Y.-w. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-j. Lin, "Training and Testing Low-degree Polynomial Data Mappings via Linear SVM," *The Journal of Machine Learning Research*, vol. 11, pp. 1471–1490, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1859899%5Cnhttp://jmlr.org/papers/v11/chang10a.html