UNIVERSITY OF SOUTHAMPTON

Faculty of Physical Sciences and Engineering
Electronics and Computer Science

Comprehensive Review of Classification Algorithms for High Dimensional Datasets

by

Iwan Syarif

Thesis for the degree of Doctor of Philosophy

March 2014

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Thesis for the degree of Doctor of Philosophy

Comprehensive Review of Classification Algorithms for High Dimensional Datasets

by Iwan Syarif

Machine Learning algorithms have been widely used to solve various kinds of data classification problems. Classification problem especially for high dimensional datasets have attracted many researchers in order to find efficient approaches to address them. However, the classification problem has become very complicated and computationally expensive, especially when the number of possible different combinations of variables is so high. In this research, we evaluate the performance of four basic classifiers (naïve Bayes, k-nearest neighbour, decision tree and rule induction), ensemble classifiers (bagging and boosting) and Support Vector Machine. We also investigate two widely-used feature selection algorithms which are Genetic Algorithm (GA) and Particle Swarm Optimization (PSO).

Our experiments show that feature selection algorithms especially GA and PSO significantly reduce the number of features needed as well as greatly reduce the computational cost. Furthermore, these algorithms do not severely reduce the classification accuracy and in some cases they can improve the accuracy as well. PSO has successfully reduced the number of attributes of 9 datasets to 12.78% of original attributes on average while GA is only 30.52% on average. In terms of classification performance, GA is better than PSO. The datasets reduced by GA have better classification performance than their original ones on 5 of 9 datasets while the datasets reduced by PSO have their classification performance improved in only 3 of 9 datasets. The total running time of four basic classifiers (NB, kNN, DT and RI) on 9 original datasets is 68,169 seconds while the total running time of the same classifiers on GA-reduced datasets is

3,799 seconds and on PSO-reduced dataset is only 326 seconds (more than 209 times faster).

We applied ensemble classifiers such as bagging and boosting as a comparison. Our experiment shows that bagging and boosting do not give a significant improvement. The average improvement of bagging when applied to nine datasets is only 0.85% while boosting average improvement is 1.14%. Ensemble classifiers (both bagging and boosting) outperforms single classifier in 6 of 9 datasets.

SVM has been proven to perform much better when dealing with high dimensional datasets and numerical features. Although SVM work well with default value, the performance of SVM can be improved significantly using parameter optimization. Our experiment shows SVM parameter optimization using grid search always finds near optimal parameter combination within the given ranges. SVM parameter optimization using grid search is very powerful and it is able to improve the accuracy significantly. Unfortunately, grid search is very slow; therefore it is very reliable only in low dimensional dataset with few parameters. SVM parameter optimization using Evolutionary Algorithm (EA) can be used to solve the problem of grid search. EA has proven to be more stable than grid search. Based on average running time, EA is almost 16 times faster than grid search (294 seconds compare to 4680 seconds). Overall, SVM with parameter optimization outperforms other algorithms in 5 of 9 datasets. However, SVM does not perform well in datasets which have non-numerical attributes.

Keywords: *high dimensional data, feature selection, ensemble classifiers, Support Vector Machine, Evolutionary Algorithms, parameter optimization*

Contents

List of Tables

List of Figures

Declaration of Authorship

I, Iwan Syarif, declare that the thesis entitled *Comprehensive Review of Classification Algorithms for High Dimensional Datasets*, the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: (Syarif et al., 2012c)(Syarif et al., 2012b)(Syarif et al., 2012a)

Signed:

Date: 23rd February 2015

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Dr. Adam Prugel-Bennett and my advisor, Dr. Gary Wills for their excellent and continued supports, guidance and patience during my study and towards the completion of this thesis.

I would like to thank my external examiner Dr. Mohamed Gaber and my internal examiner Dr. Richard A. Watson for their excellent advises, detailed review and revision during and after the viva.

I gratefully acknowledge the funding sources that supported my PhD. I was fully funded by the Ministry of Information and Communication Technology, the Republic of Indonesia for the four years scholarship and was supported by the Ministry of Education, the Republic of Indonesia for the six months extension.

My time at Southampton University was really enjoyable due to the many friends especially PhD students in Building 32 ECS that became a part of my life. Thanks to my lab mates: Betty, Tas, Alper, Vangelis, Mark, Adil, Gunawan, Saad, Victor, etc. Special thanks to my best friend Agus Djunaedy who helped me printing the thesis, PPI Soton friends : Dwi, Gunawan Ariyanto, Gunawan Budi, Husni, Didiek, Niken, Fikri, Betty,

I would not have finished this thesis if not for my parents, Bapak Ir. Hanafi Pratomo and Ibu Dr. Supartini Hanafi who raised me with never ending love and prayed days and nights for the success of their son. This thesis would also not be possible without the love and patience of my beautiful wife, Dr. Tessy Badriyah and my beloved children Daisy, Defita and Pascal. Being with you, life seems so beautiful ☺

Iwan Syarif
Southampton University
March 2014

1. Introduction and Problem Statements

Machine Learning algorithms have been widely used to solve various kinds of data classification problems. Classification problems especially for high dimensional datasets have attracted many researchers in order to find efficient approaches to address them. However, the classification problem has become very complicated and computationally intensive, especially when the number of possible different combinations of variables is so high.

1.1 Classification of High Dimensional Data

Classification is a supervised learning technique which learns a function from training data set that consists of input features/attributes and categorical output (Gaber et al., 2007)(Kotsiantis, 2007). This function will be used to predict a class label of any valid input vector. The main goal of classification is to apply machine learning algorithms to achieve the best prediction accuracy (Williams et al., 2006)(Verleysen, 2003).

In the various applications of machine learning and data mining, the use of high dimensional datasets with hundreds or thousands of features is not unusual (Braun et al., 2012). In other words, modern data sets are very often in high dimensional space. Extracting knowledge from huge data requires new approaches. The more complex the datasets, the higher the computation time and the harder they are to be interpreted and analysed. Therefore, classification on high dimensional data has become a recurring problem; since it occurs in various data mining applications for which a decision step is necessary.

The problems of high dimensional data was apparently coined by Richard Bellman (Bellman, 1957) as *"the curse of dimensionality"*. These terms refer to various phenomena that arise when analysing and organising data in a high-dimensional space which have hundreds or thousands of dimensions that do not occur in low-dimensional setting. For example, a classification algorithm such as decision tree has time complexity of $O(nd^2)$ where d is the number of

attributes and *n* is the number of samples (Su and Zhang, 2006). It means as *d* becomes large, the complexity increases quadratically and the number of samples (n) may be too small to be used as learning data to generate an accurate classification model. Insufficient number of training samples makes the classification algorithms difficult to predict the class labels of the dataset correctly. This condition is called overfitting.

High dimensional data tends to have more complex problems that low-dimensional ones and hence it is harder to make inferences. There are at least three serious problems caused by high dimensional data: complexity, over-fitting and number of samples.

The impact of high dimensionality on classification is poorly understood (Fan and Fan, 2008). Many datasets such as microarray, DNA, proteomics, etc. have thousands or more features while the sample size (number of instances) is typically tens or less than hundred. Most of basic classifiers break down when the dimensionality is high. Miller reported that there is a well-known phenomenon that a prediction model built from thousands of attributes *(d)* but has a relatively small sample size *(n)* can be quite unstable (Miller, 2002). Other reseacher (Fan and Fan, 2008) reported that the difficulty of high-dimensional classification is mostly caused by the existence of many noisy features that do not contribute to the improvement of accuracy.

The above problem reveals the importance of dimensionality reduction on high dimension data classification. Dimensionality reduction is a process for reducing the number of random variables under consideration. There are some advantages of dimensionality reduction (Fodor, 2002):

- Most machine learning and data mining techniques may not be effective for high-dimensional data
- Query accuracy and efficiency degrade rapidly as the dimension increases
- Lower computational cost
- Help avoid over-fitting (training on highly-related features rather than contingent ones)

There are two different techniques of dimensionality reduction; the first technique is feature selection which is a process that chooses an optimal

subset of features according to an objective function (Holder et al., 2005)(Williams et al., 2006). The objectives of feature selection are to reduce dimensionality, to remove noise and to improve mining performance (speed of learning and predictive accuracy). In this technique, only partial parts of the original features are selected. The second technique is feature extraction which refers to the mapping of the original high-dimensional data onto a lower-dimensional space. In this technique, all original features are used and the transformed features are linear combinations of the original features. Both feature selection and feature extraction algorithms reduce the number of features needed.

1.2 How to improve the classification performance

Classification problem can be viewed as optimisation problem where the goal is to find the best model that represents the predictive relationships in the data (Otero et al., 2012). In this research, we use four well-known classical machine learning algorithms as base classifiers which are naive Bayes (NB), decision tree (DT), k-nearest neighbour (kNN) and rule induction (RI). Beside rule induction, the other three methods were selected as the top ten algorithms in Data Mining (Wu and Kumar, 2009).

Many researchers (Schapire et al., 1997)(Lee and Cho, 2010)(Graczyk et al., 2010) have reported that ensemble classifiers (meta learning) have better accuracy than single classification techniques. An ensemble classifier is a method which uses or combines multiple classifiers to improve the classification performance from any of the constituent classifiers. (Gaber et al., 2007)(Gaber and Bader-El-Den, 2012) reported that ensemble classifier can be used to avoid over-fitting of single classifer as well as to improve the robustness. In this research we apply, analyse and evaluate two ensemble classifier techniques called bagging and boosting.

Another way to achieve better classification performance is using more sophisticated classification techniques. Other than the well-known classical data mining techniques, Support Vector Machine (SVM) and Evolutionary Algorithm (EA) have gained more attention and have been adopted in data classification problems in order to find a good solution. SVM which is an emerging data classification technique proposed by (Cortes and Vapnik, 1995),

has been widely adopted in various fields of classification (Lin et al., 2008). SVM algorithm has an advantage that it is not affected by local minima, furthermore it does not suffer from the curse of high dimensionality because of the use of support vectors (Sánchez A, 2003). SVM was also considered as the top ten algorithms in Data Mining (Wu and Kumar, 2009).

Unfortunately, the SVM performance highly depends on parameter setting and its kernel selection. The selection quality of SVM parameters and kernel functions has an effect on the learning and generalization performance (Hric et al., 2011)(Sudheer et al., 2013). Appropriate kernel functions and their parameters should be selected to obtain an optimal classification performance (Aydin et al., 2011).

Generally, most of machine learning algorithms will not achieve optimal results if their parameters are not being tuned properly. To build a high accuracy classification model, it is very important to choose a powerful machine learning algorithm as well as adjust its parameters. Parameter optimization can be very time consuming if done manually especially when the learning algorithm has many parameters (Friedrichs and Igel, 2005)(Rossi and de Carvalho, 2008).

There are two methods to adjust the SVM parameter: grid search with cross-validation and evolutionary algorithm (EA). Evolutionary algorithm (EA) is commonly used on problems which are very hard to solve in a brute force technique (Eiben and Smith, 2003)(Barros et al., 2012). EAs search the solution space (the set of all possible inputs) of a difficult problem for the best solution, but not naively like a brute-force or grid search method. An EA uses mechanisms inspired by biological evolution such as reproduction, mutation, recombination and selection. In this research, we analyse various model parameter optimization technique for SVM classification which covers grid search approach and Evolutionary Algorithms.

1.3 Motivation and Problem Statements

In this thesis, we do not propose or develop new algorithms but we investigate and analyze some well known classification algorithms which are able to handle high dimensional datasets as shown in Figure 1.1.

Transforming high dimensional data to improve the classification accuracy as well as to reduce the computational complexity is a difficult research problem (Braun et al., 2012). In this research, we evaluate the performance of two feature selection algorithms which are Genetic Algorithm (GA) and Particle Swarm Optimizations (PSO). We wish to find the dimensionality reduction algorithm that contributes to the best classification accuracy and least computation time.

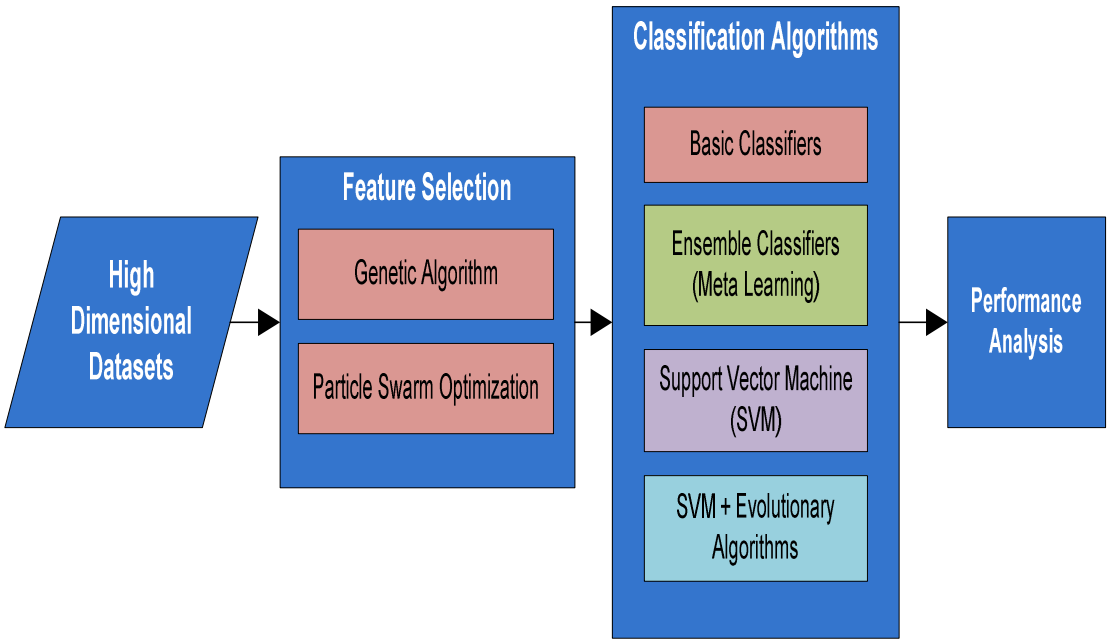

Figure 1.1 Research Outline

This research also evaluates the state of the art classification algorithms such as naïve Bayes, decision tree, k-nearest neighbour and rule induction. After that, we consider applying more sophisticated techniques which may have better performance than the basic algorithms. We used ensemble classifiers (bagging and boosting) and Support Vector Machine (SVM) to achieve better classification performance.

We also applied Support Vector Machine with four different kernels: linear, RBF, polynomial and sigmoid kernels. In order to get the best classification performance, we applied grid search and evolutionary algorithm (EA) to adjust the SVM parameters.

1.4 Objectives and Contributions of the Thesis

The main objectives of our research are explained in Figure 1.2 and the research contributions are described as follows:

- To propose the use of evolutionary algorithms for SVM parameter optimization to boost the performance of SVM. We would like to show that applying SVM with parameter optimization on GA-reduced datasets and PSO-reduced datasets outperforms other classification algorithms (basic classifiers and ensemble classifiers) which applied on both original datasets and reduced-datasets
- To apply and analyse the performance of four different kernels of SVM which are linear, RBF, polynomial and sigmoid kernels
- To apply ensemble classifiers which are bagging and boosting to improve the classification performance of four basic classifiers (naive Bayes, k-nearest neighbour, decision tree and rule induction)
- To apply and analyse the performance of GA and PSO as feature selection algorithms that significantly reduce the number of features of high dimensional datasets
- To apply and analyze the classification performance of four basic machine learning algorithm which are naive Bayes, k-nearest neighbour, decision tree and rule induction

The goal of this research is to answer the following questions:

- Does the use of SVM with parameter optimization using EA on reduced-datasets outperform other algorithms (four basic algorithms and ensemble classifiers) ?
- Is the use of ensemble classifiers such as bagging and boosting able to improve the classification performance of basic classifiers?
- What is the best dimensionality reduction algorithm that reduces the number of attributes significantly while still maintain/improve the accuracy as well as increase the speed? Which one better as feature selection algorithms, GA or PSO ?
- What is (are) the best machine learning technique(s) to handle high dimensional datasets, especially datasets which have a large number of attributes but have very limited number of examples (instances)?

- Is the use of sophisticated algorithms such as Support Vector Machine able to boost the classification performance? Which SVM kernel is the best to handle high dimensional datasets? How to adjust the SVM parameters to get the optimal results?

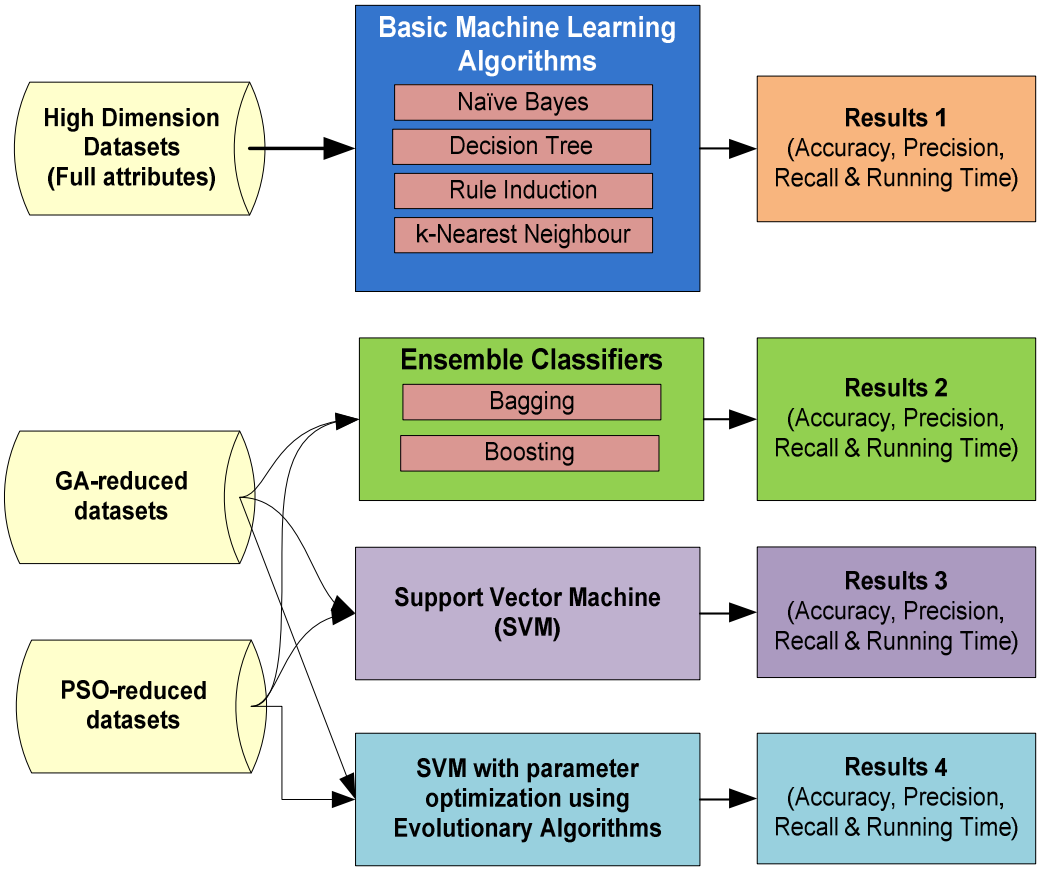

Figure 1.2 Experiments Scenario

1.5 Outline of the Thesis

The remainder of this thesis is structured as follows:
Chapter 2 presents a literature review, latest issues and research challenges of dimensionality reduction algorithms, basic machine learning classification algorithms, ensemble classifiers (bagging and boosting), Support Vector Machine, Evolutionary Algorithms and parameter optimization.

In Chapter 3, the design and implementation as well as performance analysis of two feature selection algorithms were explained in details. We selected Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) as feature

selection algorithms. These two algorithms were applied to nine high dimensional datasets.

Chapter 4 described the application of ensemble classifiers to improve the classification accuracy. We used bagging and boosting techniques to boost the performance of four basic classifiers: decision tree, rule induction, naïve Bayes and k-nearest neighbour.

Chapter 5 is about Support Vector Machine and its parameter optimization using specific techniques such as grid search and evolutionary algorithms. This chapter also compares the performance of all algorithms used in this research.

Chapter 6 presents conclusions and suggest future works.

List of Publications

Papers based on this work include:

- Syarif, Iwan, Prugel-Bennett, Adam and Wills, Gary (2012) Data mining approaches for network intrusion detection: from dimensionality reduction to misuse and anomaly detection. *Journal of Information Technology Review*, 3, (2), 70-83.
- Syarif, Iwan, Prugel-Bennett, Adam and Wills, Gary B. (2012) Unsupervised clustering approach for network anomaly detection. In, *Fourth International Conference on Networked Digital Technologies (NDT 2012), Dubai, UAE, 24 - 26 Apr 2012.* 11pp.
- Syarif, Iwan, Zaluska, Ed, Prugel-Bennett, Adam and Wills, Gary (2012) Application of bagging, boosting and stacking to intrusion detection. In, *MLDM 2012: 8th International Conference on Machine Learning and Data Mining, Berlin, Germany, 13 - 20 Jul 2012.* 10pp

2. Literature Review

This chapter consists of literature review of various important issues in this research which are dimensionality reduction, basic classification algorithms, meta-learning (ensemble classifiers), support vector machine, evolutionary algorithms and parameter optimization.

2.1 Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of random variables under consideration. This technique is a very important topic in data mining or machine learning area and it is widely used in specific applications such as image processing, bio-informatics, intrusion detection, email and web spam analysis, text classification and pattern recognition (Braun et al., 2012)(Fan and Fan, 2008).

One of the problems related to the high dimensional data is the fact that analyzing these data becomes more difficult and requires more advanced techniques. There are at least three serious problems caused by high dimensional data: complexity, over-fitting and the number of samples. (Mitchell, 1998)(Braun et al., 2012).

To build an effective classification model, dimensionality reduction is a very important issue because it will limit the number of input features in a classifier to produce a good predictive and less computationally intensive model (Huang and Dun, 2008). With a smaller feature subset, the rationale for the classification decision can be analysed and decided easier.

There are a lot of dimensionality reduction techniques but they can be divided into two categories: feature extraction and feature selection which explained in the following section.

2.1.1 Feature Extraction

In feature extraction, all available variables are used and the data is transformed using a linear transformation to a reduced dimension space. Its main goal is to replace the original variables by a smaller set of underlying variables (Tsai and Chan, 2007). Figure 2.1 gives an illustration about how the feature extraction works, x is an original data with d dimension while y is a new data with k dimension where $k<d$.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \rightarrow f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}\right) = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} = y \quad with\ k < d$$

Figure 2.1 Feature Extraction

There are many feature extraction algorithms but the most popular ones is Principal Component Analysis (PCA). PCA can be used to reduce the dimensionality of a data set by finding new variables which are smaller than the original but still retains most of the original data set information (Fodor, 2002)(Chen et al., 2006). PCA derives new variables that are linear combinations of the original variables by finding a few orthogonal linear combinations of the original variables with the largest variance (Dandpat and Meher, 2013). The new variables, called principal components (PCs), are uncorrelated and are in decreasing order of importance. So, the goal of PCA is to find a set of directions that maximizes the variances of the original data.

2.1.2 Feature Selection

Feature selection is a very important step in data pre-processing technique in data mining. It is a popular technique used to find the most important and optimal subset of features for building powerful learning models. An efficient feature selection method can eliminate irrelevant and redundant data; hence it can improve the classification accuracy (Oh et al., 2004)(Tjiong and Monteiro, 2011)(Liu et al., 2006). Feature selection problems are classified into two main categories: finding the optimal predictive features and finding all the relevant features for the class attribute.

Feature selection is actually a search problem for finding an optimal subset of n features out of an original N features (Witten and Frank, 2005)(Hall, 1999). It consists of four important parts:

1. Starting point

   Selection a point in the feature subset space is very crucial because it affects the direction of the search. There are three options; the first one is called forward selection, the search starts to proceed forward with no features and gradually add attributes. The second option is called backward elimination which is actually a converse of the previous one. The search proceeds backward through the search space, begins with all features then gradually removes them. The third option is for the search to begin somewhere in the middle and move outward from this point.

2. Search organization

   There are some search methods; the simplest one is an exhaustive search which searches all possibilities within the search space. If the dataset consists of N features, the search space will be 2^N. For large number of features (e.g. thousands attributes), an exhaustive search is infeasible. Another method is heuristic search which is more feasible than an exhaustive search but it can not guarantee to get the optimal results. More sophisticated searching techniques will be explained in more details in parameter optimization section.

3. Evaluation strategy

   The evaluation strategy is a method to evaluate the effectiveness of feature subsets. There are two different evaluation strategies which are filter and wrapper. In the wrapper method, the feature subsets are evaluated based on classifier's performance while in filter method the evaluation is based on some feature evaluation function.For example, the wrapper model proposed by (Kohavi & John, 1997) applies the classifier accuracy rate as the performance measure. Some researchers have concluded that if the purpose of the model is to minimize the classifier error rate, and the measurement cost for all the features is equal, then the classifier's predictive accuracy is the most important factor. The wrapper methods are usually slower than filter methods but they usually have better performance

because they are optimized for the particular learning algorithms used (Hall and Holmes, 2003).

4. Stopping criterion

   Every feature selection algorithm must have a stopping criterion which is used to decide when the search iteration stops. For example, a feature selector stops adding or removing features when the classification's performance does not improve after several iterations.

There are a lot of feature selection techniques, but in this paper we only select two algorithms: Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The GA and PSO algorithms will be discussed in more details in Sections 2.5.

2.2 Classification Algorithms

Classification problems have been extensively studied and it becomes one of the most popular research areas in data mining (Otero et al., 2012). The classification task consists of learning a predictive relationship between input features and a desired output. Each data point (or data instance) consists of a set of attributes and a class. The goal of classification algorithm is to create a model which represents the relationship between attributes values and class values and then use this model to predict the class label of new data. Classification problem can be viewed as optimisation problem where the goal is to find the best model that represents the predictive relationships in the data (Otero et al., 2012).

Numerous factors, such as incomplete data, and the choice of values for the parameters of a given model, may affect classification results. Classification problems have previously been solved with statistical methods such as logistic regression or discriminate analysis. Technological advances have led to the development of methods for solving classification problems, including decision trees, back-propagation neural networks, rough set theory and support vector machines (SVM). SVM which is an emerging data classification technique proposed by Vapnik, and has been widely adopted in various fields of classification (Lin et al., 2008).

The process of applying supervised machine learning algorithms to a real world problem is described in Figure 2.2 (Kotsiantis, 2007).

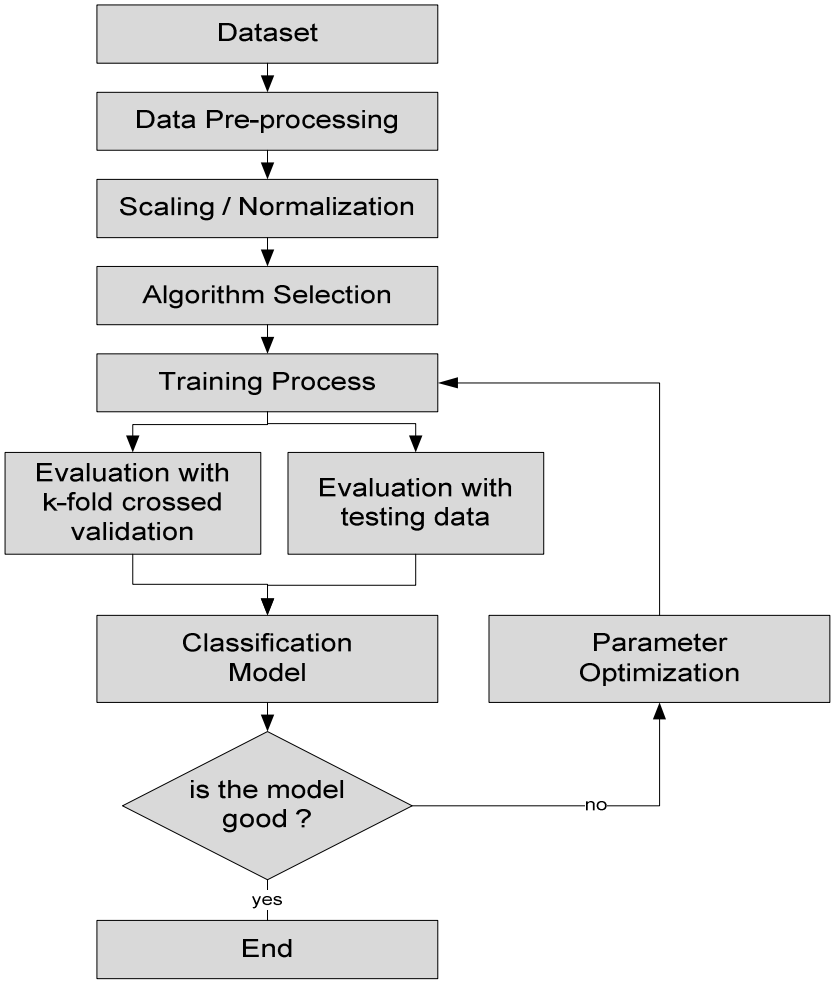

Figure 2.2 The process of supervised learning

A popular method for comparing various classification algorithms is to perform statistical comparisons of the accuracies. Comprehensive survey of classification methods can be found more details in (Gaber et al., 2007)(Fan-Zi and Zheng-Ding, 2004).

In this research, we use four well-known classical machine learning algorithms as base classifiers which are naïve Bayes, decision tree, k-nearest neighbour and rule induction.

2.2.1 Nearest Neighbour

The Nearest Neighbour (NN) algorithm was firstly introduced by J.G. Skellam (Skellam, 1952) where the ratio of expected and observed mean value of the nearest neighbour distances is used to determine if the data set is clustered. Even though it was invented more than seventy years ago, NN is still an active research area (Viswanath and Sarma, 2011). Among many supervised learning

algorithms, NN achieves consistently high performance (Islam et al., 2007)(Witten and Frank, 2005). However, this algorithm does not provide a model explicitly and it is also sensitive to the presence of irrelevant attributes (Geurts et al., 2009).

The k-Nearest Neighbour (k-NN) is a variant of NN where the result of new instance query is classified based on majority of k-NN category (Viswanath and Sarma, 2011) . *k*-NN is a type of instance-based learning or lazy learning where the function is only approximated locally and all computation is deferred until classification. The purpose of this algorithm is to classify a new object based on attributes and training samples. *k*-NN is one of the most fundamental and simplest classification methods which can be applied when there is little or no prior knowledge about the distribution of the data. The classification uses majority voting among the classification of the K objects. k-NN algorithm uses neighbourhood classification as the prediction value of the new query instance (k is a positive integer). k-NN algorithm determines the K-nearest neighbours based on the minimum distance from the query instance to the training samples. If k=1 then the new instance (data point) is simply assigned to the class of its nearest neighbour. The neighbours are taken from a set of training data for which the correct classification is already known.

The fundamental of k-NN algorithm has two important steps: Firstly, find the k instances which are nearest to the unseen data. Secondly, select the most or the majority of k neighbouring class (by referring label values). The pseudo-code for a k-NN classifier is shown inAlgorithm *1* below.

Algorithm 1 k Nearest Neighbour classifier

1: input dataset D $\{(x_1,c_1), \ldots , (x_n,c_n)\}$
2: for each instance (x_i,c_i) calculate $d(x_i,x)$
3: order $d(x_i,x)$ from lowest to highest
4: select the k nearest instance to x
5: assign to x the most frequent class

2.2.2 Decision Tree

Decision tree is a supervised learning algorithm which uses a tree to classify instances by sorting them based on features values (Kotsiantis, 2007). The goal is to create a model that predicts the value of a target variable based on several input variables (Geurts et al., 2009)(Barros et al., 2012)(Otero et al.,

2012)(Witten and Frank, 2005). The trees are generated by splitting on the values of attributes repeatedly and recursively. The main advantage of decision tree over many other classification techniques is that they produce a set of rules that are transparent, easy to understand and easily incorporated into real-time technologies. Another advantage is it does not require users to know a lot of background knowledge in the learning process. Furthermore, this algorithm is robust to noise, low computational cost for the generation model and flexible in dealing with redundant attributes (Barros et al., 2012). However, decision tree has also some disadvantages, when the dataset has too many categories the classification accuracy will significantly decrease. Furthermore, it is difficult to find rules based on the combination of several variables.

Decision tree algorithms begin with a set of examples and create a tree data structure that can be used to classify new examples. Each case is described by a set of attributes which can be numeric or nominal type. Each training data has a label which represents its class. Each internal node of this algorithm contains a test to decide what branch to follow from that node. The leaf nodes contain class labels instead of tests (Quinlan, 1993)(Kotsiantis, 2007).

At present there are a lot of decision tree algorithms, but C4.5 which was developed by Quinlan (Quinlan, 1993) is probably the most popular and the most frequently used among many researchers. The C4.5 algorithm uses an entropy-based criterion which is called the information gain ratio, in order to select the best attribute to create a node. C4.5 has been successfully applied to a wide range of classification problems and it is popularly used as an evaluation comparison of new classification algorithms (Witten and Frank, 2005). The detailed explanation of decision tree algorithms and C4.5 algorithm can be found in (Kohavi and Quinlan, 1999).

2.2.3 Rule Induction

Rule induction is a one of widely used machine learning techniques. The goal of rule induction is generally to induce a set of rules from data that captures all general knowledge within that data, and that is as small as possible at the same time (van den Bosch, 2000)(Witten and Frank, 2005)(Cohen, 1995)(Kotsiantis, 2007). During the learning phase, rules are induced from the training sample, based on the features and class labels of the training samples.

The rules that are extracted during the learning phase can easily be applied during the classification phase when new unseen test data is classified.

There are several advantages of rule induction. First of all, the rules that are extracted from the training sample are easy to understand for human beings. The rules are simple if-then rules. Secondly, rule learning systems outperform decision tree learners on many problems (Cohen, 1995). One disadvantage of rule induction, however, is that it scales relatively poorly with the sample size, particularly on noisy data.

RIPPER is a well-known rule based algorithms which was developed by Cohen (Cohen, 1995). It generates rules through an iterated growing and pruning process. Lee and Stolfo (Lee and Stolfo, 1998) applied RIPPER to learn the classification model of the normal and abnormal system call sequences. This improved rule induction algorithm is used to discover useful patterns of system features that describe program and user behaviour, then apply this feature to recognize anomalies and known intrusions.

2.2.4 Naïve Bayes

The naïve Bayes classifier is a supervised learning algorithm which widely used in data mining tasks due to its computational efficiency and competitive accuracy. This method estimates conditional class probabilities by applying Bayes theorem under the naïve assumption that the attribute values are mutually independent given the class (Geurts et al., 2009)(Kotsiantis, 2007).

The advantage of this algorithm is that it only requires a small amount of training data to predict the means and variances of the variables for classification. Because independent variables are assumed, naïve Bayes classifier only uses the variances of the variables of each class rather than the entire matrix to predict the class of new instance. Therefore, naïve Bayes algorithm is one of the fastest machine learning algorithms.

2.3 Meta Learning

An ensemble classifier is a method which uses or combines multiple classifiers to improve robustness as well as to achieve an improved classification performance from any of the constituent classifiers. Furthermore, this

technique is more resilient to noise compared to the use of a single classifier. This method uses a 'divide and conquer approach' where a complex problem is decomposed into multiple sub-problems that are easier to understand and solve.

Ensemble approaches (Schapire et al., 1997)(Dong and Han, 2004) have the advantage that they can be made to adapt to any changes in the monitored data stream more accurately than single model techniques. An ensemble classifier has better accuracy than single classification techniques. The success of the ensemble approach depends on the diversity in the individual classifiers with respect to misclassified instances (Lee and Cho, 2010). According to Polikar (Polikar, 2006), there are four ways to achieve this diversity, the first is to use different training data to train single classifiers, the second is to use different training parameters, the third is to use different features to train the classifiers and the final one is to combine different types of classifier.

(Dietterich, 1997) reported that there are three main reasons why an ensemble classifier is usually significantly better than a single classifier. Firstly, the training data does not always provide sufficient information for selecting a single accurate hypothesis. Secondly, the learning processes of the weak classifier might be imperfect, and thirdly, the hypothesis space being searched might not contain the true target function while an ensemble classifier can provide a good approximation.

(Gaber and Bader-El-Den, 2012) proposed a novel ensemble classifier called GARF (Genetic Algorithm based Random Forests) which used genetic algorithms to enhance the performance of random forests. They compared the performance of GARF against C4.5 decision tree, SVM and AdaBoost. They reported that GARF has been always superior than the original random forests and furthermore their novel approach has outperformed other classifiers on 8 of 15 datasets.

In this paper we evaluate and analyze two different ensemble classifier techniques, called bagging and boosting using various weak classifiers, such as nearest neighbour, decision tree, rule induction and naïve Bayes.

2.3.1 Bagging

Bagging, which means bootstrap aggregation, is one of the simplest but most successful ensemble methods for improving unstable classification problems. For example, weak classifiers, such as decision tree algorithms, can be unstable, especially when the position of a training point changes slightly and can lead to a very different tree. This method is usually applied to decision tree algorithms, but it also can be used with other classification algorithms such as naïve Bayes, nearest neighbour, rule induction, etc. The bagging technique is very useful for large and high-dimensional data, such as intrusion data sets, where finding a good model or classifier that can work in one step is impossible because of the complexity and scale of the problem.

Algorithm 2 Bagging pseudo-code

1: given a set of training data $D\{(x_1,y_1),....,(x_m, y_m)\}$
2: where m=the number of datasetfor each instance
3: for i = 1 to N
4: create bootstrap replicate dataset D'_i
5: $D'_i \leftarrow$ select m random examples from the training set with replacement
6: $h_i \leftarrow$ training base learning algorithm on D'_i
7: end for
8: make a plurality vote : $R(x) = \text{majority}(r_1(x), ... , r_N(x))$
9: select the highest voting score R(x) as a classification result

Bagging was first introduced by Leo Breiman (Breiman, 1996) to reduce the variance of a predictor. It uses multiple versions of a training set which is generated by a random draw with the replacement of N examples where N is the size of original training set. Each of these data sets is used to train a different model. The outputs of the models are combined by voting to create a single output. The bagging algorithm is explained in Algorithm *2* (Zhou, 2009).

2.3.2 Boosting

Boosting, which was introduced by Schapire et al. (Schapire et al., 1997), is an ensemble method for boosting the performance of a set of weak classifiers into a strong classifier. This technique can be viewed as a model averaging method and it was originally designed for classification, but it can also be applied to regression. Boosting provides sequential learning of the predictors. The first one learns from the whole data set, while the following learns from training sets based on the performance of the previous one. The misclassified

examples are marked and their weights increased so they will have a higher probability of appearing in the training set of the next predictor. It results in different machines being specialized in predicting different areas of the dataset (Graczyk et al., 2010).

In this research, we select AdaBoost algorithm, which is one of the most widely used boosting techniques for constructing a strong classifier as a linear combination of weak classifiers. The AdaBoost algorithm was first introduced by Freund and Schapire (Freund and Schapire, 1997) and has been shown to solve many of the practical difficulties of earlier boosting algorithms, since it has solid theoretical foundation and produces very accurate predictions. Details of the boosting algorithm and its pseudo-code were given in (Zhou, 2009).

2.4 Support Vector Machine

Support Vector Machine (SVM) which was firstly proposed by Vladimir Vapnik and Corinna Cortes (Cortes and Vapnik, 1995), is a supervised learning technique based on statistical learning theory that can be applied to classification, regression and pattern recognition. A SVM is a kind of binary classifiers which takes a set of input data and then classifies each input into two possible classes or categories. The idea is to map the *n*-dimensional input space into a higher dimensional feature space and then the new feature space is classified by constructing a linear classifier. In SVM, a data point is viewed as a p-dimensional vector and SVM will separate them with a (p-1) dimensional hyperplane. The SVM algorithm has an advantage that it is not affected by local minima, furthermore it does not suffer from the curse of high dimensionality because of the use of support vectors (Sánchez A, 2003).

2.4.1 How SVM works

Figure 2.3 shows that there are many possible hyperplanes that might separate two classes perfectly but we must find the best hyperplane that represents the largest separation between the two classes. SVM constructs a hyperplanein a high dimensional space which maximises the margin between the hyperplane and the two classes.

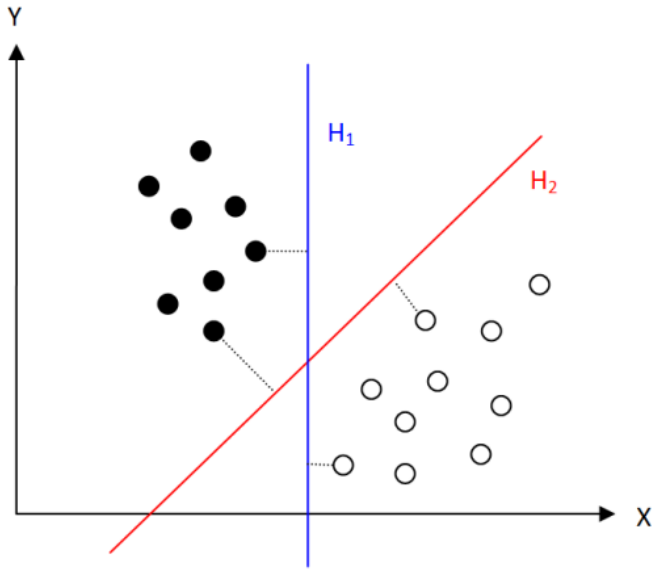

Figure 2.3 Two classes separated by hyperplanes

In 2 dimensions, two groups can be separated by a line using ax + by ≥ c for the first group and ax + by≤ c for the second group. There are a lot of possible solutions (hyperplanes) as shown in Figure 2.4.

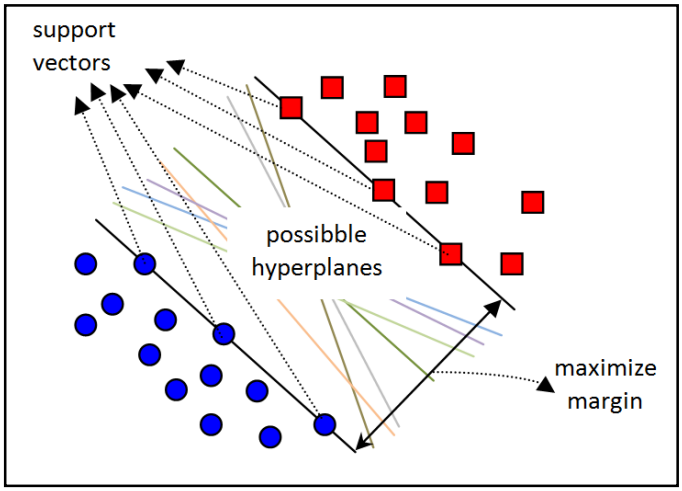

Figure 2.4 Possible hyperplanes and support vectors

In order to choose the best possible hyperplane and minimize the risk of over-fitting, it is very important to find the one with the maximal margin between the two classes. How to find the optimal hyperplane is an optimization problem which can be solved by Lagrangian formula. Once the optimal hyperplane is found, only the data points nearest to the hyperplane will be given a positive weight while others are set to zero. The data points where their distances are the closest to the decision surface are called support

vectors and they are the most critical elements of the training data. The position of dividing hyperplane would be changed or shifted if the support vectors removed.

The distance between a data point (x_0,y_0) and a line ax+by+c=0 can be measured using Formula 2.1 below:

$$\frac{|ax_0 + by_0 + c|}{\sqrt{(a^2 + b^2)}} \tag{2.1}$$

We have *L* training data where each instance X_i has *D* attributes and has 2 classes -1 and 1. We assume that the training data is linearly separable, therefore we can draw a hyperplane separating the two classes. This hyperplane can be described as *x.w-b=0* where w is normal to the hyperplane. H_1 is the hyperplane for class 1 and H_2 is the hyperplane for class 2. H_1: x_i.w-b = 1 and H_2: x_i.w-b = -1. The perpendicular distance from the hyperplane to the origin is $\frac{b}{\|w\|}$. All points which closest to H_1 and H_2 are support vectors.

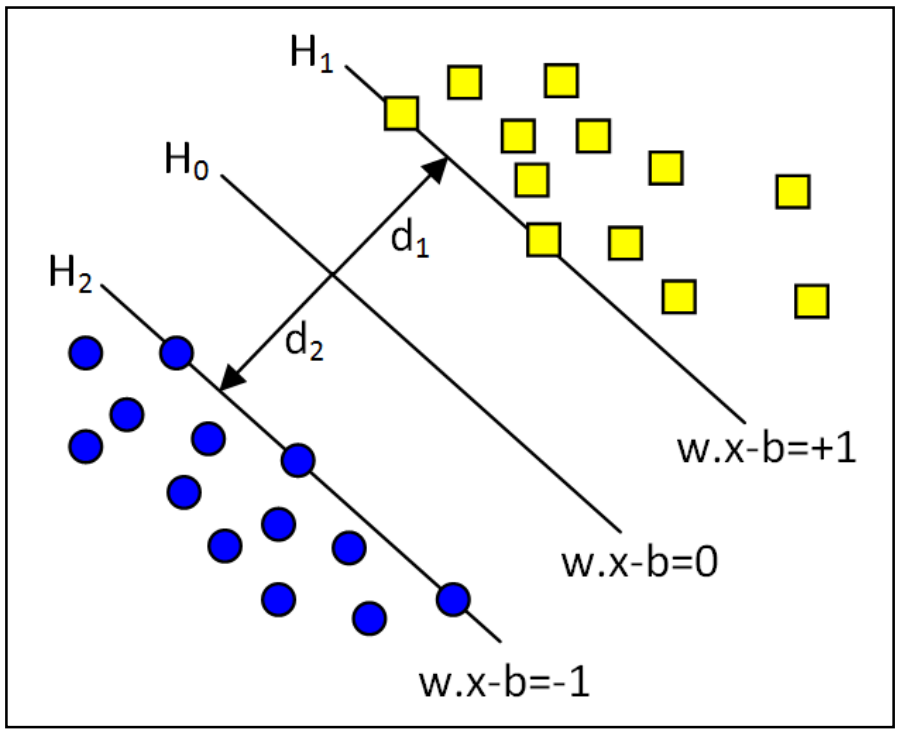

Figure 2.5 Finding the optimal separating hyperplane in SVM

Based on Figure 2.5 above, we define d_1 is the distance from H_1 to the hyperplane and d_2 from H_2 to it. The SVM margin is the distance from H_1 to H_2 which is d_1+d_2.

The distance between H0 and H1 is:

$\frac{|w.x+b|}{\|w\|} = \frac{1}{\|w\|}$, hence the distance between H1 and H2 is $\frac{2}{\|w\|}$

The distance between two hyperplanes (H1 and H2) could be maximized by minimizing the value of $\|w\|$. The margin is equal to $\frac{1}{\|w\|}$ and can be maximized using this following formula:

$$min\|w\| \text{ such that } y_i(x_i.w + b) - 1 \geq 0 \, \forall_i \tag{2.2}$$

Minimizing $\|w\|$ is equivalent to minimizing $\frac{1}{2}\|w\|^2$ and then we can apply Quadratic Programming (QP) optimization. We need to find$min \frac{1}{2}\|w\|^2$ such that $y_i(x_i.w + b) - 1 \geq 0 \, \forall_i$. Minimization could be proceed by applying Lagrange multipliers α, where $\alpha_i \geq 0 \, , \forall i$

$$L = \frac{1}{2}\|w\|^2 - \alpha[y_i(x_i.w + b) - 1 \, , \forall i] \tag{2.3}$$

$$L = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} \alpha_i[y_i(x_i \, .w + b) - 1] \tag{2.4}$$

$$L = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} \alpha_i y_i(x_i \, .w + b) + \sum_{i=1}^{L} \alpha_i \tag{2.5}$$

From the derivates = 0, we get:

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i, \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.6}$$

2.4.2 Kernel Trick

In many cases the data points are not linearly separable, in this case the input data can be transformed using a nonlinear mapping (φ) into another dimension space (Hric et al., 2011). In this new mapping, a linear boundary can be found. When mapping data into a higher dimension space, the computational complexity of the algorithm increases. To build a classification model, the learning process iterates through all the data points and to update the weights for the model, a large number of operations need to be made. Fortunately, the

calculation of the dot product between all instances can be calculated in the lower-dimension space by substituting a kernel function into the equation, this technique is called *kernel trick*. A kernel trick is a technique that depends on the training data only through dot-products. Kernel function can be interpreted as a measuring the similarity of x and x' (Sánchez A, 2003).

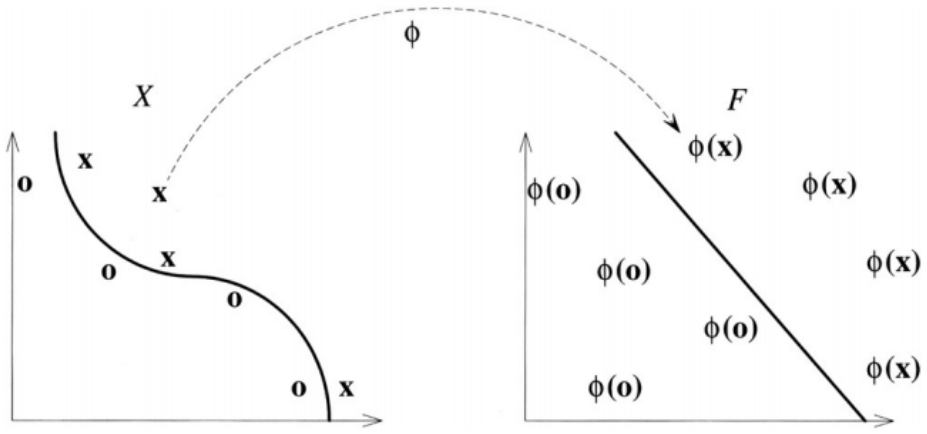

Figure 2.6 Moving a dataset into higher dimension[1]

Suppose there are *l* data points (instances) *x* and each instance consists of a binary label *y* (-1, 1), the SVM algorithm classifies the instances based on this following formula:

$$y(x) = sign\left[\sum_{i=i}^{l} \alpha_i y_i K(x_i, x) + b\right] \qquad (2.7)$$

Where α_i and b are real constant and $K(x_i, x)$ is the inner product operation.

If we assume that there are no data points between H1 and H2 then we can define the SVM hard margin which is only applicable to handle a linearly separable dataset:

$$minimize_{w,b}\frac{1}{2}\|w\|^2 \text{ subject to : } y_i[w^T x_i + b] \geq 1 \quad i = 1, \ldots, n \qquad (2.8)$$

In reality, most of the data is often not linearly separable so applying the equation above may produce classification errors. If the separating hyperplane is not possible, we can use a *soft margin* method which will select the hyperplane that split the training data as good as possible. This method

[1] http://www.music.mcgill.ca/~alastair/621/porter11svm-summary.pdf

introduces new variables called slack variables (ξ) to allow for finding a hyperplane that misclassifies some of the data points because many datasets are not linearly separable. The slack variables measure the degree of misclassification for each data point.

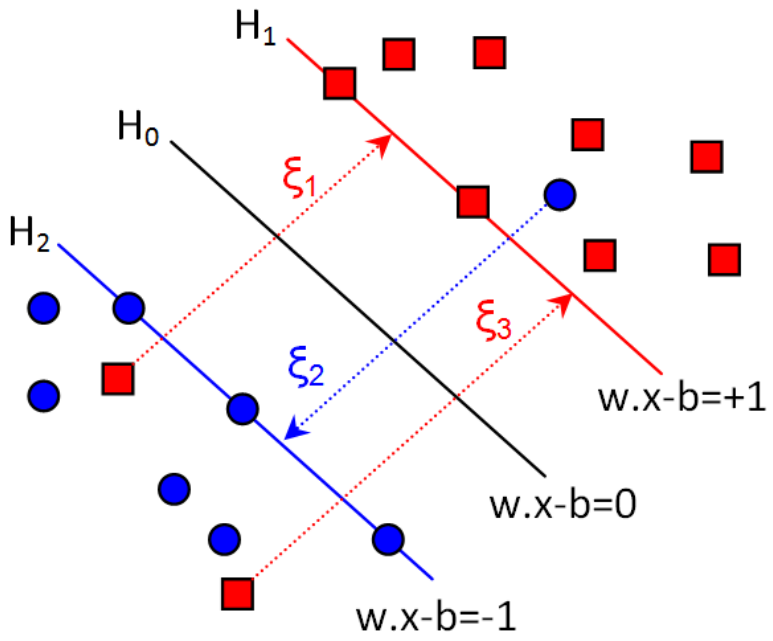

Figure 2.7 Calculate the degree of misclassification using slack variables

The above equation will be:

$$minimize_{w,b}\frac{1}{2}\|w\|^2 + C\sum_{i=i}^{N}\xi_i \,\text{subjecttoy}_i[w^T x_i + b] \geq 1 - \xi_i \qquad (2.9)$$

$1 \geq \xi_i \geq 0$ point is between margin and correct side of the hyperplane while $\xi_i >$ 1 point is misclassified.

A penalty function is added to the problem as the total of all slack variables. The constant C is used to control the trade-off between the margin and the size of the slack variables (Howley and Madden, 2005).

Using the Langrangian multipliers we will get a dual formulation as follow:

$$maximize_{\alpha}\sum_{\mathrm{i=1}}^{\mathrm{n}}\alpha_{\mathrm{i}} - \frac{1}{2}\sum_{\mathrm{i,j=1}}^{\mathrm{n}}\alpha_{\mathrm{i}}\,\alpha_{\mathrm{j}}\mathrm{y_i y_j K}(\mathrm{x_i},\mathrm{x_j}) \text{ subject to } \sum_{i=1}^{n} y_i\alpha_i = 0, 0 \leq \alpha_i \leq C \qquad (2.10)$$

The data points with $\alpha_i > 0$ are located on the margin or within the soft margin are *support vectors*. The SVM formula written in Equation2.10 depends on the data only through dot products which usually called kernel function. In Equation 2.10, the kernel function is $\mathrm{K}(\mathrm{x_i}, \mathrm{x_j}) = \langle \phi(x_i), \phi(x_j) \rangle$. The performance of a SVM classifier is highly dependent on the choice of a proper kernel function (Hussain et al., 2011)(Sánchez A, 2003) .

2.4.3 SVM Kernels

SVM can efficiently perform non-linear classification using the kernel trick by mapping their input into high-dimensional feature spaces.The most frequently used kernel functions are the linear, radial basis function (RBF)), sigmoid and polynomial kernel.

2.4.3.1 Linear Kernel

Linear kernel function is described below:

$$k(x_i, x_j) = (x_i^T x_j) + c \qquad (2.11)$$

The linear kernel has only one tuneable parameter which is c. Linear kernel performs very well and very fast on linearly separable datasets, unfortunately most real world problems are not linearly separable.

2.4.3.2 RBF (Gaussian) Kernel

The Gaussian or RBF kernel produces a mapping equivalent to an infinite dimensional Hilbert space. Therefore this function is able to map a wider variety of data sets. The RBF kernel is described below:

$$k(x_i, x_j) = exp\left(-\frac{1}{2\sigma^2}\|x_i - x_j\|\right) \qquad (2.12)$$

Alternatively, it could also be implemented using

$$k(x_i, x_j) = exp\left(-\gamma\|x_i - x_j\|^2\right) \qquad (2.13)$$

The RBF is generally applied most frequently, because it can classify non-lineary separable data, unlike a linear kernel function. Additionally, the RBF has fewer parameters to set than a polynomial kernel. The adjustable parameter γ plays a major role in the peformance of the kernel.

RBF and other kernel functions have similar overall performance. Consequently, RBF is an effective option for kernel function. (Sánchez A, 2003)(Mierswa, 2006)(Chang and Lin, 2011) reported that the RBF kernel performs well if the number of features is much larger than the size of dataset but it does not work well on noisy data.

2.4.3.3 Sigmoid Kernel

A sigmoid kernel function is equivalent to a two-layer percepton neural network. The sigmoid kernel comes from the neural network field where the sigmoid function is often used as an activation function for aritificial neurons. The sigmoid kernel function is described below:

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + c) \qquad (2.14)$$

There are two adjustable parameters in the sigmoid kernel: c and γ

2.4.3.4 Polynomial Kernel

The polynomial kernel function is described in the equation below:

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0 \qquad (2.15)$$

Compare to other SVM kernels, the polynomial kernel has more parameters that need to be optimized. Beside C andγ (gamma), it has at least 2 more important parameters: the polynomial degree *d* and the degree coefficient *r*. The parameter d should be set carefully, if the value of d is too large then the kernel values may go to infinity or zero.

2.5 Evolutionary Algorithms

Evolutionary algorithm (EA) is commonly used on problems which are very hard to solve in a brute force technique. EAs search the solution space (the set of all possible inputs) of a difficult problem for the best solution, but not naively like a brute-force or grid search.

An EA uses mechanisms inspired by biological evolution such as reproduction, mutation, recombination and selection. In nature, individuals are continuously developing and adapting to their environment while in EA, each individual is a candidate solution to the target problem which is evaluated by a fitness function. At each generation, the best individuals have a higher probability of

being selected for reproduction (Barros et al., 2012). The selected individual then produces new offspring or a new generation through crossover and mutation. This process is continuously repeated until a terminating condition achieved.

There are two important operators of EA, the first one is variation operators (recombination and mutation) which enrich the diversity as well as facilitate novelty. The second one is selection operator which reduces diversity and acts as force pushing quality. The use of both operators is able to improve the fitness values in consecutive populations. The fundamental of EA is explained in Algorithm *3* below (Eiben and Smith, 2003).

Algorithm 3*Evolutionary Algorithms pseudo-code*

```
1: CREATE an initial population (usually at random)
2: EVALUATE each candidate
3: REPEAT
4: SELECT some pairs to be parents (SELECTION)
5: COMBINE pairs of parents to create offspring (RECOMBINATION)
6: MUTATE the offspring (MUTATION)
7: SELECT some population members to be replaced
8: by the new offspring (REPLACEMENT)
9: UNTIL exit criteria is satisfied
```

There are many different variants of EAs but the general concept behind all these methods is the same: given a population of individual, natural selection (survival of the fittest) and the fitness of population (Eiben and Smith, 2003). Most of the present implementation of EA comes from these three basic types: Genetic Algorithms (GA), Evolutionary Programming (EP) and Evolutionary Strategies (ES).These variant techniques are quite similar but different in the implementation details and the nature of the particular applied problem. These algorithms have different representations (type of internal data structure) used to store the individuals): genetic algorithm (GA) uses binary strings, genetic programming (GP) uses trees, evolution strategies (ES) uses real-valued vectors, evolutionary programming uses finite state machine (Eiben and Smith, 2003).

Representation: The candidate solutions (individuals) are encoded in chromosomes which contain genes. Genes are usually in fixed position called loci and have a value. In order to find the global optimum, every feasible solution must be represented in genotype space. Selecting an appropriate

genotype representation is critical work for reducing the processing time of an EA.

Evaluation (Fitness Function): is a function used to measure the quality of phenotype which forms the selection process. The fitness function represents the requirement that the population should adapt to and it is usually optimization or minimization problems.

Population: the population of chromosomes is randomly initialized and it is representation of possible solutions.

Parent Selection Mechanism: the parents are usually selected based on their finesses or high quality individuals but it could not guarantee achieve optimal solution. It needs a special trick to avoid getting trapped in local optima.

Variation Operators: operators are used to generate new candidate solutions, they are usually divided based on number of inputs: mutation, recombination and crossover operator.

Mutation: mutation is a unary operator applied on one genotype. It is very essential to maintain the randomness and to create the diversity.

Recombination: used to merge information from parents into offspring. Some of offspring may be worse while others are the same as the parents.

Survivor Selection: there are three methods of survival selection, the first one is called fitness based which ranks the parents and offspring based on their fitness value then take the best. The second one is age based, make offspring as parents and then delete all previous parents. The third one is the combination of both techniques which is usually called *elitism*.

In this research, we only focused on GA and PSO. We used both GA and PSO for feature selection algorithms and we also used GA to optimize the SVM parameters.The use of GA in feature selection has been investigated by several researchers (Oh et al., 2004)(Hall and Holmes, 2003)(Roy and Bhattacharya, 2008)(Syarif et al., 2012a).

2.5.1 Genetic Algorithm (GA)

The Genetic Algorithm (GA) technique was originally proposed by John Holland in the 1975 as an experiment to see if the computer programs could eveolve in the Darwinian sense. GA has been applied to many function optimization problems and has been shown to be good in finding optimal and near optimal solutions. Its robustness of search in large search spaces and its domain independent nature motivated its applications in various fields (Krishna and Murty, 1999). GA can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly non-linear (Malhotra et al., 2011) .

The GA is a method for solving optimization problems that is based on natural selection, the process that drives biological evolution. GA repeatedly modifies a population of individual solutions. At each step, the GA selects individuals at random from the current population to be parents and uses them produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution.

The GA uses three main types of rules at each step to create the next generation from the current population:

1. *Selection rules* select the fitter individuals called *parents* that contribute to the population at the next generation.
2. *Crossover rules* combine two parents to form children for the next generation.
3. *Mutation rules* apply random changes to individual parents to form children.

There are two main differences betwen standard optimization algorithm and GA. First, classical algorithm generates a single point at each iteration where the sequence of points approaches an optimal solution. GA generates a population at each iteration where the best point in the population approaches an optimal solution. Second, classical algorithm selects the next point in the sequence by a deterministic computation while GA uses random number generators (Krishna and Murty, 1999).

The detail of GA is explained in the Algorithm *4* below (Mitchell, 1998).

Algorithm 4 *Genetic Algorithm pseudo-code*

1. **[Start]** Generate random population of *n* chromosomes
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. **[New population]** Create a new population by repeating following steps
4. **A. [Selection]** Select two parent chromosomes from a population according to
5. their fitness (the better fitness, the bigger chance to be selected)
6. **B. [Crossover]** With a crossover probability cross over the parents to form
7. a new offspring (children).
8. If no crossover was performed, offspring is an exact copy of parents.
9. **C. [Mutation]** With a mutation probability mutate new offspring
10. at each locus (position in chromosome).
11. **D. [Accepting]** Place new offspring in a new population
12. **[Replace]** Use new generated population for a further run of algorithm
13. **[Test]** If the end condition is satisfied, **stop**,
14. and return the best solution in current population
15. **[Loop]** Go to step **2**

2.5.2 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is an evolutionary computation technique that was first developed by Kennedy and Eberhart (1995) and is inspired by the behaviour of bird flocking to reach destination not completely known. PSO is powerful, easy to implement and computationally efficient (Huang and Dun, 2008). PSO is also an effective and flexible technique to explore the search space of a problem (Schuh et al., 2012). Like other evolutionary algorithms, PSO performs searches using a population (called swarm) of individuals (called particles) that are updated from iteration to iteration (Tjiong and Monteiro, 2011). To discover the optimal solution, each particle changes its searching direction according to two factors, its own best previous experience (called personal best or *pbest*) and the best experience of the whole swarms (called global best or *gbest*). The local best of a particle can be considered as the cognitive part while the global best particle is considered as the social part (Schuh et al., 2012)(Tjiong and Monteiro, 2011)(Korürek and Doan, 2010).

Each particle in the swarm represents one possible solution to the problem. At first, the swarm of particles are given a random initial location and velocity and are updated based on these following equations:

$$v_{i,j}^{t+1} = \omega v_{i,j}^{t} + c_1 r_1 \left(p_{i,j} - x_{i,j}^{t}\right) + c_2 r_2 \left(p_{g,j} - x_{i,j}^{t}\right) \quad (2.14)$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1} \quad (2.15)$$

Where x is the position of the particle i,v is its velocity, j is the dimension, t is time and ω is the inertial weight which represents how much of the previous velocity is retained while exploring. C_1 and c_2 are learning factor, r_1 and r_2 are weighting parameters, $p_{i,j}$ is local best while $p_{g,j}$ is global best particle. For each iteration, the fitness of each particle is calculated then the personal best and global best are also updated using Equation 2.14 and 2.15. Once the termination criteria is achieved, PSO will have good fitness, a set number of generations or a convergence factor such as a threshold for minimum population change.

The detail of PSO is explained in theAlgorithm *5* below (Mitchell, 1998)(Moraglio et al., 2008).

Algorithm 5 *Particle Swarm Optimization pseudo-code*

1. for all particle *i* do
2. initialize position x_i and velocity v_i
3. end for
4. while stop criteria not met do
5. for all particle *i* do
6. set personal best x_i as best position found so far by the particle
7. set global best g as best position found so far by the whole swarm
8. end for
9. for all particle *i* do
10. update velocity using: $v_{i,j}^{t+1} = \omega v_{i,j}^{t} + c_1 r_1 \left(p_{i,j} - x_{i,j}^{t}\right) + c_2 r_2 \left(p_{g,j} - x_{i,j}^{t}\right)$
11. update position using: $x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1}$
12. end for
13. end while

2.6 Parameter Optimisation

Generally, most of machine learning algorithms will not achieved optimal results if its parameters are not being tuned properly. To build a high accuracy classification model, it is very important to choose a powerful machine learning algorithm as well as adjust its parameters. Parameter optimization can be very time consuming if done manually especially when the learning algorithm has many parameters (Friedrichsand Igel, 2005)(Rossi and de Carvalho, 2008).

The SVM performance highly depends on parameter setting and kernel selection. The selection quality of SVM parameters and kernel functions has an effect on the learning and generalization performance. Appropriate kernel function and its parameters should be selected to obtain an optimal classification performance (Aydin et al., 2011)(Subasi, 2013). Many researchers showed (Hric et al., 2011)(Subasi, 2013)(Chang and Lin, 2011)(Huang and Dun, 2008) that the SVM classification accuracy among the four most popular kernels (linear, RBF, polynomial, sigmoid) can vary significantly based on a given training dataset. The selection of SVM parameters is actually an optimization problem in which search algorithms are used to find the best configuration of parameters for given problem (de Miranda et al., 2012)(Gaspar et al., 2012). To select a proper parameter for a specific SVM kernel is an important research issue in the machine learning area.

Aydin et. al. (Aydin et al., 2011) reported that there are two crucial points in SVM, the first one is how to choose the optimal input feature subset and the second one is how to set the best kernel and penalize its parameters. (Zhou and Chun-De, 2006) proposed a hybrid optimization selection method for SVM parameters and features selection using an immune algorithm. An immune algorithm (IA) is one of evolutionary algorithms which have abilities of learning, memorizing and self-adaptive control. They argued that IA can effectively solve the conflict between local search and global search in the feature selection and parameter selection process. They used IA to optimize the SVM kernel, to penalize its parameters and to select the most important features.

(Schuh et al., 2012) argued that the SVM kernel parameters not only need to be optimized, but the kernel itself might also contain both implicit and explicit domain knowledge which requires an expert's interpretation. They proposed to use Particle Swarm Optimization (PSO) and Genetic Programming (GP) as evolutionary approaches to find effective SVM kernel functions for various training data. Another researcher Subasi (Subasi, 2013) also applied PSO to optimize SVM parameters for electro-myography (EMG) signal classification. He reported that his approach achieved better performance compare to Nearest Neighbour algorithm, RBF classifier and a conventional SVM.

(de Miranda et al., 2012) used the combination of meta learning and PSO to optimize γ and C parameters. They used PSO to maximize the success rate and minimize the number of support vectors of the model and they also argued that meta-learning can be used to treat parameter selection as a supervised learning task. The parameter configuration suggested by meta-learning is used as initial population of the search technique, hence the search process would converge faster and be less expensive. However, meta learning is very dependent on the quality of its meta examples, unfortunately the number of problems available for meta-example generation is very limited and usually contains noisy data which needs to be cleaned.

Huang et. al. (Huang and Dun, 2008) proposed a novel PSO-SVM model to improve the classification accuracy by doing feature selection and SVM kernel parameter setting. They implemented a distributed architecture using web service technology to reduce the computational time. (Sudheer et al., 2013) also applied hybrid PSO-SVM for forecasting monthly stream flow. The PSO was used to optimize the SVM parameter and they reported this technique has successfully improved the forecasting performance.

There are several methods to adjust the SVM parameter: grid search with cross-validation, genetic algorithm (GA) and particle swarm optimization (PSO). Two major RBF parameters applied in SVM, C and γ must be set appropriately. Parameter C represents the cost of the penalty. The choice of value for C influences on the classification outcome. If C is too large, then the classification accuracy rate is very high in the training phase, but very low in the testing phase. If C is too small, then the classification accuracy rate unsatisfactory, making the model useless. Parameter γ has a much greater influence on classification outcomes than C, because its value affects the partitioning outcome in the feature space. An excessively large value for parameter γ results in over-fitting, while a disproportionately small value leads to under-fitting (Pardo and Sberveglieri, 2005).

Grid search is a computationally demanding process: by increasing the number of parameters and reducing the interval between discrete values, the number of possible combinations increases exponentially. The grid search algorithm is an alternative way to find the best SVM parameters. A logarithmic grid search method will be performed to find the best selection of C and γ. We will select

the C and the ɣ which produce the best average cross-validation accuracy. Selecting more SVM parameters will create very large combinations. For example there are three parameters to be optimized: C, γ (gamma) and d (degree) where each has 20 steps then the number of combinations would be 8,000 (20x20x20). This method has proven to be more effective and efficient than manual search, as it provides more accurate decision models in shorter time. The problem of a large number of calculations can be solved by parallel computing, or by employing the cloud computing techniques or by implement smarter machine learning algorithms.

The use of evolutionary algorithm for parameter optimization is very much faster and often gives better results than greedy search and grid search (Friedrichs and Igel, 2005)(Rossi and de Carvalho, 2008). Furthermore, it is very useful if ranges of the parameters are not known at all. For example, it is very difficult to find the correct ranges of SVM parameters especially C and γ (gamma).

(Sudheer et al., 2013) applied PSO to adjust the SVM parameters, other researchers (Schuh et al., 2012) applied GP and PSO into various SVM kernels on three different data sets. They reported that their approaches are feasible, effective and quite promising. However, finding the ranges of SVM kernel parameters as well as GP and PSO parameters are also difficult problems. Another issue is the combination of GP and PSO require very high computational cost.

3. Dimensionality Reduction

In this chapter, we applied GA and PSO into high dimensional datasets and analysed its affect before and after reduction based on classification accuracy and execution time.

3.1 Dimensionality Reduction System Design

(Hall and Holmes, 2003)(Syarif et al., 2012a) reported that if the data has many irrelevant, redundant and noisy features, the constructed model will have poor classification performance as well as higher computation cost. In this chapter, we do some experiments to find a more effective dimensionality reduction algorithm that produces better classification accuracy. The proposed system is shown in Figure 3.1 below.

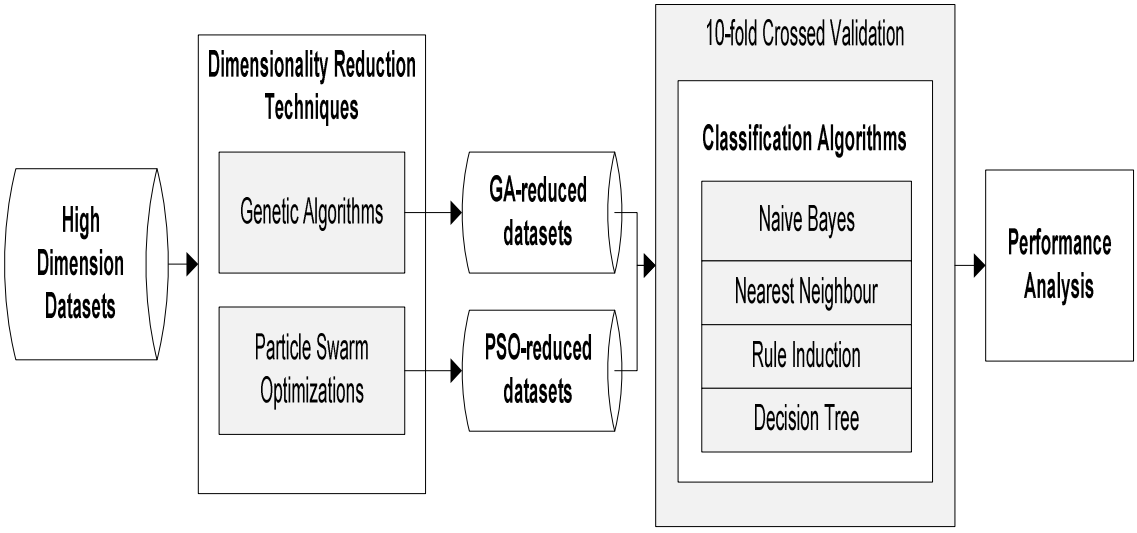

Figure 3.1 Dimensionality Reduction System Design

We used GA and PSO feature selection modules from Weka, saved the results into different files and then tested the new datasets using four classification algorithms (naïve Bayes, decision tree, rule induction and nearest neighbour) which provided by RapidMiner. The implementation of our proposed dimensionality reduction system is shown in Figure 3.2.

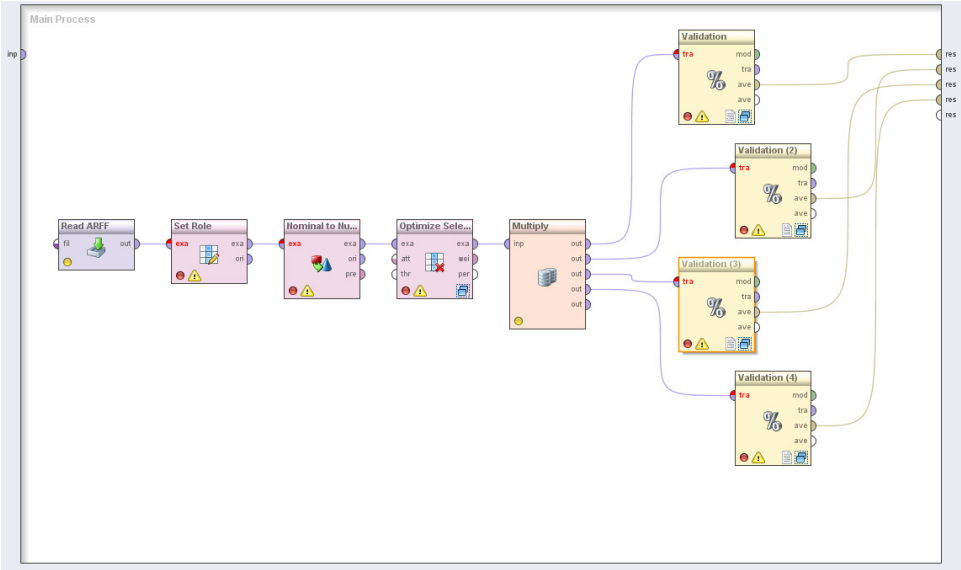

Figure 3.2 The Implementation of Dimensionality Reduction in RapidMiner

3.2 Dimensionality Reduction Algorithms

The following section explain how the two algorithms GA and PSO used to reduce the dimensionality. As described in Chapter 2 there are two types of feature selection methods which are filter and wrapper technique. We chose to apply wrapper techniques because they most often achieve better results than the filter techniques because they are trained and adjusted to the specific machine learning algorithm (Hall and Holmes, 2003)(Chen et al., 2006). In this research, GA and PSO are used as search algorithms to find the best features or subsets from original datasets as shown in Figure 3.3.

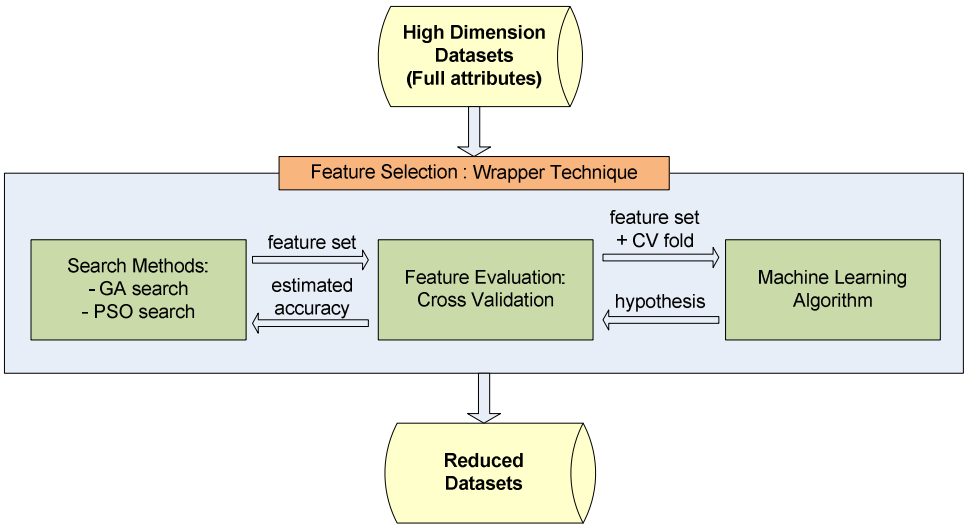

Figure 3.3 Feature selection using wrapper technique

3.2.1 Genetic Algorithm Search

We used GA-based feature selection developed by Mark Hall (Hall, 1999) which has been integrated to WEKA Data Mining Tools. A GA is used as a search technique to find the optimal subset. A solution is stored in fixed length binary string which represents a subset of original features. The value of each position in the string means presence for 1 and absence for 0. A new generation is randomly generated as an initial process then finding the optimal subset of original features is actually an iterative process. A generation is produced by applying genetic operators such as crossover and mutation to the member of population (current generation) in each iteration.

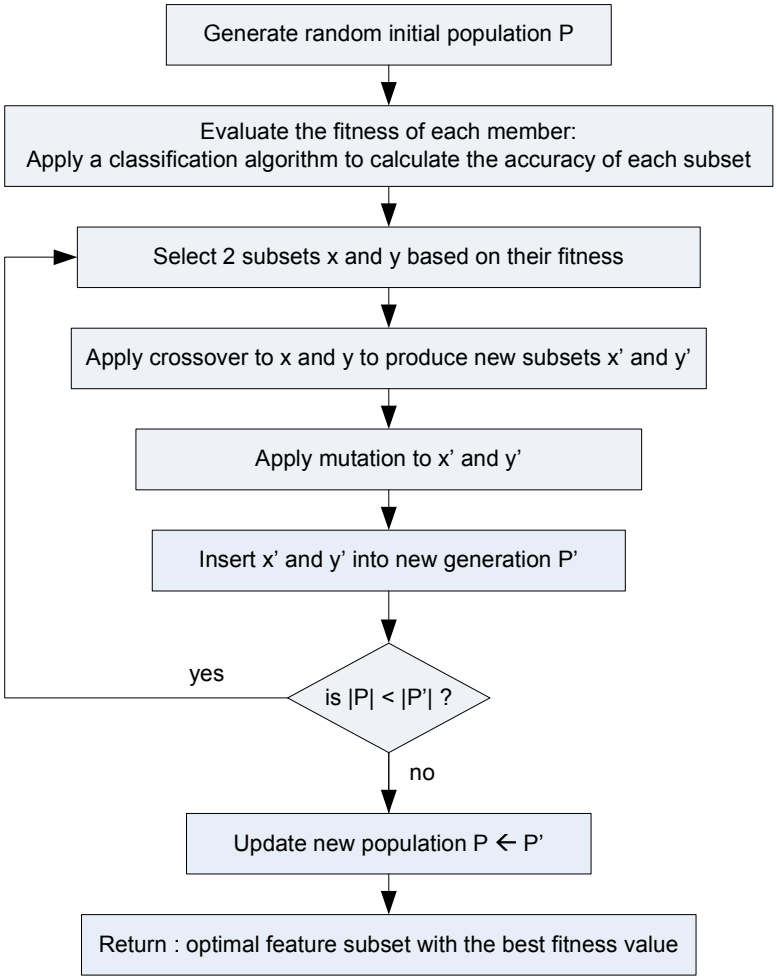

Figure 3.4 Feature selection using Genetic Algorithm

Crossover operator combines two different subsets and then generatesa new pair of subset. The mutation operator changes some of values which mean randomly adding or removing features in subset. To produce a better generation, a couple of members (usually called parents) are carefully selected

using the fitness function. The iteration will be stopped if there is no more generation to process. The flowchart of GA-based feature selection is described in the Figure 3.4 (Hall, 1999).

3.2.2 Particle Swarm Optimization Search

Beside GA, we also applied PSO algorithm for feature selection which was originally proposed by Moraglio et al. (Moraglio et al., 2008) and then the implementation in Java was written by Sebastian Luna Valero[2] . PSO uses slightly different terms compare to GA. In PSO, a solution is called a particle and the population is called swarm of particles. Similar to GA, the first process of PSO is generation of initial swarm of particles. Unlike GA, PSO does not need complex operators such as crossover and mutation; it only uses simple mathematical operators. Furthermore, PSO has less computational cost and needs a small amount of memory.

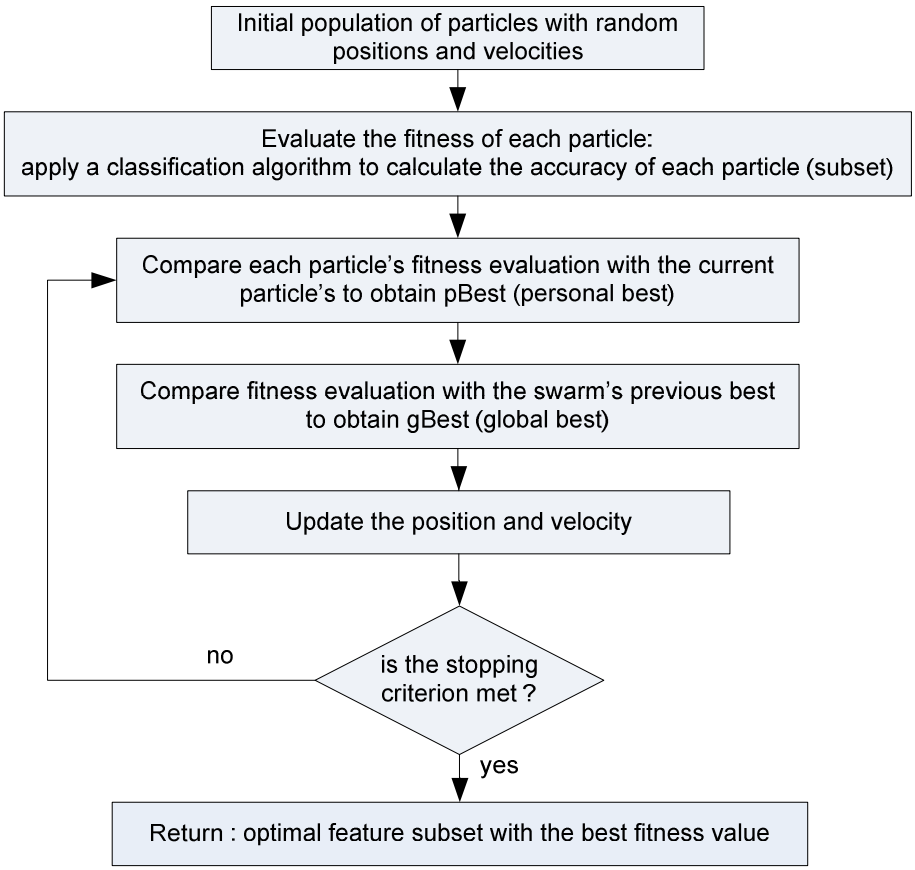

Figure 3.5 PSO search for feature selection

[2]PSOSearch, http://www.cs.waikato.ac.nz/ml/weka/packageMetaData/PSOSearch

The PSO algorithm has been described in Section 2.1.2 but there is a little modification when it applies to feature selection problem. The flow chart of PSO search is explained in Figure 3.5 (Jwo and Chang, 2009).

3.3 Performance Measurement

The metric used to evaluate the performance of classifier is given below (Davis and Goadrich, 2006):

Table 3.1 Performance metric

		Predicted Label	
		Positive	**Negative**
Actual Label	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

The accuracy rate and false positive rate are measured using the following formulae:

$$Tue\ Positive\ Rate = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad , \qquad False\ Positive\ Rate = \frac{FP}{\mathrm{FP} + \mathrm{TN}}$$

Many researchers use accuracy and false positive rate as performance measurement for classification problems, but other researchers (Davis and Goadrich, 2006)(Kotsiantis, 2007)(Williams et al., 2006)(Davis and Goadrich, 2006) argue that accuracy and false positive rates are not enough and simply using accuracy results can be misleading. They suggest accuracy, precision, recall and ROC curve as better performance measurement methods.

Precision is the percentage of positive predictions that are correct. Recall or sensitivity is the percentage of positive labelled instances that were predicted as positive. Specificity is the percentage of negative labelled instances that were predicted as negative. Accuracy is the percentage of correctly classified instances over the total number of instances.

To evaluate the performance offeature extraction and feature selection algorithms, there are other measurements called balance error rate (BER), fraction of features (FF) and fraction of probes (FP).

Table 3.2 Classification performance measurement

Measure	Formula
Precision	$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}}$
Recall / Sensitivity	$Recall/Sensitivity = \frac{\text{TP}}{\text{TP} + \text{FN}}$
Selectivity	$Selectivity = \frac{\text{TN}}{\text{FP} + \text{TN}}$
Accuracy	$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$
F-Measure	$F - Measure = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

Balanced Error Rate (BER) is the average of the errors on each class

$$BER = 0.5 * (\frac{FN}{\text{TP} + \text{FN}} + \frac{FP}{\text{FP} + \text{TN}})$$

Fraction of Features (FF) is the ratio of the number of features used by the classifier to the total number of features in the dataset.

Fraction of Probes (FP) is the ratio of the number of probes used to the number of feature used by a classifier. Probes are additional features added to the original datasets because of their similar distributions to the original features.

Area Under Curve (AUC) is an area under the ROC curve. This area is equivalent to the area under the curve obtained by plotting TP/(TP+FN) against TN/(FP+TN) for each confidence value, starting at (0,1) and ending at (1,0). The area under this curve is calculated using the trapezoid method. In the case when no confidence values are supplied for the classification the curve is given by {(0,1),(TN/(FP+TN),TP/(TP+FN)),(1,0)} and AUC = 1 - BER.

To test and evaluate various classification algorithms, we use *k*-fold cross validation (in most cases, we set *k*=10). In this method the dataset is divided into *k* subset and executed in *k* iteration. One of the *k* subsets is used as the testing data while the others are used as a training data in each iteration. The performance measurement is calculated across all *k* iterations. This technique can be used to analyse how well the classifiers will perform on unseen data.

3.4 The Datasets

We used nine high dimensional datasets which have the number of features from 45 attributes (the smallest) until 20,000 attributes (the highest). There is no exact definition of high dimensional data and also there is no agreement among researchers about the limit between low and high dimensional data. Clarke et. al. (Clarke et al., 2008) and Verleysen (Verleysen, 2003) argued that the data are high dimensional if it has hundreds or thousands of features for each example.

Since it is very difficult to get publicly available high dimensional datasets, we found 9 datasets from various websites but only 6 of them have attributes more than 100. We only used datasets which has 2 classes because we also wish to apply SVM algorithm to these datasets. The original SVM classifier is only able to handle two labelled classes. The list of datasets is shown in Table 3.16 below.

Table 3.3 High-dimensional datasets

	Dataset Name	Missing values	Number of instances	Number of attributes	Classes
1	Leukemia	no	72	7,130	all, aml
2	Embryonal Tumours	no	60	7,130	0,1
3	Dexter	no	600	20,000	1, -1
4	Internet_ads	yes	3,279	1,559	ad, nonad
5	Madelon	no	2,600	501	1, -1
6	Musk	no	6,598	168	0,1
7	Spambase	yes	4,601	58	0,1
8	SPECTF Heart	no	80	45	0,1
9	Intrusion	no	25,192	42	

From 9 datasets, there are 2 datasets (internet_ads and spambase) have missing values and there are 2 datasets have unbalanced data (internet_ads and musk). Internet_ads has 2 labelled: ad (14%) and nonad (86%), musk dataset also has 2 labeled: 0 (83%) and 1 (17%).

3.5 Experimental Results

In the first experiment, we applied four basic classifiers into original datasets. After that we applied the same classifiers into datasets that have been reduced by GA and PSO.

3.5.1 Experiments on Original Datasets

We applied four basic classifiers which are naive Bayes, k-nearest neighbour, decision tree and rule induction using 10 fold cross validation.

3.5.1.1 Naive Bayes

The naive Bayes algorithm in RapidMiner has no parameter. The results of naive Bayes experiments on original datasets are shown in Table 3.4.

Table 3.4 Naive Bayes algorithm results on original datasets

Data set	Number of instances	Number of attributes	Performance measurement with 10-fold cross validation				
			Accuracy	Precision	Recall	F Measure	Execution Time (hh:mm:ss)
Leukemia	72	7,130	98.57%	100.00%	96.67%	98.31%	00:00:01
Embryonal Tumours	60	7,130	68.33%	79.41%	70.00%	74.41%	00:00:01
Dexter	600	20,000	80.33%	77.79%	85.33%	81.39%	00:00:13
Internet_ads	3,279	1,559	96.86%	97.10%	99.33%	98.20%	00:00:10
Madelon	2,600	501	59.58%	59.81%	58.31%	59.05%	00:00:02
Musk	6,598	168	89.62%	96.74%	90.79%	93.67%	00:00:02
Spambase	4,601	58	81.81%	96.29%	72.78%	82.90%	00:00:01
SPECTF Heart	80	45	75.00%	71.17%	90.00%	79.49%	00:00:01
Intrusion	25,192	42	86.34%	99.71%	70.91%	82.88%	00:00:09

The naive Bayes algorithm is relatively fast and it only needs a small amount of training data to estimate the means and variances to build a prediction model. It takes only around 13 second to apply 10-fold cross validation technique on dexter dataset which has 20,000 attributes and it needs less than one second to execute the same technique on leukemia and embryonal tumours datasets which has 7,130 attributes. And it is also quite fast to classify a dataset with a large number of instances such as intrusion dataset which has more than 25,000 attributes.

3.5.1.2 k Nearest Neighbour

The k-NN algorithm in RapidMiner has 3 parameters:

1. k: set the number of nearest data points , the default value is 1
2. measure types: is used for selecting the type of measure to calculate the distance from the data point to the nearest neighbor. There are four options: mixed measure, nominal measure, numerical measure and Bregman divergences. The default value is mixed measure.
3. mixed measures: this parameter is only available when the measure type='mixed measures'. It has only one option: mixed Eucledian distance.

We used the default value for all k-NN parameters. The results of kNN experiments are shown in Table 3.5.

Table 3.5 k-nearest neigbour algorithm results on original datasets

Data set	Number of instances	Number of attributes	Performance measurement with 10-fold cross validation				
			Accuracy	Precision	Recall	F Measure	Execution Time (hh:mm:ss)
Leukemia	72	7,130	91.61%	96.00%	84.17%	89.70%	00:00:01
Embryonal Tumours	60	7,130	60.00%	76.00%	60.00%	67.06%	00:00:01
Dexter	600	20,000	86.33%	85.61%	87.67%	86.63%	00:00:27
Internet_ads	3,279	1,559	86.00%	96.40%	86.99%	91.45%	00:01:51
Madelon	2,600	501	65.04%	65.26%	64.54%	64.90%	00:00:24
Musk	6,598	168	95.23%	97.89%	96.43%	97.15%	00:00:59
Spambase	4,601	58	82.57%	85.73%	85.51%	85.62%	00:00:11
SPECTF Heart	80	45	63.75%	59.83%	77.50%	67.53%	00:00:01
Intrusion	25,192	42	99.54%	99.57%	99.45%	99.51%	00:08:37

kNN is one of the simplest machine learning algorithms. All of the training data are stored in an n-dimensional space. kNN classifies a new instance by a majority vote of its neighbours. If k=1, the new instance is directly assigned to the class of its nearest neighbour. There is no explicit training phase in kNN.

3.5.1.3 Decision Tree

There are 6 parameters of decision tree algorithm in RapidMiner and we used the default values for all of parameters.

1. Criterion : this parameter selects the criterion on which attributes will be selected for splitting. There are four options: information gain, gain ratio, gini index and accuracy. The default value is gain ratio.
2. Minimal size for split : this parameter sets the number of examples in its subset. The default value is 4.
3. Minimal leaf size : the tree is generated with every leaf node subset has at least the minimal leaf size number of instance.
4. Minimal gain : higher value of this parameters affects fewer split and a smaller tree. The default valus is 0.1
5. Maximal depth: set the maximal depth of the tree, the default value is 20
6. Confidence : this parameter is used to set the confidence level of prunning error, the default value is 0.25

The results of decision tree experiments are explained in Table 3.6

Table 3.6 Decision tree experiment results on original datasets

Data set	Number of instances	Number of attributes	Performance measurement with 10-fold cross validation				
			Accuracy	Precision	Recall	F Measure	Execution Time (hh:mm:ss)
Leukemia	72	7,130	82.14%	73.33%	85.00%	78.73%	00:24:44
Embryonal Tumours	60	7,130	50.00%	64.00%	54.17%	58.68%	00:40:16
Dexter	600	20,000	84.83%	85.61%	88.00%	86.79%	08:10:14
Internet_ads	3,279	1,559	78.34%	96.11%	78.11%	86.18%	00:01:19
Madelon	2,600	501	50.00%	50.00%	90.00%	64.29%	00:00:33
Musk	6,598	168	86.34%	86.26%	99.87%	92.57%	00:00:39
Spambase	4,601	58	90.72%	89.97%	95.37%	92.59%	00:02:16
SPECTF Heart	80	45	76.25%	75.00%	85.00%	79.69%	00:00:01
Intrusion	25,192	42	99.68%	99.67%	99.65%	99.66%	00:42:34

Compare to naive Bayes and k-NN, decision tree is much slower. For example, to apply 10-fold cross validation technique on dexter which has the highest dimension (20,000 attributes), decision tree takes more than 8 hours while naive Bayes only needs 13 second and k-NN needs 27 seconds. When applied to intrusion dataset which has the least dimension (42 attributes) but has the highest number of instances (25,192 instances), decision tree needs more than 42 minutes while naive Bayes only takes 9 seconds and kNN needs 8 minutes.

3.5.1.4 Rule Induction

The rule induction module in RapidMiner has four parameters as follow:

1. Criterion: selects the criterion for attribute selection and numerical splits. There are 2 options which are information gain and accuracy. The default value is information gain.
2. Sample ratio: sets the sample ratio of training data used for pruning, the default value is 0.9
3. Pureness: specifies the pureness level, the default value is 0.9
4. Minimal prune benefit: default value is 0.25

We used the default values for all parameters and the results are explained in Table 3.7

Table 3.7 Rule induction experiment results on original datasets

Data set	Number of instances	Number of attributes	Performance measurement with 10-fold cross validation				
			Accuracy	Precision	Recall	F Measure	Execution Time (hh:mm:ss)
Leukemia	72	7,130	87.68%	84.33%	82.50%	83.40%	00:14:59
Embryonal Tumours	60	7,130	60.00%	67.17%	82.50%	74.05%	00:30:33
Dexter	600	20,000	83.50%	87.12%	79.67%	83.23%	05:07:12
Internet_ads	3,279	1,559	91.00%	91.09%	99.26%	95.00%	00:05:02
Madelon	2,600	501	73.08%	72.72%	73.92%	73.32%	00:43:37
Musk	6,598	168	91.47%	91.60%	99.01%	95.16%	01:57:12
Spambase	4,601	58	91.42%	91.66%	94.48%	93.05%	00:01:51
SPECTF Heart	80	45	60.00%	59.09%	65.00%	61.90%	00:00:01
Intrusion	25,192	42	92.21%	92.00%	91.22%	91.61%	00:00:51

Rule induction algorithm is also much slower than naive Bayes and kNN especialy when applied to datasets with high number of instances or high number of attributes such as dexter, musk and intrusion dataset.

3.5.1.5 Basic classifiers results comparison

We summarized the results of previous experiments in Table 3.8. We selected F-Measure as main performance indicator. The yellow colour means the best results of four methods.

Table 3.8 Classification performance of original datasets

Datasets	Number of instances	Number of attributes	Classification Performance (F Measure)			
			NB	kNN	DT	RI
Leukemia	72	7,130	98.31%	89.70%	78.73%	83.40%
Embryonal Tumours	60	7,130	74.41%	67.06%	58.68%	74.05%
Dexter	600	20,000	81.39%	86.63%	86.79%	83.23%
Internet_ads	3,279	1,559	98.20%	91.45%	86.18%	95.00%
Madelon	2,600	501	59.05%	64.90%	64.29%	73.32%
Musk	6,598	168	93.67%	97.15%	92.57%	95.16%
Spambase	4,601	58	82.90%	85.62%	92.59%	93.05%
SPECTF Heart	80	45	79.49%	67.53%	79.69%	61.90%
Intrusion	25,192	42	82.88%	99.51%	99.66%	91.61%

Table 3.8 shows that there is no algorithm achieved the best results for all datasets. NB achieved the best results on 3 datasets: leukemia, embryonal tumour and internet_ads while other algorithms (kNN, DT and RI) achieved the best results in 2 datasets each.

Table 3.9 Learning time of NB, kNN, DT and RI

Dataset Name	Number of instances	Number of attributes	Learning time (second)			
			NB	k-NN	DT	RI
Leukemia	72	7,130	0.09	0	0.33	0.33
Embryonal Tumours	60	7,130	0.08	0	0.39	0.61
Dexter	600	20,000	1.38	0	8.60	16.11
Internet_ads	3,279	1,559	0.10	0	16.78	16.99
Madelon	2,600	501	0.17	0	2.40	5.99
Musk	6,598	168	0.22	0	0.86	13.43
Spambase	4,601	58	0.05	0	0.42	1.28
SPECTF Heart	80	45	0.10	0	0.30	0.16
Intrusion	25,192	42	0.10	0	1.12	5.53
Average learning time			0.25	0	3.47	6.71

Beside classification performance, we also compare the learning time and execution time of four methods in Table 3.9 and Table 3.10. As explained in the previous section, kNN does not require explicit training step, therefore kNN has the fastest learning time followed by naive Bayes (0.25 seconds in average), decision tree (3.47 seconds in average) and rule induction (6.71 seconds in average).

Even though kNN has no explicit learning phase, it does not mean that kNN is the fastest algorithm for classification. Table 3.10 shows that naive Bayes is actually faster than kNN when applied into 9 different datasets.

Table 3.10 Execution time of 4 classifiers on original datasets

Dataset Name	Execution Time using 10 fold cross validation (hh:mm:ss)			
	NB	k-NN	DT	RI
Leukemia	00:00:01	00:00:01	00:24:44	00:14:59
Embryonal Tumours	00:00:01	00:00:01	00:40:16	00:30:33
Dexter	00:00:13	00:00:27	08:10:14	05:07:12
Internet_ads	00:00:10	00:01:51	00:01:19	00:05:02
Madelon	00:00:02	00:00:24	00:00:33	00:43:37
Musk	00:00:02	00:00:59	00:00:33	01:57:12
Spambase	00:00:01	00:00:11	00:02:16	00:01:51
SPECTF Heart	00:00:01	00:00:01	00:00:01	00:00:01
Intrusion	00:00:09	00:08:37	00:42:34	00:00:51

Decision tree and rule induction requires much more time than naive Bayes and kNN, especially when they are applied to datasets with high number of attributes. For example, in dexter dataset which has 20,000 attributes NB needs 13 seconds and kNN needs 27 second to finish 10 fold cross validatian while RI takes more than 5 hours and DT takes more than 8 hours.

3.5.2 The GA-based Feature Selection Experiments

The WEKA feature selection algorithms have two important components: attribute evaluator and search method. 'Attribute evaluator' is a technique used to evaluate the performance of feature subsets and 'search method' is an algorithm used to search through the space of feature subsets.

We applied GA search and attribute selector called CfsSubsetEval which is a method used to evaluate the performance of an attribute subset by considering the individual predictive ability of each attribute along with the degree of redundancy between them.

We used the default parameters as follows:

- crossoverProb: set the probability of crossover, default=0.6
- maxGenerations: set the maximum number of generations, default=20

- mutationProb: set the probability of mutation, default=0.033
- populationSize: set the size of population, default=20
- seed: set the random seed
- startSet: set a start point for the search, default value=no attributes

The results of feature selection using GA are shown in Table 3.11. The longest time to reduce the number of features was on dexter dataset where GA has been successfully reduced the number of features from 20,000 to 6,133 attributes in 27 minutes 15 seconds. The other datasets only need less than 1 minute to finish the feature selection process.

Table 3.11 GA-based feature selection results

Datasets	Number of original attributes	Feature selection using GA	
		Number of reduced attributes	Time to reduce the number of attributes (hh:mm:ss)
Leukemia	7,130	2,237	00:00:50
Embryonal Tumours	7,130	619	00:00:35
Dexter	20,000	6,133	00:27:15
Internet_ads	1,559	489	00:05:02
Madelon	501	142	00:00:18
Musk	168	66	00:00:04
Spambase	58	29	00:00:02
SPECTF Heart	45	11	00:00:01
Intrusion	42	16	00:00:03

3.5.3 The PSO-based Feature Selection Experiments

We continue our feature selection experiment using PSO search and attribute selector called CfsSubsetEval. The PSO search module in Weka has more parameters than the GA search.The PSO search parameters are:

- C1 default value=1.0
- C2 default value=2.0
- maxGeneration: set the maximal number of generation, default value=50

- numParticles: set the population size, default value=100
- prune: boolean parameter, prune the subset after search to remove redundant features, default value=false
- reportFrequency: set how frequently reports are generated, default value=50
- seed: set the random seet, default value=1
- Start set: default value=no attributes

The parameter 'start set = no attributes' means this technique is to begin with no attributes and then it is successively add attributes. In this case, PSO search will proceed forward through the search space. The total of inertia weight, social weight and individual weight must be one and all these weights should be greater than or equal to zero.

We ran PSO-based feature selection module using default paramaters and the results are explained in Table 3.12.

Table 3.12 PSO-based feature selection results

Datasets	Number of original attributes	Feature selection using PSO	
		Number of reduced attributes	Time to reduce the number of attributes (hh:mm:ss)
Leukemia	7,130	109	00:01:17
Embryonal Tumours	7,130	202	00:01:08
Dexter	20,000	279	00:41:13
Internet_ads	1,559	302	00:04:50
Madelon	501	5	00:00:15
Musk	168	16	00:00:03
Spambase	58	27	00:00:03
SPECTF Heart	45	9	00:00:03
Intrusion	42	8	00:00:01

Table 3.12 shows that feature selection using PSO is sligthly slower than GA but PSO can reduce the unimportant attributes much more than GA. For example, in Dexter datasets GA reduced the attributes from 20,000 to 6,111 in 27 minutes 15 seconds while PSO reduced the attributes from 20,000 to only 279 in 41 minutes 13 seconds.

3.5.4 Results analysis of GA and PSO as feature selection algorithms

To compare the performance of GA and PSO more clearly, we summarized the GA and PSO results in Table 3.13. Fraction of features (FF) is the ratio of the number of features used by the classifier to the total number of features in the dataset. The average FF of GA is 31.36% while the average FF of PSO is 13.47%. It means that PSO reduced the number of attributes much more than GA. In madelon dataset, PSO reduced the number of attributes from 501 to 5 (fraction of features= 1%) while GA reduced to 142 (FF=28.34). In 4 of 8 datasets, PSO has successfully reduced the number of attributes to less than 5% of their original attributes (embryonal tumours 2.83%, leukemia 1.53%, dexter 1.40% and madelon 1.00%).

Table 3.13 The results comparison of GA and PSO feature selection

Datasets	Number of original attributes	Feature selection using GA			Feature selection using PSO		
		Number of reduced attributes	Fraction of features (FF)	Time to reduce the number of attributes (hh:mm:ss)	Number of reduced attributes	Fraction of features (FF)	Time to reduce the number of attributes (hh:mm:ss)
Leukemia	7,130	2,237	31.37%	00:00:50	109	1.53%	00:01:17
Embryonal Tumours	7,130	619	8.68%	00:00:35	202	2.83%	00:01:08
Dexter	20,000	6,133	30.67%	00:27:15	279	1.40%	00:41:13
Internet_ads	1,559	489	31.37%	00:05:02	302	19.37%	00:04:50
Madelon	501	142	28.34%	00:00:18	5	1.00%	00:00:15
Musk	168	66	39.29%	00:00:04	16	9.52%	00:00:03
Spambase	58	29	50.00%	00:00:02	27	46.55%	00:00:03
SPECTF Heart	45	11	24.44%	00:00:01	9	20.00%	00:00:03
Intrusion	42	16	38.10%	00:00:03	8	19.05%	00:00:01
			Average FF	Total time		Average FF	Total time
			31.36%	00:34:10		13.47%	00:48:53

Based on the execution time, GA is slightly faster than PSO. GA takes 34 minutes and 10 seconds to select the important features of 9 datasets while PSO needs 48 minutes and 53 seconds.

However, fraction of features (FF) and execution time are not the only performance indicator of feature selection algorithms. The low number of FF

and the low number of execution time are useless if the selected subsets reduce the classification accuracy. Therefore, we need to find feature selection algorithms which can reduce the number of attributes while in the same time maintain or improve the accuracy.

In the following experiment, we applied four basic classifiers to the nine datasets which have been reduced by GA and PSO, then compare the results to the classification performance of original datasets.The results are shown in Table 3.14 and Table 3.15.

Table 3.14 The classification performance of GA-reduced datasets

Data set	Reduced by GA	NB	k-NN	DT	RI
	#attributes	F-measure	F-measure	F-measure	F-measure
Leukemia	2,237	98.31%	78.55%	81.16%	82.45%
Embryonal Tumours	619	65.42%	78.82%	58.58%	81.00%
Dexter	6,133	73.30%	60.04%	87.16%	81.84%
Internet_ads	489	98.07%	78.49%	88.32%	95.02%
Madelon	142	59.35%	65.23%	64.29%	68.15%
Musk	66	95.23%	96.48%	91.96%	95.93%
Spambase	29	80.34%	90.33%	91.69%	92.90%
SPECTF Heart	11	88.50%	74.57%	73.24%	73.52%

NB and RI achieved the best results on 3 of 8 datasets while k-NN and DT achieved the best results on 1 dataset only. Even though with much less attributes,4 of 8 datasets have better classification performance than using full attributes. In the embryonal tumours dataset, rule induction (RI) has successfully increased the F-measure from 74.41% to 81.00% but with less attributes (from 7,130 original attributes reduced to 619 attributes).

In dexter dataset, decision tree (DT) was slightly increased the F-measure from 86.79% to 87.16% but with 31% of original attributes (the number of attributes was reduced from 20,000 to 6,133). In SPECTF heart dataset, naïve Bayes (NB) has also significantly increased the F-measure from 79.69% to 88.50% with 25% attributes (the number of attributes has been reduced by GA from 45 to 11). Unfortunately, all four classifiers were unable to improve the classification performance in 4 datasets: internet_ads (slightly dropped from 98.20% to 98.07%), madelon (from 73.32% to 68.15%), musk (from 97.15% to 96.48%) and spambase (from 93.05% to 92.90%).

The Table 3.15 shows the results of four basic classifiers applied to PSO-reduced dataset. NB achieved the best classification performance on 4 of 8 datasets, followed by DT (3 datasets) and k-NN (1 dataset). In this experiment, RI was not as good as other algorithms. PSO has successfully significantly reduced the number of attributes to 13% on average (GA is only 31% average). However, it does not always improve or maintain the classification performance, only 3 of 8 datasets have their performance improved. The accuracy of embryonal tumour dataset was improved from 74.41% to 76.61% with only 8% attributes, musk dataset was improved from 97.15% to 99.92% with 39% attributes and SPECTF heart dataset was improved from 79.69% to 85.89% with 24% attributes. The other 5 datasets (leukemia, dexter, internet_ads, madelon and spambase) have their classification performance slightly reduced from 0.13% (spambase) to 12.81% (dexter).

Table 3.15 The classification performance of PSO-reduced datasets

Data set	Reduced by PSO	NB	k-NN	DT	RI
	#attributes	F-measure	F-measure	F-measure	F-measure
Leukemia	109	96.55%	89.10%	69.12%	70.61%
Embryonal Tumours	202	65.40%	70.74%	76.61%	68.34%
Dexter	279	73.13%	73.98%	44.56%	70.72%
Internet_ads	302	97.77%	73.18%	96.96%	95.12%
Madelon	5	60.24%	64.25%	64.29%	63.07%
Musk	16	99.92%	96.45%	91.65%	95.29%
Spambase	27	90.29%	90.91%	92.92%	92.25%
SPECTF Heart	9	85.89%	81.42%	77.77%	76.82%

We compare the running time of four basic classifiers on original datasets, GA-reduced datasets and PSO-reduced datasets in Table 3.16. Even though kNN has the fastest learning time but it does not guarantee it has the fastest running time. Based on experiments using 9 datasets, Naive Bayes was the fastest algorithm with 4 seconds running time in average when applied in original datasets, 2 seconds in average when applied to GA-reduced dataset and 1 second in average on PSO-reduced datasets. The second fastest running time was kNN algorithms with 78 seconds in average on original datasets, 8 seconds in average on GA-reduced dataset and 4 seconds in average on PSO-reduced datasets.

Table 3.16 The running time of four basic classifiers: NB, kNN, DT and RI

Dataset Name	Number of instances	Number of attributes	NB	kNN	DT	RI
Running time on original datasets (seconds)						
Leukemia	72	7,130	1	1	1,484	899
Embryonal Tumours	60	7,130	1	1	2,416	1,833
Dexter	600	20,000	13	27	29,414	18,432
Internet_ads	3,279	1,559	10	60	79	302
Madelon	2,600	501	2	24	33	2,617
Musk	6,598	168	2	59	33	7,032
Spambase	4,601	58	1	11	136	111
SPECTF Heart	80	45	1	1	1	1
Intrusion	25,192	42	9	517	2,554	51
Average running time			**4**	**78**	**4,017**	**3,475**
Total running time			**40**	**701**	**36,150**	**31,278**
Running time on GA-reduced datasets (seconds)						
Leukemia	72	2,237	1	1	23	15
Embryonal Tumours	60	619	1	1	11	3
Dexter	600	6,133	5	6	57	2,573
Internet_ads	3,279	489	2	2	4	11
Madelon	2,600	142	1	2	9	821
Musk	6,598	66	2	8	11	58
Spambase	4,601	29	1	1	15	14
SPECTF Heart	80	11	1	1	1	1
Intrusion	25,192	16	1	53	73	9
Average running time			**2**	**8**	**23**	**389**
Total running time			**15**	**75**	**204**	**3,505**
Running time on PSO-reduced datasets (seconds)						
Leukemia	72	109	1	1	1	1
Embryonal Tumours	60	202	1	1	2	1
Dexter	600	279	1	1	2	113
Internet_ads	3,279	302	1	1	3	6
Madelon	2,600	5	1	1	1	88
Musk	6,598	16	1	2	2	16
Spambase	4,601	27	1	1	14	13
SPECTF Heart	80	9	1	1	1	1
Intrusion	25,192	8	1	26	15	2
Average running time			**1**	**4**	**5**	**27**
Total running time			**9**	**35**	**41**	**241**

Both DT and RI were very slow compare to NB and kNN especially when they applied to very big datasets. The average running time of RI was 3,472 seconds when applied to original datasets but it dropped by 27 seconds only when applied to PSO-reduced datasets. DT has longer average running time than RI when applied to original datasets, it requires 4,017 seconds to run all original datasets. But it became a little bit faster than RI when applied to PSO-reduced datasets with 5 seconds average running time.

3.6 Summary

We summarize all of the experiments in this chapter in the Table 3.17. In terms of dimensionality reduction or feature selection, PSO is much better than GA. PSO has successfully reduced the number of attributes of 9 datasets to 13.79% on average while GA is only 31.19% on average. The most extreme cases were in dexter dataset where PSO reduced the number of attributes to 1.40% (from 20,000 to 279 attributes), and in the madelon dataset, PSO reduced the number of attribute to 1% (from 501 to 5 attributes). In terms of classification performance, GA is better than PSO. GA-reduced datasets have better performance than their original ones on 5 of 9 datasets while PSO is only 3 of 9 datasets. In SPECTH heart dataset, GA has successfully improved the accuracy up to 8.81% (from 79.79% to 88.50%) by using only 24.44% of original attributes.

However, feature selection algorithms do not always improve the classification performance, there are three datasets (internet_ads, madelon and spambase) that both GA and PSO failed to improve the performance. Therefore, we need to find other strategies to solve this problem. We need to apply more powerful classifiers to these reduced datasets (both by GA or PSO) rather than using basic classifiers such as naïve Bayes, k-nearest neighbour, decision tree and rule induction. We will apply two different techniques to improve the classification performance by using ensemble classifiers (meta-learners) and support vector machine which will be discussed in the next two chapters.

In term of running time, Table 3.16 has shown that applying four basic classifiers into reduced datasets were much faster than applying them into original datasets which have much more attributes.

Table 3.17 Summary of dimensionality reduction algorithms

Data set	Full atributes			Feature Selection using GA					Feature Selection using PSO				
	#attrib.	Algorithm	F Measure	#attrib.	FF	Algorithm	F Measure	+ / -	#attrib.	FF	Algorithm	F Measure	+ / -
Leukemia	7,130	NB	98.31%	2,237	31.37%	NB	98.31%	0.00%	109	1.53%	NB	96.55%	-1.76%
Embryonal Tumours	7,130	NB	74.41%	619	8.68%	RI	81.00%	6.59%	202	2.83%	DT	76.61%	2.20%
Dexter	20,000	DT	86.79%	6,133	30.67%	DT	87.16%	0.37%	279	1.40%	k-NN	73.98%	-12.81%
Internet_ads	1,559	NB	98.20%	489	31.37%	NB	98.07%	-0.14%	302	19.37%	NB	97.77%	-0.43%
Madelon	501	RI	73.32%	142	28.34%	RI	68.15%	-5.17%	5	1.00%	DT	64.29%	-9.03%
Musk	168	k-NN	97.15%	66	39.29%	k-NN	96.48%	-0.68%	16	9.52%	NB	99.92%	2.77%
Spambase	58	RI	93.05%	29	50.00%	RI	92.90%	-0.15%	27	46.55%	DT	92.92%	-0.13%
SPECTF Heart	45	DT	79.69%	11	24.44%	NB	88.50%	8.81%	9	20.00%	NB	85.89%	6.21%
Intrusion	41	DT	99.48%	15	36.59%	k-NN	99.70%	0.22%	9	21.95%	k-NN	99.26%	-0.22%
					31.19%					13.79%			

The average running time of rule induction algorithm was reduced from 3,475 seconds to 27 seconds when applied to PSO-reduced datasets. In similar case, the average running time of decision tree was also reduced from 4,017 seconds to 5 seconds.

Our experiments have shown that the fastest learning time was achieved by kNN algorithm with 0 second in average, followed by naive Bayes (0.25 second in average), decision tree (3.47 seconds in average) and rule induction (6.71 seconds in average). Even though kNN has the fastest learning time, kNN's average running time (78 seconds in average) was quite slower than naive Bayes (4 seconds in average) but it was much faster than decision tree (4,017 second in average) and rule induction (3,475 in average).

The total running time of 4 basic classifiers (NB, kNN, DT and RI) when applied to 9 original datasets was 68,169 seconds. The total running time dropped significantly by 3,799 seconds when these four classifiers applied to GA-reduced datasets and was only 326 seconds in PSO-reduced datasets.

4. Ensemble Classifiers

Ensemble classifiers are one of several techniques in machine learning where multiple classifiers are trained to solve the same problem. This technique constructs a set of classification models the combine them or select the best one to use. In this chapter, we applied ensemble classifiers to datasets which features are already reduced by GA and PSO, and then we compared the ensemble classifiers performance with the results of basic classifers which applied to the original datasets.

We divided the experiment into three steps:

1. First, we applied four basic machine learning algorithms (NB, kNN, DT and RI) on three different datasets: original datasets, GA-reduced datasets and PSO-reduced datasets
2. Second, we applied bagging algorithm to the GA-reduced datasets and PSO-reduced datasets
3. Third, we applied boosting algorithm to the GA-reduced datasets and PSO-reduced datasets.

The goal of this experiment is to find whether bagging and boosting when applied to the reduced dataset which has fewer dimension is able to achieve better performance than the basic classifiers applied to the full-features dataset.

4.1 Basic Classifiers

We applied four basic algorithms (NB, kNN, DT and RI) to eight high dimension datasets with three variations: full features datasets, GA-reduced datasets and PSO-reduced datasets. Table 4.1 shows that the use of basic classifiers on reduced datasets was not only able to reduce the computation time, but also it was able to improve the accuracy. For example, decision tree (DT) when applied to the embryonal tumour dataset which has 7,130 original features, the F-measure value was 58.68%. But when it was applied to similar dataset which the number of features reduced by PSO from 7,130 into 202, the F-measure value was significantly increased to 81.15%.

Table 4.1 Classification performance of NB, kNN, DT and RI

NB, kNN, DT and RI applied to original datasets					
Dataset	**#attrib**	**NB**	**kNN**	**DT**	**RI**
Leukemia	7,130	**98.31%**	89.70%	78.73%	83.40%
Embryonal Tumours	7,130	74.41%	67.06%	58.68%	74.05%
Dexter	20,000	81.39%	86.63%	86.79%	83.23%
Internet_ads	1,559	**98.20%**	91.45%	86.18%	95.00%
Madelon	501	59.05%	64.90%	64.29%	**73.32%**
Musk	168	93.67%	97.15%	92.57%	95.16%
Spambase	58	82.90%	85.62%	92.59%	**93.05%**
SPECTF Heart	45	79.49%	67.53%	79.69%	61.90%
NB, kNN, DT and RI appliedto GA-reduced datasets					
Dataset	**#attrib**	**NB**	**kNN**	**DT**	**RI**
Leukemia	2,237	**98.31%**	78.55%	81.16%	82.45%
Embryonal Tumours	619	65.42%	78.82%	58.58%	**81.00%**
Dexter	6,133	73.30%	60.04%	**87.16%**	81.84%
Internet_ads	489	98.07%	78.49%	88.32%	95.02%
Madelon	142	59.35%	65.23%	64.29%	68.15%
Musk	66	95.23%	96.48%	91.96%	95.93%
Spambase	29	80.34%	90.33%	91.69%	92.90%
SPECTF Heart	11	**88.50%**	74.57%	73.24%	73.52%
NB, kNN, DT and RI appliedto PSO-reduced datasets					
Dataset	**#attrib**	**NB**	**kNN**	**DT**	**RI**
Leukemia	109	96.55%	89.10%	69.12%	70.61%
Embryonal Tumours	202	65.40%	70.74%	76.61%	68.34%
Dexter	279	73.13%	73.98%	44.56%	70.72%
Internet_ads	302	97.77%	73.18%	96.96%	95.12%
Madelon	5	60.24%	64.25%	64.29%	63.07%
Musk	16	**99.92%**	96.45%	91.65%	95.29%
Spambase	27	90.29%	90.91%	92.92%	92.25%
SPECTF Heart	9	85.89%	81.42%	77.77%	76.82%

Another good example, naïve Bayes (NB) when applied to the musk dataset which has 168 original features, the F-measure value was 93.67%. When NB was applied to the GA-reduced dataset, the F-measure value was slightly increase to 95.23% eventhough the number of features were reduced from 168 to 66. Furthermore, when NB was applied to PSO-reduced dataset where the number of features was reduced from 168 to 16, the F-measure value was significantly improved to 99.91%.

The PSO-reduced dataset is not always guarantee to have better accuracy or better in F-measure value. There was one case where the GA-reduced dataset was much better than PSO-reduced dataset. Decision tree (DT) when applied to the dexter full features dataset, the F-measure was 86.79%. When DT was applied to the same dataset which the number of features was reduced by PSO from 20,000 to 279, the F-measure was very bad (44.45%). In this case, the feature selection using PSO does not run well. When DT was applied to the dexter dataset which the number of features reduced by GA from 20,000 to 6,133, the F-measure was slightly better (87.16%).

However, there are two cases that the use of reduced datasets both GA and PSO could not improve the F-measure value. For example, when applying NB to the internet_ads full features dataset, the F-measure value was 98.20%. But when NB was applied to internet_ads where the number of features was reduced by GA from 1,559 to 489, the F-measure value becomes 98.07%. And when NB was applied to the same dataset where the number of features was reduced by PSO from 1,559 to 302, the F-measure values was slightly reduced to 97.77%.

4.2 Bagging Ensemble Classifier

The bagging learning process uses a different bootstrap sample which is randomly retrieved from the original data. A bootstrap sample is constructed by sub-sampling the training data with replacement where the size of bootstrap data is similar with the original one. Bagging is good to be applied to unstable algorithms where small changes in the training data highly affect their performance.

We used a bagging operator provided by RapidMiner. The bagging operator is used to build a better model using the weak learner selected in its sub process. We used four basic machine learning algorithms (naïve Bayes, decision tree, nearest neighbour and rule induction) as base classifiers. The bagging process is explained in Figure 4.1.

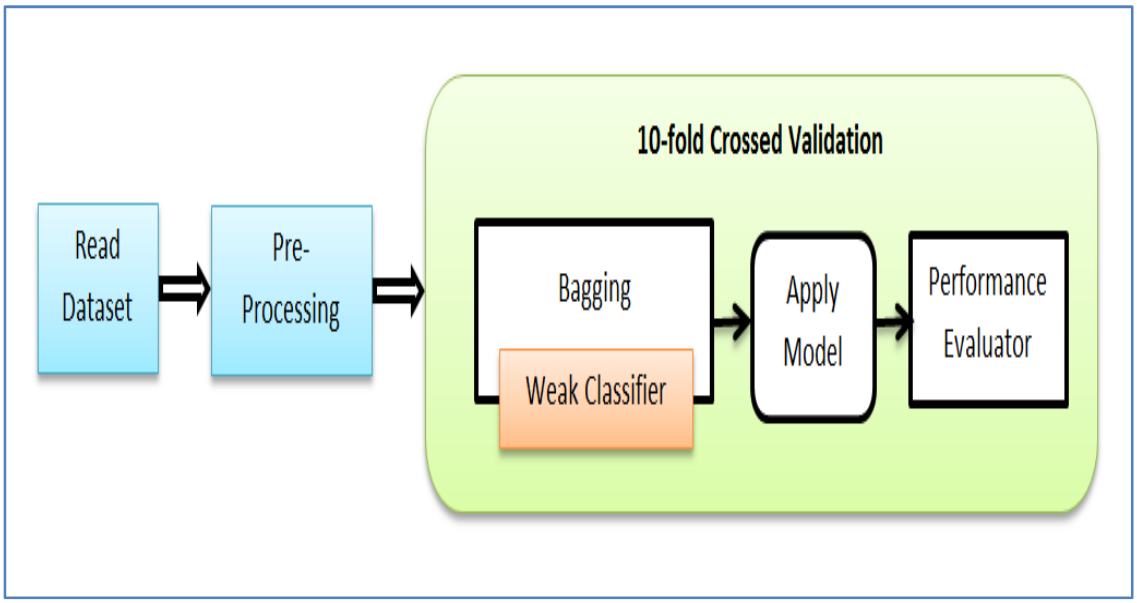

Figure 4.1. Bagging Diagram Experiment

The bagging operator in RapidMiner has 2 parameters:

- Sample ratio: set the number of examples to be used for training, the default value is 0.9
- Iteration: set the maximum number of iterations, the default value is 10

We applied bagging to a dataset four times with four different base classifiers: NB, kNN, DT and RI using the default parameters. The learning time of bagging algorithm is shown in Table 4.2

Table 4.2 The learning time of bagging algorithm

Dataset Name	Number of instances	Number of attributes	Learning Time (second)			
			Bagging with base classifier NB, kNN, DT and RI			
			NB	kNN	DT	RI
Leukemia	72	7,130	0.39	0.02	0.62	1.43
Embryonal Tumours	60	7,130	0.45	0	2.60	3.93
Dexter	600	20,000	15.04	0.03	118.38	285.71
Internet_ads	3,279	1,559	0.71	0.01	122.68	112.49
Madelon	2,600	501	2.36	0.01	50.65	134.34
Musk	6,598	168	1.79	0.02	6.65	107.34
Spambase	4,601	58	0.33	0	4.14	20.50
SPECTF Heart	80	45	0.17	0.02	0.52	0.44
Intrusion	25,192	42	1.60	0.06	17.94	81.67
Average learning time			2.54	0.02	36.02	83.09

As explained in the previous chapter, kNN algorithm does not require learning time, as consequence the bagging-kNN learning time is also almost zero for all datasets. The second fastest learning time is bagging-NB which has 2.54 seconds in average, followed by bagging-DT in third place with 36.02 seconds average learning time. The slowest learning time is bagging-RI which has 83.09 seconds in average.

To analyze the performance of bagging algorithm, we conduct many experimentswhich divided into 2 steps; the first one we use 8 datasets which features has been reduced by GA algorithm while the second one was reduced by PSO algorithm.The results of bagging experiment are shown in Table 4.3 for GA-reduced datasets and in Table 4.4 for PSO-reduced datasets. In Leukemia dataset, the highest F-measure score was achieved by bagging-NB with F-measure of 98.57%. This result was exactly the same as the results of naive Bayes as single classifier when applied to GA-reduced dataset. In this case, bagging did not improve the classification performance. In embryonal tumours dataset, the best results was achieved by bagging-DT when applied to PSO-reduced dataset (202 attributes) where the F-measure score=81.15%. This result was much better than applying decision tree on original dataset (7.130 attributes) where the F-measure=58.58%.

Table 4.3 shows that the use of bagging on GA-reduced datasets is not always guarantee better than the use of single classifier. The bagging-NB algorithm when applied to the SPECTF Heart dataset which reduced by GA (consists of 11 attributes) has achieved the best result with F-measure = 89.22%. This result is slightly better than the use of NB as a single classifier which has an F-measure of 88.50%. Furthermore, this result is also much better than the use of NB (as a single classifier) on the SPECTF Heart full-features dataset (consists of45 attributes) which achieve F-measure of 79.49%.

In the madelon dataset, the use of bagging-RI is also able to achieve the best result with F-measure=73.79% even though the number of features was significantly reduced by GA from 501 attributes to 142 attributes. This result is slightly better compare to the use of RI (as a single classifier) on full-features datasets (consists of 501 attributes) which has an F-measure of 73.32%.

Table 4.3 Classification performance of Bagging on GA-reduced datasets

	Datasets	Bagging -Naive Bayes			Bagging Nearest Neighbour			Bagging - Decision Tree			Bagging - Rule Induction		
		Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
		F Measure			F Measure			F Measure			F Measure		
1	Leukemia	98.57%	100.00%	96.67%	87.32%	91.67%	70.00%	90.36%	92.50%	79.17%	89.11%	90.17%	84.17%
		98.31%			79.38%			85.32%			87.07%		
2	Embryonal Tumours	58.33%	74.17%	55.83%	66.67%	72.67%	77.50%	58.33%	67.50%	68.33%	70.00%	72.67%	89.17%
		63.71%			75.01%			67.91%			80.08%		
3	Dexter	71.17%	69.75%	78.67%	72.00%	97.82%	45.00%	85.00%	84.49%	87.67%	86.33%	86.50%	86.33%
		73.94%			61.64%			86.05%			86.41%		
4	Internet_ads	96.46%	96.85%	99.11%	68.83%	96.80%	65.92%	71.91%	95.93%	70.36%	91.22%	91.22%	99.36%
		97.97%			78.43%			81.18%			95.12%		
5	Madelon	60.42%	60.92%	58.31%	66.50%	67.22%	64.38%	50.00%	50.00%	50.00%	74.38%	75.77%	71.92%
		59.59%			65.77%			50.00%			73.79%		
6	Musk	92.24%	99.05%	91.70%	93.94%	97.40%	95.38%	87.09%	86.84%	100.00%	93.33%	93.02%	99.61%
		95.23%			96.38%			92.96%			96.20%		
7	Spambase	79.68%	96.84%	68.72%	88.68%	91.95%	89.13%	90.87%	89.35%	96.49%	91.68%	91.27%	95.41%
		80.39%			90.52%			92.78%			93.29%		
8	SPECTF Heart	87.50%	86.17%	92.50%	71.25%	66.88%	85.00%	81.25%	84.33%	77.50%	78.75%	84.67%	75.00%
		89.22%			74.86%			80.77%			79.54%		
9	Intrusion	89.02%	90.65%	85.27%	99.77%	99.80%	99.69%	99.31%	99.55%	98.95%	92.21%	92.00%	91.22%
		87.88%			99.74%			99.25%			91.61%		

Table 4.4 Classification performance of Bagging on PSO-reduced datasets

	Datasets	Bagging -Naive Bayes			Bagging Nearest Neighbour			Bagging - Decision Tree			Bagging - Rule Induction		
		Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
		F Measure			F Measure			F Measure			F Measure		
1	Leukemia	98.57%	96.67%	100.00%	91.79%	91.00%	85.00%	79.11%	70.83%	67.50%	87.68%	92.67%	75.83%
		98.31%			87.90%			69.12%			83.41%		
2	Embryonal Tumours	60.00%	70.83%	60.83%	65.00%	75.17%	75.00%	73.33%	78.33%	84.17%	66.67%	70.83%	85.00%
		65.45%			75.08%			81.15%			77.27%		
3	Dexter	75.67%	83.03%	64.67%	71.83%	72.17%	72.00%	64.33%	100.00%	28.67%	75.83%	80.31%	69.33%
		72.71%			72.08%			44.56%			74.42%		
4	Internet_ads	96.10%	96.39%	99.18%	63.28%	97.87%	58.62%	94.75%	94.50%	99.72%	91.22%	91.22%	99.36%
		97.77%			73.32%			97.04%			95.12%		
5	Madelon	60.77%	61.13%	59.00%	63.35%	63.11%	64.46%	50.00%	50.00%	50.00%	67.38%	67.77%	66.77%
		60.05%			63.78%			50.00%			67.27%		
6	Musk	99.86%	99.87%	99.96%	94.35%	96.87%	96.43%	84.59%	84.59%	100.00%	92.62%	92.61%	99.19%
		99.91%			96.65%			91.65%			95.79%		
7	Spambase	88.57%	95.08%	85.58%	89.22%	91.63%	90.49%	91.52%	89.86%	96.99%	92.33%	92.19%	95.44%
		90.08%			91.06%			93.29%			93.79%		
8	SPECTF Heart	86.25%	84.17%	92.50%	77.50%	73.55%	90.00%	76.25%	73.50%	85.00%	78.75%	77.17%	85.00%
		88.14%			80.95%			78.83%			80.90%		
9	Intrusion	86.48%	92.64%	77.14%	99.27%	99.65%	98.78%	98.63%	99.71%	97.35%	92.21%	92.00%	91.22%
		84.18%			99.21%			98.52%			91.61%		

In GA-reduced musk dataset which has 66 attributes, the best F-measure score was achieved by bagging-kNN with F-measure of 96.38%. Interestingly, the bagging result especially bagging-naïve Bayes was much better when applied to PSO-reduced musk dataset which has only 16 attributes where the F-measure score was 99.91%.

In GA-reduced datasets, the best F-scores were achieved by bagging-RI in 4 of 9 datasets, followed by bagging-NB (3 of 9 datasets) and bagging-kNN (2 of 9 datasets). The bagging-DT did not perform as well as others. However, in PSO-reduced datasets the ranking was quite different. The best F-scores were achieved by bagging-NB in 4 of 9 datasets, followed by bagging-RI (3 of 9 datasets). Bagging-kNN and bagging-DT had the best results in only 1 dataset each.

4.3 Boosting Ensemble Classifier

The boosting algorithm uses a majority vote on top of the prediction of the base learners. Boosting consists of sequential production of classifiers where each classifier is dependent of the previous one; hence it trains one base classifier at a time. Boosting focuses on the previous classifier's errors. Instances or data which are predicted incorrectly are given higher weight, so they will be selected more often. In contrast, instances that are predicted correctly will have lower weights.

Boosting is actually a family of algorithms since there are many variants. One of the most popular boosting algorithms is AdaBoost which short for "Adaptive Boosting" (Graczyk et al., 2010)(Syarif et al., 2012c). AdaBoost is an algorithm for constructing a strong classifier as linear combination of simple or weak classifiers. This algorithm adapts weights on the base learners and training examples. It constructs classifier in an iterative process.In each iteration, AdaBoost executes a base leaner which then returns a classifier with its weight. The final classification will be decided based on the weight of the base classifiers. If the base classifier has smaller error, it will have higher weight in the final vote and vice versa. The AdaBoost algorithm is explained in Figure 4.2.

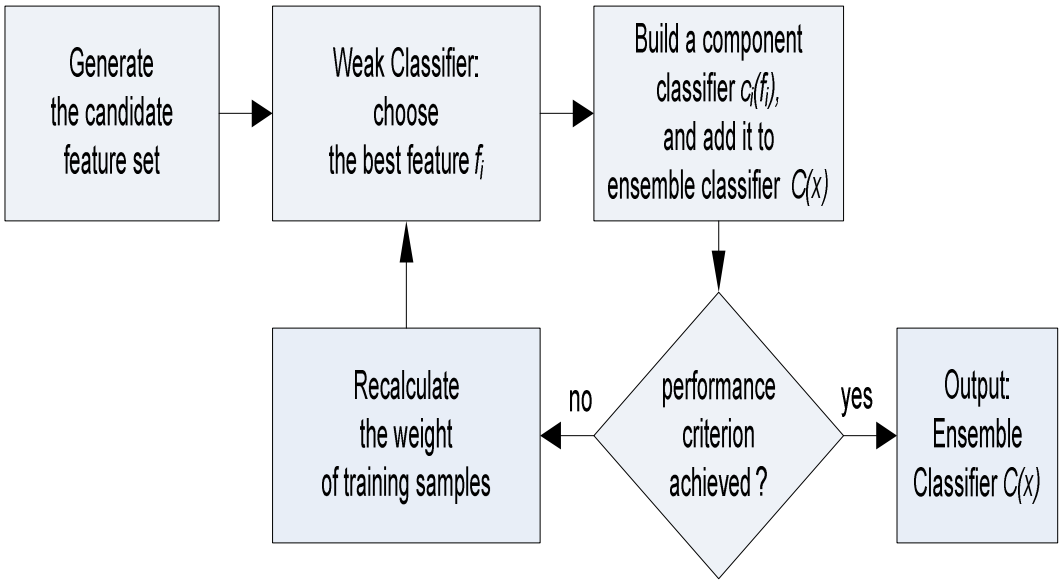

Figure 4.2 AdaBoost Algorithm

We used one of several bagging operators in RapidMiner called AdaBoost which only has 1 parameter called 'iteration'. This parameter is used to set the maximum number of iterations, the default value is 10. We applied AdaBoost algorithm to each dataset four times with four different base classifiers: NB, kNN, DT and RI using default parameter. The diagram of boosting experiment is described in the Figure 4.3.

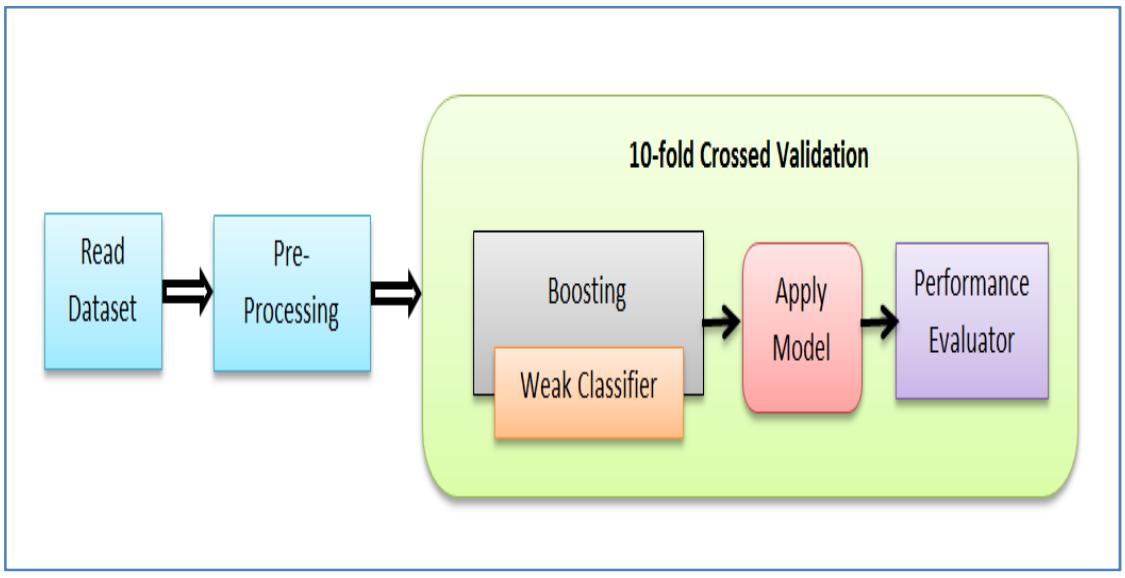

Figure 4.3 Boosting Experiments Diagram

The learning time of AdaBoost algorithm is shown in Table 4.7.

Table 4.5 The learning time of boosting algorithm

Dataset Name	Number of instances	Number of attributes	Learning Time (second)			
			Boosting Algorithm			
			AdaBoost NB	AdaBoost kNN	AdaBoost DT	AdaBoost RI
Leukemia	72	7,130	0.36	0.27	1.98	2.52
Embryonal Tumours	60	7,130	1.26	0.14	2.57	2.92
Dexter	600	20,000	143.05	61.03	143.15	262.24
Internet_ads	3,279	1,559	18.29	133.63	301.44	75.97
Madelon	2,600	501	9.72	20.89	60.15	31.00
Musk	6,598	168	8.17	15.23	0.67	63.32
Spambase	4,601	58	0.65	61.41	3.95	4.71
SPECTF Heart	80	45	0.85	0.32	0.75	0.63
Intrusion	25,192	42	7.58	985.47	11.79	29.26
Average learning time			**21.10**	**142.04**	**58.49**	**52.51**

Unlike kNN as a single classifier and kNN as base classifier of bagging algorithm that have almost zero learning time, AdaBosst-kNN has the slowest learning time. When we applied AdaBoost-kNN into 9 different datasets, the average learning time is 142.04 seconds which is the slowest compare other three boosting methods. AdaBoost-NB has the fastest learning time with 21.10 seconds in average, followed by AdaBoost-RI in second place with 52.51 seconds in average and then AdaBoost-DT in third place with 58.49 seconds in average.

To analyze the classification performance of boosting algorithm, we conduct several experiments which divided into 2 steps; the first one we use 8 datasets which features has been reduced using GA algorithms while the second one was reduced using PSO algorithm.The results of AdaBoost experiment are shown in Table 4.6 for GA-reduced datasets and in Table 4.7 for PSO-reduced datasets. In experiments using GA-reduced datasets (Table 4.6), boosting-NB outperformed other algorithms in 4 of 9 datasets, followed by boosting-kNN which had the best results in 4 of 9 datasets. In other experiments using PSO-reduced datasets, once again boosting-NB outperformed other algorithms in 4 of 9 datasets, followed by boosting-RI which had the best results in 3 of 9 datasets.

Table 4.6 Classification performance of Boosting on GA-reduced datasets

	Datasets	Boosting -Naive Bayes			Boosting Nearest Neighbour			Boosting - Decision Tree			Boosting - Rule Induction		
		Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
		F Measure			F Measure			F Measure			F Measure		
1	Leukemia	98.57%	100.00%	96.67%	90.00%	91.67%	80.00%	80.54%	72.00%	75.00%	95.71%	100.00%	85.00%
		98.31%			85.44%			73.47%			91.89%		
2	Embryonal Tumours	66.67%	71.00%	81.67%	71.67%	79.00%	82.50%	61.67%	69.50%	74.17%	70.00%	72.00%	86.67%
		75.96%			80.71%			71.76%			78.66%		
3	Dexter	71.67%	68.95%	79.00%	71.17%	97.75%	43.43%						
		73.63%			60.14%			0.00%			0.00%		
4	Internet_ads	95.58%	97.18%	7.69%	68.86%	96.82%	65.96%	94.33%	94.18%	99.57%	90.55%	90.57%	99.40%
		14.25%			78.46%			96.80%			94.78%		
5	Madelon	59.62%	59.31%	61.46%	66.38%	67.10%	64.62%	50.00%	-	0.00%	65.77%	66.15%	64.69%
		60.37%			65.84%			0.00%			65.41%		
6	Musk	99.88%	99.93%	99.93%	93.97%	97.32%	95.50%	85.28%	85.22%	100.00%	97.89%	97.91%	99.64%
		99.93%			96.40%			92.02%			98.77%		
7	Spambase	79.64%	96.74%	68.72%	88.63%	91.75%	89.31%	89.70%	89.54%	94.01%	93.35%	94.60%	94.44%
		80.36%			90.51%			91.72%			94.52%		
8	SPECTF Heart	81.25%	91.00%	72.50%	71.25%	71.83%	80.00%	77.50%	83.00%	72.50%	72.50%	72.67%	72.50%
		80.70%			75.70%			77.40%			72.58%		
9	Intrusion	93.81%	92.58%	94.27%	99.73%	99.75%	99.68%	99.28%	99.39%	99.06%	99.41%	99.45%	99.29%
		93.42%			99.71%			99.22%			99.37%		

Table 4.7 Classification performance of Boosting on PSO-reduced datasets

	Datasets	Boosting - Naive Bayes			Boosting - Nearest Neighbour			Boosting - Decision Tree			Boosting - Rule Induction		
		Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
		F Measure			F Measure			F Measure			F Measure		
1	Leukemia	98.57%	96.67%	100.00%	93.04%	93.00%	86.67%	84.82%	85.00%	71.67%	83.04%	83.33%	67.50%
		98.31%			89.72%			77.77%			74.58%		
2	Embryonal Tumours	61.67%	71.50%	69.17%	61.67%	69.33%	72.50%	60.00%	70.00%	71.67%	65.00%	72.17%	76.67%
		70.32%			70.88%			70.83%			74.35%		
3	Dexter	73.67%	77.51%	69.33%	73.83%	74.18%	73.33%	62.67%	95.58%	33.00%	72.00%	76.69%	64.67%
		73.19%			73.75%			0.00%			0.00%		
4	Internet_ads	95.88%	96.80%	98.48%	63.65%	97.90%	59.01%	94.66%	94.40%	99.72%	91.22%	91.22%	99.36%
		97.63%			73.64%			96.99%			95.12%		
5	Madelon	62.04%	60.80%	67.77%	63.81%	63.70%	64.23%	50.00%	-	0.00%	64.96%	65.05%	64.85%
		64.10%			63.96%			0.00%			64.95%		
6	Musk	99.95%	99.98%	99.96%	94.30%	96.81%	96.45%	84.59%	84.59%	100.00%	98.77%	98.92%	99.64%
		99.97%			96.63%			91.65%			99.28%		
7	Spambase	88.87%	95.06%	86.12%	89.52%	91.74%	90.92%	90.76%	89.79%	95.66%	93.00%	94.17%	94.30%
		90.37%			91.33%			92.63%			94.23%		
8	SPECTF Heart	87.50%	86.50%	90.00%	80.00%	77.50%	90.00%	82.50%	80.55%	90.00%	76.25%	78.50%	77.50%
		88.22%			83.28%			85.01%			78.00%		
9	Intrusion	86.75%	92.98%	77.44%	99.24%	99.61%	98.77%	98.54%	99.62%	97.23%	98.36%	98.80%	97.67%
		84.50%			99.19%			98.41%			98.23%		

In musk dataset, boosting-NB achieved the best F-measure score of 99.93% when applied to GA-reduced dataset which has 168 attributes. Interestingly, boosting-NB had a better F-measure score of 99.97% when applied to PSO-reduced dataset which has 16 attributes only. It also happened to internet_ads dataset, boosting-DT had F-measure score of 96.80% when applied to GA-reduced dataset which has 1,559 attributes, but boosting-NB had better F-measure score of 97.63 when applied to PSO-reduced dataset which has 302 attributes only.

4.4 Summary

We compare the performance of bagging and boosting based on learning time (time to build a classification model) and classification performance. Table 4.8 shows that boosting was much slower than bagging. Naïve Bayes as a single classifier only needs learning time of 0.25 seconds in average, bagging-NB requires 2.54 seconds and boosting-NB needs 21.10 seconds. kNN and bagging-kNN have an almost zero learning time, in contrast boosting-kNN have the longest learning time with 142.04 seconds in average.

Decision tree as a single classifier has the longest learning time with 6.71 seconds in average. Among four bagging base classifiers, bagging-RI also has the longest learning time with 83.09 seconds in average. Overall, boosting is much slower than bagging in term on learning time.

Table 4.8 shows that the feature selection algorithms work well on almost all (7 of 8) datasets even with fewer attributes. Only in 1 dataset (internet_ads), the F-measure was slightly decreased from 98.20% (with 1,559 original features) to 98.07% (GA-reduced dataset with 489 features) and 97.77% (PSO-reduced dataset with 302 features). We summarized the results of all bagging and boosting experiments in Table 4.9.

Table 4.8 Learning time of single base classifier, bagging and boosting

Dataset Name	Learning Time (second)											
	Single Classifier				Bagging Algorithm				Boosting Algorithm			
	NB	k-NN	DT	RI	Bagging NB	Bagging kNN	Bagging DT	Bagging RI	AdaBoost NB	AdaBoost kNN	AdaBoost DT	AdaBoost RI
Leukemia	0.09	0	0.33	0.33	0.39	0.02	0.62	1.43	0.36	0.27	1.98	2.52
Embryonal Tumours	0.08	0	0.39	0.61	0.45	0	2.60	3.93	1.26	0.14	2.57	2.92
Dexter	1.38	0	8.60	16.11	15.04	0.03	118.38	285.71	143.05	61.03	143.15	262.24
Internet_ads	0.10	0	16.78	16.99	0.71	0.01	122.68	112.49	18.29	133.63	301.44	75.97
Madelon	0.17	0	2.40	5.99	2.36	0.01	50.65	134.34	9.72	20.89	60.15	31.00
Musk	0.22	0	0.86	13.43	1.79	0.02	6.65	107.34	8.17	15.23	0.67	63.32
Spambase	0.05	0	0.42	1.28	0.33	0	4.14	20.50	0.65	61.41	3.95	4.71
SPECTF Heart	0.10	0	0.30	0.16	0.17	0.02	0.52	0.44	0.85	0.32	0.75	0.63
Intrusion	0.10	0	1.12	5.53	1.60	0.06	17.94	81.67	7.58	985.47	11.79	29.26
Average learning time	0.25	0.00	3.47	6.71	2.54	0.02	36.02	83.09	21.10	142.04	58.49	52.51

Table 4.9 Summary of bagging and boosting performance

Dataset	Single Classifier on original datasets				Bagging							
					GA-reduced datasets				PSO-reduced datasets			
	NB	kNN	DT	RI	NB	kNN	DT	RI	NB	kNN	DT	RI
Leukemia	98.31%	89.70%	78.73%	83.40%	98.31%	79.38%	85.32%	87.07%	98.31%	87.90%	69.12%	83.41%
Emb. Tumours	74.41%	67.06%	58.68%	74.05%	63.71%	75.01%	67.91%	80.08%	65.45%	75.08%	81.15%	77.27%
Dexter	81.39%	86.63%	86.79%	83.23%	73.94%	61.64%	86.05%	86.41%	72.71%	-	44.56%	74.42%
Internet_ads	98.20%	91.45%	86.18%	95.00%	97.97%	78.43%	81.18%	95.12%	97.77%	73.32%	97.04%	95.12%
Madelon	59.05%	64.90%	64.29%	73.32%	59.59%	65.77%	50.00%	73.79%	60.05%	63.78%	50.00%	67.27%
Musk	93.67%	97.15%	92.57%	95.16%	95.23%	96.38%	92.96%	96.20%	99.91%	96.65%	91.65%	95.79%
Spambase	82.90%	85.62%	92.59%	93.05%	80.39%	90.52%	92.78%	93.29%	90.08%	91.06%	93.29%	93.79%
SPECTF Heart	79.49%	67.53%	79.69%	61.90%	89.22%	74.86%	80.77%	79.54%	88.14%	80.95%	78.83%	80.90%
Intrusion	89.47%	99.14%	99.47%	91.61%	87.88%	99.74%	99.25%	91.61%	84.18%	99.21%	98.52%	91.61%

Dataset	Single Classifier on original datasets				Boosting							
					GA-reduced datasets				PSO-reduced datasets			
	NB	kNN	DT	RI	NB	kNN	DT	RI	NB	kNN	DT	RI
Leukemia	98.31%	89.70%	78.73%	83.40%	98.31%	85.44%	73.47%	91.89%	98.31%	89.72%	77.77%	74.58%
Emb. Tumours	74.41%	67.06%	58.68%	74.05%	75.96%	80.71%	71.76%	78.66%	70.32%	70.88%	70.83%	74.35%
Dexter	81.39%	86.63%	86.79%	83.23%	-	-	62.07%	-	73.19%	73.75%	49.06%	70.17%
Internet_ads	98.20%	91.45%	86.18%	95.00%	14.25%	78.46%	96.80%	94.78%	97.63%	73.64%	96.99%	95.12%
Madelon	59.05%	64.90%	64.29%	73.32%	60.37%	65.84%	-	65.41%	64.10%	63.96%	-	64.95%
Musk	93.67%	97.15%	92.57%	95.16%	99.93%	96.40%	92.02%	98.77%	99.97%	96.63%	91.65%	99.28%
Spambase	82.90%	85.62%	92.59%	93.05%	80.36%	90.51%	91.72%	94.52%	90.37%	91.33%	92.63%	94.23%
SPECTF Heart	79.49%	67.53%	79.69%	61.90%	80.70%	75.70%	77.40%	72.58%	88.22%	83.28%	85.01%	78.00%
Intrusion	89.47%	99.14%	99.47%	91.61%	93.42%	99.71%	99.22%	99.37%	84.50%	99.19%	98.41%	98.23%

In leukemia dataset, the NB algorithm when applied to full features dataset with 7,130 attributes achieved 98.31% F-measure. Interestingly, the F-measure score is still the same (98.31%) when NB applied to the GA-reduced dataset with less attributes (2,237 attributes).

The use of bagging-NB or boosting-NB to PSO-reduced dataset is successfully able to maintain the F-measure score (98.31%) even though with much less attributes (109 attributes). In the other six datasets (E. tumours, dexter, madelon, musk, spambase and SPECTF heart), the use of reduced-datasets is able to improve the classification accuracy as well as to reduce the computation time.

Bagging achieved the best results on 4 of 9 datasets: emb. tumours (81.15%), madelon (73.79%), SPECTF heart (89.22%) and intrusion (99.74%). Boosting achieved the best results in only 2 of 9 datasets: musk (99.97%) and spambase (94.52%). Overall, ensemble classifiers (both bagging and boosting) outperformed single classifier in 6 of 9 datasets.

5. Support Vector Machine and Parameter Optimization

This chapter focuses on the application of SVM on various high dimensional datasets. To build a powerful SVM model, we need to know how to pre-process the data, what kernel to use and most importantly is how to set the SVM parameters. The SVM kernels and their regularization parameters are called the hyper-parameters of SVM.

We conduct several SVM experiments, first we applied SVM using default parameters then we investigate the effects of normalization and its affect to the SVM performance. After that, we implement SVM parameter optimization with grid search and more sophisticated technique such as evolutionary algorithm.

5.1 SVM with default parameters and un-scaled data

There are a lot of SVM variations since it was firstly developed by Vapnik (Cortes and Vapnik, 1995). We compared some SVM implementation such as mySVM[3] developed by Stefan Ruping, Sequential Minimal Optimization (SMO) developed by John Platt (Platt, 1999) and LibSVM developed by Chang and Lin (Chang and Lin, 2011). Our initial experiment showed that actually all of these SVM applications have given satisfactory results in terms of accuracy but only LibSVM has an ability and stability to handle very large datasets such as dexter dataset (20,000 attributes), leukemia dataset (7,130 attributes), embryonal tumour (7,130 attributes), etc. Therefore, we decided to use LibSVM (Chang and Lin, 2011) for our experiments which has been integrated to Weka and RapidMiner data mining tools.

[3] http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html

We used nine high-dimensional datasets and applied 10-fold crossed validation. In each dataset, we applied SVM with four different kernels (linear, RBF, polynomial and sigmoid kernel).

Table 5.1 LibSVM default parameters

Parameters	Kernels	Type	Default
C	linear, RBF, sigmoid, polynomial	real	0.0
γ (gamma)	RBF, sigmoid, polynomial	real	0.0
degree	polynomial	integer	3

In LibSVM module both in RapidMinar and Weka, the linear kernel has several parameters but the most important is C (cost), the penalty parameter of the error. The default value of C is 0. For a large value of C, a large penalty is assigned to margin errors while a smaller value of C allows to ignore points close to the boundary and increases the margin.

The RBF kernel and sigmoid kernel have 2 important parameters: C and γ (gamma). The value of γ strongly affects the classification performance of SVM model. The default value of $\gamma$is 0. The polynomial kernel has more parameters than other kernels but the most important ones are C, $\gamma$and degree (the default value is 3).

In the first experiment, we applied LibSVM to 9 datasets without normalisation or scaling and the results are shown in Table 5.2.

Table 5.2 The results of SVM with default parameters and un-scaled data

No	PSO-reduced datasets	SVM kernels			
		Linear	RBF	Polynomial	Sigmoid
		F Measure	F Measure	F Measure	F Measure
1	Leukemia	74.11%	84.25%	80.94%	66.67%
2	Embryonal Tumours	74.50%	76.67%	74.50%	76.67%
3	Dexter	74.92%	68.70%	63.00%	53.18%
4	Internet_ads	96.81%	92.19%	96.81%	95.08%
5	Madelon	61.45%	65.59%	60.55%	66.67%
6	Musk	91.29%	96.58%	93.03%	78.74%
7	Spambase	79.70%	84.91%	73.90%	64.84%
8	SPECTF Heart	74.11%	84.25%	80.94%	66.67%
9	Intrusion NSLKDD	26.70%	94.41%	40.03%	84.44%

From nine datasets, the RBF kernel achieved the best results on 5 datasets (leukemia 84.25%, embryonal tumours 76.67%, spambase 84.91%, SPECTF heart 84.25% and intrusion 94.41). Compared to other kernels, the RBF kernel is able to handle un-scaled or un-normalized data much better. Linear, polynomial and sigmoid kernel achieved the best results in 2 datasets only.

5.2 The effect of normalization

In order to achieve higher accuracy rates, it is very important to do standarization or Z-score normalization or scaling to avoid attributes with greater numeric ranges dominating other attributes with smaller ranges. It is also useful to avoid numerical difficulties during the calculation because kernel values highly depend on the inner products of feature vectors, therefore large attribute values might cause numerical errors. In many datasets, the available attributes are continuous type, where each attribute is measured in a different scale and has a different set of possible values. In this case, scaling or normalization will convert all attributes into the same scale using this following formula:

$$z = \frac{x - \mu}{\sigma} \tag{5.1}$$

where x is a data point, μ is the mean (average), σ is the standard deviation from the mean and z is the value after normalization.

We ran another experiment, similar to the previous one (Table 5.2) but now with scaled or normalized datasets and the results are shown in Table 5.3. In the previous experiment, RBF kernel achieved the best results on 5 of 9 un-scaled datasets, but in the latest experiment with scaled/normalized datasets linear kernel achieved the best results on 6 of 9 datasets. RBF kernel has the best results on intrusion dataset only. The results comparison between before and after normalization is explainedin more details in Table 5.4.

Table 5.4 shows that the linear kernel is very sensitive to the un-scaled or un-normalized dataset. Scaling or normalization significantly improved the performance especially in linear kernel as well as polynomial and sigmoid kernel. The highest improvement of linear kernel performance is on the

intrusion dataset where the F-measure increased from 26.70% to 88.38% (61.69% improvement), followed by leukemia (24.31% improvement), SPECT heart (15% improvement), spambase (12.77% improvement), musk (8.57% improvement) and embryonal tumours (2.17% improvement).The other three datasets (dexter, internet_ads and madelon) did not have significant improvement because they are already scaled or normalized.

Table 5.3 The SVM results on normalized data

No	PSO-reduced datasets	SVM kernels			
		Linear	RBF	Polynomial	Sigmoid
		F Measure	F Measure	F Measure	F Measure
1	Leukemia	98.41%	89.82%	98.41%	96.27%
2	Embryonal Tumours	76.67%	75.05%	76.67%	74.89%
3	Dexter	74.92%	68.70%	63.00%	53.18%
4	Internet_ads	97.41%	97.27%	95.88%	95.08%
5	Madelon	61.83%	57.13%	57.04%	62.04%
6	Musk	99.87%	89.73%	94.97%	99.83%
7	Spambase	92.46%	88.14%	74.48%	92.56%
8	SPECTF Heart	89.11%	82.71%	69.29%	89.11%
9	Intrusion NSLKDD	88.38%	89.33%	88.07%	88.02%

Overall, normalization increases the performance of linear kernel to 13.94% in average, polynomial kernel is 6.01% in average and sigmoid kernel is 10.89% in average. In contrast, normalisation could not improve the performance of RBF kernel. There are only 3 of 9 datasets (leukemia, internet_ads and spambase) that have slightly improvements, 1 dataset (dexter) has the same accuracy and 5 other datasets (embryonal tumours, madelon, musk, SPECTF heart and intrusion) have worse performance. Overall the performance of RBF kernel on scaled data was -5.08%. Based on a report published by Li (Li et al., 2012), RBF kernel is a kind of normalized kernel function, therefore this kernel does not need a normalization on input dataset.

Table 5.4 The effect of normalization to the SVM's classification performance

No	PSO-reduced datasets	Linear			RBF			Polynomial			Sigmoid		
		before	after	+/-	before	after	+/-	before	after	+/-	before	after	+/-
1	Leukemia	74.11%	98.41%	24.31%	84.25%	89.82%	5.57%	80.94%	98.41%	17.48%	66.67%	96.27%	29.60%
2	Embryonal Tumours	74.50%	76.67%	2.17%	76.67%	75.05%	-1.61%	74.50%	76.67%	2.17%	76.67%	74.89%	-1.78%
3	Dexter	74.92%	74.92%	0.00%	68.70%	68.70%	0.00%	63.00%	63.00%	0.00%	53.18%	53.18%	0.00%
4	Internet_ads	96.81%	97.41%	0.60%	92.19%	97.27%	5.08%	96.81%	95.88%	-0.93%	95.08%	95.08%	0.00%
5	Madelon	61.45%	61.83%	0.38%	65.59%	57.13%	-8.46%	60.55%	57.04%	-3.51%	66.67%	62.04%	-4.63%
6	Musk	91.29%	99.87%	8.57%	96.58%	89.73%	-6.85%	93.03%	94.97%	1.94%	78.74%	99.83%	21.09%
7	Spambase	79.70%	92.46%	12.77%	84.91%	88.14%	3.23%	73.90%	74.48%	0.58%	64.84%	92.56%	27.72%
8	SPECTF Heart	74.11%	89.11%	15.00%	84.25%	82.71%	-1.54%	80.94%	69.29%	-11.65%	66.67%	89.11%	22.44%
9	Intrusion NSLKDD	26.70%	88.38%	61.69%	94.41%	89.33%	-5.08%	40.03%	88.07%	48.05%	84.44%	88.02%	3.58%
				13.94%			-1.07%			6.01%			10.89%

5.3 SVM parameter optimization

The largest problems encountered in setting up the SVM model are how to select the kernel function and its parameter values. Inappropriate parameter settings lead to poor classification results (Keerthi & Lin, 2003). Furthermore, current SVM algorithm can not easily handle high dimensional datasets. A standard SVM algorithm requires solving linear or quadratic program, Therefore, there is a need to improve this basic SVM algorithm in order to improve its ability to handle high dimensional datasets.

To improve the SVM performance, we conducted various experiments where we independently test each kernel and tune its parameters. The design of SVM parameter optimization module is shown in Figure 5.1 below.

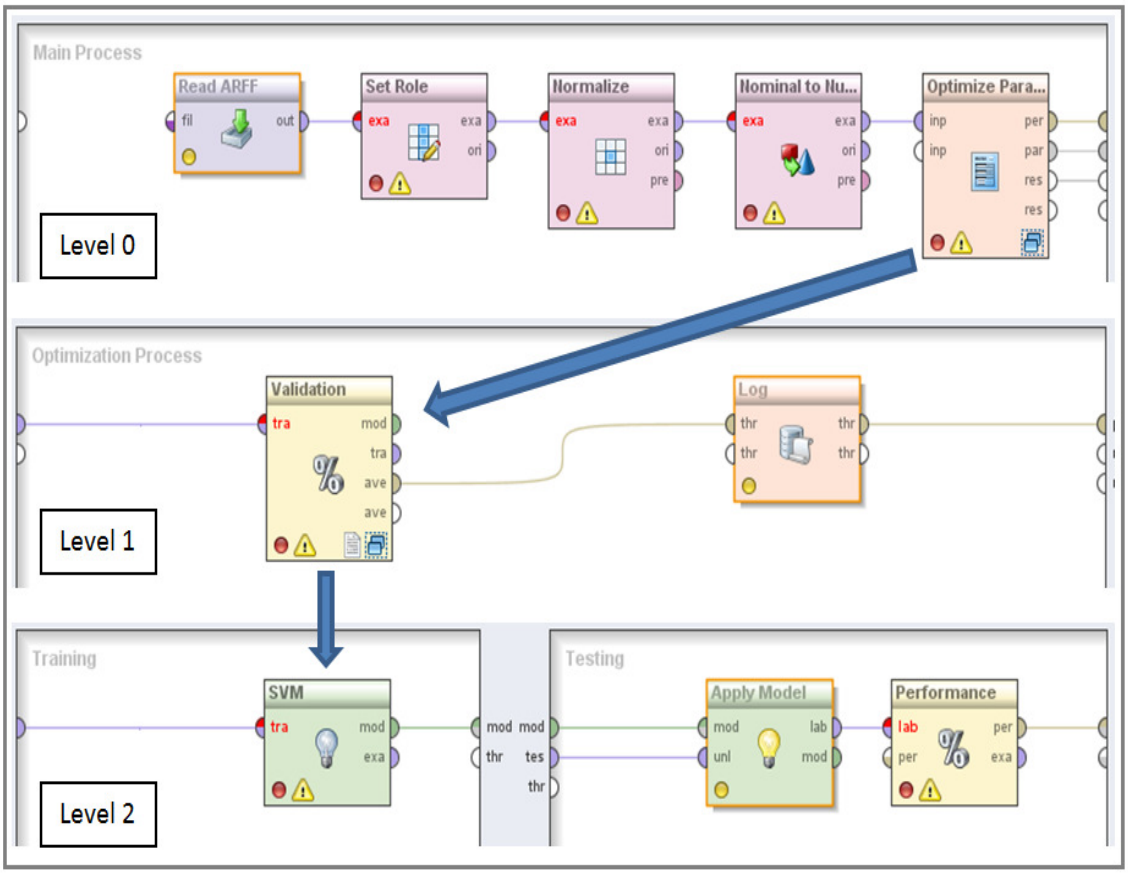

Figure 5.1 The design of the SVM parameter optimization module

The SVM parameter optimization is the problem of selecting a set of hyper-parameters for an SVM algorithm with the goal of obtaining the best

generalization. The accuracy results are very sensitive to kernel parameters especially C and γ. To get the proper values of C and γ, we need to simultaneously optimize both C and γ since optimal C typically depends on γ and other parameters.

In this research, we used grid search and evolutionary algorithms to optimize the SVM parameters.

5.3.1 Grid search

The grid search is originally an exhaustive search based on defined subset of the hyper-parameter space. The hyper-parameters are specified using minimal value (lower bound), maximal value (upper bound) and number of steps. There are four different scales that can be used: linear scale, quadratic scale, logarithmic scale and logarithmic legacy scale. The performance of every combination is evaluated using some performance metrics.

Grid search optimizes the SVM parameters (C, γ, degree, etc.) using a cross validation (CV) technique as a performance metric. The goal is to identify good hyper-parameter combination so that the classifier can predict unknown data accurately. According to (Chang and Lin, 2011), the cross-validation technique can prevent the over-fitting problem.

To choose C and γ using k-fold CV, we first split the available data into k subsets (in most experiments we set k=10). One subset is used as a testing data and then evaluated using the remaining k-1 training subsets. Then we calculate the CV error using this split error for the SVM classifier using different values of C, $\gamma$and other parameters. Various combination of hyper-parameters value are entered and the one with the best cross-validation accuracy (or the lowest CV error) is selected and used to train an SVM on the whole dataset.

In linear kernel there is only one important parameter to be optimized which is C, in RBF kernel and sigmoid kernel there are 2 parameters: C and γ while polynomial kernel has 3 parameters: C, γ and degree. Actually there are many parameters than we have not yet mentioned, but selecting more parameters and a large number of steps (or possible values of parameters) results in a huge number of combinations. For example, if we choose to optimize 5

parameters and 25 steps for each parameter, then the total combinations would be 25^5 or 9,765,625 which requires a huge (impractical) amount of time.

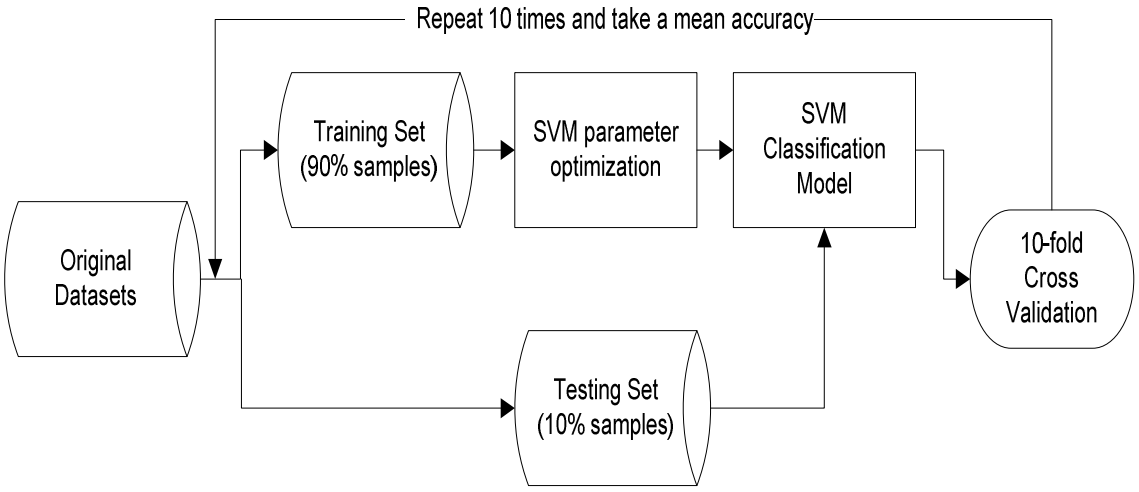

Figure 5.2 SVM parameter optimization using 10-fold cross validation

One of the biggest problems of SVM parameter optimization is that there are no exact ranges of C and γ values. We believe that the wider the parameter range is, the more possibilities the grid search method has of finding the best combination parameter. Therefore, in our experiment we decided to make the range of C and γ from 0.001 to 10,000.

Table 5.5 Hyper parameters range for experiments

Parameters	Kernel	Min	Max	Type	Steps	Scale
C	linear	0.001	10,000	real	10	logarithmic or logarithmic legacy
gamma	linear, RBF, sigmoid,	0.001	10,000	real	10	logarithmic or logarithmic legacy
degree	polynomial	1	5	integer	1	linear (1,2,3,4,5)

The SVM parameters used in our experiment are explained in Table 5.5. If we set C parameter from 0.001 (minimal) to 10,000 (maximal) with 10 steps using a logarithmic scale, the C value is initially 0.001 then it is increased logarithmically until it reaches 10,000 in the last iteration. The C values are 0.001, 0.005, 0.025, 0.126, 0.631, 3.162, 15.849, 79.433, 398.107, 1995.262 and 10000. If we use logarithmic legacy, the values are 5.311, 14.850, 38.813, 99.006, 250.205, 630.002, 1584.021, 3980.431 and 10000.

The Figure 5.3 below explains how the grid search works to find the best SVM parameters.

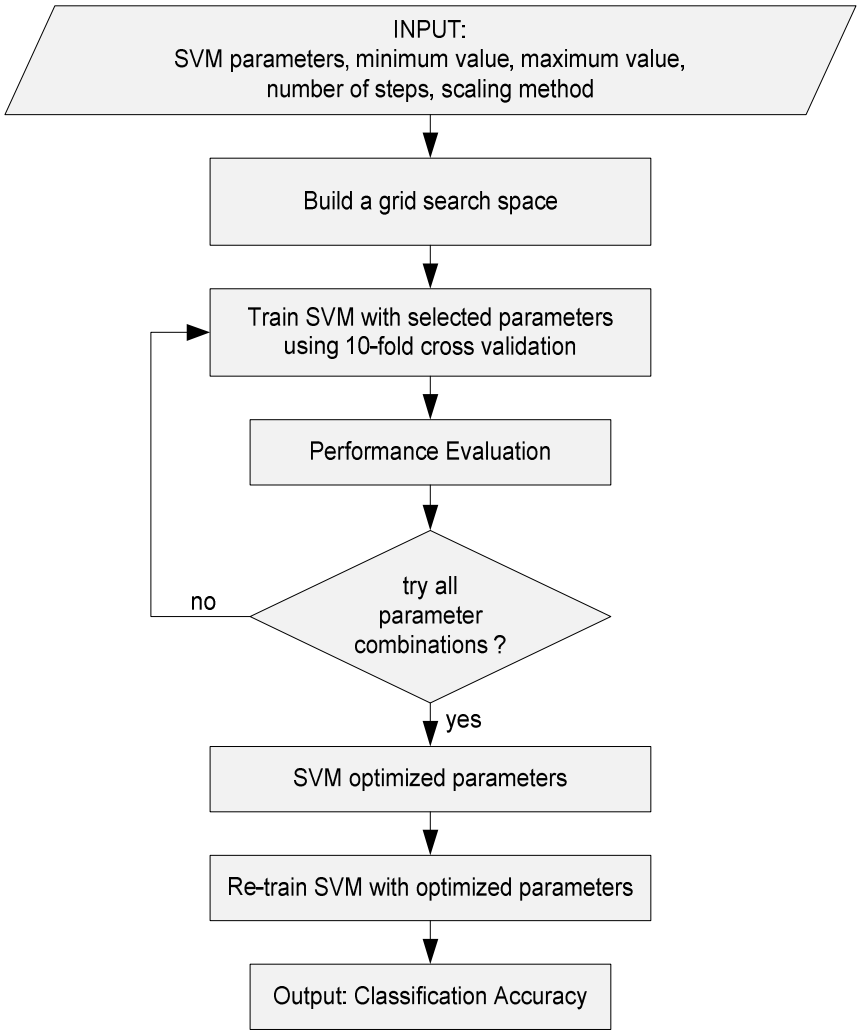

Figure 5.3 SVM parameter using GRID search

The results of SVM parameter optimization using grid search are shown in the Table 5.6, this table shows that parameter optimization using grid search is very powerful and it is able to improve the accuracy significantly. In leukemia and musk datasets, this technique achieved 100% accuracy for all four kernels, these results are amazing especially as it was applied to PSO-reduced datasets which have many fewer attributes compare to the original ones. The 100% accuracy is much better than the results of 4 basic machine learning algorithms (decision tree, naïve Bayes, nearest neighbour and rule induction) applied to full attributes and 3 ensemble classifiers (bagging, boosting and stacking) applied to GA-reduced datasets and PSO-reduced datasets (please see Chapter 4).

However, grid search has an disadvantage which is extremely slow. In Table 5.6 we can see that the intrusion dataset (all kernels), spambase (polynomial and sigmoid kernel), madelon (polynomial kernel), internet_ads (polynomial kernel) and dexter (polynomial kernel) were failed. The program executions were forced to stop after running for 1 week or 2 weeks without any results.

In this experiment, linear kernel achieved best results in 5 of 7 datasets, not including 2 datasets (spambase and intrusion) that grid search failed to finish. RBF kernel produced best results on 2 datasets (madelon and spambase) while sigmoid kernel achieved best results only on 1 dataset (SPECTF heart).

Table 5.6 shows that the best SVM parameters found by grid search have various values. There is no clear relationship between the value of C and γ, in some datasets the C values are larger than γbut in other datasets it is vice versa.

This experiment shows that the grid search always finds near optimal parameter combination within the given ranges, unfortunately it is very time consuming where the computation time scale is K^M. If the dimension of datasets is quite high or the number of parameter combinations is huge, the grid search might be never finished. Therefore, grid search is very reliable only in low dimensional dataset with few parameters.

5.3.2 Evolutionary algorithm

Evolutionary algorithm (EA) can also be used for SVM parameter optimization. EAs search the best parameters but not naively like a brute-force or grid search. EA is very useful to be implemented when the best ranges and dependencies of various SVM parameters is not known at all. EA is more appropriate than grid search which is very time consuming because it tries too many combinations of parameters.

Beside SVM parameters (C, γ, degree, etc.), there are other EA parameters that need to be optimized in order to achieve the best results:

- **Max generations**: sets the number of generations for process termination, the default value is 50
- **Population size**: specifies the population size or the number of individuals per generation, the default value is 5

Table 5.6 The results of parameter optimization using grid search

No	PSO-reduced datasets	Linear		RBF		Polynomial		Sigmoid	
		F Measure	Best parameters	F Measure	Best parameters	F Measure	Best parameters	F Measure	Best parameters
1	Leukemia	100.00%	C = 31.622776	100.00%	C=31.6228 gamma=0.001	100.00%	C =7.0454 gamma=250.4695 degree =1	100.00%	C=31.62277 gamma=0.001
2	Embryonal Tumours	84.95%	C=0.762	76.67%	C=0.0999 gamma=0.0999	81.56%	C=3.082 gamma=125.072 degree=1	81.61%	C = 3.082 gamma=125.072 degree=1
3	Dexter	78.68%	C=6.9466	75.13%	C=63.0957344 gamma=0.003981	failed, no results	forced to stop after 1 week running	78.22%	C=63.095734 gamma=0.00398
4	Internet_ads	97.54%	C=1.0	97.44%	C=0.9965 gamma=0.9956	failed, no results	forced to stop after 1 week running	95.60%	C=1000.0 gamma=0.000099999
5	Madelon	62.28%	C=250.3904	66.07%	C=0.9965 gamma=0.9956	failed, no results	forced to stop after 1 week running	62.02%	C=1000.0 gamma=0.000099999
6	Musk	100.00%	C=0.25118864	100.00%	C=63.095 gamma=0.001	100.00%	C=0.00398 gamma=125.072 degree=1	100.00%	C=251.18864 gamma=0.001
7	Spambase	failed, no results		94.36%	C=31.62277 gamma=0.01	failed, no results		failed, no results	
8	SPECTF Heart	86.81%	C=220.499	89.97%	C=63.0957 gamma=0.015848	90.36%	C=1.0 gamma=0.0630 degree=1	91.75%	C=1.0 gamma=15.8489
9	Intrusion	failed, no results	forced to stop after 2 weeks running	failed, no results	forced to stop after 2 weeks running	failed, no results	forced to stop after 2 weeks running	failed, no results	forced to stop after 2 weeks running

- **Tournament fraction**: specifies the fraction of the current population which should be used as tournament members, the default value is 0.25
- **Crossover prob**: specifies the probability for an individual to be selected for crossover, the default value is 0.9
- **Mutation type**: there are three mutation types which are Gaussian mutation, switching mutation and sparsity mutation. We used the default value: Gaussian mutation
- **Selection type**: there are eight different selection types which are union, cut, roulette wheel, stochastic universal sampling, Bolztmann, rank and tournament (default value).

The parameter optimization using EA algorithm is explained in the Figure 5.4.

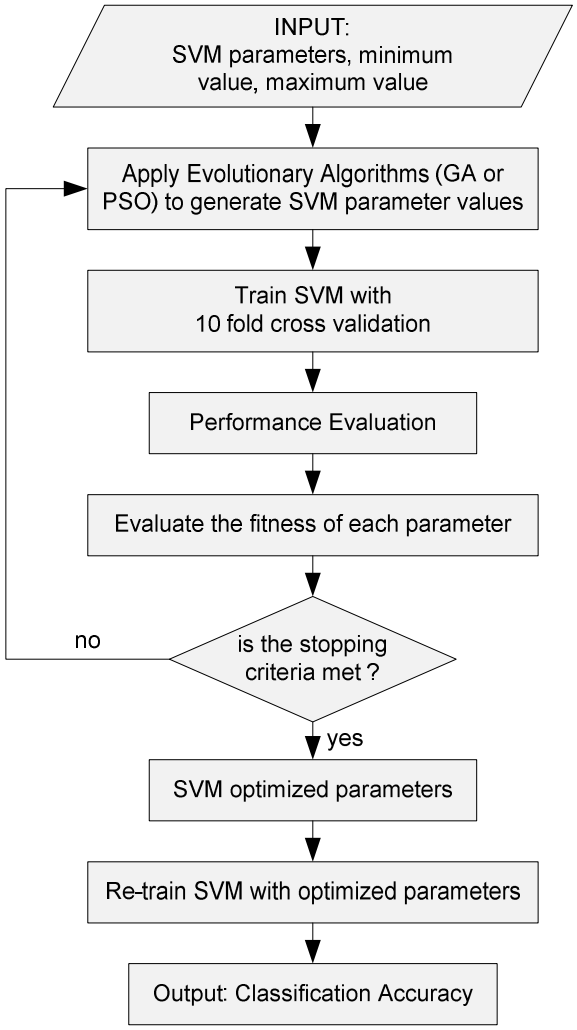

Figure 5.4 Parameter Optimization using Evolutionary Algorithm

We now optimized the SVM parameters using evolutionary algorithm which much faster than the grid search. We trained SVM classifiers using four different kernels (linear, RBF, polynomial and sigmoid) and the results are shown in Table 5.7.

In the previous experiment, grid search had failed because of very long execution time and returned no results when applied to intrusion (all four kernels), spambase (linear, polynomial and sigmoid kernel), madelon (polynomial kernel), internet_ads (polynomial kernel) and dexter (polynomial kernel) datasets. In the current experiment, evolutionary search has proven to be more stable than grid search.

Table 5.7 shows that evolutionary search has successfully given satisfactory results in almost all datasets. Linear kernel achieved the best results in 4 of 9 datasets which are on leukemia, dexter, internet_ads and musk datasets. RBF kernel outperforms other kernels in spambase and intrusion datasets while the polynomial kernel achieved the best results in embryonal tumours and musk datasets. Sigmoid kernel achieved 100% accuracy in leukemia dataset and achieved the best results in SPECTF heart dataset.

To compare the performance of grid search and evolutionary search, we summarized both experiments in Table 6.3. In leukemia and musk datasets, grid search achieved 100% accuracy in 4 kernels while evolutionary search achieved 100% accuracy in 2 kernels. From these 2 datasets results, we can see that linear kernel is much faster than other kernels (RBF, polynomial and sigmoid kernels). In embryonal tumours, dexter, internet_ads, SPECTF Heart and intrusion datasets, evolutionary search has slightly better accuracy but much faster execution time. Only in 1 dataset (spambase) grid search has better accuracies than the evolutionary one.

In the madelon and the intrusion datasets evolutionary search could not guarantee good results for all kernels because the classification performances were not so good (in madelon datasets the F-measure is only 66.67% and in intrusion dataset the F-measure is only 61.31%).

Table 5.7 Parameter optimization using Evolutionary Algorithm

No	PSO-reduced datasets	Linear		RBF		Polynomial		Sigmoid	
		F Measure	Best parameters	F Measure	Best parameters	F Measure	Best parameters	F Measure	Best parameters
1	Leukemia	100.00%	C=68.773	21.85%	C=0.5625 gamma=0.94	98.75%	C=220.5046 gamma=484.66 degree=1	100.00%	C=564.6153 gamma=0.07
2	Embryonal Tumours	79.56%	C=0.5681	84.23%	C=5620.701 gamma=936.64	85.33%	C=0.0488 gamma=0.69 degree=1	83.33%	C=5646.186 gamma=687.63
3	Dexter	78.88%	C=69.7642	67.64%	C=564.6229 gamma=68.76	77.81%	C=0.59089 gamma=0.95 degree=1	77.01%	C=0.56692 gamma=0.07
4	Internet_ads	97.58%	C=623.381	94.33%	C=182.4549 gamma=68.76	90.67%	C=0.5625 gamma=0.94 degree=1	90.67%	C=0.5625 gamma=0.94
5	Madelon	66.67%	unoptimal results, premature convergence	66.67%	unoptimal results, premature convergence	66.67%	unoptimal results, premature convergence	66.67%	unoptimal results, premature convergence
6	Musk	100.00%	C=220.505	99.63%	C=0.62033 gamma=0.47	100.00%	C=182.45498 gamma=968.49 degree=1	81.69%	C=623.38429 gamma=863.35
7	Spambase	73.73%	C=220.504	83.42%	C=220.498 gamma=68.755	73.73%	c=240.516 gamma=0.40 degree=2	78.50%	C=182.45 gamma=968.49
8	SPECTF Heart	87.31%	C=2205.035	88.64%	C=0.562 gamma=936.64	90.31%	C=0.2233 gamma=0.48 degree=1	93.34%	C=623.38 gamma=863.36
9	Intrusion	61.31%	unoptimal results, premature convergence	95.43%	C=564.62 gamma=68.763	61.31%	unoptimal results, premature convergence	61.31%	unoptimal results, premature convergence

When we applied SVM to the madelon and intrusion datasets, the program executions were trapped in a local minimum and were terminated very early, therefore the results were not satisfactory. This problem is called premature convergence, the condition where a population for optimization process converged too early which results an un-optimal solutions.

As shown in Table 5.6, Table 5.7 and Table 5.8, in madelon dataset SVM failed to give satisfactory results even with parameter optimization techniques. When using grid search, three kernels (linear, RBF and sigmoid) only achieved accuracy less than 70% (maximum accuracy was only 66.07%) and even worse the polynomial kernel failed to give any results due to very long program execution (the program execution was forced to stop after 1 week).

Unfortunately, the evolutionary search has also failed to improve the performance where all four kernels (linear, RBF, polynomial and sigmoid kernel) have very poor results. The program executions were trapped in a local minimum and then terminated very early, therefore the results were not satisfactory.

Similar case also happened to intrusion dataset, the grid search also failed to get the results in all kernels where the program executions were forced to stop after running for 2 weeks. In the beginning, evolutionary search also failed to give satisfactory results because all kernels had premature convergence which leads to unoptimal solutions. Then we decided to adjust some evolutionary algorithm parameters such as maximum generations, population size and mutation type with some different values and it worked very well especially on RBF kernel where the accuracy (or F-measure) was significantly improved from 61.31% to 95.43%.

We tried to adjust evolutionary search parameters to improve the SVM performance on madelon dataset as we have successfully done in intrusion dataset, unfortunately it failed. We were really curious to know about the main reasons why the SVM classifier failed on the madelon dataset. We decided to run more experiments applying various algorithms to madelon dataset only. The results are explained in Table 5.9.

Table 5.8 Grid search and evolutionary search results comparison

No	Datasets	Number of instances	Number of original attributes	Number of attributes after reduced by PSO		SVM with grid search			SVM with evolutionary search		
						F Measure	Kernels	Exec. Time (hh:mm:ss)	F Measure	Kernels	Exec. Time (hh:mm:ss)
1	Leukemia	72	7,130	109	1.53%	100.00%	linear	00:00:05	100.00%	linear	00:00:02
							RBF	00:00:34		sigmoid	00:00:03
							polynomial	00:00:28			
							sigmoid	00:00:14			
2	Embryonal Tumours	60	7,130	202	2.83%	84.95%	linear	00:00:02	85.33%	polynomial	00:00:03
3	Dexter	600	20,000	279	1.40%	78.68%	linear	05:56:03	78.88%	linear	00:20:05
4	Internet_ads	3,279	1,559	302	19.37%	97.54%	linear	00:20:13	97.58%	linear	00:16:15
5	Madelon	2,600	501	5	1.00%	66.07%	RBF	00:26:32	66.67%	linear	00:00:02
										RBF	00:00:02
										polynomial	00:00:02
										sigmoid	00:00:02
6	Musk	6,598	168	16	9.52%	100.00%	linear	00:21:20	100.00%	linear	00:19:32
							RBF	16:31:02		polynomial	00:30:12
							polynomial	00:46:59			
							sigmoid	04:13:21			
7	Spambase	4,601	58	27	46.55%	94.36%	RBF	01:37:30	83.42%	linear	
										RBF	00:47:44
										polynomial	
8	SPECTF Heart	80	45	9	20.00%	91.75%	sigmoid	00:00:20	93.34%	sigmoid	00:00:04
9	Intrusion	25192	42	8	19.05%	no results	all kernels were failed	program was forced to stop after running for 2 weeks without any results	95.43%	linear	
										RBF	17:13:36
										polynomial	
										sigmoid	

Table 5.9 Experiment results on madelon dataset

Machine Learning Algorithms		Madelon dataset (2,600 examples)		
		Original attributes	GA-reduced	PSO-reduced
		501 attributes	142 attributes	5 attributes
Single classifiers				
	k Nearest Neighbour (kNN)	64.90%	65.23%	64.25%
	Naïve Bayes (NB)	59.05%	59.35%	60.24%
	Decision Tree (DT)	64.28%	64.29%	64.29%
	Rule Induction (RI)	73.32%	68.15%	63.07%
Ensemble Classifiers				
	Bagging-kNN	-	65.77%	63.78%
	Bangging-NB	-	59.59%	60.05%
	Bangging-DT	-	50.00%	50.00%
	Bangging-RI	-	73.79%	67.27%
	Adaboost-kNN	-	65.84%	63.96%
	Adabost-NB	-	60.37%	64.10%
	Adaboost-DT	-	failed	failed
	Adaboost-RI	-	65.41%	64.95%
Support Vector Machine (SVM)				
	linear kernel	66.67%	66.67%	61.83%
	RBF kernel	66.67%	66.67%	57.13%
	polynomial kernel	66.67%	66.67%	57.04%
	sigmoid kernel	66.67%	66.67%	62.04%
SVM with parameter optimization using grid search				
	linear kernel	66.67%	66.67%	62.28%
	RBF kernel	66.67%	66.67%	66.07%
	polynomial kernel	66.67%	66.67%	failed
	sigmoid kernel	66.67%	66.67%	62.02%
SVM with parameter optimization using evolutionary algorithm				
	linear kernel	66.67%	66.67%	66.67%
	RBF kernel	66.67%	66.67%	66.67%
	polynomial kernel	66.67%	66.67%	66.67%
	sigmoid kernel	66.67%	66.67%	66.67%

We applied various machine learning algorithms from the basic ones to ensemble classifiers and SVM with parameter optimization but none of them achieved satisfactory results. The highest accuracy was only 73.79% which produced by the bagging-rule induction algorithm. In this case, the bagging technique hadsuccessfully improved the accuracy from 73.32% (rule induction

as a single classifier) to 73.79% (bagging-RI), but this result was still less than our expectation.

Actually the madelon dataset is an artificial dataset that is generated by a computer program which was used in the NIPS 2003 feature selection challenge[4]. The goal of this dataset is to classify random data which have 2 classes and sparse binary input variable. We reckon that the poor classification performance of madelon dataset was strongly related to the feature selection algorithms to be used. We used GA and PSO as feature selection algorithms and they performed very well in almost all datasets except madelon. We speculate that if we use more advance feature selection techniques on madelon dataset, most of the classifiers will give better results. The winner of NIPS 2003 feature selection challenge used a combination of Bayesian neural network and diffusion tree (Guyon et al., 2004).

[4] http://www.nipsfsc.ecs.soton.ac.uk/papers/NIPS2003-Datasets.pdf

6. Summary and Discussion

In this chapter we would like to compare the performance of basic classifiers when applied to high dimensional datasets, and the performance of ensemble classifiers as well as SVM with parameter optimizations which applied to reduced datasets.

The key question when dealing with the classification problem is not to find a learning algorithm which is superior to others, but under which conditions a specific method can significantly outperform others on a given application problem. The following sections are the summary of our experiment on basic machine learning algorithms, ensemble classifiers (bagging and boosting) and SVM with parameter optimization.

6.1 Summary of Feature Selection Algorithms

We summarized all feature selection algorithms experiments in Table 6.1. Feature selection algorithms have successfully removed irrelevant or redundant features as well as reduced computational cost. From 8 datasets, the GA has successfully reduced the number of attribute to 30.52% of the original attributes on average. The highest reduction rate was on embryonal tumours dataset where GA reduced the number of attributes from 7,130 to 619 or 8.68% of original attributes. PSO reduced the number of attributes much better than the GA with 12.78% on average. The highest reduction rate was in madelon dataset where PSO reduced the number of attributes from 501 to 5 or 1.00% of original attributes. In dexter dataset, PSO also reduced the number of attributes from 20,000 to 279 or 1.40% while in leukemia dataset, PSO successfully reduced the number of attributes from 7,130 to 109 or 1.53%.

Even though the feature selection algorithms have successfully reduced the number of attributes significantly, they do not always improve the classification performance. There are 4 of 8 datasets that when reduced by GA and PSO, have positive classification improvement for almost all classifiers, they are embryonal tumours, musk, spambase and SPECTF heart dataset. The highest average accuracy improvement was on SPECTH-heart PSO-reduced dataset with 10.05% improvement average. In embryonal tumours dataset,

decision tree algorithm achieved 58.68% accuracy when applied to original dataset, but its accuracy increased to 81.15% when applied to PSO-reduced dataset. It means the accuracy improvement was 22.47% which is the highest improvement.

Unfortunately, the reduction of attributes both by GA and PSO has also reduced the classification accuracy in 4 of 8 datasets which are leukemia, dexter, internet_ads and madelon datasets. The worst accuracy reduction was on dexter dataset where in the GA-reduced dataset the classification accuracy was dropped by 8.93% on average and in PSO-reduced dataset, the classification accuracy was dropped by 19.91% on average.

In the GA-reduced datasets, 3 of 4 classifiers achieved worse accuracy than if they applied to original datasets with full attributes. naïve Bayes, k-NN and decision tree had negative average improvement while rule induction was the only algorithm that has average positive improvement with 1.46% average. Contrarily, in PSO-reduced datasets there are three classifiers (naïve Bayes, k-NN and rule induction) achieved average positive improvement and only one classifier (decision tree) had negative average improvement.

As a conclusion, both GA and PSO as feature selection algorithms have successfully reduced the number of attributes significantly but they do not always improve the classification performance especially if basic machine learning algorithms such as naïve Bayes, k nearest neighbour, decision tree and rule induction was used as classifier.

6.2 Summary of Ensemble Classifiers

Table 6.2 shows that in the PSO reduced datasets, both bagging and boosting have slightly improved the accuracy. The average improvement varies from 0.38% (Bagging k-NN) to 4.47% (bagging rule induction). The highest improvement was 13.18% where the boosting decision tree increased the accuracy from 58.58% to 71.76%. From four basic classifiers, only rule induction constantly has positive improvement, bagging rule induction has average improvement of 2.59% (on GA-reduced datasets) and 4.47% (on PSO-reduced datasets) and boosting rule induction has average improvement of 1.09% (GA-reduced datasets) and 2.31% (PSO-reduced datasets).

Table 6.1 Summary of feature selection algorithms performance

Feature selection using GA																	
Dataset	Number of attributes			Classification Performance (F-Measure)												Maximum Improvement	Average Improvement
	Original data	reduced by GA		Naïve Bayes			k Nearest Neighbour			Decision Tree			Rule Induction				
		attributes	FF	Original	GA reduced	Improvement	Original	GA reduced	Improvement	Original	GA reduced	Improvement	Original	GA reduced	Improvement		
Leukemia	7,130	2,237	31.37%	98.31%	98.31%	0.00%	89.70%	78.55%	-11.15%	78.73%	81.16%	2.43%	83.40%	82.45%	-0.95%	2.43%	-2.42%
Emb. Tumours	7,130	619	8.68%	74.41%	65.42%	-8.99%	67.06%	78.82%	11.76%	58.68%	58.58%	-0.10%	74.05%	81.00%	6.95%	11.76%	2.41%
Dexter	20,000	6,133	30.67%	81.39%	73.30%	-8.09%	86.63%	60.04%	-26.59%	86.79%	87.16%	0.37%	83.23%	81.84%	-1.39%	0.37%	-8.93%
Internet_ads	1,559	489	31.37%	98.20%	98.07%	-0.13%	91.45%	78.49%	-12.96%	86.18%	88.32%	2.14%	95.00%	95.02%	0.02%	2.14%	-2.73%
Madelon	501	142	28.34%	59.05%	59.35%	0.30%	64.90%	65.23%	0.33%	64.29%	64.29%	0.00%	73.32%	68.15%	-5.17%	0.33%	-1.14%
Musk	168	66	39.29%	93.67%	95.23%	1.56%	97.15%	96.48%	-0.67%	92.57%	91.96%	-0.61%	95.16%	95.93%	0.77%	1.56%	0.26%
Spambase	58	29	50.00%	82.90%	80.34%	-2.56%	85.62%	90.33%	4.71%	92.59%	91.69%	-0.90%	93.05%	92.90%	-0.15%	4.71%	0.28%
SPECTF Heart	45	11	24.44%	79.49%	88.50%	9.01%	67.53%	74.57%	7.04%	79.69%	73.24%	-6.45%	61.90%	73.52%	11.62%	11.62%	5.31%
				Average Improvement		-1.11%	Average Improvement		-3.44%	Average Improvement		-0.39%	Average Improvement		1.46%		
Feature selection using PSO																	
Dataset	Number of attributes			Classification Performance (F-Measure)												Maximum Improvement	Average Improvement
	Original data	reduced by PSO		Naïve Bayes			k Nearest Neighbour			Decision Tree			Rule Induction				
		attributes	FF	Original	PSO reduced	Improvement	Original	PSO reduced	Improvement	Original	PSO reduced	Improvement	Original	PSO reduced	Improvement		
Leukemia	7,130	2,237	31.37%	98.31%	98.31%	0.00%	89.70%	87.90%	-1.80%	78.73%	69.12%	-9.61%	83.40%	83.41%	0.01%	0.01%	-2.85%
Emb. Tumours	7,130	619	8.68%	74.41%	65.45%	-8.96%	67.06%	75.08%	8.02%	58.68%	81.15%	22.47%	74.05%	77.27%	3.22%	22.47%	6.19%
Dexter	20,000	6,133	30.67%	81.39%	72.71%	-8.68%	86.63%	-	-	86.79%	44.56%	-42.23%	83.23%	74.42%	-8.81%	-8.68%	-19.91%
Internet_ads	1,559	489	31.37%	98.20%	97.77%	-0.43%	91.45%	73.32%	-18.13%	86.18%	97.04%	10.86%	95.00%	95.12%	0.12%	10.86%	-1.90%
Madelon	501	142	28.34%	59.05%	60.05%	1.00%	64.90%	63.78%	-1.12%	64.29%	50.00%	-14.29%	73.32%	67.27%	-6.05%	1.00%	-5.12%
Musk	168	66	39.29%	93.67%	99.91%	6.24%	97.15%	96.65%	-0.50%	92.57%	91.65%	-0.92%	95.16%	95.79%	0.63%	6.24%	1.36%
Spambase	58	29	50.00%	82.90%	90.08%	7.18%	85.62%	91.06%	5.44%	92.59%	93.29%	0.70%	93.05%	93.79%	0.74%	7.18%	3.52%
SPECTF Heart	45	11	24.44%	79.49%	88.14%	8.65%	67.53%	80.95%	13.42%	79.69%	78.83%	-0.86%	61.90%	80.90%	19.00%	19.00%	10.05%
				Average Improvement		0.63%	Average Improvement		0.76%	Average Improvement		-4.24%	Average Improvement		1.11%		

Naïve Bayes and k-NN algorithm has positive average improvement on PSO reduced datasets but they have negative average results on GA reduced datasets.

Ensemble classifiers achieved the best classification performance on 5 of 8 datasets. In embryonal tumours dataset, the best accuracy was 81.15% which was achieved by bagging decision tree applied to PSO reduced dataset, in madelon dataset the best accuracy was 73.79% achieved by bagging rule induction applied to GA reduced dataset. In the musk dataset, the best accuracy was 99.97% achieved by boosting naïve Bayes. In the spambase dataset, the highest accuracy was 94.52% achieved by boosting rule induction and in SPECTF heart dataset, the best accuracy was 89.22% achieved by bagging naïve Bayes.

Unfortunately, in three datasets (leukemia, dexter and internet_ads) the ensemble classifiers failed to improve the classification performance. In the leukemia dataset, the best accuracy was 98.31% achieved by naïve Bayes as a single classifier. Both bagging and boosting were unable to improve the accuracy on this dataset. In dexter, the best classification performance was achieved by the decision tree with 87.16% and in the internet_ads dataset, the best result was 98.07% achieved by naïve Bayes.

Now we would like to compare the classification performance between single classifier and ensemble classifiers. Table 6.2 shows that actually bagging and boosting did not give significant improvement. The use of bagging on GA-reduced datasets gave 0.64% average improvement and on PSO-reduced datasets bagging had an average improvement of 1.06%. So, the average improvement of bagging when applied to eight different datasets and four base classifiers is only 0.85%.

Boosting had an average improvement of 0.70% when applied to GA-reduced datasets and it had an average improvement of 1.59% when applied to PSO-reduced datasets. Overall the average improvement of boosting when applied to eight different datasets and four base classifiers is 1.14%.

Table 6.2 Summary of Ensemble Classifiers

GA-reduced datasets

Dataset	#attrib	Naïve Bayes					k Nearest Neighbour					Decision Tree					Rule Induction					Highest Improvement
		Single Classifier	Bagging	+/-	Boosting	+/-	Single Classifier	Bagging	+/-	Boosting	+/-	Single Classifier	Bagging	+/-	Boosting	+/-	Single Classifier	Bagging	+/-	Boosting	+/-	
Leukemia	2,237	98.31%	98.31%	0.00%	98.31%	0.00%	78.55%	79.38%	0.83%	85.44%	6.89%	81.16%	85.32%	4.16%	73.47%	-7.69%	82.45%	87.07%	4.62%	91.89%	9.44%	9.44%
Emb. Tumours	619	65.42%	63.71%	-1.71%	75.96%	10.54%	78.82%	75.01%	-3.81%	80.71%	1.89%	58.58%	67.91%	9.33%	71.76%	13.18%	81.00%	80.08%	-0.92%	78.66%	-2.34%	13.18%
Dexter	6,133	73.30%	73.94%	0.64%	-	-	60.04%	61.64%	1.60%	-	-	87.16%	86.05%	-1.11%	62.07%	-25.09%	81.84%	86.41%	4.57%	-	-	4.57%
Internet_ads	489	98.07%	97.97%	-0.10%	97.43%	-0.64%	78.49%	78.43%	-0.06%	78.46%	-0.03%	88.32%	81.18%	-7.14%	97.08%	8.76%	95.02%	95.12%	0.10%	94.77%	-0.25%	8.76%
Madelon	142	59.35%	59.59%	0.24%	60.37%	1.02%	65.23%	65.77%	0.54%	65.84%	0.61%	64.29%	50.00%	-14.29%	-	-	68.15%	73.79%	5.64%	65.41%	-2.74%	5.64%
Musk	66	95.23%	95.23%	0.00%	99.93%	4.70%	96.48%	96.38%	-0.10%	96.40%	-0.08%	91.96%	92.96%	1.00%	92.02%	0.06%	95.93%	96.20%	0.27%	98.77%	2.84%	4.70%
Spambase	29	80.34%	80.39%	0.05%	80.36%	0.02%	90.33%	90.52%	0.19%	90.51%	0.18%	91.69%	92.78%	1.09%	91.72%	0.03%	92.90%	93.29%	0.39%	94.52%	1.62%	1.62%
SPECTF Heart	11	88.50%	89.22%	0.72%	80.70%	-7.80%	74.57%	74.86%	0.29%	75.70%	1.13%	73.24%	80.77%	7.53%	77.40%	4.16%	73.52%	79.54%	6.02%	72.58%	-0.94%	7.53%
Average Improvement				-0.02%		1.12%			-0.07%		1.51%			0.07%		-0.94%			2.59%		1.09%	

PSO-reduced datasets

Dataset	#attrib	Naïve Bayes					k Nearest Neighbour					Decision Tree					Rule Induction					Average Improvement
		Single Classifier	Bagging	+/-	Boosting	+/-	Single Classifier	Bagging	+/-	Boosting	+/-	Single Classifier	Bagging	+/-	Boosting	+/-	Single Classifier	Bagging	+/-	Boosting	+/-	
Leukemia	109	96.55%	98.31%	1.76%	98.31%	1.76%	89.10%	87.90%	-1.20%	89.72%	0.62%	69.12%	69.12%	0.00%	77.77%	8.65%	70.61%	83.41%	12.80%	74.58%	3.97%	12.80%
Emb. Tumours	202	65.40%	65.45%	0.05%	70.32%	4.92%	70.74%	75.08%	4.34%	70.88%	0.14%	76.61%	81.15%	4.54%	70.83%	-5.78%	68.34%	77.27%	8.93%	74.35%	6.01%	8.93%
Dexter	279	73.13%	72.71%	-0.42%	73.19%	0.06%	73.98%	-	-	73.75%	-0.23%	44.56%	44.56%	0.00%	49.06%	4.50%	70.72%	74.42%	3.70%	70.17%	-0.55%	4.50%
Internet_ads	302	97.77%	97.77%	0.00%	97.63%	-0.14%	73.18%	73.32%	0.14%	73.64%	0.46%	96.96%	97.04%	0.08%	96.99%	0.03%	95.12%	95.12%	0.00%	95.12%	0.00%	0.46%
Madelon	5	60.24%	60.05%	-0.19%	64.10%	3.86%	64.25%	63.78%	-0.47%	63.96%	-0.29%	64.29%	50.00%	-14.29%	-	-	63.07%	67.27%	4.20%	64.95%	1.88%	4.20%
Musk	16	99.92%	99.91%	-0.01%	99.97%	0.05%	96.45%	96.65%	0.20%	96.63%	0.18%	91.65%	91.65%	0.00%	91.65%	0.00%	95.29%	95.79%	0.50%	99.28%	3.99%	3.99%
Spambase	27	90.29%	90.08%	-0.21%	90.37%	0.08%	90.91%	91.06%	0.15%	91.33%	0.42%	92.92%	93.29%	0.37%	92.63%	-0.29%	92.25%	93.79%	1.54%	94.23%	1.98%	1.98%
SPECTF Heart	9	85.89%	88.14%	2.25%	88.22%	2.33%	81.42%	80.95%	-0.47%	83.28%	1.86%	77.77%	78.83%	1.06%	85.01%	7.24%	76.82%	80.90%	4.08%	78.00%	1.18%	7.24%
Average Improvement				0.40%		1.61%			0.38%		0.39%			-1.03%		2.05%			4.47%		2.31%	
Average improvement of Bagging for all datasets										0.85%		Average improvement of Boosting for all datasets								1.14%		

6.3 Summary of SVM Parameter optimization

We summarized all experiments from basic classifiers, ensemble classifiers to SVM with parameter optimization in Table 6.3 in the following page. Table 6.3 shows that SVM outperforms other algorithms in 5 of 9 datasets. In leukemia and musk datasets, SVM (either using grid search or evolutionary search as parameter optimization technique) obtained 100% accuracy which can not achieved by other algorithms. In leukemia dataset, SVM only uses 1.53% (109 of 7,130) attributes while in musk, SVM uses 9.52% (16 of 168 attributes).

In the embryonal tumours dataset, we applied SVM into PSO reduced dataset where the number of attribute is only 2.83% from its original (202 of 7130 attributes) and the best results achieved was 85.33%. This result is much better than applying naive Bayes algorithms to the full dataset with 7,130 attributes where the F-measure was only 74.41%. It is also quite a bit better than the use of bagging decision tree classifier on PSO-reduced dataset which had anF-measure of 81.15%.

In the dexter dataset, the SVM with parameter optimization using evolutionary algorithm when applied to PSO-reduced dataset has an F-measure of only 78.88%. This result could not beat the result of decision tree algorithm when applied to dexter full dataset with 20,000 attributes where the F-measure was 86.79%. Both ensemble classifiers bagging and boosting, have relatively poor performance on PSO-reduced dexter dataset which has 279 attributes or only 1.40% of original attributes. But bagging and boosting have better performance when applied to the GA-reduced dataset which has 6,133 attributes (30.67% of original attributes). Bagging rule induction on GA-reduced dataset achieved 86.41% which beats the SVM result (78.88%) but is still below the decision tree result (86.79%). We were curious to know how to improve the performance of SVM while the parameter optimization result (78.88%) was worse than basic algorithms such as decision tree (86.79%). We decided to apply SVM to GA-reduced dataset which has more attributes than PSO-reduced one (6,133 compare to 279 attributes).

We used evolutionary algorithm to optimize the SVM parameters and used four different kernels but only the linear kernel achieved the best result. With the

linear kernel and variable C set to 562.07, the SVM achieved the best results for the dexter dataset with an F-measure of 89.39%.

In this latest experiment, we can see that the classification performance of various algorithms on the PSO-reduced dexter dataset was slightly decreased, but it is very understandable because it used only 1.40% of original attributes. On the other hand, the GA-reduced dexter dataset has many more attributes than the PSO-reduced one but it provided better classification performance.

In the internet_ads dataset, both ensemble classifiers (bagging and boosting) and the SVM could not improve the accuracy when they were applied to the GA-reduced datasets and the PSO-reduced datasets. The highest F-measure was 98.20% achieved by naïve Bayes algorithm when applied to original dataset with 1,559 attributes. The highest score of ensemble classifiers was 97.77% achieved by bagging naïve Bayes which applied to PSO-reduced dataset with 302 attributes or 19.37% of original attributes. SVM was also unable to produce higher score even using an evolutionary algorithm to perform the parameter optimization; it achieved only 97.58% when applied to PSO-reduced dataset. Unfortunately, SVM could not be applied to GA-reduced dataset because it has nominal or categorical attributes. Even though we have done some pre-processing techniques to convert nominal or categorical values to numerical attributes, the classification performance could not be improved. Furthermore, this dataset has missing values, where some attributes have missing values in around 28% of the examples.

As discussed in previous section, in the madelon dataset the SVM could not perform well, all kernels were trapped into premature converge which lead to suboptimal solution with 66.67% of F-measure. In contrast, ensemble classifiers have worked very well especially bagging rule induction which achieves 73.79% of F-measure with only 142 attributes or 28.34% of original dataset.

In the spambase dataset, SVM with the grid search algorithm achieved 94.36% of F-measure. This result was slightly lower than boosting rule induction which had 94.25% of F-measure but much higher than rule induction results which was 93.05%.

Table 6.3 Classification performance of all methods

No	Dataset	Base Classifiers			Ensemble Classifiers			SVM with grid search			SVM with evolutionary search		
		dataset used number of attributes	Algorithm	F-measure	dataset used number of attributes	Algorithm	F-measure	dataset used number of attributes	SVM kernels	F-measure	dataset used number of attributes	SVM kernels	F-measure
1	Leukemia	original dataset 7,130	naive Bayes	98.31%	PSO reduced 109 1.53%	Boosting Naïve Bayes	98.31%	PSO reduced 109 1.53%	linear, RBF, sigmoid, polynomial	100%	PSO reduced 109 1.53%	linear, sigmoid	100%
2	Embryonal Tumours	original dataset 7,130	naive Bayes	74.41%	PSO reduced 202 2.83%	Bagging Decision Tree	81.15%	PSO reduced 202 2.83%	linear	84.95%	PSO reduced 202 2.83%	polynomial	85.33%
3	Dexter	original dataset 20,000	Decision Tree	86.79%	GA reduced 6,133 30.67%	Bagging Rule Induction	86.41%	PSO reduced 279 1.40%	linear	78.68%	GA reduced 6,133 30.67%	linear	89.39%
4	Internet Ads	original dataset 1,559	naive Bayes	98.20%	PSO reduced 302 19.37%	Bagging Naïve Bayes	97.77%	PSO reduced 302 19.37%	linear	97.54%	PSO reduced 302 19.37%	linear	97.58%
5	Madelon	original dataset 501	Rule Induction	73.32%	GA reduced 142 28.34%	Bagging Rule Induction	73.79%	PSO reduced 5 1.00%	RBF	66.07%	PSO reduced 5 1.00%	linear, RBF, sigmoid, polynomial	66.67%
6	Musk	original dataset 168	k-Nearest Neighbour	97.15%	PSO reduced 16 9.52%	Boosting Naïve Bayes	99.97%	PSO reduced 16 9.52%	linear, RBF, sigmoid, polynomial	100%	PSO reduced 16 9.52%	linear, polynomial	100.00%
7	Spambase	original dataset 58	Rule Induction	93.05%	GA reduced 29 50.00%	Boosting Rule Induction	94.52%	PSO reduced 27 46.55%	RBF	94.36%	PSO reduced 27 46.55%	linear, RBF, sigmoid, polynomial	83.42%
8	SPECTF Heart	original dataset 45	Decision Tree	79.69%	GA reduced 11 24.44%	Bagging Naïve Bayes	89.22%	PSO reduced 9 20.00%	sigmoid	91.75%	PSO reduced 9 20.00%	sigmoid	93.34%
9	Intrusion	original dataset 42	Decision Tree	99.48%	GA reduced 16 38.10%	Bagging k-NN	99.74%	PSO reduced 8 19.05%	failed	no results	PSO reduced 8 19.05%	RBF	95.43%

The SVM was applied to PSO-reduced dataset with 27 attributes, the boosting rule induction algorithm was applied to GA-reduced dataset with 29 attributes and rule induction as basic classifier was applied to original dataset with 58 attributes.

In intrusion dataset, the SVM results could not be better than other algorithms because the GA-reduced dataset and the PSO-reduced dataset have some nominal values. From 16 attributes of GA-reduced dataset, there are 3 nominal attributes and from 8 attributes of PSO-reduced dataset, there are 2 nominal attributes. When we applied SVM to both PSO-reduced and GA-reduced datasets, we ignored these nominal attributes and just used the numerical ones. Therefore, the classification performances were not optimal. We have tried to converting these nominal attributes using 'nominal to numerical function', but it could not improve the performance. This is the reason why for intrusion dataset, ensemble classifiers performed better than the SVM.

6.4 Time complexity of classification algorithms

In this section, we compared the time complexity of all algorithms used in our experiments. The time complexity quantifies the amount of time required by an algortihm to run. The most common metric for describing time complexity is big O notation. The O expression is also called Landau's symbol. Apart from time complexity there is space complexity which is the number of memory required by an algorithm.

The theoretical time complexity for learning a naive Bayes classifier is O(nd) where d is the number of attributes and n is the number of samples (instances). The space complexity is O(ndv) where v is the average number of values per attribute (Webb et al., 2005).

kNN classifier is quite different from other classification algorithms. The training phase in kNN simply consists of determining k and preprocessing the dataset. Its time complexity is O(kn) where k is the number of nearest neighbour and n is the number of examples (Zhou and Chen, 2006). If we do not need to preprocess the dataset and we have already predefined the value of k, then kNN does not need any training. In our experiments, we predefined

the value of k to 1, therefore the learning time complexity becomes O(1). The other classification algorithms are more complex than kNN.

Table 6.4 Time complexity of classification algorithms

Algorithms	Time Complexity	Variables
Naive Bayes	O(nd)	d=number of attributes, n=number of samples
k Nearest Neighbour	O(kn)	k = number of nearest points n = number of samples
Decision Tree	$O(nd^2)$	d=number of attributes, n=number of samples
Rule Induction	$O(n\log^2 n)$	n=number of samples
Bagging	O(mns)	m=number of sample of every subset n=number of samples s=number of subsets
Boosting (AdaBoost)	O(bnc)	b=number of base classifiers n=number of samples c=the complexity of base classifier
SVM (without parameter optimization)	$O(n^3)$	n = number of samples

We used an improved decision tree algorithm called C4.5 which has $O(nd^2)$ time complexity where d is number of attributes and n is number of examples (Su and Zhang, 2006). The time complexity of original rule induction algorithm is $O(n^4)$ which is very high. We used an improved rule induction algorithm which is called Ripper that has less time complexity of $O(n \log^2 n)$(Cohen, 1995).

The time complexity of bagging is O(mns) where m is the number of sample of every subset, n is the number of the whole training dataset and s is the number of subsets (Zheng et al., 2011). The AdaBoost algorithm is quite simple but its learning time complexity is high. The time complexity of AdaBoost is O(bnc) where b is the number of base/weak classifiers, n is the number of instances/examples and c is the complexity of base classifier (Sochman and Matas, 2003).

The standard SVM classifier has time complexity of $O(n^3)$ and space complexity of $O(n^2)$ (James et al., 2005). The time complexity is more than quadratic that makes SVM hard to classify a dataset which consists of more than 10,000 instances/examples.

We summarized the learning time of all classification methods in Table 6.5. This table shows that k-NN which has the simplest time complexity, had the fastest learning time. However, kNN average running time was 3.89 seconds which was slightly slower than naive Bayes (1 second), bagging-NB (1.22 seconds) and boosting-NB (2 seconds) but it was much faster than decision tree (56.78 seconds), rule induction (209.33 seconds) and SVM (130.22 seconds) as shown in Table 6.6. In term of classification performance, kNN could not achieve the best accuray (or F-measure) in any of 9 datasets.

Even though naive Bayes average learning time was below kNN (0.25 seconds), it had the fastest average running time compare to all other methods. Naive Bayes only required 1 second or less when applied to 9 PSO-reduced datasets. This algorithm was very fast when applied to datasets which have a large number of attributes such as dexter (20,000 attributes) or leukemia (7,130 attributes) and also when it was applied to datasets with high number of examples/instances such as intrusion dataset (25,192 instances). Based on classification performance, naive Bayes achieved the best result in 1 of 9 datasets. Decision tree's average learning time was 3.47 seconds which was slower than NB and kNN but was still faster than rule induction (6.71 seconds) and SVM (24.70 seconds). Its running time was also slower than NB and kNN but faster than RI and SVM. Decision tree could not achieve the best classification performance in any of 9 datasets as shown in Table 6.5.

The average learning time of rule induction was 6.71 seconds and its average running time was 26.78 seconds. Rule induction's learning time and running time are slower than kNN, NB and DT, but faster than SVM. Like kNN and DT, Rule Induction could not achieve the best results in any of 9 datasets.

Table 6.5 Learning time of classification algorithms

Dataset Name	Number of instances	Number of attributes	Learning Time Comparison, tested on original datasets (seconds)												
			Single Classifier				Bagging with a base classifier				Boosting with a base classifier				SVM
			NB	k-NN	DT	RI	NB	kNN	DT	RI	NB	kNN	DT	RI	
Leukemia	72	7,130	0.09	0	0.33	0.33	0.39	0.02	0.62	1.43	0.36	0.27	1.98	2.52	0.03
Embryonal Tumours	60	7,130	0.08	0	0.39	0.61	0.45	0	2.60	3.93	1.26	0.14	2.57	2.92	0.05
Dexter	600	20,000	1.38	0	8.60	16.11	15.04	0.03	118.38	285.71	143.05	61.03	143.15	262.24	7.05
Internet_ads	3,279	1,559	0.10	0	16.78	16.99	0.71	0.01	122.68	112.49	18.29	133.63	301.44	75.97	9.19
Madelon	2,600	501	0.17	0	2.40	5.99	2.36	0.01	50.65	134.34	9.72	20.89	60.15	31.00	26.51
Musk	6,598	168	0.22	0	0.86	13.43	1.79	0.02	6.65	107.34	8.17	15.23	0.67	63.32	117.39
Spambase	4,601	58	0.05	0	0.42	1.28	0.33	0	4.14	20.50	0.65	61.41	3.95	4.71	0.33
SPECTF Heart	80	45	0.10	0	0.30	0.16	0.17	0.02	0.52	0.44	0.85	0.32	0.75	0.63	0.24
Intrusion	25,192	42	0.10	0	1.12	5.53	1.60	0.06	17.94	81.67	7.58	985.47	11.79	29.26	61.52
Average learning time			0.25	0.00	3.47	6.71	2.54	0.02	36.02	83.09	21.10	142.04	58.49	52.51	24.70
Standard deviation			0.43	0.00	5.65	6.99	4.75	0.02	50.43	92.75	46.11	319.29	102.68	83.36	40.17
Learning time rank			3	1	5	6	4	2	9	12	7	13	11	10	8

Table 6.6 The Running Time Comparison

Dataset Name	Running Time Comparison : tested on PSO-reduced datasets using 10 fold cross validation (seconds)														
	Single Classifier				Bagging with a base classifier				Boosting with a base classifier				SVM with default paramater (linear kernel)	SVM with parameter optimization (linear kernel)	
	NB	k-NN	DT	RI	NB	kNN	DT	RI	NB	kNN	DT	RI		grid search	evol. search
Leukemia	1	1	1	1	1	1	5	2	1	1	4	1	1	3	1
Embryonal Tumours	1	1	2	1	1	1	17	10	1	1	19	6	1	3	2
Dexter	1	1	2	113	1	2	24	886	1	5	8	402	1	21,363	2
Internet_ads	1	1	3	6	3	91	21	56	9	238	2	60	390	2,479	613
Madelon	1	1	1	88	1	1	1	603	1	18	1	348	229	6,750	602
Musk	1	2	2	16	1	20	4	186	1	225	7	1,270	225	680	1,172
Spambase	1	1	14	13	1	14	144	114	1	50	16	231	20	5,850	42
SPECTF Heart	1	1	1	1	1	1	1	1	1	1	1	1	2	313	189
Intrusion	1	26	15	2	1	212	294	17	2	763	38	458	303	-	26
Average running time	1.00	3.89	4.56	26.78	1.22	38.11	56.78	208.33	2.00	144.67	10.67	308.56	130.22	4680.13	294.33
Standard deviation	0.00	8.30	5.68	42.61	0.67	71.38	99.69	318.02	2.65	250.98	12.10	403.97	156.03	7247.07	413.86
Running time rank	1	4	5	7	2	8	9	12	3	11	6	14	10	15	13

The time complexity of bagging and boosting were strongly correlated with the selection of base classifier. The average learning time of Bagging-kNN (0.02 seconds) and bagging-NB (2.54 seconds) were relatively fast while bagging-DT (36.02 seconds) and bagging-RI (83.09 seconds) were quite slow. Bagging has achieved the best classification performance in 2 of 9 datasets. Bagging-RI achieved the best result on madelon dataset with F-measure of 73.79% and bagging-kNN produced the best F-measure of 99.74% on intrusion dataset.

Compare to bagging, boosting was much slower. The average learning time of boosting-NB was 2 seconds, boosting-kNN was 144.67 seconds, boosting-DT was 10.67 seconds and boosting-RI was 308.56 seconds. Boosting had the best classification result on only 1 of 9 dataset where boosting-RI achieved F-measure of 94.52% on spambase dataset.

SVM which has time complexity of $O(n^3)$, was actually not the slowest classifier. Based on average learning time, SVM was number 8 of 13 classifiers with kNN was the fastest (0 second) and boosting-kNN was the slowest (142.04 seconds). Based on average running time, SVM (with default parameters) was number 10 of 15 classifiers with naïve Bayes was the fastest (1 second) and SVM with parameter optimization using grid search was the slowest (4680.13 seconds).

However, SVM with default parameters could not achieve the best classification performance in any of 9 datasets. The use of parameter optimization both using grid search and evolutionary search have been successfully improved the classification performance. Even though SVM parameter optimization (both using grid search and evolutionary algorithm) required high computation time as shown in Table 6.6, they have successfully improved the classification performance and achieved the best results in 5 of 9 datasets.

7. Conclusions and Future Works

7.1 Conclusions

The main questions that we attempted to answer with our research work are: what are the best machine learning techniques to handle high dimensional datasets and how to improve their classification performance. Our experiments on nine high dimensional datasets show that there is no single classifier that always achieves the best accuracy in all domains or all applications, but our results show that SVM with parameter optimization is very powerful classifier and outperforms other algorithms in 5 of 9 datasets. However, SVM is only good in handling datasets with numerical attributes. This algorithm does not perform well on datasets which have nominal attributes. In this case, ensemble classifiers such as bagging and boosting can be used as an alternative. Their performances are slightly better than basic classifier such as naïve Bayes, k-Nearest Neighbour, decision tree and rule induction.

Dimensionality reduction algorithms both GA and PSO, significantly reduces the number of features or attributes needed as well as greatly reduce the computational cost. Furthermore, these algorithms do not severely reduce the classification accuracy and in some cases they can improve the accuracy as well. We use these two algorithms to select the most important features in nine high dimensional datasets. It takes around 34 minutes and 10 seconds to finish the feature selection process on 9 datasets using GA and 48 minutes and 53 seconds using PSO.

The total running time of four basic classifiers (NB, kNN, DT and RI) on 9 original datasets is 68,169 seconds while the total running time of the same classifiers on GA-reduced datasets is 3,799 seconds which means 17.9 times faster. The total running time of the same four classifiers on PSO-reduced datasets is only 326 seconds which is more than 200 times faster than the total running time on original datasets.

In terms of dimensionality reduction, PSO is much better than GA. PSO has successfully reduced the number of attributes of 9 datasets to 12.78% of the original attributes on average while GA is only 30.52% on average. In terms of classification performance, GA is better than PSO. GA-reduced datasets have better classification performance than their original ones on 5 of 9 datasets while PSO is better only in 3 of 9 datasets.

We applied two ensemble classifiers called bagging and boosting. Our experiment shows that actually bagging and boosting did not give significant improvement for all basic classifiers. In some datasets, both bagging and boosting were able to improve the accuracy more than 10%. The average improvement of bagging when applied to nine different datasets is only 0.85% while boosting improvement is 1.14%. Ensemble classifiers (both bagging and boosting) outperforms single/base classifier in 7 of 8 PSO-reduced datasets and 4 of 8 GA-reduced datasets.

SVM has been proven to perform much better when dealing with high dimensional datasets and continuous/numerical features. The performance of SVM highly depends on the slack variable penalty weight (C) and γ (gamma). Finding the proper C and γ value is a kind of searching for the best trade-off between allowing misclassification errors and generalizing the model.

The linear kernel gives better results if the number of features is very large where the use of nonlinear mapping (RBF kernel, polynomial kernel and sigmoid kernel) does not improve the performance. The RBF does not perform well when the number of features is very large. If the number of feature is large, it is not possible to map data to a higher dimensional space. Therefore, the nonlinearity can not improve the classification performance, in this case linear kernel is the best solution.

Although SVM work well with default value, the performance of SVM can be improved significantly using parameter optimization. One of the biggest problems of SVM parameter optimization is there is no exact ranges of C and γ values. We believe that the wider the parameter range is, the more possibilities the grid search method finds the best combination parameter.

Our experiment shows that the grid search always finds near optimal parameter combination within given ranges. SVM parameter optimization using

grid search is very powerful and it is able to improve the accuracy significantly. In leukemia and musk datasets, this technique achieved 100% accuracy for all four kernels (linear, RBF, polynomial and sigmoid kernel), these results are amazing especially it was applied to PSO-reduced datasets which have much less number of attributes (only 1.53% of original attributes for leukemia dataset and only 9.52% of original attributes for musk dataset). These results are much better than the performance of four basic classifiers (naïve Bayes, k-nearest neighbour, decision tree and rule induction) which are applied to the original datasets and also slightly better than the performance of ensemble classifiers which are applied to GA-reduced and PSO-reduced datasets.

However, grid search has several disadvantages, it is extremely slow and furthermore it may lead to very long execution time. For example, grid search has been failed in finding optimal SVM parameters for intrusion dataset which have a large number of instances. The process was forced to stop after 2 weeks running. Therefore, grid search is very reliable only in low dimensional dataset with few parameters. To solve this problem, we use Evolutionary Algorithm (EA) which is very useful to be implemented when the best ranges and dependencies of various SVM parameters is not known at all. EA has proven to be more stable than grid search. When applied to 9 datasets, EA has an average running time of 294 seconds while grid search is around 4,680 seconds (it does not include intrusion dataset which was failed). It means, SVM parameter optimization using EA is more than 15.9 times faster than using grid search.

7.2 Future Work

The parameter optimization using Evolutionary Algorithm (EA) has shown satisfactory results, it is more stable and much faster than the grid search but it has a drawback. In some datasets, SVM parameter optimization using EA did not achieve good results because the program executions were trapped in a local minimum and then terminated very early. This problem is called premature convergence, the condition where a population for optimization process converged too early which results a sub-optimal solution. In the future work, we would like to investigate some methods to avoid a premature converge problem in EA optimization.

Bibliography

Aydin, I., Karakose, M., Akin, E., 2011. A multi-objective artificial immune algorithm for parameter optimization in support vector machine. Applied Soft Computing 11, 120–129.

Barros, R.C., Basgalupp, M.P., De Carvalho, A.C.P.L.F., Freitas, A.A., 2012. A Survey of Evolutionary Algorithms for Decision-Tree Induction. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42, 291–312.

Bellman, R., 1957. Dynamic Programming. Princeton University Press.

Braun, A.C., Weidner, U., Hinz, S., 2012. Classification in High-Dimensional Feature Spaces #x2014;Assessment Using SVM, IVM and RVM With Focus on Simulated EnMAP Data. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 5, 436 –443.

Breiman, L., 1996. Bagging predictors. Machine Learning Journal 24, 123–140.

Chang, C.-C., Lin, C.-J., 2011. LIBSVM: A library for support vector machines. ACM Trans. Intell. System Technology 2, 27:1–27:27.

Chen, Y., Li, Y., Cheng, X.-Q., Guo, L., 2006. Survey and taxonomy of feature selection algorithms in intrusion detection system, in: Proceedings of the Second SKLOIS Conference on Information Security and Cryptology, Inscrypt'06. Springer-Verlag, Berlin, Heidelberg, pp. 153–167.

Clarke, R., Ressom, H.W., Wang, A., Xuan, J., Liu, M.C., Gehan, E.A., Wang, Y., 2008. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. Nat Rev Cancer 8, 37–49. doi:10.1038/nrc2294

Cohen, W.W., 1995. Fast Effective Rule Induction, in: In Proceedings of the Twelfth International Conference on Machine Learning. Morgan Kaufmann, pp. 115–123.

Cortes, C., Vapnik, V., 1995. Support-Vector Networks. Machine Learning 20, 273–297.

Dandpat, S.K., Meher, S., 2013. Performance improvement for face recognition using PCA and two-dimensional PCA, in: 2013 International Conference on Computer Communication and Informatics (ICCCI). Presented at the 2013 International Conference on Computer Communication and Informatics (ICCCI), pp. 1–5.

Davis, J., Goadrich, M., 2006. The relationship between Precision-Recall and ROC curves, in: Proceedings of the 23rd International Conference on Machine Learning, ICML '06. ACM, New York, NY, USA, pp. 233–240.

De Miranda, P.B.C., Prudencio, R.B.C., Carvalho, A.C.P.L.F., Soares, C., 2012. Combining a multi-objective optimization approach with meta-learning for SVM parameter selection, in: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Presented at the 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2909–2914.

Dietterich, T.G., 1997. Machine Learning Research: Four Current Directions. AI Magazine Vol 18, pp. 97-136.

Dong, Y.-S., Han, K.-S., 2004. A comparison of several ensemble methods for text categorization, in: 2004 IEEE International Conference on Services Computing, 2004. (SCC 2004). Proceedings. Presented at the 2004 IEEE International Conference on Services Computing, 2004. (SCC 2004). Proceedings, pp. 419–422.

Eiben, A.E., Smith, J.E., 2003. Introduction to Evolutionary Computing. SpringerVerlag.

Fan, J., Fan, Y., 2008. High dimensional classification using features annealed independence rules. Ann. Statist.

Fan-Zi, Z., Zheng-Ding, Q., 2004. A survey of classification learning algorithm, in: 2004 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. Presented at the 2004 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04, pp. 1500–1504 vol.2.

Fodor, I., 2002. A Survey of Dimension Reduction Techniques.

Freund, Y., Schapire, R.E., 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences 55, 119–139.

Friedrichs, F., Igel, C., 2005. Evolutionary tuning of multiple SVM parameters. Neurocomputing 64, 107–117.

Gaber, M.M., Bader-El-Den, M.B., 2012. Optimisation of Ensemble Classifiers using Genetic Algorithm, in: Graña, M., Toro, C., Posada, J., Howlett, R.J., Jain, L.C. (Eds.), Advances in Knowledge-Based and Intelligent Information and Engineering Systems - 16th Annual KES Conference, San Sebastian, Spain, 10-12 September 2012, Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 39–48.

Gaber, M., Zaslavsky, A., Krishnaswamy, S., 2007. A Survey of Classification Methods in Data Streams, in: Aggarwal, C. (Ed.), Data Streams, Advances in Database Systems. Springer US, pp. 39–59.

Gaspar, P., Carbonell, J., Oliveira, J.L., 2012. On the parameter optimization of Support Vector Machines for binary classification. J Integr Bioinform 9, 201.

Geurts, P., Irrthum, A., Wehenkel, L., 2009. Supervised learning with decision tree-based methods in computational and systems biology. Mol Biosyst 5, 1593–1605.

Graczyk, M., Lasota, T., Trawiski, B., Trawiski, K., 2010. Comparison of Bagging, Boosting and Stacking Ensembles Applied to Real Estate Appraisal, in: Nguyen, N., Le, M., Swiatek, J. (Eds.), Intelligent Information and Database Systems, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 340–350.

Guyon, I., Hur, A.B., Gunn, S., Dror, G., 2004. Result analysis of the NIPS 2003 feature selection challenge, in: Advances in Neural Information Processing Systems 17. MIT Press, pp. 545–552.

Haddadi, F., Khanchi, S., Shetabi, M., Derhami, V., 2010. Intrusion Detection and Attack Classification Using Feed-Forward Neural Network, in: 2010

Second International Conference on Computer and Network Technology (ICCNT). Presented at the 2010 Second International Conference on Computer and Network Technology (ICCNT), pp. 262–266.

Hall, M.A., 1999. Correlation-based Feature Selection for Machine Learning.

Hall, M.A., Holmes, G., 2003. Benchmarking attribute selection techniques for discrete class data mining. IEEE Transactions on Knowledge and Data Engineering 15, 1437–1447.

Holder, L.B., Russell, I., Markov, Z., Pipe, A.G., Carse, B., 2005. CURRENT AND FUTURE TRENDS IN FEATURE SELECTION AND EXTRACTION FOR CLASSIFICATION PROBLEMS. International Journal of Pattern Recognition and Artificial Intelligence 19, 133–142.

Howley, T., Madden, M.G., 2005. The Genetic Kernel Support Vector Machine: Description and Evaluation. Artificial Intelligence Review 24, 379–395.

Hric, M., Chmulik, M., Jarina, R., 2011. Model parameters selection for SVM classification using Particle Swarm Optimization, in: Radioelektronika (RADIOELEKTRONIKA), 2011 21st International Conference. Presented at the Radioelektronika (RADIOELEKTRONIKA), 2011 21st International Conference, pp. 1–4.

Huang, C.-L., Dun, J.-F., 2008. A distributed PSO–SVM hybrid system with feature selection and parameter optimization. Applied Soft Computing 8, 1381–1391.

Hussain, M., Wajid, S.K., Elzaart, A., Berbar, M., 2011. A Comparison of SVM Kernel Functions for Breast Cancer Detection, in: 2011 Eighth International Conference on Computer Graphics, Imaging and Visualization (CGIV). Presented at the 2011 Eighth International Conference on Computer Graphics, Imaging and Visualization (CGIV), pp. 145 –150.

Islam, M.J., Wu, Q.M.J., Ahmadi, M., Sid-Ahmed, M.A., 2007. Investigating the Performance of Naive- Bayes Classifiers and K- Nearest Neighbor Classifiers, in: International Conference on Convergence Information Technology, 2007. Presented at the International Conference on Convergence Information Technology, 2007, pp. 1541–1546.

James, I.T., Kwok, J.T., Bay, C.W., 2005. Very Large SVM Training using Core Vector Machines, in: In Proc. 10th Int. Workshop Artif. Intell. Stat. pp. 349–356.

Jwo, D.-J., Chang, S.-C., 2009. Particle swarm optimization for GPS navigation Kalman filter adaptation. Aircraft Engineering and Aerospace Technology 81, 343–352.

Kohavi, R., Quinlan, R., 1999. Decision Tree Discovery, in: IN HANDBOOK OF DATA MINING AND KNOWLEDGE DISCOVERY. University Press, pp. 267–276.

Korürek, M., Doan, B., 2010. ECG beat classification using particle swarm optimization and radial basis function neural network. Expert Systems with Applications 37, 7563–7569.

Kotsiantis, S.B., 2007. Supervised Machine Learning: A Review of Classification Techniques, in: Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word

AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies. IOS Press, Amsterdam, The Netherlands, The Netherlands, pp. 3–24.

Krishna, K., Murty, M.N., 1999. Genetic K-means algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 29, 433–439.

Lee, K.C., Cho, H., 2010. Performance of Ensemble Classifier for Location Prediction Task: Emphasis on Markov Blanket Perspective. . International Journal of u- and e- Service, Science and Technology 3.

Lee, W., Stolfo, S.J., 1998. Data mining approaches for intrusion detection, in: Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7, SSYM'98. USENIX Association, Berkeley, CA, USA, pp. 6–6.

Li, C.-H., Ho, H.-H., Liu, Y.-L., Lin, C.-T., Kuo, B.-C., Taur, J.-S., 2012. An Automatic Method for Selecting the Parameter of the Normalized Kernel Function to Support Vector Machines. J. Inf. Sci. Eng. 28, 1–15.

Lin, S.-W., Ying, K.-C., Chen, S.-C., Lee, Z.-J., 2008. Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Systems with Applications 35, 1817–1824.

Liu, Y., Wang, G., Chen, H., Dong, H., Zhu, X., Wang, S., 2006. An Improved Particle Swarm Optimization for Feature Selection. Engineering 8, 924–928.

Malhotra, R., Singh, N., Singh, Y., 2011. Genetic Algorithms: Concepts, Design for Optimization of Process Controllers. Computer and Information Science 4, p39.

Mierswa, I., 2006. Evolutionary Learning with Kernels: A Generic Solution for Large Margin Problems. Proceeding of the 8th annual conference on Genetic and Evolutionary Computation, pp. 1553-1560

Miller, A.J., 2002. Subset selection in regression. Chapman & Hall/CRC, Boca Raton.

Mitchell, M., 1998. An introduction to genetic algorithms. MIT Press, Cambridge, Mass.

Moraglio, A., Chio, C.D., Togelius, J., Poli, R., 2008. Geometric Particle Swarm Optimization.

Mukkamala, S., Sung, A.H., Abraham, A., 2005. Intrusion detection using an ensemble of intelligent paradigms. J. Network and Computer Applications 28, 167–182.

Oh, I.-S., Lee, J.-S., Moon, B.-R., 2004. Hybrid genetic algorithms for feature selection. Pattern Analysis and Machine Intelligence, IEEE Transactions on 26, 1424 –1437.

Otero, F.E.B., Freitas, A.A., Johnson, C.G., 2012. Inducing decision trees with an ant colony optimization algorithm. Applied Soft Computing 12, 3615–3626.

Pardo, M., Sberveglieri, G., 2005. Classification of electronic nose data with support vector machines. Sensors and Actuators B: Chemical 107, 730–737.

Platt, J.C., 1999. Advances in kernel methods, in: Schölkopf, B., Burges, C.J.C., Smola, A.J. (Eds.), . MIT Press, Cambridge, MA, USA, pp. 185–208.

Polikar, R., 2006. Ensemble based systems in decision making. IEEE Circuits and Systems Magazine 6, 21–45.

Quinlan, J.R., 1993. C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rossi, A.L.D., de Carvalho, A.C.P., 2008. Bio-inspired Optimization Techniques for SVM Parameter Tuning, in: 10th Brazilian Symposium on Neural Networks, 2008. SBRN '08. Presented at the 10th Brazilian Symposium on Neural Networks, 2008. SBRN '08, pp. 57–62.

Roy, K., Bhattacharya, P., 2008. Improving Features Subset Selection Using Genetic Algorithms for Iris Recognition, in: Prevost, L., Marinai, S., Schwenker, F. (Eds.), Artificial Neural Networks in Pattern Recognition, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 292–304.

Sánchez A, V.D., 2003. Advanced support vector machines and kernel methods. Neurocomputing 55, 5–20.

Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S., 1997. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods.

Schuh, M.A., Angryk, R.A., Sheppard, J., 2012. Evolving Kernel Functions with Particle Swarms and Genetic Programming, in: Youngblood, G.M., McCarthy, P.M. (Eds.), Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference, 2012. AAAI Press, Marco Island, Florida, pp. 80–85.

Skellam, J.G., 1952. Studies in Statistical Ecology Spatial Pattern. Biometrika 39, 346–362.

Sochman, J., Matas, J., 2003. AdaBoost and Face Detection (Version 1.0).

Subasi, A., 2013. Classification of EMG signals using PSO optimized SVM for diagnosis of neuromuscular disorders. Computers in Biology and Medicine.

Sudheer, C., Maheswaran, R., Panigrahi, B.K., Mathur, S., 2013. A hybrid SVM-PSO model for forecasting monthly streamflow. Neural Computing and Applications.

Su, J., Zhang, H., 2006. A Fast Decision Tree Learning Algorithm, in: Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI'06. AAAI Press, Boston, Massachusetts, pp. 500–505.

Syarif, I., Prugel-Bennett, A., Wills, G., 2012a. Data mining approaches for network intrusion detection: from dimensionality reduction to misuse and anomaly detection. Journal of Information Technology Review 3, pp. 70–83.

Syarif, I., Prugel-Bennett, A., Wills, G.B., 2012b. Unsupervised clustering approach for network anomaly detection. Fourth International Conference on Networked Digital Technologies (NDT) 2012. 11 pages.

Syarif, I., Zaluska, E., Prugel-Bennett, A., Wills, G., 2012c. Application of bagging, boosting and stacking to intrusion detection. Presented at the MLDM 2012: 8th International Conference on Machine Learning and Data Mining. 10 pages.

Tjiong, A.S.J., Monteiro, S.T., 2011. Feature selection with PSO and kernel methods for hyperspectral classification, in: 2011 IEEE Congress on Evolutionary Computation (CEC). Presented at the 2011 IEEE Congress on Evolutionary Computation (CEC), pp. 1762 –1769.

Tsai, F.S., Chan, K.-L., 2007. Dimensionality reduction techniques for data exploration, in: 2007 6th International Conference on Information, Communications Signal Processing. Presented at the 2007 6th International Conference on Information, Communications Signal Processing, pp. 1–5.

Van den Bosch, A., 2000. Using induced rules as complex features in memory-based language learning, in: Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7, ConLL '00. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 73–78.

Verleysen, M., 2003. Learning High-Dimensional Data, in: Limitations and Future Trends in Neural Computation 186. IOS Press, pp. 141–162.

Viswanath, P., Sarma, T.H., 2011. An improvement to k-nearest neighbor classifier, in: 2011 IEEE Recent Advances in Intelligent Computational Systems (RAICS). Presented at the 2011 IEEE Recent Advances in Intelligent Computational Systems (RAICS), pp. 227–231.

Webb, G.I., Boughton, J.R., Wang, Z., 2005. Not So Naive Bayes: Aggregating One-Dependence Estimators. Mach Learn 58, 5–24.

Williams, N., Z, S., Armitage, G., 2006. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. Computer Communication Review 30.

Witten, I.H., Frank, E., 2005. Data mining: practical machine learning tools and techniques. Morgan Kaufman, Amsterdam; Boston, MA.

Wu, X., Kumar, V. (Eds.), 2009. The Top Ten Algorithms in Data Mining, 1 edition. ed. Chapman and Hall/CRC, Boca Raton.

Zheng, Y., Sun, C., Li, J., Yang, Q., Chen, W., 2011. Entropy-Based Bagging for Fault Prediction of Transformers Using Oil-Dissolved Gas Data. Energies 4, 1138–1147.

Zhou, C.Y., Chen, Y.Q., 2006. Improving nearest neighbor classification with cam weighted distance. Pattern Recognition, Graph-based Representations 39, 635–645.

Zhou, H.-G., Chun-De, Y., 2006. Using Immune Algorithm to Optimize Anomaly Detection Based on SVM, in: 2006 International Conference on Machine Learning and Cybernetics. Presented at the 2006 International Conference on Machine Learning and Cybernetics, pp. 4257–4261.

Zhou, Z.-H., 2009. Ensemble Learning, in: Li, S.Z., Jain, A. (Eds.), Encyclopedia of Biometrics. Springer US, pp. 270–273.