AN ADAPTIVE MULTI-LEVEL SEQUENTIAL FLOATING FEATURE SELECTION

Knitchepon Chotchantarakun

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science and Information Systems)
School of Applied Statistics
National Institute of Development Administration
2020

AN ADAPTIVE MULTI-LEVEL SEQUENTIAL FLOATING FEATURE SELECTION

Knitchepon Chotchantarakun

School of Applied Statistics

.... Major Advisor
(Associate Professor Ohm Sornil, Ph.D.)

The Examining Committee Approved This Dissertation Submitted in Partial Fulfillment of Requirements for the Degree of Doctor of Philosophy (Computer Science and Information Systems).

.... Committee Chairperson
(Associate Professor Surapong Auwatanamongkol, Ph.D.)

.... Committee
(Associate Professor Ohm Sornil, Ph.D.)

.... Committee
(Assistant Professor Tanasai Sucontphunt, Ph.D.)

.... Committee
(Assistant Professor Orawan Chaowalit, Ph.D.)

.... Dean
(Assistant Professor Pramote Luenam, Ph.D.)

_____/_____/_____

ABSTRACT

Title of Dissertation	AN ADAPTIVE MULTI-LEVEL SEQUENTIAL FLOATING FEATURE SELECTION
Author	Knitchepon Chotchantarakun
Degree	Doctor of Philosophy (Computer Science and Information Systems)
Year	2020

Dealing with a large amount of available data becomes a major challenge in data mining and machine learning. Feature selection is a significant preprocessing step for selecting the most informative features by removing irrelevant and redundant features, especially for large datasets. These selected features play an important role in information searching and enhancing the performance of machine learning models such as classification and prediction. There have been several strategies proposed in the past few decades.

In this dissertation, we propose a new technique called An Adaptive Multi-level Sequential Floating Feature Selection (AMFFS). AMFFS consists of three proposed algorithms, which are One Level Forward Inclusion (OLFI), One-level Forward Multi-level Backward Selection (OFMB) and Multi-level Forward Inclusion (MLFI). Our proposed methods are considered to be deterministic algorithms related to sequential feature selection under the supervised learning model.

The OFMB algorithm consists of two parts. The first part aims to create preliminarily selected subsets. These subsets have similar performance to the Improved Forward Floating Selection (IFFS). This part contains the same procedure as the OLFI algorithm. The second part provides an improvement on the previous result using the multi-level backward searching technique. The idea is to apply an improved step during the feature addition and the adaptive search method on the backtracking step. However, we need to limit the level of backwards-searching to maintain lower execution time by introducing an adaptive variable called the generalization limit.

The MLFI algorithm also consists of two parts. The first part aims to search for the maximum classification accuracy by applying the multi-level forward-searching technique. The second part provides an improvement on the previous result by replacing the week feature technique. The idea is to apply an adaptive multi-level forward search method with the replacement step during the feature addition without any backtracking search. Similar to OFMB, we also need to limit the level of forward-searching by the generalization limit.

In the experiments, we applied KNN, Naive Bayes, and Decision Tree for our criterion functions. We tested our algorithms on fourteen standard UCI datasets and compared their classification accuracy with other popular methods. Our proposed algorithms showed better results than the other sequential feature selection techniques for the majority of the tested datasets. The OFMB and MLFI algorithms spend more computational time than the other methods due to the complexity of the program.

ACKNOWLEDGEMENTS

I would like to express my greatest appreciation to my advisor, Associate Professor Dr Ohm Sornil, for all the very good advice and his guidance in finding a good research topic. I have learnt many techniques for conducting good research and how to write papers for submission. Without his generous support, this dissertation would not be accomplished. I would also like to thank my committee chairperson, Associate Professor Dr Surapong Auwantanamongkol for the valuable advice on various aspects of the dissertation. Furthermore, many thanks to my dissertation committee, Assistant Professor Dr Tanasai Sucontphunt and Assistant Professor Dr Orawan Chaowalit for their advice and helpful suggestions. Last but not least, I would like to express my special thanks to my family for all the help and continuous support throughout the completion of my doctoral degree.

Knitchepon Chotchantarakun
July 2021

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

SYMBOLS AND ABBREVIATIONS

Symbols

J	Criterion function
Y	Original feature set
X	Candidate set
k	Number of features in X
d	Predefined number of selected feature
D	Total number of original features
x^+	Included feature
x^-	Excluded feature
r	Generalization limit
r_{max}	Maximum value of r
s	Generalization level

Abbreviations	**Equivalence**
KNN	K-Nearest Neighbour
NB	Naïve Bayes
DT	Decision Tree
CV	Cross Validation
SFS	Sequential Forward Search
SBS	Sequential Backward Search
SFFS	Sequential Forward Floating Search

ASFFS	Adaptive Sequential Forward Floating Search
IFFS	Improved Forward Floating Search
SDFFS	Sequential Deep Floating Forward Search

CHAPTER 1

INTRODUCTION

In the past few decades, the available data is growing extremely large due to modern technology and internet applications. Dealing with a large amount of data is the major challenge these days. In the data analysis task, a large amount of data can be a high dimensional dataset which directly affects performance because some irrelevant and redundant features also make some contribution to the analysis. To overcome the problem, those irrelevant and redundant features should be eliminated which lead to more effective dimensions. This data preprocessing step is called feature selection. Generally, the goal of feature selection is to determine the best subsets of features for conducting statistical analysis or building a machine learning model (Huang, 2015). Feature selection assists in selecting the minimum features from the whole dataset. These features are useful for finding accurate data models. To ensure the optimal feature subset, a feature selection method has to evaluate a total of $2^n - 1$ subsets, where n is the total number of features in the dataset. Even though an exhaustive search for optimal feature subset results in an optimal solution, but it is not practical especially for a moderately large n. This type of problem is said to be an NP-hard problem, as a result, many search strategies have been proposed in the literature for suboptimal solutions.

Feature selection is interesting research areas that deal with data mining and machine learning since it provides more accuracy, faster computational time, and is also cost-effective. The accuracy of the classifier depends not only on the classification algorithm but also on the feature selection method used. Irrelevant features may affect the classifier and lead to incorrect results, therefore feature selection is necessary in order to improve the efficiency and accuracy of the classifier.

Feature selection offers advantages such as reducing storage requirements, avoiding overfitting, facilitating data visualization, speeding up the execution of mining algorithms and reducing the training times. For example, a data set named "DOROTHEA" use for drug discovery (Sutha & Tamilselvi, 2015) contains 1,950 instances and 100,000 features. Many of these relevant features are useful for information discovery, but they also contain a lot of irrelevant features in the dataset. This is where the feature selection steps in to improve the computational efficiency.

1.1 Data Representation

In machine learning and pattern recognition, a feature is referred to as an individual measurable property or characteristic of a phenomenon being observed. To improve the learning performance we need to select an informative, discriminating and independent feature using an effective algorithm. Features are usually numeric, but structural features also occur sometimes. These features are represented as an *n*-dimensional vector of numeric features that indicate some object. Many machine learning algorithms require a numerical representation of objects for further processing and statistical analysis. Some types of data such as images can be represented by the pixel, while text features might be represented by the frequencies of occurrence of the textual terms. These feature vectors are similar to the variables used in statistical analysis such as linear regression. They are normally combined with weights using dot product as a preliminary result for making a prediction. To reduce the dimensionality of the feature space, various dimensionality reduction techniques are applied.

1.2 Feature Selection

Feature selection is one of the most important preprocessing techniques in data mining. This technique uses to eliminate the irrelevant and redundant features from the dataset. The goal of feature selection for classification tasks is to maximize classification accuracy. Therefore, the computational time of the classifier to process data will decrease whereas the classification accuracy will increase since the irrelevant features are removed. Feature selection is also known as attributes selection or variable selection (Beniwal & Arora, 2012).

1.2.1 Feature Selection Process

The process of finding the feature subset consists of four basic steps (Liu & Yu, 2005):

1) Subset generation
2) Subset evaluation
3) Stopping criterion
4) Validation of the results

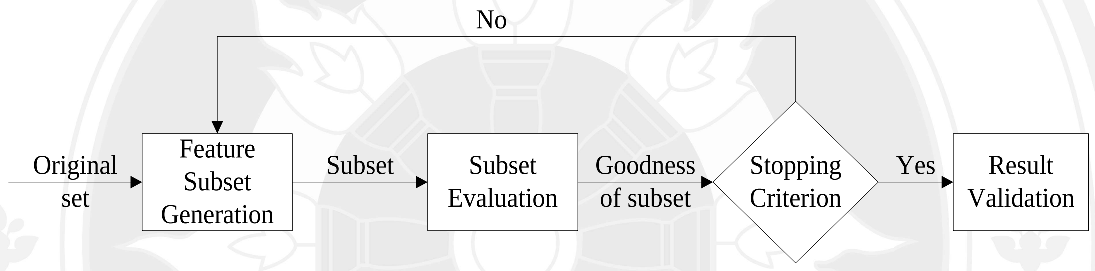


Figure 1.1 Feature Selection Process

From figure 1.1, begins with an original dataset by inserting features into the process. Feature subset generation produces candidate feature subsets for evaluation based on searching strategies. These searching strategies are used to preselect subsets for further evaluation. Subset evaluation is aimed to evaluate the subset generated from the previous procedure. At this step, the classifiers such as K-Nearest Neighbor, Naïve Bayes or Random Forest are applied to calculate the classification accuracy. Continue selecting or removing features that yield the highest accuracy until the process reaches the stopping criterion. Finally, the result validation is the selection of the best results for all subset sizes.

Stopping criteria could be any of the following:

1) Selected subset with number of features equal to the predefined value
2) New subset of the feature does not yield a better result
3) Number of iteration is reached

Cai, Chang and He (Cai, Zhang, & He, 2010) stated that "Various data mining and machine learning tasks, such as classification and clustering, that are analytically or computationally manageable in low dimensional spaces may become completely intractable in spaces of several hundred or thousand dimensions". High dimensional space leads to low performance in machine learning algorithms particularly when the samples are small. This difficulty is known as the curse of dimensionality. Feature selection or dimensionality reduction plays an important role to solve this kind of problem.

1.2.2 Search Strategies

There are four usual search strategies (Jovic, Brkic, & Bogunovic, 2015):

1) Forward selection
2) Backward elimination
3) Bidirectional selection
4) Heuristic feature subset selection

Forward selection starts with an empty set and then adding one or more features to the set. Oppositely, backward elimination is removing one or more features from the set that start with all the features. Bidirectional selection starts from both sides. Heuristic selection uses a heuristic search such as a genetic algorithm to explore the feature subset.

1.2.3 Types of Feature Selection Algorithm

Feature selection algorithms can be classified in many different ways. The most common one can be categorized into three types (Pavya & B.Srinivasan, 2017):

1) Filter approach
2) Wrapper approach
3) Embedded approach

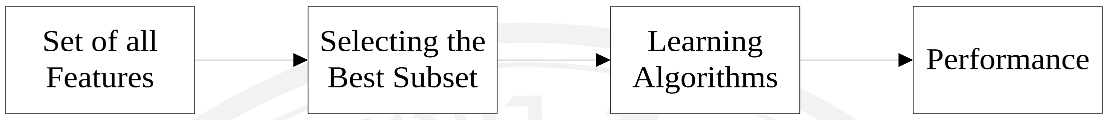


Figure 1.2 Filter Approach

The filter approach or filter method (Figure 1.2) uses an independent criterion function to select the feature without depending upon the type of classifier used which leads to the simplicity of the method, whereas the interactions with classifier and feature dependencies are ignored. The filter method ranks each individual feature according to the measurement such as information, distance, or similarity. It only considers the association between the feature and the class label. The nature of this method results in a drawback that each feature is considered separately.

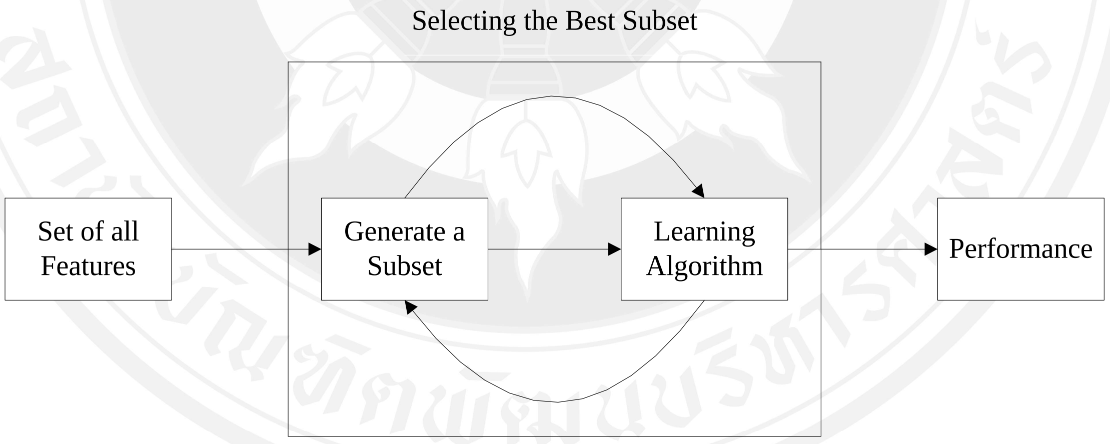


Figure 1.3 Wrapper Approach

The wrapper approach or wrapper method (Figure 1.3) uses the result of the classifier to determine the goodness of the given feature, therefore the selected features are dependent on the classification algorithm. This method removes the disadvantage of the filter approach by the consideration of feature dependency whereas it is more time consuming than the filter approach. The quality of the feature is directly related to the performance of the classifier.

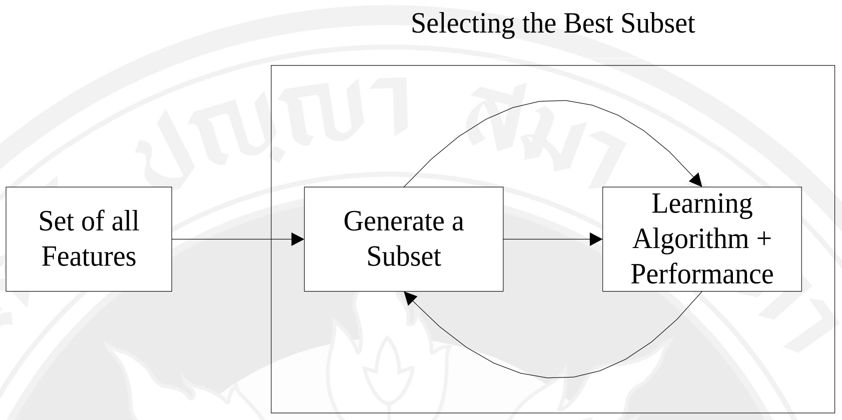


Figure 1.4 Embedded Approach

The embedded approach or embedded method (Figure 1.4) searches for an optimal feature subset during the model training that is built into the classifier construction. It returns both the learned model and selected features simultaneously. The benefit of this method is that it takes less computational time than the wrapper approach. This method is also called the hybrid model. It incorporates a learning algorithm and is optimized for higher accuracy. The embedded approach utilizes a filter-based technique to select highly representative features and then apply a wrapper-based technique to add candidate features. The candidate subsets are evaluated for selecting the best ones. It does not only reduce the dimensionality of the dataset but also decreases the computational time and improves the performance. Somol, Novovicova, and Pudil (Somol, Novovicova, & Pudil, 2006) proposed a flexible hybrid sequential forward floating selection (hSFFS) by employing an evaluation function to filter some features and using a wrapper criterion to identify the optimal feature subset. The main benefit of this method is the ability to trade off the resulting quality with the computational cost in order to enable the wrapper-based selection in high dimensional datasets. Their experimental results show promising classification accuracy.

1.3 Feature Learning Methods

Feature selection methods can also be divided into a supervised, unsupervised and semi-supervised model. Figure 1.5 shows a framework for feature selection.

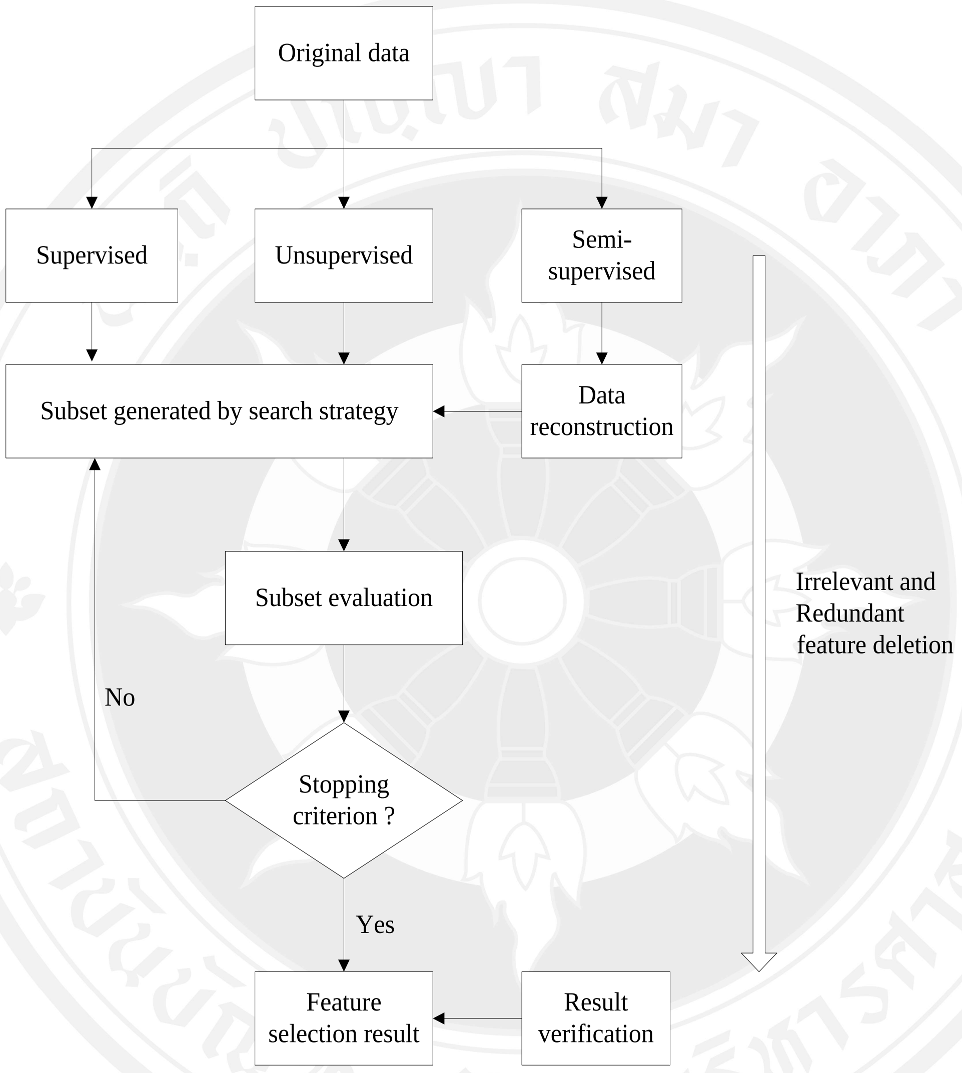


Figure 1.5 Feature Selection Framework

According to Mwadulo (Mwadulo, 2016), supervised feature selection is normally used in the classification problem by calculating the correlation between the feature and the class label. The supervised model aims to find an optimal feature subset that maximizes the classification accuracy. In the filter method, to analyze the relevance and redundancy of feature-class and feature-feature respectively, we need to use a model such as Euclidean distance, information measures, and Pearson correlation. A classical criterion for feature selection is MRMR (Max-Relevance and Min-Redundancy), which uses mutual information as the evaluation measure. For wrapper models, the classification error or accuracy rate is used as the feature evaluation. The wrapper model tends to have higher classification accuracy than the filter model.

Unsupervised feature selection is dealing with how to arrange the objects into natural classes whose members are similar to each other. This procedure is particularly difficult due to the absence of class labels for feature relevance estimation. Normally, unsupervised feature selection applies to the clustering processes which aim to maximize intra-cluster similarity and minimize inter-cluster similarity. The problem of selecting features in unsupervised learning scenarios is considered to be a much harder problem due to the absence of class labels that would guide the search for relevant information. Both filter and wrapper approach can be useful in unsupervised feature selection.

For the semi-supervised feature selection, the dataset (*D)* is divided into two groups where the first group is the sample set with class labels that use to train the learning model. The second group is the sample set without class labels that use to improve the learning performance of the learning model trained by the first sample group. The semi-supervised feature selection method is mainly based on the filter approach using score functions such as variance score, Laplacian score, Fisher score, and Constraint score (Cai, Luo, Wang, & Yang, 2018).

1.4 Performance Validation

The performance of the feature selection method is usually evaluated by the machine learning model. The commonly used machine learning models include Naïve Bayes, K-Nearest Neighbor, C4.5, Support Vector Machine, Decision Tree, Random Forest, Artificial Neural Network and K-means. A good feature selection method should have high learning accuracy but less computational time. The model should also avoid overfitting and underfitting. Overfitting occurs when the model results in good accuracy for the training data set but has a poor result on the new data set. Underfitting occurs when a machine learning model cannot capture the underlying trend of data that is it does not fit the data well enough. To solve these problems we need to apply the cross-validation technique, which is introduced in the next section.

1.5 Cross Validation

In machine learning, cross-validation is a statistical method used for model generation on a limited data sample. The idea is to separate the sample into 2 groups called the training set and the test set. The training set used to train the model, while the test set is for model evaluation. The most common procedure uses a single parameter called *k* which refers to the number of groups that a given data sample is spitted into and is often called *k*-fold cross-validation. This method is quite popular due to the simplicity of understanding and less biased than other methods. The general procedure starts by shuffle the dataset randomly then split the dataset into *k* groups and then takes one group as a hold out for testing data. The remaining groups are assigned to be the training datasets. Fit a model on the training set and evaluate it on the testing set. Retain the evaluation score for each observation and find the average. Each sample is allowed to be used in the holdout set once and used to train the model $(k - 1)$ times.

1.6 Overview

In this dissertation, we explored an adaptive sequential feature selection algorithm, which was used under supervised learning for classification problems. The paper was organized into five chapters. Some background introduction on feature selection explained in chapter 1. In chapter 2, related works in the literature were reviewed. In chapter 3, we discussed the proposed method which improved sequential feature selection using an adaptive multi-level backwards selection. In chapter 4, we introduced an adaptive multi-level forward inclusion technique with replaced the weak feature to select more effective features and improved the classification performance. Conclusions and recommendation for future works are concluded in chapter 5.

CHAPTER 2

LITERATURE REVIEW

Due to the increase in features domain from tens to thousands of variables used in the application, several techniques are developed to extract only those relevant and non-redundant variables which help in understanding data, lower computational time, and improve performance. This is not similar to other dimension reduction methods such as Principal Component Analysis (PCA) because good features are independent of the rest of the data. The exhaustive evaluation results in an NP-hard problem, therefore a suboptimal procedure can be used for an exceptional reason.

Feature selection becomes a necessary step in the data mining process because the high dimensionality and vast amount of data give rises to a challenge to the learning task. Many irrelevant features do not add much value during the learning process, hence learning models tend to become highly complicated and decrease learning accuracy. Feature selection is one effective way to identify relevant features for dimensionality reduction. However, the benefit of feature selection comes with an extra effort by trying to get an optimal subset that represents the original dataset.

Jovic, et al. (Jovic et al., 2015) categorized feature selection methods into three common search strategies. Exponential algorithms evaluate subsets that grow exponentially with the feature space size, for example, Exhaustive search and Branch-and-bound. Sequential algorithms such as Sequential Forward Floating Selection (SFFS) (Pudil, Novovicova, & Kittler, 1994) include or exclude features from the active subset sequentially. Random algorithms incorporate randomness into the search process to optimize the solution. An example of a random algorithm is Evolutionary computation algorithms using genetic or ant colony optimization. This research focuses on sequential feature selection regarding deterministic algorithms.

Since from the utilized training data, the feature selection method can be divided into supervised, unsupervised, and semi-supervised learning, while in this research, we focus on feature selection methods using supervised learning. According to their relationship with learning methods, feature selection methods can be categorized into the filter, wrapper, and embedded models which are reviewed in the following section.

2.1 Type of Feature Selection

2.1.1 Filter Method

Filter methods use variable ranking techniques by ordering features according to the principle criteria such as correlation criteria, mutual information, or information distance. After the ranking, a threshold is used to remove a feature that has a score below the threshold. The name filter method comes from the nature of the method since the process of filtering out the less relevant features is applied before the classification step. Therefore, the filter method consists of the selection algorithm and a criterion function. For the selection algorithm, all features are ranked by an ascending order regarding the criterion value, meanwhile, the searching strategy will be applied to generate subsets until the process meets the stopping criterion. The filter method is more preferable for real-world problems due to its characteristics in terms of time and space over the wrapper and embedded method, while the performance is compatible with the other two. The advantages of filter methods are that they easily scale to high-dimensional datasets, are computationally simple, and also are independent of data mining. The subset of features selected is presented as input to the classification algorithm. Therefore, the accuracy of this method heavily depends on the quality measure.

The basis of the filter method is relevance and redundancy. Relevance is the relationship between feature and class, whereas redundancy is the relationship between feature and feature. Features can be divided from the original set into four groups (Liu & Yu, 2005): 1) completely irrelevant and noisy features, 2) weakly relevant and redundant features, 3) weakly relevant and non-redundant features, and 4) strongly relevant features. Supervised feature selection results should include groups 3) and 4).

A classical criterion for feature selection based on relevance and redundancy analysis is Max-Relevance and Min-Redundancy (MRMR), which uses mutual information as the quality measure. This technique (Peng, Long, & Ding, 2005) studied how to select good features according to the maximal statistical dependency criterion based on mutual information. The experiments on many datasets (handwritten digits, arrhythmia, NCI cancer cell lines, and lymphoma tissues) showed that MRMR leads to improvement in feature selection and classification accuracy. However, mutual information only minimizes feature-feature mutual information and ignored the classification performance of candidate features. To overcome this problem, conditional mutual information was introduced.

2.1.2 Wrapper Method

Wrapper methods use the result of the data mining algorithm to determine how good a given subset is. During the search process, the space of possible feature subsets is defined to generate and evaluate until we get the satisfied subset for the preprocessing step. The main advantage is that the quality of the generated subset is directly measured by the performance of the data mining algorithm. As a result, wrapper methods seem to be much slower than the filter methods as the data mining algorithm is applied to each attribute subset considered by the search. The application of several different data mining algorithms leads to the wrapper method becoming even more computationally expensive.

Since the evaluation of all the possible subsets is an NP-hard problem, suboptimal subsets become more focused and are found by employing search algorithms that find a subset heuristically. Various algorithms either sequential search or evolutionary algorithms such as genetic algorithm (GA) and particle swarm optimization (PSO) are studied thoroughly which produce local optimum results. These results are exceptional in many applications and are computationally feasible. Wrapper methods can be classified into sequential selection algorithms and heuristic search algorithms.

2.1.3 Embedded Method

Embedded methods can also be called hybrid methods. The aim is to reduce the computational time on the classification step of the wrapper methods. They try to combine the advantages of both filter and wrapper methods. One example is the Hybrid Floating Sequential Search (hSFFS) (Somol et al., 2006), by applying a filter criterion function to filter some features and generate a candidate set before applying a wrapper criterion function to select the best one from the candidate set. This technique also consists of the forward and backward phases. The forward phase allows the current subset to add one feature at a time by first using the filter-based technique to pre-select several candidate features. Secondly, apply the wrapper-based technique to identify the best feature among those candidate features. The backward phase removes one feature conditionally by pre-select several candidate features in the current subset using the filter approach, then identifying the best one to remove using the wrapper approach. This hybrid scheme uses only a fraction of the full wrapper computing time to obtain the results. The advantage of hSFFS is the possibility to deal with the quality of results versus the computational time trade-off. The results showed that it was possible to trade a significant reduction in search time for a little decrease in classification accuracy.

2.2 Sequential Forward Search (SFS)

A.W. Whitney (Whitney, 1971) introduced a Sequential Forward Search (SFS) by starting with an empty set and adding one feature at a time to the selected subset so that the new subset maximizes the criterion function value. During each iteration process, the remaining features are added individually to the current subset and a new subset is evaluated. After the number of features is satisfied, the selection process will be terminated.

The SFS process in a forward direction and is essential for constructing other more complex algorithms. Large datasets normally contain a lot of features whereas only some of them are significant for model training. The idea is to select a feature that gives the highest learning accuracy. Assume we have a set $Y = \{y_1, y_2, \ldots, y_D\}$, where D is the number of input dimensions. We want to find a subset $X_k = \{xj \mid j = 1, 2, \ldots, k; x_j \in Y\}$, where $k = (0, 1, 2, \ldots, D)$, and d is the required subset size. Initialize $X_0 = \{\}$ and $k = 0$, and $x+$ is an included feature where $x \in Y - X_k$.

The algorithm is described below:

Step 1: Inclusion step.

$x^{+} = \arg\max J(x_k = x)$, where $x \in Y - X_k$

$X_{k+1} = X_k + x^{+}$

$k = k + 1$

(Add a selected feature x^{+} to the subset X_k, where x^{+} is a feature that maximizes the criterion function (J).)

Step 2: Continue step 1 until d features are selected.

2.3 Sequential Backward Search (SBS)

T. Marill and D. Green (Marill & Green, 1961) have presented a Sequential Backward Search (SBS) which processes in an opposite direction with the SFS. This method begins with all input features and keeps removing one feature at a time from the current subset until the resulted subset maximizes the criterion function value. The idea is to remove the feature whose removal gives the lowest decrease in predictor performance. These two techniques combined to form a generalized version of the SFS and SBS by adding or removing several features in each sequential step. These methods have one major drawback, for example when the best five features subset is selected it must contain the best four features subset, but in practice, the best four features subset does not necessarily be part of the best five features subset. This problem is called the 'nesting effect'. As a result of not being allowed to add or to remove later, they are sensitive to feature interaction, hence they can easily be trapped into local minima.

2.4 Sequential Forward Floating Search (SFFS)

One of the most significant innovations of sequential feature selection is the Sequential Forward Floating Search (SFFS) algorithm (Pudil et al., 1994). This method combines the concept of SFS and SBS giving it more flexibility and effective than SFS by introducing a backtracking step. The simplified flowchart of the SFFS algorithm is given in figure 2.1 where *k* is the current subset size and *d* is the required dimension.

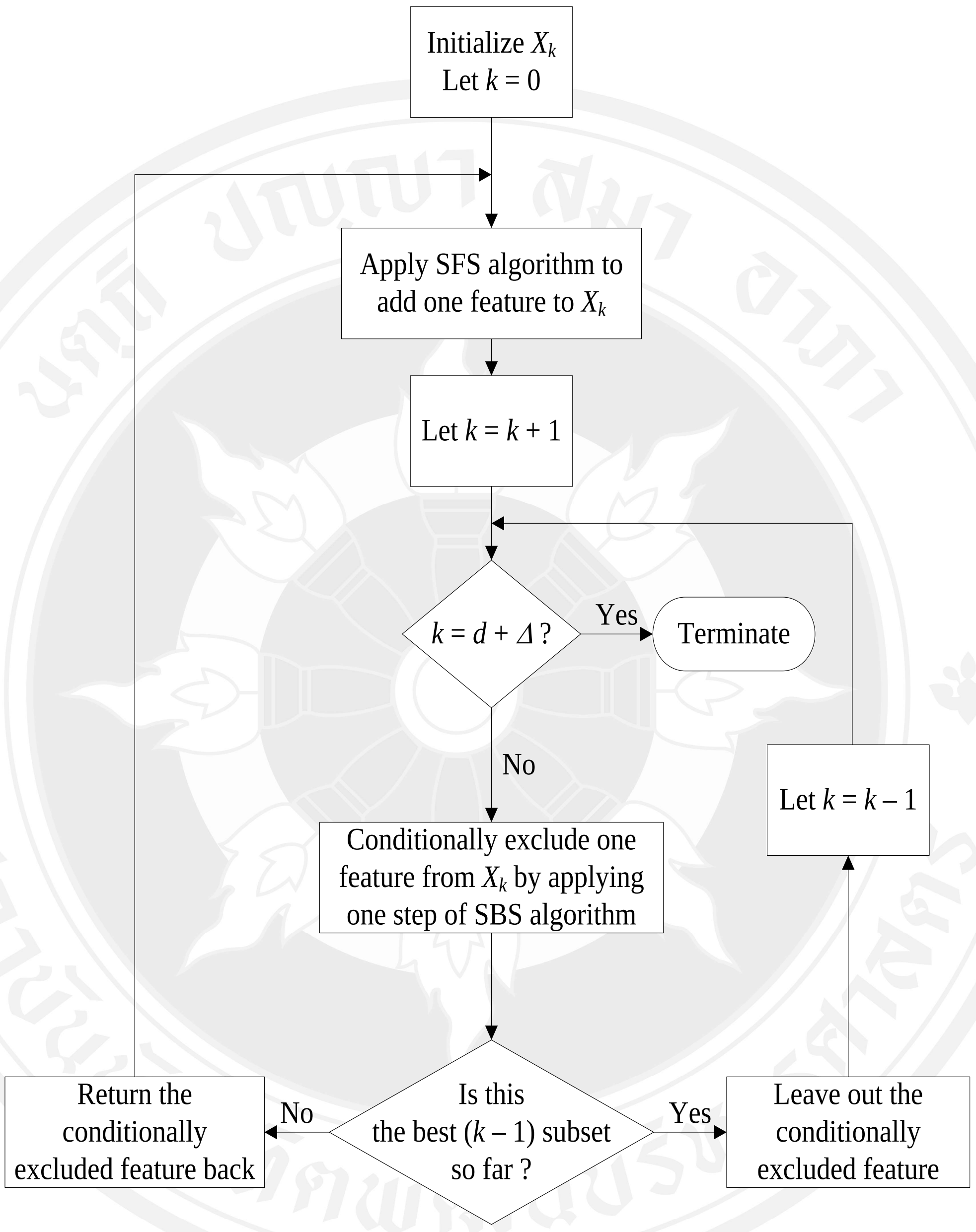

Initialize X_k
Let $k = 0$
Apply SFS algorithm to add one feature to X_k
Let $k = k + 1$
$k = d + \Delta$?
Yes
Terminate
No
Let $k = k - 1$
Conditionally exclude one feature from X_k by applying one step of SBS algorithm
Return the conditionally excluded feature back
No
Is this the best $(k - 1)$ subset so far ?
Yes
Leave out the conditionally excluded feature

Figure 2.1 Structure of the SFFS Algorithm

SFFS algorithm consists of two parts, forward search and backward search. The forward search selects the best-unselected feature according to a criterion function (*J*) to form a new subset. This step is the same as the SFS algorithm. The

SBS method starts with a full feature subset and eliminates a feature in each iteration until a predetermined criterion is satisfied. The backtracking step is the conditional step where improvement can be made during the search process. As mentioned earlier, the best four features do not necessarily lead to the best five features because the forward step is unconditional. Therefore, SFFS is suffering from the nesting problem as well. However, SFFS is said to be a state-of-the-art method that is widely used in several applications. Researchers in sequential feature selection normally extend their method using SFFS as the standard method to compare their results. To describe the SFFS algorithm below, let x^- be an excluded feature where $x \in X_k$:

Step 1: Inclusion step. (Apply SFS algorithm.)

Step 2: Conditional exclusion step. (This step is similar to the SBS algorithm.)

$x^- = arg\ max\ J(x_k = x),\ where\ x \in X_k$

If $J(x_k - x^-) > J(x_{k-1})$:

$X_{k-1} = X_k - x^-$

$k = k - 1$

(Remove a feature if the resulting subset improves the performance. If $k \le 2$ or there is no improvement on X_{k-1} then go to step 1, or else, repeat step 2.)

Step 3: Continue steps 1 and 2 until d features are selected.

2.5 Adaptive Sequential Forward Floating Search (ASFFS)

After the introduction of SFFS, several improved versions have been proposed to obtain better performance. An adaptive version of the floating search method (Somol, Pudil, Novovicova, & Paclik, 1999) was presented to improve the performance of the SFFS algorithm. The idea behind Adaptive SFFS (ASFFS) is selecting features to add or remove more than one feature in each sequential step in order to search for a better subset. The number of search features in each step can be varied depending on the remaining features in the dataset. The result is a more thorough search with a better chance to find an optimal solution by setting a higher generalization level. The algorithm of ASFFS can be illustrated in figure 2.2.

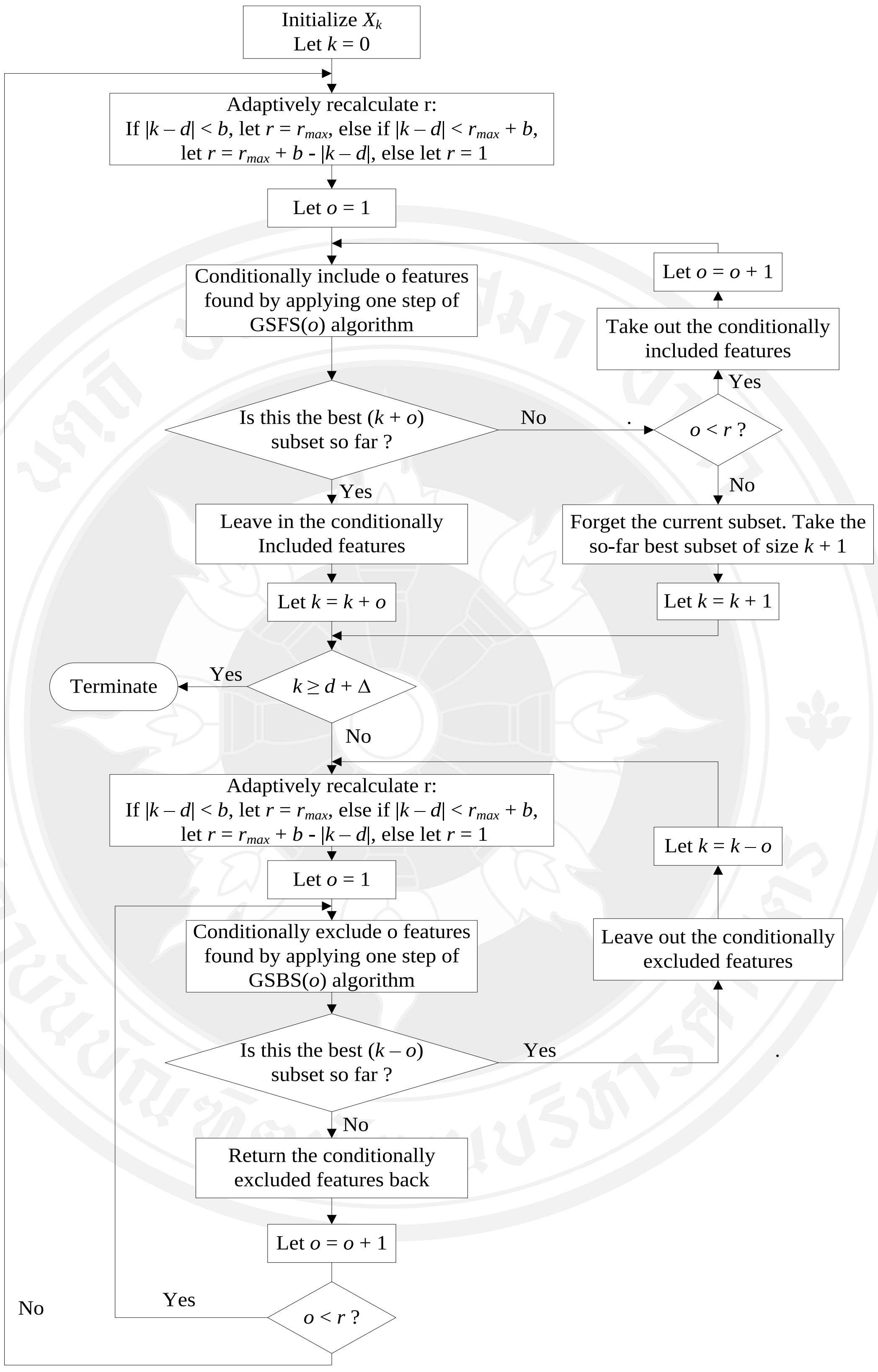

Initialize X_k
Let $k = 0$
Adaptively recalculate r:
If $|k - d| < b$, let $r = r_{max}$, else if $|k - d| < r_{max} + b$,
let $r = r_{max} + b - |k - d|$, else let $r = 1$
Let $o = 1$
Conditionally include o features found by applying one step of GSFS(o) algorithm
Is this the best $(k + o)$ subset so far ?
No
$o < r$?
Yes
Take out the conditionally included features
Let $o = o + 1$
No
Forget the current subset. Take the so-far best subset of size $k + 1$
Let $k = k + 1$
Yes
Leave in the conditionally Included features
Let $k = k + o$
$k \geq d + \Delta$
Yes
Terminate
No
Adaptively recalculate r:
If $|k - d| < b$, let $r = r_{max}$, else if $|k - d| < r_{max} + b$,
let $r = r_{max} + b - |k - d|$, else let $r = 1$
Let $o = 1$
Conditionally exclude o features found by applying one step of GSBS(o) algorithm
Is this the best $(k - o)$ subset so far ?
Yes
Leave out the conditionally excluded features
Let $k = k - o$
No
Return the conditionally excluded features back
Let $o = o + 1$
$o < r$?
Yes
No

Figure 2.2 Structure of the ASFFS Algorithm

The ASFFS method attempts to obtain a less redundant subset than the SFFS algorithm. The forward step can lead to finding a subset that is worse than the best one of a given dimension that has been found so far. If this occurs, the current one is forgotten and the best one so far becomes the current one. The two free parameters, r_{max} and b, in the ASFFS specify the generalization limit and range of the adaptive search. The parameter r specifies the number of features to be added in the forward phase or inclusion phase which is calculated adaptively. In the backward phase or exclusion phase, remove o features if it increased the performance. With $r_{max} = 1$, the ASFFS is identical to SFFS. The suggestions for the two values are 4 and 3, respectively. The nearer the current subset size to d, the higher is the generalization limit (Figure 2.3). The reason behind this characteristic is to save the computing time by limiting the generalization level while the current subset is still far from the desired one. The generalization level (r) increases when the number of features (k) in the current subset gets close to d until it reaches r_{max}. ASFFS has shown better results than SFFS due to a more thorough search. Theoretically, better results are also depending on the criterion function and distribution of the data.

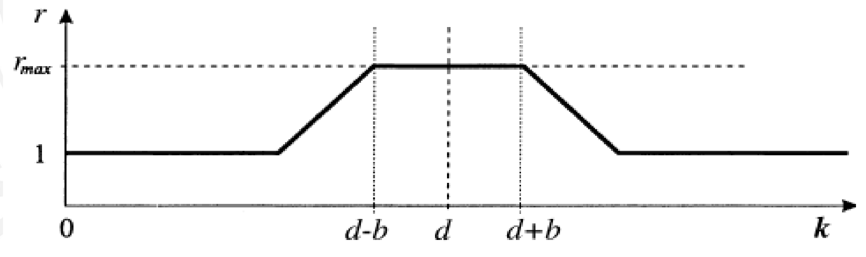


Figure 2.3 An Adaptive Determination of r-values for ASFFS

The calculation of the r-value occurs at the beginning of every forward and backward phase using the following conditions:

1) If $|k – d| < b$, let $r = r_{max}$
2) Else if $|k – d| < b + r_{max}$, let $r = r_{max} + b – |k – d|$
3) Else let $r = 1$

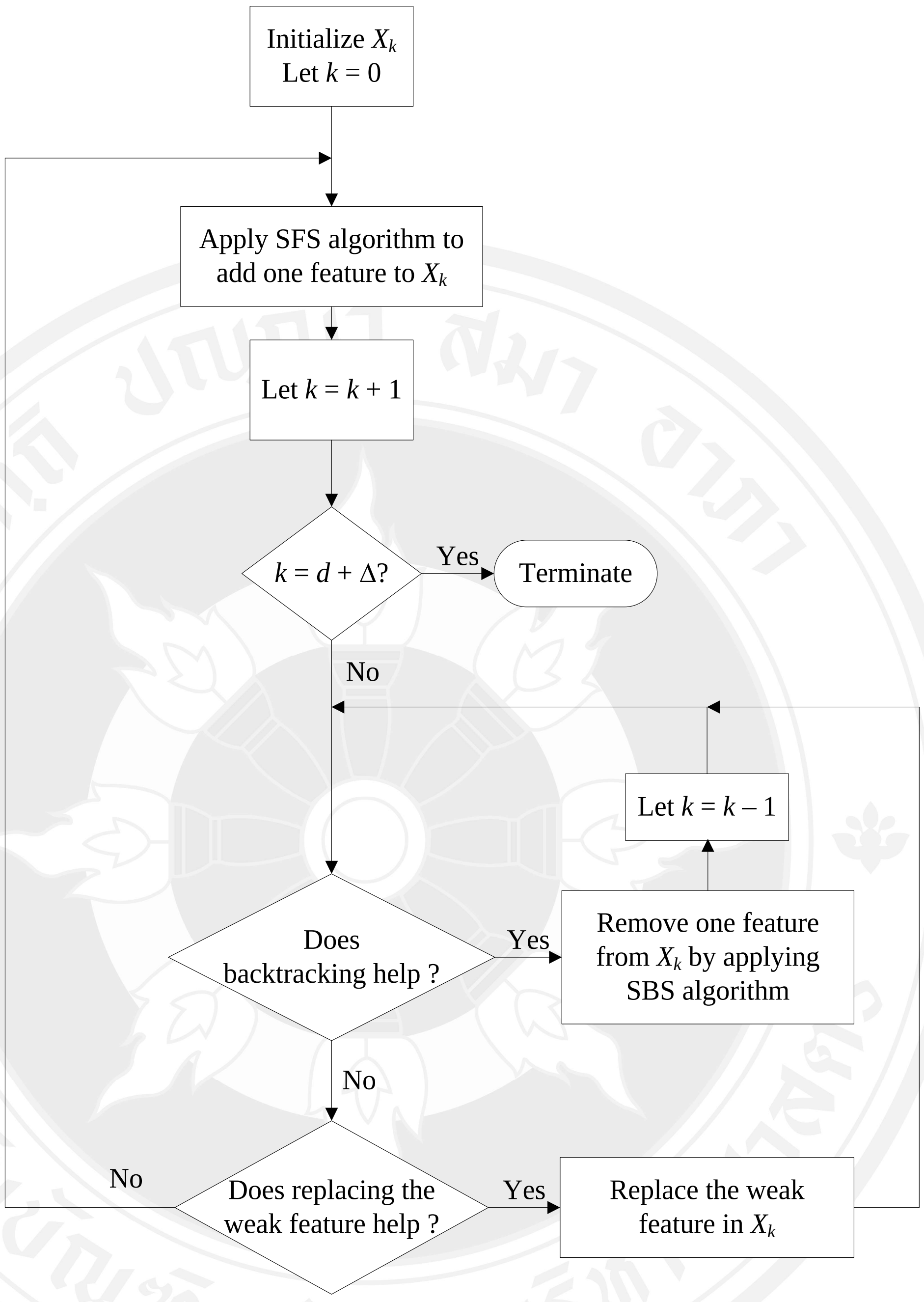
While the number of features k is far from the required subset size (d), r is assigned a value of 1, which is exactly the same as SFFS’s procedure. When k gets closer to d, the value for r increases but no more than r_{max}. Even though ASFFS has shown slightly better results than SFFS, it takes more computational time due to the complexity of the algorithm. The adaptive step leads to additional work to the SFFS structure in both in the forward and the backward direction. Elements of the current feature subset can be increased or decreased along the searching process, which is another reason for the longer time required. The backtracking step explores features within the current subset without considering the unselected features that are located outside the boundary. The generalization level can be helpful during the search only when k in the current subset is getting close to the target size, thus the detailed search concept works only when k almost reaches the end of the process. An adaptive calculation of r can be illustrated by an example. Assume we specify $r_{max} = 4$ and $b = 3$, then the value of r is shown in Table 2.1.

Table 2.1 Example of r-values from ASFFS

$\|k - d\|$	1	2	3	4	5	6	7	8
r	4	4	4	3	2	1	1	1

2.6 Improved Forward Floating Search (IFFS)

One of the most remarkable improvements on SFFS was the IFFS algorithm (Nakariyakul & Casasent, 2009), which had successfully removed the weakness of SFFS by adding an additional step to improve the criterion function value. Based on the fact that it is not necessary that the best k-subset contains all features from the best (k–1)-subset, IFFS was introduced to solve this nesting problem. This improved step is called ‘replacing the weak feature’ which is to check whether removing any feature in the currently selected feature subset and adding a new one at each sequential step can improve the current feature subset. IFFS can impressively prevent the nesting effect of SFFS and the algorithm is simpler than ASFFS with exceptionally short computing time. IFFS yields better performance than both SFFS and ASFFS with a little more process time than SFFS. Figure 2.4 demonstrates the IFFS algorithm using a flowchart.



Initialize X_k
Let $k = 0$
Apply SFS algorithm to add one feature to X_k
Let $k = k + 1$
$k = d + \Delta$?
Yes
Terminate
No
Let $k = k - 1$
Does backtracking help ?
Yes
Remove one feature from X_k by applying SBS algorithm
No
No
Does replacing the weak feature help ?
Yes
Replace the weak feature in X_k

Figure 2.4 Structure of the IFFS Algorithm

The IFFS algorithm is described below:

Step 1: Inclusion step. (Apply SFS algorithm.)

Step 2: Conditional exclusion step. (Apply SBS algorithm.)

Step 3: Check if replacing the weak feature helps.

```
For $x_i$ in $X_k$ :
    $X_{k-1} = X_k - x_i$
    For $x_j$ in $Y - X_{k-1}$ :
        $x_j = \arg\max J(x_j)$
        If $J(X_{k-1} + x_j) > J(X_k)$:
            $X_k = X_{k-1} + x_j$
```

(Generate *k* new subsets of *k* features by removing one feature and adding one feature using SFS. Calculate the *J*-values of *k* subsets. If the subset with the largest *J*-value gives an improvement, then replace the new subset with the current subset, and go to step 2. Otherwise, go to step 1.)

Step 4: Continue steps 1, 2 and 3 until *d* features are selected.

2.7 Sequential Deep Floating Forward Search (SDFFS)

Another recent sequential search algorithm in this field is the Sequential Deep Floating Forward Search (SDFFS) (Lv, Peng, & Sun, 2015). This algorithm also tries to improve the state-of-the-art SFFS algorithm. The deep searching step aims to confirm whether there exists a subset with *k* features being better than the current one that has been found so far using the SFS step which cannot be found in the SFFS algorithm. In the experiment, SDFFS uses eight datasets by pre-selecting 100 features from each subset. Those 100 features are filtered out using MRMR as a criterion function. The criterion function value used in this paper is the classification accuracy computed from the KNN classifier where $K = 5$. The SDFFS algorithm shows in figure 2.5.

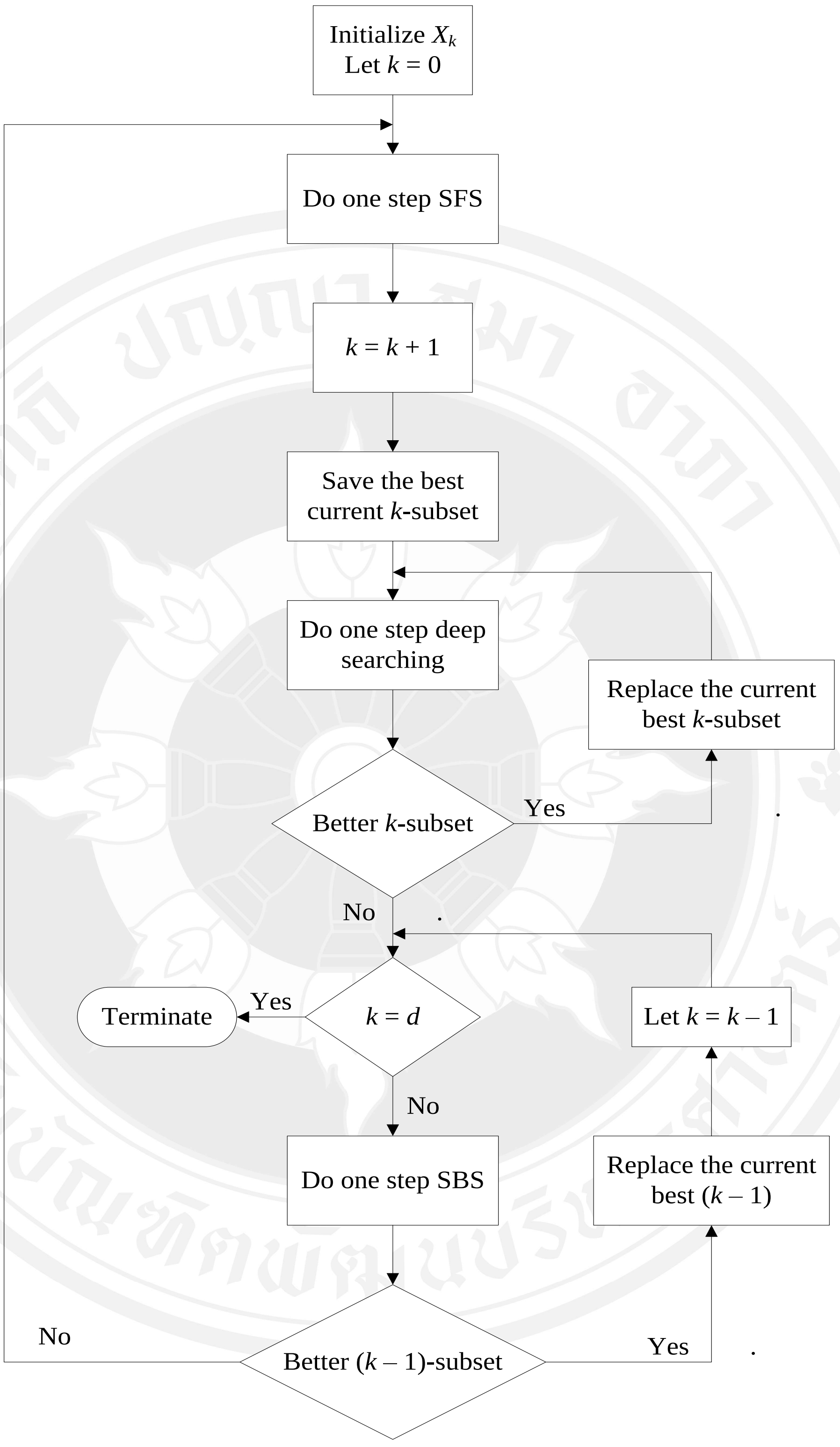


Figure 2.5 Structure of the SDFFS Algorithm

The SDFFS algorithm is described below:

Step 1: Adding a feature to the selected subset using the basic SFS method by selecting the most significant feature concerning X_k and include it in X_k, $k = k + 1$.

Step 2: Deep searching step. For each iteration, remove one feature except the previous feature added to get the new (k–1)-subset for k–1 subsets. To every new (k–1)-subset, select N features from the candidate set and each time add one into the new (k–1)-subset, which leads to N new k-subsets. Calculate all the new potential k-subsets' criterion function values and select the one with the highest criterion function value. If the selected k-subset is better than the current k-subset, replace it with the current one. Repeat step 2. Otherwise, go to step 3.

Step 3: Check if backtracking helps. Remove the least significant feature k in X_k. If the backtracking step results in an improvement, then decrease k by 1 and repeat step 3. Otherwise, go to step 1.

The experimental results show that SDFFS does not perform the best all the time for all feature's sizes. However, the results are relatively high accuracy and more stable. This led to the overall performance being quite remarkable. Moreover, one major drawback for SDFFS is the computational time that is far greater than those previous methods. SDFFS searches through the first 100 features with only slight chances to gain accuracy especially those with low criterion function value, and thus modification of the algorithm should be concerned to make it more effective.

2.8 Other Related Works

Chaiyakarn proposed (Chaiyakarn, 2013) a Filter-Based Feature Selection Using Two Criterion. The algorithm relies on a criterion function by applying CMI as the first criterion and selects one of the information measures; which are mutual information, Bhattacharyya distance, Jeffreys-Matusita distance bound to the Bayes error, and Mahalanobis distance, as a second criterion. The two functions can complement different characteristics of data together for selecting features more effectively. The evaluation is independent of the data mining algorithm. The experimental results show that this technique takes less time to select a significant features subset, particularly on high-dimensional data, whereas an optimal feature set cannot guarantee.

The two criteria filter-based approach provides an option for users to select two suitable criterion functions since each function has a unique characteristic and it has shown that Jeffreys-Matusita distance bound to the Bayes error as the second criterion function yields the best result. This method outperforms the original filter-based approach with one criterion function and also provides more opportunity to get higher accuracy. Due to the same structure as the original filter-based method, this technique also suffered from the nesting problem.

A recent study from Homsapaya and Sornil (Homsapaya & Sornil, 2017) introduced a floating search technique employing a genetic algorithm (GA) to improve the quality of the selected feature subset. The results showed that GA improved the performance for the majority of sample datasets. Kadhum, et al. (Kadhum, manaseer, & Dalhoum, 2021) have proposed a new model for evolutionary wrapper feature selection by applying GA to explore the space of feature combinations from a set of features that already has its priorities assigned. Extreme Learning Machines (ELM) and Support Vector Machine (SVM) was used as the classifiers based on the Chronic Kidney Disease dataset (CKD) from the UCI repository (Dheeru & Efi, 2017). The application of the proposed model affected the classification performance by improving the accuracy rate while also reducing the computing time.

Other recent works in the feature selection domain focused on the application of feature selection techniques to other areas of work such as face recognition, text classification and medical science (Bolon-Canedo & Alonso-Betanzos, 2019), (Cisotto, Capuzzo, Guglielmi, & Zanella, 2020) and (Raj et al., 2020). The improvement of sequential feature selection tends to focus on non-deterministic algorithms like particle swarm optimization, genetic algorithms or deep neural networks (Liu & Wang, 2019) and (Huda & Banka, 2019), while our study concerned a deterministic algorithm.

CHAPTER 3

ONE LEVEL FORWARD/MULTI-LEVEL BACKWARD SELECTION (OFMB)

Feature selection using wrapper approach is more of interest due to the high classification accuracy when compared with other approaches. From the past researches, several methods applied the wrapper approach to sequential feature selection. One of the most popular sequential search algorithms is Sequential Forward Floating Selection (SFFS), which represents the state-of-the-art method. Other techniques were usually developed from SFFS in order to improve classification accuracy with a reasonable time complexity and also overcome the effect of the nesting problem. The development of Adaptive Sequential Forward Floating Selection (ASFFS) and an Improved Forward Floating Selection (IFFS) has been shown to be superior to the standard SFFS. In this dissertation, we attempted to take advantages of ASFFS and IFFS to combine them together to form new sequential floating feature selection algorithms that produced better results than the earlier works. We improved our algorithm based on the IFFS by removing the backtracking step and inserting an adaptive step to give the higher chances for discovering better solutions with less computational time.

In the first half of this research, we present One-level Forward Multi-level Backward Selection (OFMB), which is a sequential forward selection that explores possible subsets several levels deeper in order to maximize the classification accuracy of the learning dataset. The idea is to explore backward after feature inclusion since newly included features may affect smaller subsets. This backward search can examine many levels by excluding more than one feature in each iteration. This method considers a wider range of features when searching backwards deeper. OFMB is similar to the backtracking step of SFFS but it can explore feature subsets to much greater depth. Subsequently, a new, smaller subset with a higher criterion function can be discovered, whereas the standard SFFS or even the IFFS are not capable of finding such subsets.

The first part of OFMB is a result of our proposed technique called One Level Forward Inclusion (OLFI). The idea of OLFI is to relocate the ‘replacing the weak feature’ step from the last part of the IFFS algorithm at the beginning of our selection process. As a result, improvement can be made during the operation and the backtracking phase can be ignored because if no improvement occurs, the backward step cannot proceed further than (*k*–1) feature anyway. Therefore OLFI is the construction of feature subsets that have relatively high classification accuracy close to the IFFS algorithm. The second part is to explore the recently selected subset deeper backwards up to some specified point. After the inclusion and improvement steps from the OLFI technique, we remove one or more features from the currently selected subset to form many subsets of size (*k*–*s*), where *s* refers to the number of removed features range from 1 to *r*, and *r* is the generalization limit. The searching target is a subset with a higher *J*-value for a particular subset size. We propose the conditions used to calculate the value of *r* in the following subsection. As a result of applying the OFMB algorithm, there are higher chances to find a better feature subset of size (*k*–*s*).

3.1 One Level Forward Inclusion (OLFI)

Most sequential search techniques consist of two parts which are the forward phase and the backward phase. Related to this combination, the number of features keeps increasing or decreasing throughout the searching process as long as an improvement of the criterion function value (*J*-value) can be made. While the number of features almost reaches the required subset size (*d*) but it may decrease during the backtracking part. Our work comes up with an idea of whether it would be possible to remove the backtracking step and determine the improvement only on the forward step. Consequently, our first proposed algorithm tries to manage this idea into reality. This is the motivation of the OLFI algorithm which seems to give the algorithm to perform better than the standard SFFS.

Normally, most sequential search algorithms start with an empty set. First we add features using SFS until we get more than two features and then the process can continue to exclude or include features according to which direction can give a higher *J*-value for a subset of size *k*. As opposed to the other sequential search algorithms, OLFI allows the number of features to be either increased or remain the same without a backtracking step. The insertion of 'replacing the weak feature' during the forward phase makes it possible to improve the feature subset. While using SFS to include one feature, we try to find a better feature subset by removing one feature in that subset for every element except the one that just has been added. If an improvement can be made, replace a new subset with the current one. If there is no improvement, we keep on adding a feature for the next iteration. This is a feature improvement step applying a technique from the IFFS algorithm. The OLFI algorithm is explained below with the flowchart in figure 3.1.

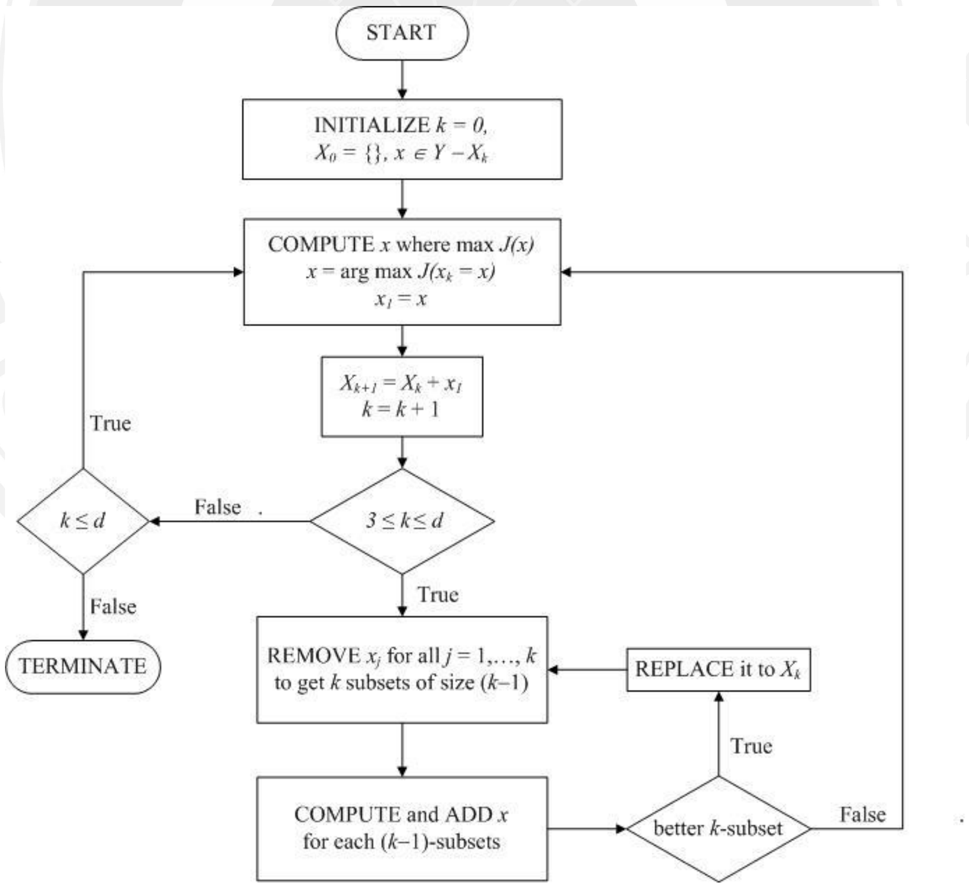

START
INITIALIZE $k = 0$, $X_0 = \{\}$, $x \in Y - X_k$
COMPUTE x where max $J(x)$
$x = \arg\max J(x_k = x)$
$x_1 = x$
$X_{k+1} = X_k + x_1$
$k = k + 1$
True
False
$k \leq d$
$3 \leq k \leq d$
False
True
TERMINATE
REMOVE x_j for all $j = 1,\dots, k$ to get k subsets of size $(k-1)$
REPLACE it to X_k
True
COMPUTE and ADD x for each $(k-1)$-subsets
better k-subset
False

Figure 3.1 Structure of the OLFI Algorithm

The OLFI algorithm is described below:

Step 1: Apply SFS to select one feature from the remaining feature set. Add this feature to the selected feature subset. Continue step 2 with the feature subset X_k where $k = k + 1$.

Step 2: From the selected feature subset size k, remove 1 feature iteratively we have X_{k-1}, and use SFS to select a new feature from the remaining feature set $(Y - X_{k-1})$ for adding to each feature subset. Then calculate whether there is an improvement. If there is an improvement, replace that previous feature subset with the newly selected feature subset and repeat step 2. Otherwise, continue step 3 with the feature subset X_k.

Step 3: Continue steps 1 and 2 until d features are selected.

3.1.1 An Example using Wine Dataset

To demonstrate the OFMB algorithm, we selected the Wine dataset from the UCI repository based on the KNN classifier. First, assume we have a dataset $Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ with 13 features; the required subset size (d) is 13 features.

3.1.1.1 Feature Inclusion

At the beginning, assume we apply SFS for the first 3 features, thus for k = 1, 2 and 3 we have $X_1 = \{6\}$, $X_2 = \{6, 10\}$ and $X_3 = \{6, 10, 2\}$ respectively. Now, the current subset of $k = 3$ is $X_3 = \{6, 10, 2\}$. This subset is the best 3-subset that has been found so far.

3.1.1.2 Feature Improvement

Assume we continue the process up to $k = 4$. We have $X_4 = \{6, 10, 2, 7\}$ with 90.09% classification accuracy. Remove one feature except $x_4 = 7$ and we have {6, 10, 7}, {6, 2, 7} and {10, 2, 7}. Then, select one feature from the remaining set that produces the best J value with those 3 subsets. Now we have new subsets of size 4 for consideration. After calculation we find that $J(\{10, 6, 7, 9\})$ produces the highest J value with 92.84% accuracy. Therefore, replace {6, 10, 2, 7} with {10, 6, 7, 9} as the best subset of size 4 that has been found so far. Repeat the same process for $X_4 = \{10, 6, 7, 9\}$ and we cannot find any better subset of size 4. Then return to step 1, continue adding the next best feature for $k = 5$ to get $X_5 = \{10, 6, 7, 9, x_5\}$ where x_5 is the newly selected feature from the remaining feature set ($Y - X_k$). This process produces similar solutions with the IFFS algorithm.

3.1.1.3 Termination Condition

The OLFI algorithm processes sequentially until the subset size k reaches the required subset size (d). The best of all feature subsets are copied into X_k and then terminate the program.

We can see that the subset is increasing in size along with an improvement across the process until it reaches the required number of d. This method also solves the nesting problem that occurs in SFFS and produces a result similar to IFFS without backtracking step, which performs better than the SFFS algorithm that was known as the state-of-the-art method.

3.2 Multi-level Backward Selection

This section is the explanation of the proposed OFMB algorithm on the second part of our method after we already have preliminary results from the OLFI algorithm. This second part applies the multi-level backward tracking to improve the performance on the classification accuracy. A flowchart of the OFMB algorithm has shown in figure 3.2 followed by a pseudo-code.

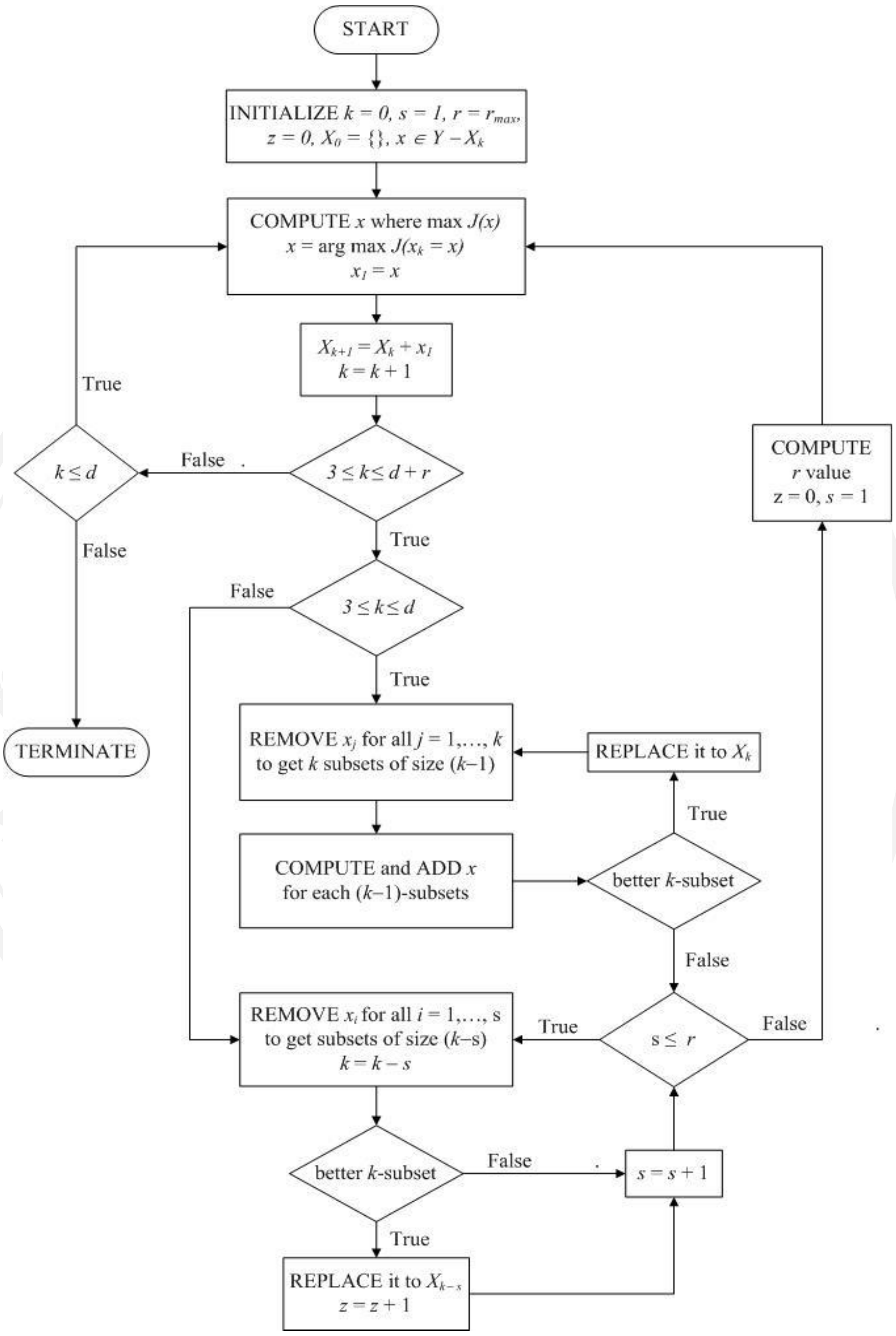

START
INITIALIZE $k = 0$, $s = 1$, $r = r_{max}$, $z = 0$, $X_0 = \{\}$, $x \in Y - X_k$
COMPUTE x where max $J(x)$
$x = \arg\max J(x_k = x)$
$x_1 = x$
$X_{k+1} = X_k + x_1$
$k = k + 1$
$3 \leq k \leq d + r$
False
$k \leq d$
True
False
TERMINATE
True
$3 \leq k \leq d$
False
True
REMOVE x_j for all $j = 1,\ldots, k$ to get k subsets of size $(k-1)$
COMPUTE and ADD x for each $(k-1)$-subsets
better k-subset
True
REPLACE it to X_k
False
$s \leq r$
True
False
REMOVE x_i for all $i = 1,\ldots$, s to get subsets of size (k−s) $k = k - s$
better k-subset
False
$s = s + 1$
True
REPLACE it to X_{k-s}
$z = z + 1$
COMPUTE r value
$z = 0$, $s = 1$

Figure 3.2 Structure of OFMB Algorithm

Algorithm: One-level Forward Multi-level Backward Selection (OFMB)

Input: A set of feature $Y = \{y_1, y_2, \ldots, y_D\}$, where D is the number of input dimension; J is a criterion function; d is the required subset size; r is the generalization level which is limited by r_{max};

Output: A feature subset $X_k = \{x_j \mid j = 1, 2, \ldots, k;\ x_j \in Y\}$, where $k = (0, 1, 2, \ldots, d)$.

Initialize: Initialize $X_0 = \{\}$; $k = 0$; $s = 1$; $r = r_{max}$; $z = 0$.

```
(1) Feature Inclusion
  #Find the best feature and update $X_k$
     $x^+ = \arg\max J(x_k = x)$, where $x \in Y - X_k$
     $X_{k+1} = X_k + x^+$
     $k = k + 1$
     $\max(X_k) = X_{k+1}$
(2) Feature Improvement
  #Replace a weak feature by trying to remove one feature and added one feature
     Repeat
          For $x_j$ in $X_k$ : #where  $j = 1, 2, \ldots, k$
               $X_{k-1} = X_k - x_i$
               For $x_i$ in $Y - X_{k-1}$ : #where  $i = 1, 2, \ldots, d - (k-1)$
                    $x_i = \arg\max J(x_i)$
                    If $J(X_{k-1} + x_i) > J(X_k)$:
                         $X_k = X_{k-1} + x_i$
                         $\max(X_k) = X_k$
     Until   $J(X_{k-1} + x_i) \le J(X_k)$
(3) Multi-level Backward Selection
  #Searching for better subsets by multiple backtracking step
     Repeat
          $x_s$ in $X_k$ : #where  $s = 1, \ldots, r$ and $x_s$ are the features from 1 to $r$
          $X_{k-s} = X_k - x_s$
```

```
            If $J(X_{k-s}) > J(\max(X_{k-s}))$:
                $\max(X_{k-s}) = X_{k-s}$
                $z = z + 1$
            $s = s + 1$
        Until   $s > r$
(4) *Compute r-value*
        If $z < r_{max}$ :
            $r = r_{max} - z$
        Else :
            $r = 1$
        $z = 0$
        $s = 1$
(5) *Termination Condition*
    #Terminate when $k > d$
        If $k \leq d$
            Go to step 1
        $X_k = \max(X_k)$ #for all $k$
        Return the best individual subset $X_k$
```

This is a description of the OFMB Algorithm:

Step 1: Apply SFS to select one feature from the remaining feature set. Add this feature to the selected feature subset. Continue step 2 with the feature subset X_k where $k = k + 1$.

Step 2: From the selected feature subset size k, remove 1 feature iteratively we have X_{k-1}, and use SFS to select a new feature from the remaining feature set $(Y - X_{k-1})$ for adding to each feature subset. Then calculate whether there is an improvement. If there is an improvement, replace that previous feature subset with the newly selected feature subset and repeat step 2. Otherwise, continue step 3 with the feature subset X_k.

Step 3: From the selected feature subset (X_k), remove s features iteratively from 1 to r. Then, searching for the best (k–s)-subset. If there is a better subset $X_{k\text{–s}}$, replace it to the previous $X_{k\text{–s}}$. Repeat steps 3 until $s > r$, then continue step 4.

Step 4: Compute the r-value, then continue step 5.

Step 5: Continue steps 1, 2, 3 and 4 until d features are selected.

3.2.1 Computation of *r*-value

The generalization limit (r) needs to be carefully specified since a larger value of r results in a more thorough search and also increases the time complexity. We introduce a user-defined parametric limit r_{max} to restricting the maximum generalization level. This number can be any integer depending on how deep we need to search but normally it is only a small integer. The suggestion of r_{max} from Somol et al. (Somol et al., 1999) is 4. In our experiments, we assigned the value of r_{max} to be 5 for all tested datasets. The level s is similar to the level o in ASFFS but s is determined dynamically according to the r calculation technique we have proposed.

The generalization limit can be changing adaptively depending on the number of times we have found better k-subsets. If we have found a few better subsets in the previous iteration, the next iteration we should try a deeper search and that will increase the value of r. On the other hand, if the previous iteration has found many better subsets, the next iteration may not need to go too deep that will decrease the value of r. This adaptive nature by adjusting the generalization limit automatically is aimed to save computing time. Therefore, the search should go deeper when the algorithm cannot find a better subset. The application of this calculation technique leads to better performance than the previous techniques via our algorithm. We have selected the first 20 features from the whole dataset for the experiments. OFMB considers a wider range of features that lead to a thorough search. As a result, there are higher chances to improve the current feature subset.

Assume $r_{max} = 5$, thus $1 \leq r \leq 5$. Let z be the number of times the algorithm has found a better subset for that particular iteration. We have assigned the relationship of z and r by $r_{max} – z$. Adaptive determination of r is defined as follows:

1) If $z < r_{max}$, let $r = r_{max} - z$
2) Else, let $r = 1$

From the condition above, we can build the following graph (Figure 3.3) that shows the value of r for the first 20 features based on the values from the Ionosphere dataset in table 3.1. The value for r decreases while z increases.

Table 3.1 The *r*-values and the *z*-values from the Ionosphere Dataset

No. of features (*k*)	***z*-values**	***r*-values**
1	0	5
2	0	5
3	2	5
4	0	3
5	3	5
6	0	2
7	2	5
8	0	3
9	0	5
10	0	5
11	3	5
12	0	2
13	2	5
14	1	3
15	0	4
16	0	5
17	3	5
18	0	2
19	4	5
20	0	1

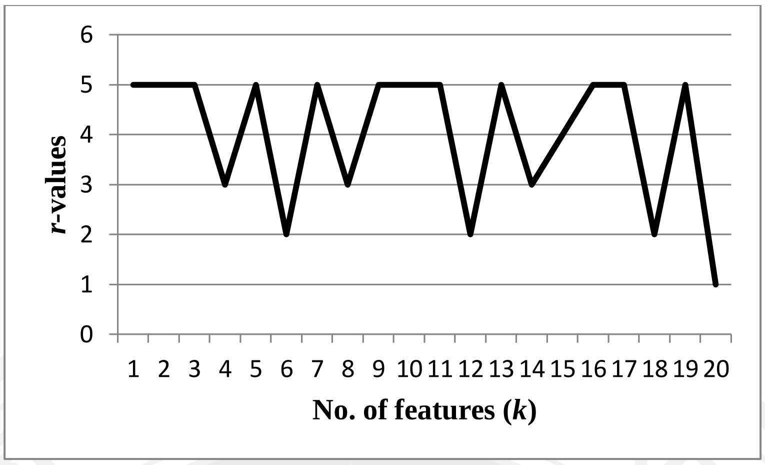


Figure 3.3 Graph of the *r*-values from the Ionosphere Dataset

3.2.2 An Example using Wine Dataset

To demonstrate the OFMB algorithm, we selected the Wine dataset from the UCI repository based on the KNN classifier. First, assume we have a dataset $Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ with 13 features; the required subset size (d) is 20 features, and we assign $r_{max} = 5$. Since the Wine dataset contains only 13 features, we need to process until $d = 13$, and we have $z = \{0, 0, 0, 0, 0, 2, 0, 1, 2, 0, 0, 0, 0\}$.

3.2.2.1 Feature Inclusion

At the beginning, assume we apply SFS for the first 3 features, thus for $k = 1$, 2 and 3 we have $X_1 = \{6\}$, $X_2 = \{6, 10\}$ and $X_3 = \{6, 10, 2\}$ respectively. Now, the current subset of $k = 3$ is $X_3 = \{6, 10, 2\}$. This subset is the best 3-subset that has been found so far.

3.2.2.2 Feature Improvement

Assume we continue the process up to $k = 4$. We have $X_4 = \{6, 10, 2, 7\}$ with 90.09% classification accuracy. Remove one feature except $x_4 = 7$ and we have {6, 10, 7}, {6, 2, 7} and {10, 2, 7}. Then, select one feature from the remaining set that produces the best J value with those 3 subsets. Now we have new subsets of size 4 for consideration. After calculation we find that $J(\{10, 6, 7, 9\})$ produces the highest J value with 92.84% accuracy. Therefore, replace {6, 10, 2, 7} with {10, 6, 7, 9} as the best subset of size 4 that has been found so far. Repeat the same process for $X_4 = \{10, 6, 7, 9\}$ and we cannot find any better subset, thus we continue to the next step with $X_4 = \{10, 6, 7, 9\}$. The next step will be an optimization of this solution.

3.2.2.3 Multi-level Backward Selection

Assume we continue the process until we reach k = 9 and we have X_9 = {0, 1, 2, 5, 6, 9, 10, 7, 8} with 92.25% accuracy. After the feature improvement step we have X_9 = {0, 1, 2, 5, 6, 7, 8, 9, 11} with 92.82% accuracy that is the best 9-subset that has been found so far. For Multi-level Backward Selection, starting with s = 1, remove one feature to find a better 8-subset. Now we consider only subsets containing the feature x_9 = {8}, which are {0, 1, 2, 5, 6, 7, 8, 11}, {0, 1, 2, 6, 7, 8, 9, 11}, {0, 1, 2, 5, 6, 7, 8, 9}, {1, 2, 5, 6, 7, 8, 9, 11}, {0, 1, 2, 5, 7, 8, 9, 11}, {0, 1, 2, 5, 6, 8, 9, 11}, {0, 2, 5, 6, 7, 8, 9, 11}, {0, 1, 5, 6, 7, 8, 9, 11}. We calculate the J values for all the combinations of 8-subset but cannot find a better 8-subset. The process continues to the next inner loop for s = 2. Remove two features from X_9 = {0, 1, 2, 5, 6, 7, 8, 9, 11} and we have {0, 2, 5, 6, 8, 9, 11}, {2, 5, 6, 7, 8, 9, 11}, {0, 2, 5, 7, 8, 9, 11},…, {0, 5, 6, 7, 8, 9, 11} for 28 subsets of size 7 to be considered. The calculation has shown no better result, thus continue to the next inner loop for s = 3. Remove three features from X_9 = {0, 1, 2, 5, 6, 7, 8, 9, 11} and we have {1, 2, 6, 7, 8, 11}, {0, 1, 2, 6, 8, 11}, {1, 2, 5, 6, 8, 11},…, {2, 5, 6, 7, 8, 11}. There are 56 subsets of size 6 to be considered. At this point, we can find a better 6-subset, which is {0, 6, 7, 8, 9, 11} with 93.36% accuracy. Replace X_6 with {0, 6, 7, 8, 9, 11} as the best 6-subset that has been found so far. A subset X_6 now has the highest accuracy, which cannot be found by other sequential searching techniques.

3.2.2.4 Compute r-value

An adaptive determination of r is applied to find the value of r for the next iteration. There are two input variables: one is the maximum value of r (r_{max}), which is 5 for this particular example. The other one is z, which has recently been acquired from the multi-level backward selection step. If k = 6, the value of z would be 2. Apply an adaptive determination of r that matches with the first condition, which is ‘If $z < r_{max}$, let $r = r_{max} – z$’. Thus we have r = 5 – 2 = 3. Now, r = 3 will be applied to the algorithm for k = 7. The value of r can vary from 1 to 5 depending on the value of z. Therefore, r changes adaptively in different iterations.

3.2.2.5 Termination Condition

The OFMB algorithm processes sequentially until the subset size (k) reaches the required subset size (d). The best of all feature subsets are copied into X_k and then the program is terminated. This method applies the idea of adaptive search in order to explore the potential subset thoroughly, in other words, it provides a better chance of finding the optimal solution via a more detailed search by adjusting the generalization limit adaptively.

The OFMB algorithm gives a chance to explore smaller subsets similar to the backtracking step in SFFS but it also takes a look at some more subsets that have not come across before. There are possibilities that we may find a better subset that increases the classification accuracy from what we previously received from the OLFI step. The results from OFMB lead to even closer to the optimal solutions.

This proposed method is classified as the sequential floating selection methods which are considered to be data-dependent with the other data and as a result of their heuristic behavior they cannot jump across to a new solution regardless of the previous searching steps. Even though these searching techniques cannot guarantee the optimal solution since they are focusing on the suboptimal solutions but they provide similar results with those exhaustive searches in most cases (Somol et al., 1999).

3.3 The Classifiers

There are several classifiers used for data mining and machine learning. One of the simplest and most popular classifiers is K-Nearest-Neighbors (KNN). The other ones that we interest in for our application include the Naive Bayes and Decision Tree classifiers. To compare the performance of our algorithms against other searching algorithms, we calculate the criterion function (J) for each subset that is chosen by each algorithm for the different number of selected features subset sized (k).

3.3.1 K-Nearest-Neighbors (KNN)

Due to the robustness and versatility, KNN is often used in various applications such as economic forecasting, data compression and genetics which can outperform other more powerful classifiers. KNN falls in the supervised learning family and is selected to calculate the classification accuracy in both IFFS and SDFFS. Therefore, we have decided to use KNN to compare our performance on different sequential floating feature selection algorithms. We have applied 5-fold cross-validation for all tested datasets.

The algorithm of KNN is described below:

1) Load the training and test data
2) Choose the value of *k*
3) For each point in the test data
   (1) find the distance to all training data points
   (2) store the distances in a list and sort it
   (3) choose the first *k* points
   (4) assign a class to the test point based on the majority of classes present in the chosen points

There are many different ways to compute distances. The two popular ones are Euclidean distance and Cosine similarity. Euclidean distance is probably seemed to be more familiar with and is the default measurement in the python library for the KNN classifier.

3.3.2 Naïve Bayes (NB)

The NB Classifier is a classification model based on probability theory by applying Bayes theorem. It was widely used in machine learning research since the 1950s because of its effectiveness and ease of implementation without complicated iterative parameter estimation. Frequently, the NB classifier outperforms those more sophisticated classification methods. Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. See the equation below:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \ldots \times P(x_n|c) \times P(c)$$

where

$P(c|x)$ is the posterior probability of class (target) given predictor (feature or attribute)

$P(c)$ is the prior probability of class

$P(x|c)$ is the likelihood which is the probability of predictor given class

$P(x)$ is the prior probability of predictor

In our experiments, regarding the Scikit learn which is a python library, we have decided to use the Gaussian Naïve Bayes for the classification task due to the simplicity and it is the most popular one.

3.3.3 Decision Tree (DT)

DT algorithm belongs to the family of supervised learning algorithms. It can be used for solving classification and regression problems. The goal of DT is to create a training model that can be used to predict the class or value of the target variable by learning simple decision rules inferred from the training data. There are two types of DT based on the target variable. The first one is the categorical variable DT that has a categorical target variable. The second one is the continuous variable DT that has a continuous target variable. DT is an effective machine learning model. The model is defined by a series of questions that lead to a class label when applying to any observation. In the trees, each leaf node represents class labels while the branches represent conjunctions of features leading to class labels.

One example of the application of DT is for a bank to decide whether or not to offer someone a loan by asking a series of questions to figure out if it is safe to allow a load to an individual. These questions are like the income of a person, how long they occupy this job, what is their credit card payment behavior and so on. The final decision can be either yes or no which represent the class label for the leaf nodes. DT is also our selected classifier for the experiments.

3.4 Datasets

The datasets used in the experiments are 14 standard datasets with various sizes from the UCI machine learning repository (Dheeru & Efi, 2017). Some of these datasets are also used in earlier works for sequential floating search. Detail of all data sets is shown in table 3.2.

Table 3.2 Datasets Used in the Experiments

Name	Feature Type	No. of instances	No. of features	No. of classes
Wine	Integer, Real	178	13	3
Thoracic Surgery	Integer, Real	470	17	2
Online Shoppers	Integer, Real	12330	17	2
Lymphography	Categorical	148	18	2
Image Segmentation	Real	2310	19	7
Crowdsourced	Real	10546	29	6
Breast Cancer	Real	569	32	2
Ionosphere	Integer, Real	351	34	2
Soybean	Categorical	307	35	15
Waveform 2	Real	5000	40	3
Spectf Heart	Integer	267	44	2
Spambase	Integer, Real	4601	57	2
Sonar	Real	208	60	2
Urban Land Cover	Real	675	147	9

3.5 Experimental Setup

To compare our method with other algorithms we developed an experimental environment similar to the previous works. The performance of the feature selection methods is usually evaluated by a machine learning model. Some popular models include Naïve Bayes, C4.5, SVM, Decision Tree and K-means clustering. One of the most popular classifiers is K-Nearest-Neighbors (KNN). We applied KNN to compare our performance on different algorithms based on 5-fold cross-validation. The other two classifiers that also used in these experiments are NB and DT. Data normalization is preferred as a preprocessing step. We selected Python as the programming language, using the Jupyter notebook editor for program development. We randomly selected some instances for a large dataset and also eliminated some missing values if necessary. We applied the same randomly selected instances to all techniques to ensure that they received the same input.

3.6 Results and Discussion

In this section, we discuss our results for the OFMB algorithm compared with popular suboptimal methods, which are SFS, SFFS and IFFS. This research aimed to increase the classification accuracy rather than reducing the time complexity. Normally, the improvement of the earlier technique requires a more complicated algorithm which also requires higher computing time unavoidably. The size of the dataset does not affect the algorithm. We considered only the first 20 features for all datasets in order to limit the operation cost. There were some datasets with less than 20 features, therefore we considered the whole dataset sizes for these small datasets.

We studied the effectiveness of the proposed sequential feature selection algorithm based on the three classification methods which were KNN, NB and DT on 14 standard UCI machine learning repositories. The performances were evaluated by classification accuracy and the minimum number of selected features that produced the maximum accuracy. The classification accuracy was the first priority for the best performance. If the results on accuracy for different algorithms were equal, then the smaller number of selected features would be in consideration.

Table 3.3 The Comparison of Maximum Accuracy Using KNN

Dataset	Previous Methods (KNN)			Proposed Method (KNN)
	SFS	SFFS	IFFS	OFMB
Wine (13)	92.82 (10)	**93.38 (7)**	**93.38 (7)**	**93.38 (7)**
Thoracic Surgery (17)	84.89 (5)	85.96 (9)	85.96 (10)	**86.96 (10)**
Online Shopper (17)	90.43 (7)	90.59 (7)	**90.67 (5)**	**90.67 (5)**
Lymphography (18)	88.00 (15)	88.76 (13)	90.14 (11)	**90.81 (10)**
Image Segmentation (19)	80.95 (10)	80.95 (7)	81.43 (8)	**81.43 (7)**
Crowdsourced (29)	89.46 (20)	88.98 (20)	90.13 (19)	**90.42 (20)**
Breast Cancer (32)	95.44 (18)	**95.44 (12)**	95.44 (16)	95.44 (13)
Ionosphere (34)	93.45 (5)	94.02 (12)	94.59 (12)	**94.89 (11)**
Soybean (35)	89.1 (18)	90.23 (18)	90.23 (19)	**90.23 (17)**
Waveform 2 (40)	85.22 (14)	**86.8 (18)**	85.39 (13)	86.17 (17)
Spectf Heart (44)	81.65 (11)	**98.33 (9)**	**98.33 (9)**	85.37 (12)
Spambase (57)	90.43 (12)	90.43 (12)	**93.04 (19)**	**93.04 (19)**
Sonar (60)	78.56 (11)	77.44 (6)	80.88 (20)	**81.76 (19)**
Urban land cover (147)	60.49 (9)	60.48 (9)	**61.37 (6)**	**61.37 (6)**

The results in table 3.3 were the comparison of maximum classification accuracy (%) and resulted number of selected features in parenthesis using KNN from different feature selection algorithms where the highest accuracy for each dataset was in bold. It shows that the classification accuracy was noticeably enhanced by the proposed algorithm compared to the previous works using KNN as performance validation method. OFMB had the best performance in the majority of the datasets because it produced either the highest accuracy and/or a lower number of features. With the Wine dataset, OFMB achieved the same optimal solutions as SFFS and IFFS due to the size of the dataset being small. With the Breast Cancer dataset, SFFS was the best method among the other three with the same maximum accuracy, but with a lower number of selected features. SFFS was also the best method for the Waveform 2 dataset. With the Spectf Heart dataset, both SFFS and IFFS produced the best solutions. For the rest of the resulted datasets, the OFMB algorithm showed the best performance among the other techniques. Only for the Online Shopper, Spambase and Urban Land Cover datasets, IFFS had equal solutions to the OFMB algorithm.

Table 3.4 The Comparison of Maximum Accuracy Using NB

Dataset	**Previous Methods (NB)**			**Proposed Method (NB)**
	SFS	**SFFS**	**IFFS**	**OFMB**
Wine (13)	**93.35 (5)**	**93.35 (5)**	**93.35 (5)**	**93.35 (5)**
Thoracic Surgery (17)	85.11 (1)	85.11 (1)	85.11 (1)	**85.32 (5)**
Online Shopper (17)	**90.67 (2)**	**90.67 (2)**	**90.67 (2)**	**90.67 (2)**
Lymphography (18)	86.48 (7)	86.52 (10)	87.33 (9)	**88.05 (8)**
Image Segmentation (19)	81.91 (5)	81.91 (5)	**82.86 (5)**	**82.86 (5)**
Crowdsourced (29)	82.92 (18)	83.01 (16)	**83.4 (19)**	**83.4 (19)**
Breast Cancer (32)	95.44 (8)	95.44 (8)	**96.14 (6)**	**96.14 (6)**
Ionosphere (34)	92.58 (14)	93.44 (11)	93.72 (14)	**93.72 (11)**
Soybean (35)	83.86 (20)	84.61 (15)	**91.73 (12)**	**91.73 (12)**
Waveform 2 (40)	85.2 (18)	85.61 (15)	85.81 (17)	**86 (20)**
Spectf Heart (44)	79.4 (1)	80.12 (6)	79.4 (1)	**80.15 (3)**
Spambase (57)	79.89 (15)	80.65 (18)	81.84 (12)	**82.29 (18)**
Sonar (60)	81.4 (7)	81.4 (7)	**81.45 (13)**	**81.45 (13)**
Urban land cover (147)	71.48 (12)	75.24 (16)	**76.14 (15)**	**76.14 (15)**

The results in table 3.4 also show that the classification accuracy was enhanced by the OFMB algorithm compared to the previous works using the NB classifier. Only the Wine and Online Shopper datasets had equal results for all techniques. Apart from the two datasets mentioned above, IFFS produced the same maximum accuracy as OFMB with the Image Segmentation, Crowdsourced, Breast Cancer, Soybean, Sonar and Urban Land Cover datasets. The rest of the tested datasets provide the best results obtained by the proposed algorithm. Therefore, OFMB had the best performance with all datasets because it produced the highest classification accuracy with the smallest number of selected features equal to or better than the other methods.

Table 3.5 The Comparison of Maximum Accuracy Using DT

Dataset	Previous Methods (DT)			Proposed Method (DT)
	SFS	SFFS	IFFS	OFMB
Wine (13)	91.66 (10)	92.27 (6)	93.95 (5)	**93.98 (8)**
Thoracic Surgery (17)	80.43 (3)	80.64 (6)	80.85 (6)	**81.06 (3)**
Online Shopper (17)	88.81 (2)	89.05 (2)	89.13 (10)	**89.3 (11)**
Lymphography (18)	85.14 (6)	86.05 (10)	**87.29 (5)**	87.29 (6)
Image Segmentation (19)	84.76 (11)	86.19 (12)	85.71 (7)	**88.57 (9)**
Crowdsourced (29)	80.84 (12)	82.16 (13)	82.16 (10)	**83.22 (16)**
Breast Cancer (32)	95.61 (5)	96.32 (6)	**96.67 (8)**	96.5 (14)
Ionosphere (34)	92.3 (20)	92.89 (18)	93.72 (19)	**94.3 (20)**
Soybean (35)	90.22 (20)	91.75 (17)	**92.11 (18)**	91.36 (19)
Waveform 2 (40)	77.61 (18)	**78.39 (16)**	**78.39 (16)**	78 (18)
Spectf Heart (44)	80.55 (10)	80.17 (11)	**86.9 (15)**	83.54 (20)
Spambase (57)	90.32 (20)	90.65 (17)	90.43 (20)	**90.65 (16)**
Sonar (60)	83.27 (17)	**86.6 (13)**	86.18 (19)	84.3 (13)
Urban land cover (147)	78.07 (14)	77.33 (12)	80.42 (20)	**81.31 (20)**

As shown in table 3.5, the majority of the best results obtained by the OFMB algorithm using the DT classifier. SFFS gave the best solution only for Waveform 2 and Sonar datasets whereas SFS did not perform well for all datasets. IFFS produced the best accuracy with the smallest subset sizes for Lymphography, Breast Cancer, Soybean, Waveform 2 and Spectf Heart datasets. Even though IFFS gave the highest accuracy for the Lymphography dataset, but this solution was also equal to OFMB with only one feature less than from the IFFS for the selected subset. However, OFMB provided the maximum classification accuracy for the other eight datasets in the result table.

Table 3.6 The Comparison of Maximum Accuracy From the Three Different Classifiers for the OFMB Algorithm

Dataset	Proposed Method (OFMB)		
	KNN	NB	DT
Wine (13)	93.38 (7)	93.35 (5)	**93.98 (8)**
Thoracic Surgery (17)	**86.96 (10)**	85.32 (5)	81.06 (3)
Online Shopper (17)	90.67 (5)	**90.67 (2)**	89.3 (11)
Lymphography (18)	**90.81 (10)**	88.05 (8)	87.29 (6)
Image Segmentation (19)	81.43 (7)	82.86 (5)	**88.57 (9)**
Crowdsourced (29)	**90.42 (20)**	83.4 (19)	83.22 (16)
Breast Cancer (32)	95.44 (13)	96.14 (6)	**96.5 (14)**
Ionosphere (34)	**94.89 (11)**	93.72 (11)	94.3 (20)
Soybean (35)	90.23 (17)	**91.73 (12)**	91.36 (19)
Waveform 2 (40)	**86.17 (17)**	86 (20)	78 (18)
Spectf Heart (44)	**85.37 (12)**	80.15 (3)	83.54 (20)
Spambase (57)	**93.04 (19)**	82.29 (18)	90.65 (16)
Sonar (60)	**81.76 (19)**	81.45 (13)	84.3 (13)
Urban land cover (147)	61.37 (6)	76.14 (15)	**81.31 (20)**

Table 3.6 shows the comparison of the results from the OFMB algorithm using different criterion functions. The performances were validated by KNN, NB and DT classifiers. The majority of the best performances were from KNN with eight sample datasets, whereas NB provided the best results with only two datasets which are Online Shopper and Soybean. The other four tested datasets with maximum classification accuracy were from the DT classifier. Different criterion functions yielded different results because each function has a unique character and we can see that KNN as the criterion function yielded the best result followed by DT and NB respectively. Thus, KNN is the most preferable classifier for getting the best solutions since it provides more opportunity to get the highest accuracy.

The proposed algorithm based on a sequential feature selection algorithm produced effective feature subsets with higher classification accuracy with several different datasets. Our proposed algorithm can extract a more relevant and effective feature subset from the source dataset using multi-level backward tracking selection with an adaptive generalization level technique. From the experiments, the maximum accuracies with the smallest subsets produced by our proposed method on most of the tested datasets. This improvement was the result of the multi-level backwards tracking technique that leads to a more thorough search on the smaller feature subsets. Some smaller subsets with higher accuracy were discovered by our in-depth searching method.

In this chapter, we proposed an algorithm called One-level Forward Multi-level Backward Selection (OFMB) algorithm. We aimed to develop a feature selection method that surpasses previous works in terms of accuracy. We proposed a feature selection algorithm based on the sequential searching technique by improving the performance of the SFFS algorithm. Incorporating a feature improvement step in our method produces similar results with the IFFS algorithm. The addition of multi-level backtracking was done to discover relevant subsets that cannot be discovered by SFFS or IFFS. The algorithm employs an adaptive generalization limit to indicate the level of backward searching. A higher limit leads to a better chance of finding a better subset. In the experiments, we compared our method with SFS, SFFS and IFFS. Results on the 14 standard datasets showed that OFMB performed better than the other suboptimal sequential feature selection algorithms for most of the tested datasets. Some results from previous methods might be better than from our proposed method due to the relationship between smaller subsets with larger subsets. Higher accuracy in the smaller subset might lead to a trap in the local optimum solution. Therefore, while the subset size increased, there was a chance that the searching process could not gain maximum accuracy. In the next chapter, we proposed another sequential searching technique that focused on looking ahead in the forward direction rather than looking backwards by applying adaptive generalization limit and also introduced two new methods to calculate the value of the generalization limit.

Related to the time complexity of the OFMB algorithm, we can derive from the two main steps. Firstly, a feature improvement step where it computes n subsets after removing one feature from the selected subset and add one feature from the remaining set. This operation continues no greater than n loop, therefore the feature improvement step requires no greater than n^2 for the time complexity. Secondly, a multi-level backward selection step processes up to r_{max} for the inner loop. The selected subsets of size k are considered and s is a constant from 1 to r_{max}. For each inner loop, the total subsets that need to be calculated are $C(k, s)$ minus the subsets that do not contain the newly selected feature. This operation repeats no greater than n times, which is the total number of features in the dataset. Therefore, the number of subsets that need to be evaluated for the inner loop can be described by an expression below:

$$\frac{k!}{(k-s)!s!} - \frac{(k-1)!}{((k-1)-(k-s))!(k-s)!},$$

where k is the number of features in the subset and s is the generalization level for that particular iteration.

For $s = 1$, the number of subsets that need to evaluate is;

$$\frac{k!}{(k-1)!1!} - \frac{(k-1)!}{((k-1)-(k-1))!(k-1)!} = k - 1 \approx k$$

For $s = 2$, the number of subsets that need to evaluate is;

$$\frac{k!}{(k-2)!2!} - \frac{(k-1)!}{((k-1)-(k-2))!(k-2)!} = \frac{k(k-1)}{2} - (k-1) \approx k^2$$

For $s = 3$, the number of subsets that need to evaluate is;

$$\frac{k!}{(k-3)!3!} - \frac{(k-1)!}{((k-1)-(k-3))!(k-3)!} = \frac{k(k-1)(k-2)}{6} - \frac{(k-1)(k-2)}{2} \approx k^3$$

Repeat this calculation for $s = 4$ and 5, so we have the computing time of k^4 and k^5 respectively. We can see that the time complexity for this second part is $n\sum_{s=1}^{5} k^s$. Other steps are constant time so they can be ignored. Combine the two steps we have $n^2 + n\sum_{s=1}^{5} k^s$ time complexity for the OFMB algorithm. We can conclude that OFMB requires higher computational time than IFFS for which IFFS bounded by $O(n^2)$, moreover, SFS and SFFS are bounded by $O(n)$.

CHAPTER 4

MULTI-LEVEL FORWARD INCLUSION (MLFI)

The Multi-level Forward Inclusion (MLFI) algorithm is similar to the One-Level Forward Inclusion (OLFI) with the addition of an adaptive floating search concept. Instead of adding one feature, we use SFS to add more than one feature but to some specified point called the generalization limit, which is denoted by *r*. The value of *r* is varying using the condition we proposed in section 3.3.3. From the *r* value calculation method, we can add features ranging from 1 to r_{max}, where r_{max} is the maximum value of *r*, which is needed to be defined. The higher value of *r* leads to more computational time, therefore we need to keep *r* as a small number such as 4 or 5. The level of *r* in each loop uses *s* as a variable. The level *s* is similar to the level *o* in ASFFS where *s* is determined dynamically according to the previous searching situation. This method considers a wider range of feature subset for consideration. As a result, there are higher chances to improve the current feature subset. Next, we will discuss the MLFI algorithm in more detail.

In our study, we focused on the wrapper approach based on a sequential selection algorithm. It used the result of a data mining algorithm to determine the goodness of a given feature subset. During the search process, the space of possible feature subsets is defined to generate and evaluate features until we get the satisfied subset. For the sequential floating search methods, the number of features dynamically increases and decreases until we reached the desired target. The variables allow floating forward or backwards so that they can be flexibly changed without pre-setting any parameters. For this reason, a floating search is possible to be trapped at a local optimum since the best *k*-subset does not necessarily contain the best (*k*–1)-subset. Therefore, we present an alternative improvement to the floating search algorithm to remove some of its drawbacks and try to find a solution as much closer to the optimal solution as possible.

Our study attempted to explore a new sequential feature selection algorithm that produced better results than the earlier works. We proposed a Multi-level Forward Inclusion (MLFI) algorithm where the idea is to remove the backtracking step and modify the 'replacing a weak feature' step from the IFFS method with an addition of the adaptive floating search technique. Instead of adding one feature, we can add more than one feature to a defined point using the generalization limit (r). The appropriate value of r depends on the specified conditions described in section 4.1. MLFI considers a wider range of features that lead to a more thorough search. As a result, it has higher chances to maximize the current feature subset according to the classification accuracy. The MLFI algorithm can be described below with the flowchart in figure 4.1 follow by the pseudo code.

This is a description of the MLFI Algorithm:

Step 1: Apply SFS to select one feature from the remaining feature set. Add this feature to the selected feature subset. Continue step 2 with the feature subset X_k where $k = k + 1$.

Step 2: From the selected feature subset (X_k), remove s features iteratively from 1 to r. Search for the best (k–s)-subset. If there is a better subset ($X_{k\text{–s}}$), replace the previous subset with the new subset for $X_{k\text{–s}}$. Repeat steps 1 and 2 until $s > r$, then continue step 3.

Step 3: From the selected feature subset size k, remove 1 feature iteratively then we have X_{k-1}, and use SFS to select a new feature from the remaining feature set ($Y - X_{k-1}$) to add to each feature subset. Then calculate whether there is an improvement. If there is an improvement, replace that previous feature subset with the newly selected feature subset and repeat step 3. Otherwise, continue step 4 with the feature subset X_k.

Step 4: Compute the r-value, then continue step 5.

Step 5: Continue steps 1, 2, 3 and 4 until d features are selected.

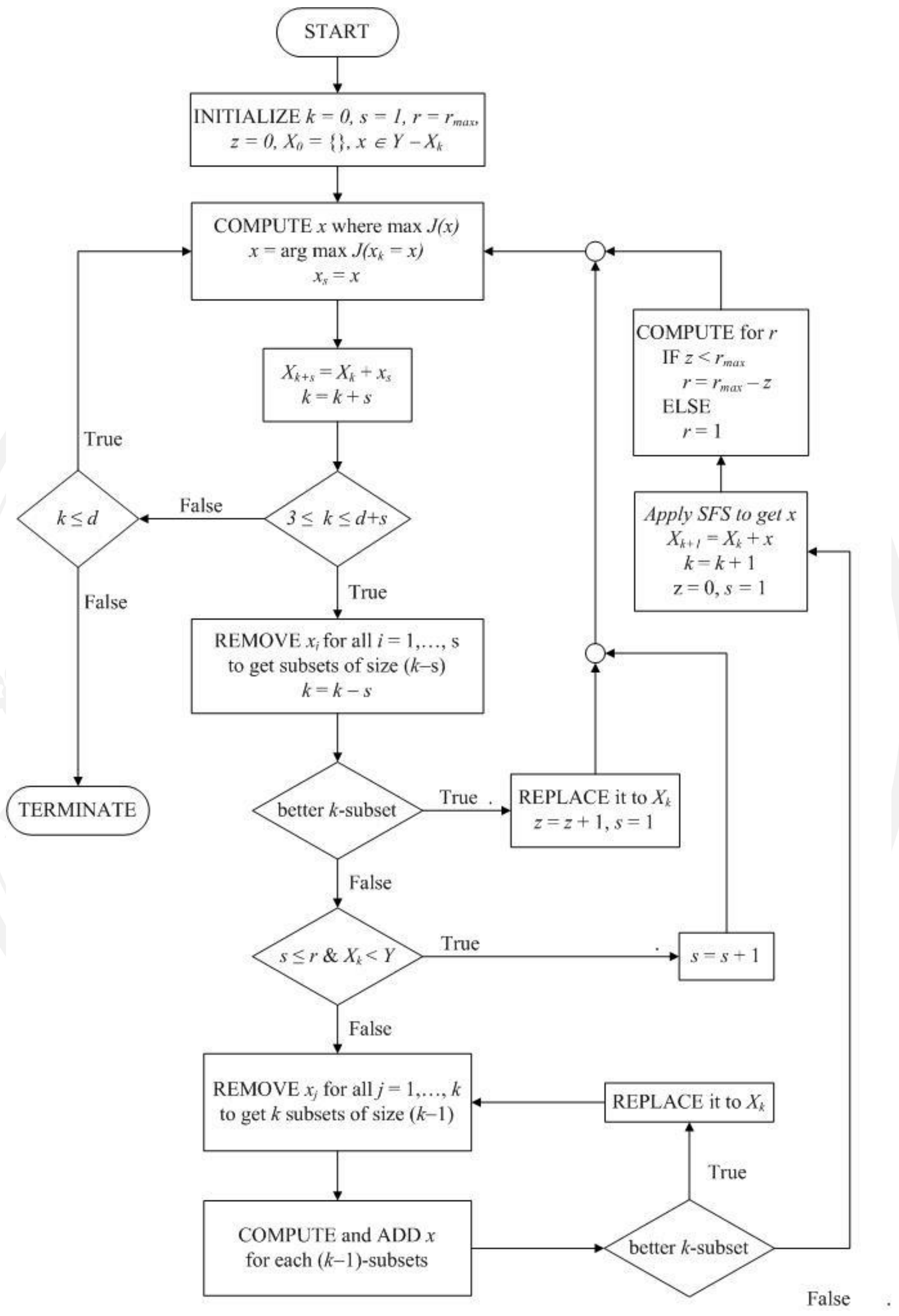


Figure 4.1 Structure of the MLFI Algorithm

Algorithm: Multi-level Forward Inclusion (MLFI)

Input: A set of features $Y = \{y_1, y_2,\ldots, y_D\}$, where D is the number of input dimensions; J is a criterion function; d is the required subset size; r is the generalization level which is limited by r_{max}.

Output: A feature subset $X_k = \{x_j \mid j = 1, 2,\ldots, k; x_j \in Y\}$, where $k = (0, 1, 2,\ldots, d)$.

Initialize: Initialize $X_0 = \{\}$; $k = 0$; $s = 1$; $r = r_{max}$; $z = 0$.

```
(1) Feature Inclusion
    #Find the best feature and update $X_k$
        $x^+ = \arg\max J(x_k = x)$, where $x \in Y - X_k$
        $X_{k+1} = X_k + x^+$
        $k = k + 1$
        $\max(X_k) = X_{k+1}$
(2) Multi-level Forward Inclusion
    #Searching for better $k$-subset by multi-level forward searching step
        Repeat
                $x_s$ in $X_k$ : #where  $s = 1,\ldots, r$
                $X_{k-s} = X_k - x_s$
                If $J(X_{k-s}) > J(\max(X_{k-s}))$:
                        $\max(X_{k-s}) = X_{k-s}$
                        $z = z + 1$
                Else
                        $s = s + 1$
                        Go to step 1
        Until   $s > r$
(3) Feature Replacement
    #Replace a weak feature by removing one feature and adding one feature that
      maximizes the criterion function
        Repeat
                For $x_j$ in $X_k$ : #where  $j = 1, 2,\ldots, k$
                        $X_{k-1} = X_k - x_i$
```

```
                    For $x_i$ in $Y - X_{k-1}$ : #where  $i = 1, 2,\ldots, d - (k-1)$
                        $x_i = \arg\max J(x_i)$
                        If $J(X_{k-1} + x_i) > J(X_k)$:
                            $X_k = X_{k-1} + x_i$
                            $\max(X_k) = X_k$
        Until  $J(X_{k-1} + x_i) \leq J(X_k)$
        $x^+ = \arg\max J(x_k = x)$, where $x \in Y - X_k$
        $X_{k+1} = X_k + x^+$
        $k = k + 1$
        $\max(X_k) = X_{k+1}$
    (4) Compute the r-value
        If $z < r_{max}$ :
            $r = r_{max} - z$
        Else :
            $r = 1$
        $z = 0$
        $s = 1$
    (5) Termination Condition
      #Terminate when $k > d$
        If $k \leq d$
            Go to step 1
        $X_k = \max(X_k)$ #for all $k$
        Return the best individual subset $X_k$
```

4.1 Computation of the *r*-value

The generalization limit (r) needs to be carefully specified because a larger value of r results in a more thorough search and also increases the time complexity. We introduced a user-defined parametric limit r_{max} to restricting the maximum generalization level. This number can be any integer depending on how deep we need to search through. Normally, r_{max} is a small integer while the suggestion of r_{max} from ASFFS is 4. In our experiments, we assigned the value of r_{max} to be 5 for all tested datasets. The level s is determined dynamically according to the r calculation technique we proposed.

4.1.1 Method I

The generalization limit can change adaptively depending on the number of times we found a better k-subset. In our experiments, we selected the first 20 features from the whole dataset. MLFI considers a wider range of features that lead to a thorough search. Seeing that, there are higher chances to improve the current feature subset. We applied the same calculation method for the r-value as in chapter 3 and called it method I.

Assume $r_{max} = 5$, thus $1 \le r \le 5$. Let z be the number of times the algorithm has found a better subset for that particular iteration. Thus, z is related to r by $r_{max} - z$. Adaptive determination of r is defined as follows:

1) If $z < r_{max}$, $r = r_{max} - z$
2) Else , $r = 1$

4.1.2 Method II

The generalization limit increases step by step starting from 1 to r_{max} along with the feature subset sizes. The calculation defined by the equation below:

1) $r = \lceil k / \lceil d/r_{max} \rceil \rceil$

For example, if we let $d = 20$ and $r_{max} = 5$, thus $r = \lceil k/4 \rceil$. Now, the value for r from the Ionosphere dataset shows in table 4.1 and figure 4.2 for all subset size k, where $1 \le k \le 20$.

Table 4.1 The *r*-values and the *z*-values from the Ionosphere Dataset

No. of features (*k*)	***r*-values**
1	1
2	1
3	1
4	1
5	2
6	2
7	2
8	2
9	3
10	3
11	3
12	3
13	4
14	4
15	4
16	4
17	5
18	5
19	5
20	5

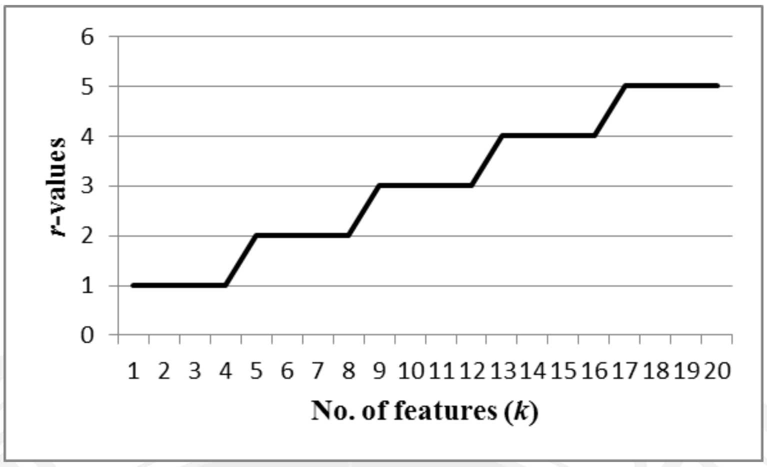


Figure 4.2 *r*-values for Method II

4.1.3 Method III

The idea of the third method is to avoid the solution being trapped by local optima since MLFI starting with a similar subset to IFFS for the first half of the *k*-subsets where we have $r = 1$. For the second half, we increase the *r*-value until it reaches the maximum generalization limit. Then the process continues by applying r_{max} through the end of the required subset sizes. The calculation of method III is described below:

1) If $k <= \lfloor d / 2 \rfloor$, $r = 1$
2) Else if $\lfloor d / 2 \rfloor < k < \lfloor d / 2 \rfloor + r_{max}$, $r = r + 1$
3) Else , $r = r_{max}$

For example, let assume $d = 20$ and $r_{max} = 5$, then we have; if $k <= 10$: $r = 1$, elif $10 < k < 15$: $r = r + 1$, else: $r = 5$. Table 4.2 and figure 4.3 show an example of *r*-value for method III.

Table 4.2 The *r*-values and the *z*-values from the Ionosphere Dataset

No. of features (*k*)	***r*-values**
1	1
2	1

No. of features (*k*)	*r*-values
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	2
12	3
13	4
14	5
15	5
16	5
17	5
18	5
19	5
20	5

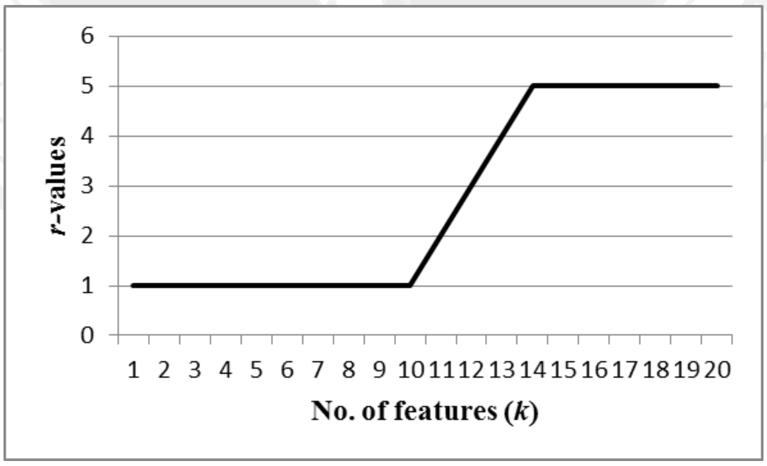


Figure 4.3 *r*-values for Method III

4.2 An Example using Wine Dataset

To demonstrate the MLFI algorithm, we selected the Wine dataset from the UCI repository based on the KNN classifier. First, assume we have a dataset $Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ with 13 features; the required subset size (d) is 20 features, and we assign $r_{max} = 5$. Since the Wine dataset contains only 13 features, we need to process until $d = 13$, and we have $z = \{0, 0, 0, 0, 0, 2, 0, 1, 2, 0, 0, 0, 0\}$.

4.2.1 Feature Inclusion

At the beginning, assume we apply SFS for the first 3 features, thus for $k = 1$, 2 and 3 we have $X_1 = \{6\}$, $X_2 = \{6, 10\}$ and $X_3 = \{6, 10, 2\}$ respectively. Now, the current subset of $k = 3$ is $X_3 = \{6, 10, 2\}$. This subset is the best 3-subset that has been found so far.

4.2.2 Multi-level Forward Inclusion

Assume we continue the process up to $k = 4$, we have $X_4 = \{6, 10, 2, 7\}$ with 90.09% classification accuracy. For $k = 4$, $s = 1$, remove one feature except $x_4 = 7$ to find a better 3-subset, now we have ({10, 6, 7}, {2, 6, 7} and {2, 10, 7}). After calculate the criterion function values (J), we have found that J({10, 6, 7}), J({2, 6, 7}) and J({2, 10, 7}) are not greater than J({6, 10, 2}), therefore we continue to the next inner loop for $s = 2$ with J({6, 10, 2}) = 90.04% as the best 3-subset that has been found so far.

For $k = 5$, $s = 2$, we add the forth and the fifth features into X_5 we get {6, 10, 2, 7, 5}. Remove two features to find a better 3-subset, now consider only subsets containing the feature $x_5 = \{5\}$ which are ({5, 6, 7}, {2, 10, 5}, {10, 5, 6}, {10, 5, 7}, {2, 5, 6}, {2, 5, 7}). We calculate the J values for all the combinations of 3-subset by removing any two features except x_5. We cannot find a better 3-subset, the process continues to the next inner loop for $s = 3$ with the same J({6, 10, 2}) as the best 3-subset that has been found so far.

For $k = 6$, $s = 3$, we add the forth, the fifth and the sixth features into X_6 we get {6, 10, 2, 7, 5, 11}. Remove three features to find a better 3-subset, now consider only subsets containing the feature $x_6 = \{11\}$ which are ({11, 6, 7}, {10, 11, 5}, {10, 11, 6}, {10, 11, 7}, {11, 5, 7}, {2, 11, 5}, {11, 5, 6}, {2, 11, 6}, {11, 10, 2}, {2, 11, 7}). We calculate the J values for all the combinations of 3-subset by removing any three features except x_6. From the current feature subsets, we cannot find a better 3-subset, continue to the next inner loop for s = 4 with the same J({6, 10, 2}) as the best 3-subset that has been found so far.

For k = 7, s = 4, we add the forth, the fifth, the sixth, and the seventh features into X_7 we get {6, 10, 2, 7, 5, 11, 0}. Remove four features to find a better 3-subset, now consider only subsets containing the feature x_7 = {0} which are ({0, 5, 6}, {0, 10, 7}, {0, 5, 7}, {0, 10, 6}, {0, 11, 6}, {0, 11, 7}, {0, 10, 11}, {0, 6, 7}, {0, 11, 5}, {0, 2, 5}, {0, 2, 7}, {0, 2, 6}, {0, 2, 10}, {0, 2, 11}, {0, 10, 2}, {0, 10, 5}). We calculate the J values for all the combinations of 3-subset by removing any four features except x_7. From the current feature subsets, we can find a better 3-subset, which is J({0, 11, 6}) = 90.06%. Therefore, continue the next iteration for k = 4 and s = 1 with J({0, 11, 6}) as the best 3-subset that has been found so far.

The process repeats by exploring further until an improvement cannot be made. After the adaptive search meets the condition $s > r$, the best 3-subset that we have found so far would be X_3 = {0, 11, 6}. Continue the next step in order to maximize this solution.

4.2.3 Feature Replacement

We are now continue onto the feature replacement step by removing one feature iteratively we have {11, 6}, {0, 6} and {0, 11}, then select one feature from the remaining set using SFS we have {11, 6, 0}, {0, 6, 9} and {0, 11, 6}. The J values for each subset are 90.06%, 91.17% and 90.06% respectively. At this point, we have found J({0, 6, 9}) = 91.17% as the new best 3-subset. Replace {0, 11, 6} with {0, 6, 9} thus X_3 = {0, 6, 9} is the best subset of size 3 features that we have found so far. Continue the next outer loop with k = 4 for J({0, 6, 9, 11}) = 92.27% as the best 4-subset. We can see that the number of features in the subset either increases or remains the same throughout the whole process. When the process continues up to 5-subset we get X_5 = {0, 6, 8, 9, 11} which is the highest accuracy that cannot be found by other sequential searching techniques.

4.2.4 Compute the *r*-value

An adaptive determination of *r* is applied to find the value of *r* for the next iteration. In the method I, there are two input variables, one is the maximum value of *r* (r_{max}), which is 5. The other one is *z*, which is recently acquired from the multi-level forward inclusion step. The value of *r* can be varying from 1 to 5 depending on the value of *z*. Therefore, *r* is changing adaptively on different iteration. Method II and III do not require a variable *z*, while they can lead to better accuracy than method I for some datasets.

4.2.5 Termination Condition

The MLFI algorithm processes sequentially until the selected subset size (*k*) reaches the required subset size (*d*). The best of all feature subsets are copied into X_k and then the program is terminated. This method applies the idea of adaptive search in order to explore the potential subset thoroughly, in other words, it provides a better chance to find the optimal solution via a more detailed search by adjusting the generalization limit adaptively.

4.3 The Classifiers

The classifiers used in the experiments were KNN, NB and DT, which was the same as in chapter 3. To compare the performance of our algorithm against the other sequential searching algorithms, we calculated the criterion function (*J*) according to those classifiers for every subset sizes from the different algorithms.

4.4 Datasets

The datasets used in the experiments were the 14 standard datasets with various sizes from the UCI machine learning repository. Some of these data sets were also used in the earlier works, therefore we can compare our results with the other methods. The detail of all datasets shows in Table 4.3.

Table 4.3 Datasets Used in the Experiments

Name	Feature Type	No. of instances	No. of features	No. of classes
Wine	Integer, Real	178	13	3
Thoracic Surgery	Integer, Real	470	17	2
Online Shoppers	Integer, Real	12330	17	2
Lymphography	Categorical	148	18	2
Image Segmentation	Real	2310	19	7
Crowdsourced	Real	10546	29	6
Breast Cancer	Real	569	32	2
Ionosphere	Integer, Real	351	34	2
Soybean	Categorical	307	35	15
Waveform 2	Real	5000	40	3
Spectf Heart	Integer	267	44	2
Spambase	Integer, Real	4601	57	2
Sonar	Real	208	60	2
Urban Land Cover	Real	675	147	9

4.5 Experimental Setup

To compare our proposed method with other algorithms, we developed an experimental environment similar to the previous works and in chapter 3. We used KNN, NB and DT classifiers to compare our performances on different algorithms based on 5-fold cross-validation. We randomly selected some instances for a large dataset and also eliminated some missing values if necessary. We applied the same randomly selected instances to all techniques to ensure that they received the same input.

4.6 Results and Discussion

In this section, we discussed our results on the MLFI algorithm compared with popular suboptimal methods, which were SFS, SFFS and IFFS. The MLFI algorithm was also aiming to increase the classification accuracy rather than to reduce the time complexity. We considered only the first 20 features for all datasets to limit the operation cost. Datasets with less than 20 features were considered for the whole dataset sizes.

We studied the effectiveness of the proposed sequential feature selection algorithm based on the three classification methods, which were KNN, NB and DT on 14 standard UCI machine learning repositories. We evaluated their performances by classification accuracy and the minimum number of selected features that produced the maximum accuracy. To compare our method with the previous method we used classification accuracy as the first priority for the best performance. If the results on accuracy for different algorithms are equal, then the smallest number of selected features was considered. In this chapter, we introduced three methods for the *r*-value calculation. The results of MLFI were selected from the *r*-value calculation methods that produced the best performance among the three methods.

The results in Table 4.4 was the comparison of maximum classification accuracy (%) and resulted number of selected features in parenthesis using KNN from different feature selection algorithms where the highest accuracy for each dataset was in the bold font. The classification accuracy from the MLFI technique produced the best solutions for most of the tested datasets when compared with the previous works using KNN as a performance validation method. SFS could not produce the highest solution at all. The results from MLFI were either the highest in accuracy and/or had the lowest number of features in the subset. For the Wine dataset, MLFI achieved the same optimal solutions as SFFS and IFFS due to the size of the dataset. In the Waveform 2 dataset, SFFS was the best method among the other three algorithms. IFFS had the best performance on the Spectf Heart dataset with equal solutions to the

SFFS method. Apart from Waveform 2 and Spectf Heart datasets, MLFI performed the best for all other datasets. There were only Wine, Image Segmentation, Spambase and Urban Land Cover datasets where IFFS produced maximum accuracy equal to the MILF algorithm.

Table 4.4 The Comparison of Maximum Accuracy Using KNN

Dataset	**Previous Methods (KNN)**			**Proposed Method (KNN)**
	SFS	**SFFS**	**IFFS**	**MLFI**
Wine (13)	92.82 (10)	**93.38 (7)**	**93.38 (7)**	**93.38 (7)**
Thoracic Surgery (17)	84.89 (5)	**85.96 (9)**	85.96 (10)	**85.96 (9)**
Online Shopper (17)	90.43 (7)	90.59 (7)	90.67 (5)	**90.67 (4)**
Lymphography (18)	88.00 (15)	88.76 (13)	90.14 (11)	**90.81 (10)**
Image Segmentation (19)	80.95 (10)	80.95 (7)	**81.43 (8)**	**81.43(8)**
Crowdsourced (29)	89.46 (20)	88.98 (20)	90.13 (19)	**90.42 (20)**
Breast Cancer (32)	95.44 (18)	95.44 (12)	95.44 (16)	**95.44 (10)**
Ionosphere (34)	93.45 (5)	94.02 (12)	94.59 (12)	**94.87 (8)**
Soybean (35)	89.1 (18)	90.23 (18)	90.23 (19)	**91.73 (20)**
Waveform 2 (40)	85.22 (14)	**86.8 (18)**	85.39 (13)	86.38 (18)
Spectf Heart (44)	81.65 (11)	**98.33 (9)**	**98.33 (9)**	86.53 (15)
Spambase (57)	90.43 (12)	90.43 (12)	**93.04 (19)**	**93.04 (19)**
Sonar (60)	78.56 (11)	77.44 (6)	80.88 (20)	**81.76 (19)**
Urban land cover (147)	60.49 (9)	60.48 (9)	**61.37 (6)**	**61.37 (6)**

The results in Table 4.5 also show that the classification accuracy was enhanced by the MLFI algorithm compared to the previous works using the NB classifier. Only for Wine and Online Shopper datasets that had equal results for all techniques. Apart from the two datasets mentioned earlier, IFFS produced the same maximum accuracy as MLFI on Image Segmentation, Crowdsourced, Breast Cancer, Ionosphere, Soybean and Urban Land Cover datasets. Regardless of equal solutions with the IFFS, the MLFI yielded the highest classification accuracy for 13 datasets from all 14 tested datasets. Therefore, MLFI had the best performance for almost all the tested datasets because it produced the highest classification accuracy with the smallest number of selected features equal to or better than the other methods.

Table 4.5 The Comparison of Maximum Accuracy Using NB

Dataset	Previous Methods (NB)			Proposed Method (NB)
	SFS	**SFFS**	**IFFS**	**MLFI**
Wine (13)	**93.35 (5)**	**93.35 (5)**	**93.35 (5)**	**93.35 (5)**
Thoracic Surgery (17)	85.11 (1)	85.11 (1)	85.11 (1)	**85.32 (5)**
Online Shopper (17)	**90.67 (2)**	**90.67 (2)**	**90.67 (2)**	**90.67 (2)**
Lymphography (18)	86.48 (7)	86.52 (10)	87.33 (9)	**87.33 (7)**
Image Segmentation (19)	81.91 (5)	81.91 (5)	**82.86 (5)**	**82.86 (5)**
Crowdsourced (29)	82.92 (18)	83.01 (16)	**83.4 (19)**	**83.4 (19)**
Breast Cancer (32)	95.44 (8)	95.44 (8)	**96.14 (6)**	**96.14 (6)**
Ionosphere (34)	92.58 (14)	93.44 (11)	**93.72 (14)**	**93.73 (14)**
Soybean (35)	83.86 (20)	84.61 (15)	**91.73 (12)**	**91.73 (12)**
Waveform 2 (40)	85.2 (18)	85.61 (15)	85.81 (17)	**86.01 (19)**
Spectf Heart (44)	79.4 (1)	80.12 (6)	79.4 (1)	**80.15 (2)**
Spambase (57)	79.89 (15)	80.65 (18)	81.84 (12)	**82.94 (15)**
Sonar (60)	81.4 (7)	81.4 (7)	81.45 (13)	**82.35 (9)**
Urban land cover (147)	71.48 (12)	75.24 (16)	**76.14 (15)**	76.13 (13)

As shown in Table 4.6, the majority of the best results were produced by the MLFI algorithm. SFS and SFFS could not give the highest solution for all datasets. IFFS produced the best accuracy with the smallest subset sizes for Wine and Lymphography datasets. Even though IFFS gave the highest accuracy for the Lymphography dataset, but this solution was also equal to MLFI with two features less than from MLFI for the selected subset. However, the MLFI algorithm provided the maximum classification accuracy for the other twelve datasets. Seeing that, our proposed method was more preferable for feature selection than the other methods using the DT classifier.

Table 4.6 The Comparison of Maximum Accuracy Using DT

Dataset	**Previous Methods (DT)**			**Proposed Method (DT)**
	SFS	**SFFS**	**IFFS**	**MLFI**
Wine (13)	91.66 (10)	92.27 (6)	**93.95 (5)**	93.93 (7)
Thoracic Surgery (17)	80.43 (3)	80.64 (6)	80.85 (6)	**81.28 (6)**
Online Shopper (17)	88.81 (2)	89.05 (2)	89.13 (10)	**89.38 (6)**
Lymphography (18)	85.14 (6)	86.05 (10)	**87.29 (5)**	87.29 (7)
Image Segmentation (19)	84.76 (11)	86.19 (12)	85.71 (7)	**86.67 (8)**
Crowdsourced (29)	80.84 (12)	82.16 (13)	82.16 (10)	**83.4 (15)**
Breast Cancer (32)	95.61 (5)	96.32 (6)	96.67 (8)	**96.84 (13)**
Ionosphere (34)	92.3 (20)	92.89 (18)	93.72 (19)	**94.88 (18)**
Soybean (35)	90.22 (20)	91.75 (17)	92.11 (18)	**92.11 (16)**
Waveform 2 (40)	77.61 (18)	78.39 (16)	78.39 (16)	**78.97 (17)**
Spectf Heart (44)	80.55 (10)	80.17 (11)	86.9 (15)	**89.13 (19)**
Spambase (57)	90.32 (20)	90.65 (17)	90.43 (20)	**92.39 (17)**
Sonar (60)	83.27 (17)	86.6 (13)	86.18 (19)	**88.98 (18)**
Urban land cover (147)	78.07 (14)	77.33 (12)	80.42 (20)	**81.76 (17)**

Table 4.7 shows the comparison of the results from the MLFI algorithm using different criterion functions. The performance validations included KNN, NB and DT classifiers. The majority of the best performances were from DT with seven sample datasets whereas NB provided the best results on only one dataset, which was the Online Shopper dataset. The other six tested datasets with maximum classification accuracy were from the KNN classifier. Therefore different criterion functions affected the performance of the MLFI algorithm. DT classifier as a criterion function yielded the best result followed by KNN and NB respectively. Thus, both KNN and

DT are more favorable classifiers for getting the best solutions since they provide more opportunity to get the highest accuracy.

Table 4.7 The Comparison of Maximum Accuracy From the Three Different Classifiers for the MLFI Algorithm

Dataset	**Proposed Method (MLFI)**		
	KNN	**NB**	**DT**
Wine (13)	**93.38 (7)**	93.35 (5)	93.93 (7)
Thoracic Surgery (17)	**85.96 (9)**	85.32 (5)	81.28 (6)
Online Shopper (17)	90.67 (4)	**90.67 (2)**	89.38 (6)
Lymphography (18)	**90.81 (10)**	87.33 (7)	87.29 (7)
Image Segmentation (19)	81.43(8)	82.86 (5)	**86.67 (8)**
Crowdsourced (29)	**90.42 (20)**	83.4 (19)	83.4 (15)
Breast Cancer (32)	95.44 (10)	96.14 (6)	**96.84 (13)**
Ionosphere (34)	94.87 (8)	93.73 (14)	**94.88 (18)**
Soybean (35)	91.73 (20)	91.73 (12)	**92.11 (16)**
Waveform 2 (40)	**86.38 (18)**	86.01 (19)	78.97 (17)
Spectf Heart (44)	86.53 (15)	80.15 (2)	**89.13 (19)**
Spambase (57)	**93.04 (19)**	82.94 (15)	92.39 (17)
Sonar (60)	81.76 (19)	82.35 (9)	**88.98 (18)**
Urban land cover (147)	61.37 (6)	76.13 (13)	**81.76 (17)**

The proposed algorithm based on a sequential feature selection algorithm produced effective feature subsets with higher classification accuracy on several different datasets. Our proposed algorithm can extract a more relevant and effective feature subset from the source dataset using a multi-level forward-searching technique with the application of the adaptive generalization level. The experimental results showed that our method produced the maximum accuracy with the smallest subsets on the majority of the tested datasets. This improvement is the result of the multi-level forward looks ahead technique that leads to a more thorough search with better chances to discover the smaller feature subsets. Several smaller subsets with higher accuracy are found by our in-depth searching method. Similar to the OFMB algorithm, a trap in the local optimum is also possible for some tested datasets.

To evaluate the time complexity of the MLFI algorithm, we can derive from the two main steps similar to OFMB. The first one is the multi-level forward inclusion step, which computes for *s* loops where *s* is 1 to r_{max} and the selected subsets of size *k* are considered. For each loop, the total subsets that need to be calculated are $C(k, s)$ minus the subsets that do not contain the newly selected feature. This operation repeats at most *n* times. Therefore, the number of subsets that need to be evaluated describe by the same expression as OFMB, which results in the time complexity for this step is $n\sum_{s=1}^{5} k^s$. The second step is the feature replacement step, which consists of *k* subsets. We remove one feature from the selected subset and add one feature from the remaining set. This operation continues no greater than *n* loop. Thus, the feature replacement step requires no more than n^2 of time complexity. Other steps are constant time so they can be ignored. Combine the two steps together we have $n\sum_{s=1}^{5} k^s + n^2$ time complexity for the MLFI algorithm. We can conclude that MLFI requires higher computational time than IFFS for which IFFS bounded by $O(n^2)$, moreover, SFS and SFFS are bounded by $O(n)$.

4.7 The Comparison between the OFMB and MLFI Algorithms

Table 4.8 shows the comparison of the results from the OFMB and MLFI algorithms using different criterion functions. The performance validations were KNN, NB and DT classifiers where the highest accuracy for each method and the dataset was in bold; the higher accuracy between the two proposed methods was underlined. There were four datasets from OFMB that produced the highest accuracy with less number of selected features in the subset. These four datasets included Wine, Thoracic Surgery, Image Segmentation and Ionosphere which were the results from KNN and DT classifiers. The majority of the best performances were from the MLFI method with six tested datasets. These datasets were Breast Cancer, Soybean, Waveform 2, Spectf Heart, Sonar and Urban Land Cover. Five of them provided the best accuracy using the DT classifier. Only Waveform 2 was a result of the KNN classifier. The application of NB produced a poor result among the three classifiers for both the OFMB and MLFI methods. The remaining datasets had equal maximum accuracy with the same classifier from the two proposed methods. According to the size of the dataset, OFMB provided a better solution for small datasets, while MLFI yielded the best performance for large datasets. Therefore we can conclude that the MLFI method performs better than the OFMB method in general.

Table 4.8 The Comparison of Maximum Accuracy From the Three Different Classifiers for the OFMB and MLFI Algorithms

Dataset	**Proposed Method (OFMB)**			**Proposed Method (MLFI)**		
	KNN	**NB**	**DT**	**KNN**	**NB**	**DT**
Wine (13)	93.38 (7)	93.35 (5)	**93.98 (8)**	**93.38 (7)**	93.35 (5)	93.93 (7)
Thoracic Surgery (17)	**86.96 (10)**	85.32 (5)	81.06 (3)	**85.96 (9)**	85.32 (5)	81.28 (6)
Online Shopper (17)	90.67 (5)	**90.67 (2)**	89.3 (11)	90.67 (4)	**90.67 (2)**	89.38 (6)
Lymphography (18)	**90.81 (10)**	88.05 (8)	87.29 (6)	**90.81 (10)**	87.33 (7)	87.29 (7)
Image Segmentation (19)	81.43 (7)	82.86 (5)	**88.57 (9)**	81.43 (8)	82.86 (5)	**86.67 (8)**
Crowdsourced (29)	**90.42 (20)**	83.4 (19)	83.22 (16)	**90.42 (20)**	83.4 (19)	83.4 (15)
Breast Cancer (32)	95.44 (13)	96.14 (6)	**96.5 (14)**	95.44 (10)	96.14 (6)	**96.84 (13)**
Ionosphere (34)	**94.89 (11)**	93.72 (11)	94.3 (20)	94.87 (8)	93.73 (14)	**94.88 (18)**
Soybean (35)	90.23 (17)	**91.73 (12)**	91.36 (19)	91.73 (20)	91.73 (12)	**92.11 (16)**

Dataset	Proposed Method (OFMB)			Proposed Method (MLFI)		
	KNN	**NB**	**DT**	**KNN**	**NB**	**DT**
Waveform 2 (40)	**86.17 (17)**	86 (20)	78 (18)	**86.38 (18)**	86.01 (19)	78.97 (17)
Spectf Heart (44)	**85.37 (12)**	80.15 (3)	83.54 (20)	86.53 (15)	80.15 (2)	**89.13 (19)**
Spambase (57)	**93.04 (19)**	82.29 (18)	90.65 (16)	**93.04 (19)**	82.94 (15)	92.39 (17)
Sonar (60)	**81.76 (19)**	81.45 (13)	84.3 (13)	81.76 (19)	82.35 (9)	**88.98 (18)**
Urban land cover (147)	61.37 (6)	76.14 (15)	**81.31 (20)**	61.37 (6)	76.13 (13)	**81.76 (17)**

In this chapter, we proposed an algorithm called Multi-level Forward Inclusion (MLFI) algorithm. We aimed to develop a feature selection method that overcomes the previous works in terms of classification accuracy. We proposed a feature selection algorithm based on the sequential searching technique by improving the performance of SFFS. The application of an adaptive multi-level forward search assisted the maximization of classification accuracy for the feature subset selection. With the addition of a feature replacement step, the nesting problem was solved. MLFI was able to discover important subsets that did not find by SFFS or IFFS. The algorithm employed an adaptive generalization limit to indicate the level of forwarding search. The higher the limit led to a higher chance of finding a better subset. There are three proposed methods for calculating the generalization limit and the best one was selected for our results. In the experiments, we compared our method with SFS, SFFS and IFFS. Results on the 14 standard UCI datasets showed that MLFI performed better than the other suboptimal sequential feature selection algorithms for the majority of the tested datasets. At the end of the chapter, we compared our two proposed methods under the same condition. The results showed that the MLFI method overcomes the OFMB method for large datasets, particularly on the DT classifier.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

5.1 Conclusions

Feature selection is very important for the performance of classification in the data mining process. This dissertation focused on an improvement of the early sequential feature selections. We developed feature selection methods that overcome the previous works in terms of classification accuracy. Our feature selection algorithms based on the sequential searching technique by improving the performance of the standard SFFS. We proposed two new algorithms that search for undiscovered subsets. Our proposed algorithms are the One-level Forward Multi-level Backward Selection (OFMB) and the Multi-level Forward Inclusion (MLFI). The improvement of feature selection also assists the effectiveness of the data mining algorithm. Our objective was to enhance the feature selection performances while maintaining the computational time as small as possible for the application on a large dataset. The concept of the two proposed algorithms is summarized below.

The OFMB algorithm divided into two parts. The first part is to explore feature subsets by incorporating a feature improvement step that produces a similar result to the IFFS. This part can be considered as another algorithm and we named it the One Level Forward Inclusion (OLFI). The second part is the addition of a multi-level backtracking step and is designed to discover important subsets that cannot be discovered by the SFFS or IFFS methods. The algorithm employs an adaptive generalization limit to indicate the level of backward searching. The higher the limit leads to a higher chance of finding a better subset.

The MLFI algorithm is also an improvement on the standard SFFS method. With the application of an adaptive multi-level forward search, MLFI maximizes the classification accuracy of the feature subset. An addition of a feature replacement step helps to solve the nesting problem. MLFI is able to discover important subsets that cannot be discovered by past methods. Similar to OFMB, the MLFI algorithm also uses an adaptive generalization limit to indicate the level of forwarding search.

In the experiments, we compared our proposed methods with SFS, SFFS and IFFS on various criterion functions including KNN, NB and DT classifiers. The results on the fourteen standard UCI datasets showed that our methods outperform the other suboptimal sequential feature selection algorithms for the majority of the tested datasets.

The time complexity of OFMB and MLFI is greater than the other methods, which is $n \sum_{s=1}^{5} k^{s} + n^{2}$. Whereas the IFFS bounded by $O(n^2)$, moreover, SFS and SFFS are bounded by $O(n)$.

5.2 Future Work

An adaptive multi-level sequential feature selection using a wrapper approach improves the classification performance for various well-known classifiers. Even though the level of the backward and forward-searching can explore more potential feature subsets, but it is limited by the generalization limit. This limitation leads to a reduction in the algorithms' effectiveness. The application of a deeper search can increase the classification accuracy closer to the optimal solution. However, the higher numbers of generalization limit result in the growth of the computing time. To keep this computing time as low as possible, we need to explore several techniques further, which may help with the time reduction.

One technique can be an addition of a filter approach by using some measurement such as information gain or distance measures to filter out features that are less likely to be significant for feature-class correlation. Applying our methods to the pre-selected features and ignoring the unlikely important features can greatly reduce the computational time.

Another issue for further studies is the prediction of the generalization limit (r). The number of the best r-value for each iteration is unknown. If we can find a connection of the suitable r-value for a particular iteration of the process, it will improve the performance and can also help on the time reduction since we can spend less time searching for subsets that do not gain high accuracy. A genetic algorithm or neural network may be a good choice for studying this issue.

According to our results on the comparison among different classifiers, we can see that they produce different classification accuracies. Therefore other criterion functions apart from the ones that we mentioned in the previous chapters may lead to better solutions on the same datasets.

BIBLIOGRAPHY

Beniwal, S., & Arora, J. (2012). Classification and feature selection techniques in data mining. *International Journal of Engineering Research & Technology (IJERT), 1*(6).

Bolon-Canedo, V., & Alonso-Betanzos, A. (2019). Ensembles for feature selection: A review and future trends. *Information Fusion*, *52*, 1-12.

Cai, D., Zhang, C., & He, X. (2010). Unsupervised feature selection for multi-cluster data. *International Conference on Knowledge discovery and data mining*, 333-342.

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, 70-79.

Chaiyakarn, J. (2013). *A filter-Based feature selection using two criterion functions and evolutionary fuzzification.* (Doctoral dissertation), National Institute of Development Administration, Bangkok.

Cisotto, G., Capuzzo, M., Guglielmi, A. V., & Zanella, A. (2020). Feature selection for gesture recognition in Internet-of-Things for healthcare. *IEEE Xplore*.

Dheeru, D., & Efi, K. T. (2017). UCI machine learning repository. Retrieved from http://archive.ics.uci.edu/ml

Homsapaya, K., & Sornil, O. (2017). Improving floating search feature selection using genetic algorithm. *Journal of ICT, 11*(3), 299-317.

Huang, S. H. (2015). Supervised feature selection: A tutorial. *Artificial Intelligence Research, 4*, 22-37.

Huda, R. K., & Banka, H. (2019). New efficient initialization and updating mechanisms in PSO for feature selection and classification. *Neural Computing and Applications, 32*, 3283-3294.

Jovic, A., Brkic, K., & Bogunovic, N. (2015). A review of feature selection methods with applications. *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*.

Kadhum, M., Manaseer, S., & Dalhoum, A. L. A. (2021). Evaluation feature selection technique on classification by using evolutionary ELM wrapper method with features priorities. *Journal of Advances in Information Technology, 12*(1), 21-28.

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering, 17*(4), 491-502.

Liu, W., & Wang, J. (2019). A brief survey on nature-inspired metaheuristics for feature selection in classification in this decade. *Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control*, 424-429.

Lv, J., Peng, Q., & Sun, Z. (2015). A modified sequential deep floating search algorithm for feature selection. *International Conference on Information and Automation*, 2988-2933.

Marill, T., & Green, D. M. (1961). On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 11-17.

Mwadulo, M. W. (2016). A review on feature selection methods for classification tasks. *International Journal of Computer Applications Technology and Research, 5*(6), 395-402.

Nakariyakul, S., & Casasent, D. P. (2009). An improvement on floating search algorithms for feature subset selection. *Pattern Recognition*, 1932-1940.

Pavya, K., & Srinivasan, B. (2017). Feature selection techniques in data mining: a study. *International Journal of Scientific Development and Research (IJSDR), 2*(6), 594-598.

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEETransactions on Pattern Analysis and Machine Intelligence, 27*, 1226-1238.

Pudil, P., Novovicova, J., & Kittler, J. (1994). Floating search methods in feature selection. *Pattern Recognition Letters*, 1119-1125.

Raj, R. J. S., Shobana, S. J., Pustokhina, I. V., Pustokhin, D. A., Gupta, D., & Shankar, K. (2020). Optimal feature selection-based medical image classification using deep learning model in internet of medical things. *IEEE Access, 8*, 58006-58017.

Somol, P., Novovicova, J., & Pudil, P. (2006). Flexible-hybrid sequential floating search in statistical feature selection. *Structural, Syntactic, and Statistical Pattern Recognition*, 632-639.

Somol, P., Pudil, P., Novovicova, J., & Paclik, P. (1999). Adaptive floating search methods in feature selection. *Pattern Recognition Letters*, 1157-1163.

Sutha, K., & Tamilselvi, D. J. J. (2015). A review of feature selection algorithms for data mining techniques. *International Journal on Computer Science and Engineering (IJCSE)*, 63-67.

Whitney, A. W. (1971). A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, 1100-1103.

BIOGRAPHY

Name-Surname	Mr. Knitchepon Chotchantarakun
Academic Background	Bachelor of Science (Computer Science) Mahidol University International College Year of Graduation: 2003
	Master of Science (Computer Science) Chulalongkorn University Year of Graduation: 2006
Experience	Year 2006 Project Supervisor International System House Danube
	Year 2007 - 2012 Lecturer Faculty of Information Technology Siam University
	Year 2013 - Present Lecturer Faculty of Humanities and Social Sciences Burapha University