Markov Blanket: Efficient Strategy for Feature Subset Selection Method for High Dimensionality Microarray Cancer Datasets

by

Abdala Nour

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science (MSc) in Computational Sciences

The Faculty of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE

Laurentian Université/Université Laurentienne

Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis
Titre de la thèse Markov Blanket: Efficient Strategy for Feature Subset Selection Method for High Dimensionality Microarray Cancer Datasets

Name of Candidate
Nom du candidat Nour, Abdala

Degree
Diplôme Master of Science

Department/Program
Département/Programme Computational Sciences

Date of Defence
Date de la soutenance August 28, 2017

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Kalpdrum Passi
(Supervisor/Directeur de thèse)

Dr. Ratvinder Grewal
(Committee member/Membre du comité)

Dr. Mazen Saleh
(Committee member/Membre du comité)

Dr. Sanjeena Dang
(External Examiner/Examinateur externe)

Approved for the Faculty of Graduate Studies
Approuvé pour la Faculté des études supérieures
Dr. David Lesbarrères
Monsieur David Lesbarrères
Dean, Faculty of Graduate Studies
Doyen, Faculté des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

Abstract

Currently, feature subset selection methods are very important, especially in areas of application for which datasets with tens or hundreds of thousands of variables (genes) are available. Feature subset selection methods help us select a small number of variables out of thousands of genes in microarray datasets for a more accurate and balanced classification.

Efficient gene selection can be considered as an easy computational hold of the subsequent classification task, and can give subset of gene set without the loss of classification performance. In classifying microarray data, the main objective of gene selection is to search for the genes while keeping the maximum amount of relevant information about the class and minimize classification errors. In this paper, explain the importance of feature subset selection methods in machine learning and data mining fields. Consequently, the analysis of microarray expression was used to check whether global biological differences underlie common pathological features in different types of cancer datasets and identify genes that might anticipate the clinical behavior of this disease. Using the feature subset selection model for gene expression contains large amounts of raw data that needs analyzing to obtain useful information for specific biological and medical applications. One way of finding relevant (and removing redundant) genes is by using the Bayesian network based on the Markov blanket [1]. We present and compare the performance of the different approaches to feature (genes) subset selection methods based on Wrapper and Markov Blanket models for the five-microarray cancer datasets. The first way depends on the Memetic algorithms (***MAs***) used for the feature selection method. The second way uses ***MRMR*** (Minimum Redundant Maximum Relevant) for feature subset selection hybridized by genetic search optimization techniques and afterwards compares the Markov blanket model’s performance with the most common classical classification algorithms for the selected set of features.

For the memetic algorithm, we present a comparison between two embedded approaches for feature subset selection which are the wrapper filter for feature selection algorithm (*WFFSA*) and Markov Blanket Embedded Genetic Algorithm (*MBEGA*). The memetic algorithm depends on genetic operators (crossover, mutation) and the dedicated local search procedure. For comparisons, we depend on two evaluations techniques for learning and testing data which are 10-Kfold cross validation and 30-*Bootstraping*. The results of the memetic algorithm clearly show *MBEGA* often outperforms *WFFSA* methods by yielding more significant differentiation among different microarray cancer datasets.

In the second part of this paper, we focus mainly on *MRMR* for feature subset selection methods and the Bayesian network based on Markov blanket (***MB***) model that are useful for building a good predictor and defying the curse of dimensionality to improve prediction performance. These methods cover a wide range of concerns: providing a better definition of the objective function, feature construction, feature ranking, efficient search methods, and feature validity assessment methods as well as defining the relationships among attributes to make predictions.

We present performance measures for some common (or classical) learning classification algorithms (Naive Bayes, Support vector machine [***LiBSVM***], K-nearest neighbor, and AdBoostM Ensampling) before and after using the *MRMR* method. We compare the Bayesian network classification algorithm based on the Markov Blanket model's performance measure with the performance of these common classification algorithms. The result of performance measures for classification algorithm based on the Bayesian network of the Markov blanket model get higher accuracy rates than other types of classical classification algorithms for the cancer Microarray datasets.

Bayesian networks clearly depend on relationships among attributes to make predictions. The Bayesian network based on the Markov blanket *(MB)* classification method of classifying variables provides all necessary information for predicting its value. In this paper, we recommend the Bayesian network based on the Markov blanket for learning and classification processing, which is highly effective and efficient on feature subset selection measures.

Keywords

Microarray datasets, feature selection methods, genetic algorithms, memetic algorithms, overfitting problem, fitness function, crossover, mutation, Markov Blanket, minimum redundancy-maximum relevant, support vector machine, naive Bayes, k-nearest-neighbor, ensemble classifier, Bayesian networks.

Acknowledgements

First and foremost, I would like to acknowledge the tireless and prompt help of my supervisor in the Computational Science division at Laurentian University, Dr. Kalpdrum Passi, who has allowed me complete freedom to define and explore my own directions in research. I thank him for his consistent support and encouragement through all stages of my master’s study. Dr. Passi was always kind and positive with me even when I faced problems during my study, which has given me confidence to complete this work.

In addition to Dr. Passi, I am deeply grateful for all the staff of my Computational Science department. They have provided me with very much appreciated knowledge and information during my study and they have kindly given me teaching assistantship positions during my study.

Special thanks must also go to all my family and friends in Canada. They have provided unconditional support and encouragement through both the highs and lows of my time in graduate school.

Table of Contents

Chapter 1

Introduction

1.1 Introduction to machine learning

Machine learning is a branch of artificial intelligence using a set of algorithms to build analytical models, help computers "learn," and find patterns in data. It can be applied to high dimensionality data to create exciting new applications and more accurately predict outcomes without being explicitly programmed. Moreover, machine learning is said to allow learning whether performance on a defined task (or tasks) will improve with experience. More specifically, machine learning can modify algorithms and subsequently do the same task (or tasks) more efficiently [2, 3].

In many cases, to improve the performance of learning algorithms in a supervised learning machine, feature subset selection is considered an underlying obstacle to defining the perfect model. Feature subset selection techniques help reduce noisy or irrelevant genes before applying the classification algorithm. Also, it improves the performance measure of learning classification algorithms[2].

1.2 Gene Expressions and Microarrays

DNA microarray technology is a powerful new research tool capable of an expression level of one thousand to ten thousand genes, each representing a different gene in an organism. Microarrays are used to analyze the gene expression levels in two different populations of cells (e.g., to look at gene expression in plants grown under different conditions, to look at gene expression in normal cells vs. cancer cells). This is done by labeling cDNAs from two different groups of cells with two different dyes and hybridizing them to the microarrays. Genes are "differentially regulated," meaning all cells in an organism contain the same genes, but different genes are expressed (transcribed) in different tissues under different conditions. This gives different tissues their different phenotypes, or appearance and function. DNA microarray can measure gene sequencing expression, DNA transcription, and hybridization to analyze and identify thousands of genes simultaneously. Gene expression microarrays can be used to select which genes increase or decrease activities, also referred to as transcriptional profiles or gene expression "signatures" that have since established distinct tumor types. They also allow us to determine which genes are active in different cell states. Furthermore, studying and analyzing gene expression in normal and tumor tissues will help researchers identify genes or groups of genes expressed to understand gene regulation, genetic mechanisms of disease, and function as well as response to drug treatment [4]. We can obtain gene expression data by using high-throughput technologies such as microarray and oligonucleotide chips in different tissues. Raw microarray data are images which must be transformed into gene expression matrices (or tables) where the rows represent genes' expression patterns, the columns represent various sample types such as tissues or experimental conditions, and each cell characterizes the particular gene's measured expression level in a sample [5, 6]. When we have gene expression data, annotation can be added either to the gene or to the sample. For example, the gene's function or additional details on the biology of the sample can be provided, such as "cancer state" or "normal state" [7]. There are two straightforward methods used to study the gene expression matrix:

1. comparing gene expression profiles by comparing rows in the expression matrix;
2. comparing sample expression profiles by comparing columns in the matrix.

Additionally, by studying the gene expression matrix (data) we can look for similarities and differences between genes (or samples). If the two genes are similar, we can emphasize clues that they are co-regulated and possibly functionally related. By comparing samples, we can find which genes are differentially expressed in different situations [8].

Moreover, due to the high dimensionality (curse dimensionality) of microarray datasets, they often contain many irrelevant and redundant features which increase the complexity of classification and influence the performance of most learning algorithms [9]. The main difficulties in DNA microarray classification are the availability of a very small number of instances (samples) in comparison with the number of genes (or attributes) in the sample and the experimental variation in measured gene expression levels[10]. The feature subset selection methods used on DNA microarray datasets are particularly interesting approaches since they allow removing irrelevant and redundant features (genes) from microarray datasets, which is the key problem addressed by feature subset selection methods [11]. Thus, the computational cost is reduced while the level of performance measures such as prediction accuracy is increased through using effective feature subset selection. Therefore, the task of removing redundant / irrelevant features is a one of the most important aspects of machine learning and data mining techniques[2].

Existing feature selection methods mainly fall into two main categories, those individual (single) feature evaluation and subset feature evaluation methods, based on whether they evaluate the goodness of features individually or through feature subsets. Methods of individual evaluation feature usually depend on some statistical measures are calculated for each feature, then a ranked feature list is provided in a predefined order of the statistic. The statistics used for individual feature selection include information gain, correlation coefficient, t-statistic, χ2-statistic and others. Based on those statistical measures, the rank features according to their importance in differentiating instances of different classes can be calculated and after that can only remove irrelevant features as redundant features likely have similar rankings. Methods of subset feature evaluation methods search through candidate minimum subset of features that satisfies some goodness measure of each subset and can remove irrelevant features as well as redundant ones. For example, a correlation coefficient can be used to estimate the goodness of feature subsets based on the hypothesis that a good feature subset is one that contains features highly correlated to the class, yet uncorrelated to each other[11-13].

In machine learning field, the feature subset selection methods typically fall into two broad categories which are wrapper and filter methods. The wrapper methods use an inductive learning algorithm as the evaluation function while the filter method is used essentially as a data pre-processing or data filtering method [9, 14, 15]. Many classification algorithms such as neural networks, support vector machines, ensample, and others have been used to perform classification and predictions of gene subset selection. Unfortunately, these classification techniques offer little insight into probabilistic reasoning[11].

1.3 Motivation

The limitations of existing research clearly inspired us to look for various methods of feature selection that allow efficient analysis and solve problems associated with high-dimensional data. The Bayesian network method provides an approach based on probabilistic reasoning which is used to measure the relationship among features making up the prediction and classification learning model [16]. The Bayesian network method based on the Markov blanket can be used as a tool to select features based on statistical information (probabilistic reasoning) and graphical model (DAG) [16, 17] . As a result, artificial intelligence and machine learning are concerning with Bayesian network which used in solving the problems of uncertainty in prediction and classification fields [18].

Although several researchers have recommended to use a Markov blanket model as primary type for feature subset selection method to get a high-performance level of learning classification model [1, 9, 14, 15]. The research studies in this area are considered scarce and more research work is needed. For example, high dimensional data (i.e., data sets with hundreds or thousands of features) can contain high degree of irrelevant and redundant information which may greatly degrade the performance of learning algorithms. Therefore, feature selection becomes very necessary for machine learning tasks when facing high dimensional data nowadays. Consequently, using the Markov blanket model with other types of feature subset selection methods can prove and confirm its efficiency in reducing irrelevant and redundant features. Thus, the main purpose of this paper is presenting this innovative comparison.

This work contributes to the science literature by demonstrating the importance of the Bayesian network and Markov Blanket for feature subset selection and learning classification models. In addition to the Markov blanket model, we describe how maximum relevance minimum redundancy (MRMR) can be used as a feature subset selection method hybridized with a genetic algorithm as a search optimization tool to build high quality and effective learning classification algorithms; the search for optimal features is considered another key problem of feature subset selection methods [11]. Feature subset selection based on MRMR methods use feature ranking and correlation coefficients as a principal selection mechanism because it is very simple and significant features are accessible [19]. The main reasons we are concerned about feature subset selection methods are listed briefly as follows:

1. It is cheaper to measure only a set of variables instead of all features.
2. The prediction accuracy level might be improved through exclusion of irrelevant variables.

3. The predictor to be built is usually simpler and potentially faster when less input variables are used.
4. Identifying relevant and removing irrelevant features can help us understand the nature of the prediction problem at hand [20].

1.4 Organization

The thesis is organized as follows. In Chapter 2, we present a literature review, including previous research on microarray classification algorithms and feature subset selection methods. In Chapter 3, we focus on the Bayesian network approach and Markov Blanket model. In Chapter 4, we present the methodology and experimental results of our approach on five DNA Cancer microarray datasets and compare them with existing methods covered in Chapter 3. In Chapter 5, we show the datasets and evaluation learning models. In Chapter 6, we present and discuss the results, followed by the conclusion and references.

Chapter 2

Literature Review

Supervised learning is a machine learning process based on a classification task (data analysis) which takes a known input data (the training set) and known response of the data (output), where a model (classifier) learns or is built to predict and assign the class (categorical or discrete, e.g., normal or tumor tissue in cancer datasets) label to an unknown observation or sample. So each classification technique depends on a learning algorithm to detect a model that can be considered a best fit for the relationship between the subset features and class label of the input data [2, 21, 22]. In contrast, for the unsupervised learning problem, the class label information is not known and we observe only the features and have no measurements of the outcome [22, 23]. In unsupervised learning, a machine must decide which features should be grouped together as one class based on specific criteria. Clustering, or cluster analysis, is the process of grouping a set of data cases into groups by which the cases in each group are very similar to each other and different from the cases in other groups [22] . This study is concerned with a supervised learning machine and in the following sections we explain some of the most common classification algorithms.

2.1 The Naïve Bayes Classifier

Naive Bayes is a simple statistical classifier but it is considered a powerful algorithm for predictive modeling [22]. It assigns each observation, also known as a tuble, to the most likely class based on its predictor values. The Naive Bayes classifier is obtained by using the Bayes rule and assuming features (variables) that are independent of each other given its class. This assumption is called class-conditional independence. The following equation shows the naive Bayes rule which assumes feature values are statistically independent within each class.

$$\mathbf{P(C\,|X)} = \frac{\boldsymbol{P(X|\mathrm{C})\ \mathrm{P}(\mathrm{C})}}{\boldsymbol{P(X)}}$$

- ***P*(*c*|*x*)** is the posterior probability of a class (c, target) given a predictor (x, attributes).
- **P(c)** is the prior probability of class.
- ***P*(*x*|*c*)** is the likelihood, which is the probability of a *predictor* given its *class*.
- ***P*(*x*)** is the prior probability of a *predictor*.

From the previous equation, we can see the Naïve Bayes classifier deals with different types of probabilities:

- Class probabilities, and
- Conditional probabilities.

Conditional probability means for each distinct parent node values, we need to specify the probability that the child will take each of its values.

The Naïve Bayes rule's core task is finding the probability of the previously unseen instance belonging to each class, then simply pick the most probable class. The Naïve Bayes classifier has been shown to perform well when classifying many real data sets in the machine learning field [2, 22].

2.2 Support Vector Machine (SVM)

A *support vector machine* (SVM) is another type of learning system which is a relatively promising classification method [24]. It is a margin classifier that draws an optimal hyperplane in the feature vector space; this defines a boundary that maximizes the margin between data samples in two classes, therefore leading to good generalization properties. A key factor in the SVM is using kernels to construct a nonlinear decision boundary (i.e. separating the tuples of one class from another) [22]. In this paper, we will use a linear kernels SVM.

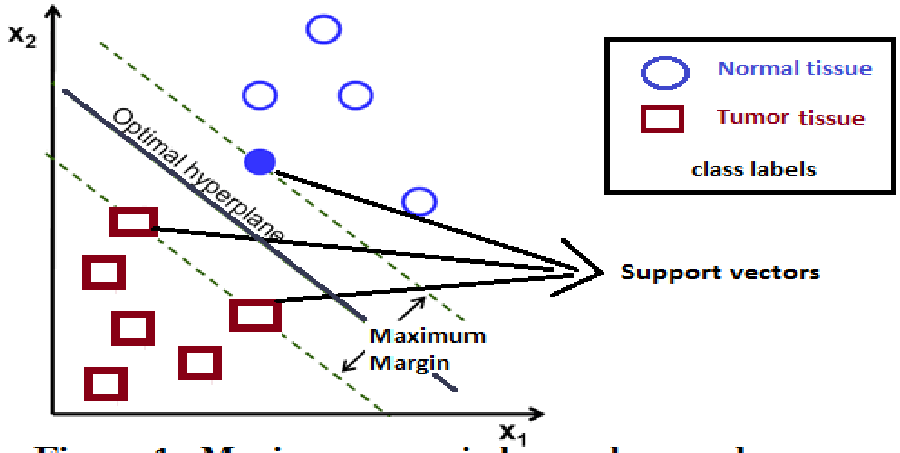


Figure 1 : Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

2.2.1 SVM Notations and terminology:

In general, an SVM is a **linear learning system** that builds two-class classifiers. Let the set of training examples *D* be $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, where the $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a ***r*-**dimensional **input vector** in a real-valued space $X \subseteq \Re^{r}$, and y_i is its **class label** (output value) and $y_i \in \{1, -1\}$z. Where 1 denotes the positive class and -1 denotes the negative class.

Similarly, each data instance is called an **input vector** and denoted by a bold face letter. In the following, we use bold face letters for all vectors. To build a classifier, a SVM finds a linear function of the form

$$\boldsymbol{f}(\mathbf{x}) = (\mathbf{w} \, . \, \mathbf{x}) + \boldsymbol{b} \qquad (1)$$

so, an input vector $\boldsymbol{x_i}$ is assigned to the positive class if $f(x_i) \geq 0$, and to the negative class otherwise, i.e.

$$\mathbf{f}(\mathbf{x}) = \begin{cases} -1 \ \text{ if } \ (\mathbf{W.Xi}) + \mathbf{b} \ < 0 \\ \ 1 \ \text{ if } \ (\mathbf{W.Xi}) + \mathbf{b} \ \geq 0 \end{cases} \qquad (\ 2)$$

Hence, f(x) is a real-valued function $\boldsymbol{f}: \ X \subseteq \Re^r \rightarrow \Re$. $\mathbf{w} = (w_1, w_2, \ldots, w_r) \in \Re^{\boldsymbol{r}}$ is called the **weight** vector. The $\boldsymbol{b} \in \Re$ is called the ***bias***. (**w.x**) is the dot product of **w** and **x** (or Euclidean inner product). We can easily extend the equation (1) to the r-dimensional setting as follows:

$$\boldsymbol{f(x1, x2, \ldots, xr) = w1 \, . \, x1 + w2 \, . \, x2 + \ldots + wr \, . \, xr + b} \qquad \mathbf{(3)}$$

Substantially, an SVM is a discriminative classifier that works as follows. It uses nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (i.e., **a “decision boundary”** or **“decision surface“** separating the tuples of one class from another and is used to make classification decisions on test instances) [21, 22, 25].

2.2.2 Define the optimal hyperplane

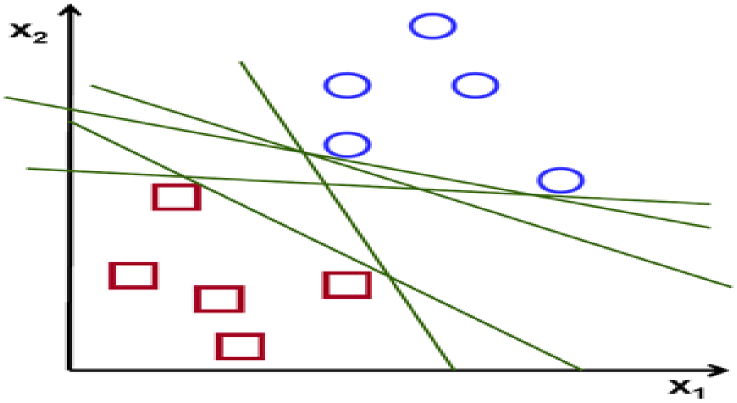
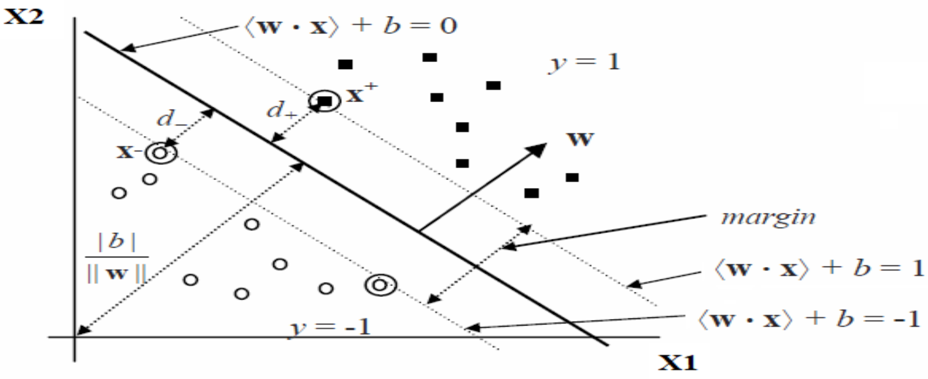


Figure 2 : severial lines of decision boundaries

In an SVM, there are an infinite number of lines (decision boundaries) that offer a classification of the problem (see figure2). How can we choose the best one? We can depend on a criterion to estimate the best lines. A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, an SVM's goal should be binding the line (hyperplane) passing as far as possible from all points, which maximizes the margin between positive and negative data points, as seen in Figure 1 [22, 25].

2.2.3 Linear Separable of SVM:



X2
$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$
$y = 1$
$\mathbf{x}^+$
d_-
d_+
$\mathbf{x}^-$
$\mathbf{w}$
margin
$\frac{|b|}{||\mathbf{w}||}$
$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 1$
$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = -1$
$y = -1$
X1

Figure3 : Separating hyperplane and margin of SVM. Support vectors implemented by small circle.

An SVM algorithm attempts to maximize the margin between positive and negative data points, let us find the margin. Let d+ (respectively d) be the shortest distance from the separating hyperplane ((w . x) + b = 0) to the closest positive (negative) data point. The margin of the separating hyperplane is (d+ (+) d-). An SVM looks for the separating hyperplane with the largest margin, which is also called the maximal margin hyperplane (also known as the maximal margin hyperplane), as the final decision boundary. The reason for choosing this hyperplane to be the decision boundary is theoretical results from structural risk minimization in computational learning theory show that maximizing the margin minimizes the upper boundary of classification errors.

Note: Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors. These observations do affect the support vector classifier.

The optimal hyperplane can be represented in an infinite number of different ways by scaling ***w and b.*** As a matter of convention, among all the possible representations of the two parallel hyperplanes (they are chosen parallel to the (w. x) + b = 0)

$$(w . x)\square + b =1 \quad or$$

$$(w . x)\square + b = -1$$

From linear algebra, let us compute the distance between the two margin hyperplanes (d+ (+) d-). Depending on the Euclidean distance from a point x_i to a hyperplane □(w . x) + b = 0 is

$$\frac{|(w.Xi) \ + b\ |}{||\mathrm{w}||}$$

where ||w|| is the Euclidean norm of w, so the ||w ||=$\sqrt{(\mathbf{w.w})}$.

Next, we use the result of geometry that gives the distance between a point x+(or d+) and a hyperplane (w. x) + b = 0 (for example $\mathbf{x_s}$ to (**w** · **x+)** + *b* = 1):

$$d+=\frac{|(w.xs)+b-1|}{||w||} = \frac{1}{||w||}$$

Likewise, we can compute the distance from $\mathbf{x_s}$ to **(< w. x > + b = -1)**to obtain (d- = 1/ ||w||). Thus, the decision boundary **((w. x) + b = 0)** lies half-way between **((w. x)> + b = +1)** and **((w. x) + b = -1)**. Therefore, we can denote the margin distance as $\mathcal{M}$, is twice the distance to the closest examples.

$$\mathcal{M}= \mathbf{d+} + \mathbf{d-} = \frac{\mathbf{2}}{||\mathbf{w}||}$$

Because an SVM looks for the separating hyperplane that maximizes the margin, this gives us an optimization problem since maximizing the margin is the same as minimizing $\mathbf{||w||^2/2 = (w.w)/2}$.

Definition (Linear SVM: Separable Case): Given a set of linearly separable training examples, ***D* = {(x1, *y*1), (x2, *y*2), …, (x*n*, *yn*)},** the learning process is used to solve the following constrained minimization problem:

$$\textit{Minimize}: <w.w>/2$$

$$\textit{Subject to}: y_i\,((w\,.\,x_{i)} + b) \geq 1 \quad , i=1,2,\ldots, n$$

Note that the constraint **(y_i ((w . x_i)+b) ≥1 , i=1,2,..., *n*)** *summarizes as :*

((w . xi)+b)□□□□□□□□□□for yi = 1

□((w .□xi) □+ b)□□-1 for yi = -1.

Since the objective function is quadratic and convex and the constraints are linear in the parameters **w and *b***, we can use the standard Lagrange multiplier method to solve it.

Instead of optimizing only the objective function, which is called unconstrained optimization, we need to optimize the ***Lagrangian*** of the problem, which considers the constraints at the same time. The need to consider constraints is obvious because they restrict the feasible solutions. Since our inequality constraints are expressed using “ ≥□”, the Lagrangian is formed by the constraints multiplied by positive Lagrange multipliers and subtracted from the objective function, i.e.

$$\boldsymbol{Lp}= \frac{1}{2}\ \boldsymbol{<w.w>}\ - \sum_{i=1}^{n} \boldsymbol{\alpha i}\ (\boldsymbol{yi}\ \ (< \boldsymbol{w.Xi} >\ +\boldsymbol{b}\ ) - \mathbf{1}\ )$$

Where **$\alpha_i \geq 0$** are the **Lagrange multipliers.**

The ***y_i*** represents each of the **labels** from the training examples. This is a problem of Lagrangian optimization that can be solved using Lagrange multipliers to obtain the **weight vector w and bias b** of the optimal hyperplane. Based on optimization theory that says an optimal solution to (Lp) must satisfy certain conditions, called ***Kuhn–Tucker*** conditions, which play a central role in constrained optimization. Only data points on the margin hyperplanes can have $\boldsymbol{\alpha i}$ > 0 since for them **y_i(<w. x_i>□+ b) – 1 = 0**. These data points are called **support vectors**; all the other data points have □i= 0. In general, Kuhn–Tucker conditions are necessary for an optimal solution, but not sufficient. However, for our minimization problem with a convex objective function and a set of linear constraints, the Kuhn–Tucker conditions are both necessary and sufficient for an optimal solution. Solving the optimization problem is still a difficult task due to the inequality constraints. However, the Lagrangian treatment of the convex optimization problem leads to an alternative ***dual*** formulation of the problem, which is easier to solve than the original problem, also called the **primal problem** (LP is called the primal Lagrangian). The concept of ***duality*** is widely used in the optimization literature. The aim is to provide an alternative formulation of the problem which is more convenient to solve computationally and/or has some theoretical significance.

In the context of an SVM, the dual problem is not only easy to solve computationally, but also crucial for using kernel functions to deal with nonlinear decision boundaries as we do not need to compute w explicitly. Transforming from the primal to its corresponding dual can be done by setting to zero the partial derivatives of the Lagrangian (Lp= $\frac{1}{2}$ <w.w> - $\sum_{i=1}^{n} \alpha i \ (yi \ \ ((w.Xi) \ + b\ ) - 1\ )$) with respect to the primal variables (i.e. w and b), and substituting the resulting relations back into the Lagrangian into the original Lagrangian equation to eliminate the primal variables, which gives us the dual objective function (denoted by LD)

$$\mathbf{LD} = \sum_{i=1}^{n} \boldsymbol{\alpha i} \ - \ \frac{\mathbf{1}}{\mathbf{2}} \ \sum_{i,j=1}^{n} \boldsymbol{yi \ yj \ \alpha i \ \alpha j} \ \ \mathbf{K(x_i.x_j)} \ \ ;$$

$$\textbf{Subject to:} \ \ \sum_{i=1}^{n} \boldsymbol{yi \ \alpha i = 0} \ \ , \ \ \boldsymbol{\alpha i} >= \mathbf{0}$$

For our convex objective function and linear constraints of the primal, it has the property that the αi's at the maximum of *LD* gives **w** and *b* occurring at the minimum of *LP* (the primal). In the above formula **K(x$_i$, x$_j$)** is the **kernel function**. Although and SVM can handle nonlinear boundaries with the kernel tricks, studies show that a linear kernel suits the text categorization problem well, and that **polynomial and RBF kernels** do not improve the performance significantly. Therefore, we stick to the simple linear kernel $\mathbf{K(x_i, x_j) = x_i \cdot x_j}$ in this study[2, 21].

2.3 K-Nearest Neighbor Learning

In the K-nearest neighbor method (k-NN), no learning model occurs from training data. Learning only occurs when a test model (example) needs to be classified. A k-NN classification algorithm is one of the simplest classification methods [21, 26]; it assigns a class label according to similarity (proximity) or distance. The basic idea of this classification method is as follows: the closest point (feature or neighbor) of the training test instance (feature vector of gene expression levels) in the training dataset determines class membership of this test instance [21, 22]. In cases where the k-NN classification method depends on a similarity measure on gene expression levels, then we depend on *Pearson* ***correlation coefficients*** as a metric measure of similarity between genes. The Pearson’s correlation coefficient has been proven effective and is widely used as a similarity measure for gene expression level [11].

$$R(i) = \frac{\sum_{k=1}^{n} (x_{k,i} - \bar{x})^2 * (y_k - \bar{y})^2}{\sqrt{(\sum_{k=1}^{n} (x_{k,i} - \bar{x})^2)} \sqrt{(\sum_{k=1}^{n} (y_k - \bar{y})^2)}}$$

Figure 4: R-correlation coefficient

Let us simplify the person correlation coefficient (1 - *R-correlation coefficient*). So, the R-coefficient is (see Figure 4). To explain this equation, we have a set of *n* instances $(\boldsymbol{x_i}, \boldsymbol{y_i})$**,** where ($i$ = 1,..n) and we have m inputs (known) $\boldsymbol{x_{k,i}}$ (i = 1,..m) and one output (unknown class label) $\boldsymbol{y_c}$ attribute. Attribute ranking for the Pearson's correlation coefficient makes use of a scoring function $\boldsymbol{R(i)}$ computed from the values $\boldsymbol{x_{k,i}}$ and $\boldsymbol{y_k}$, (k = 1,..m). Based on these ranking scores, the k-NN classifier let them weighted voting on the correct class for the test point, where **weights** reflect **priors and cost** [2, 22]. Moreover, using the correlation coefficient method as an attribute **weighting criterion** enforces a **weighting** according to goodness of linear fit of individual attributes. Then, each attribute is tested individually, and its value is calculated by computing with the class attribute.

The k-NN classification algorithm, when given an unknown instance (tuple or record), is looking for the most common class among its k-nearest neighbors. Moreover, a k-NN classifier searches in the pattern space for the k training examples that are nearest to the unknown example. These k training tuples are the k "nearest neighbors" of the unknown example. Sometimes the term "closeness" is defined in terms of a distance metric, for example, Euclidean distance. The Euclidean distance between two points or tuples is **D(X,Y)=**$\boldsymbol{\sqrt{\sum_{i=1}^{k}(xi-yi)^{2}}}$ **.** Sometimes, we need to normalize the values of each attribute before using the Euclidean distance (D(X,Y)) equation to prevent attributes with initially large values from outweighing attributes with initially smaller values (e.g., binary attributes). For example, min-max normalization can be used to transform a value v of a numeric attribute A to V` in the range [0, 1] by computing V`=$\frac{V-Min\ (A)}{Max(A)-Min(A)}$ where Min(A) and Max(A) are the minimum and maximum values of attribute A. The min-max normalization (encoding) schemes are applied to obtain a reduced or "compressed" representation of the original data. The cost of having this bounded range is we will end up with smaller standard deviations, which can suppress the effect of outliers.

Moreover, the data should be normalized or standardized to help avoid dependence on the choice of measurement units. Normalizing the data attempts to give all attributes an equal weight. Normalization is particularly useful for classification algorithms involving neural networks or distance measurements such as nearest-neighbor classification and clustering. Normalizing the input values for each attribute measured in the training tuples will help speed the learning phase. For distance-based methods, normalization helps prevent attributes with initially large ranges (e.g., income) from outweighing attributes with initially smaller ranges (e.g., binary attributes). It is also useful when given no prior knowledge of the data [22].

The k-NN can be used to predict a numeric value, meaning it can return a real-valued prediction for a given unknown tuple. In this case, the k-NN classifier will return the average value of the real-valued labels (classes) associated with the k-NN of the unknown tuple [22]. As stated previously, the k-NN classification method calculates the distance (many types of distance) between the attributes of new and previous examples to determine the class. Therefore, the term “distance” can be used based on the entire data. But how we can define the distance for those attributes that are not numeric values (categorical) such as tumor tissue or normal tissue in cancer datasets? Consequently, different types of data require different methods for finding out the distance. For example, nominal (categorical) attributes only differ regarding whether they are identical or not (=, ≠). For ordinal attributes with ordered values, we cannot compute the distance between them (e.g., tall, medium, and short for an individual's height) and the difference cannot give an exact number, so we can only apply >,<,=,≠ to them [22, 27]. However, the Euclidean distance is the most popular distance measure function; we have different types of distance functions that are used to get a good learning system.

2.3.1 Distance function

We can mention some of the most common distance functions for the two inputs of **x** and **y** tuples, and n is the number of attributes [22, 27]:

- **Manhattan (or city block) distance** is defined as

  $$\mathbf{D(X,Y)=\sum_{i=1}^{n} | X_i - Y_i|}$$

- **Minkowski h-distance** is a generalization of the Euclidean and Manhattan distances) is defined as

  $\mathbf{D(X,Y)=\sqrt[h]{\sum_{i=1}^{n} \ (Xi - Yi)}^{\,h}}$ ***,*** where ***h*** is a real number such that ***h =1*** (which is the Manhattan distance) ***or h=2*** (which is the Euclidean distance). The larger value of ***h*** has the effect of giving greater weight to the attributes on which the objects differ most.

- **weighted Euclidean distance (**if each attribute is assigned a weight based on its importance) is defined as

  $$\mathbf{D(X,Y)= \sqrt{\sum_{i=1}^{n} Wi \ |Xi - Yi|^2}}$$

- **chi-square distance function** is defined as

  $$\mathbf{D(X,Y)=\sum_{i=1}^{n} \frac{1}{sum(i)} \left(\frac{Xi}{size(X)} - \frac{Yi}{size(Y)} \right)^2}$$

- **Cosine Similarity (Sim(X,Y)),** is defined as :

  $$\mathbf{Sim(X,Y)= \frac{X.Y}{||X||\ .\ ||Y||}}$$

2.3.2 The *k* and overfitting problem

The k-NN rule is usually used, and it assigns an instance to the class which is represented mostly by its ***k*** neighbors using a pre-determined distance function. The ***k*** can be any number of its neighbors, ***k***= 1, 2, 3, 4,…,n, where ***n*** is the number of cases. The results of the K-NN algorithm depend on what values are used in its computation. The value ***k*** is the number of neighbors that will decide the class of the element in the classification process.

The next question to ask is how to choose the value number of ***k*** in the k-NN classifier method to avoid an *overfitting* problem. The problem of overfitting is considered a fundamental problem in supervised machine learning, which means the classification method learns from the training examples ‘too well’ (over-trained classifier) so it does not perform as well when it is used with data unlike the examples. In the case of a bioinformatics domain, the goal is to induce the relationship between the symptoms and their corresponding diagnosis. It is an error to put the patient ID number as a variable selection in DNA microarray cancer datasets, so if by mistake this happens, then the classification and prediction process may conclude the illness is determined by the ID number [21]. However, we use ***k*** for a test set to estimate the error rate of the classifier. So ***k*** can be determined by experiments because it is a hyperparameter of a k-NN classifier that allows us to balance between overfitting (small value of ***k***) and underfitting (large value of ***k***). For example, using ***k=1*** when beginning the classification process and then determining the error rate. This process can be repeated each time by incrementing ***k*** to allow for one more neighbor. The ***k*** value returning the lowest error rate may be selected. In general, if we have a larger number of training instances, then we need a larger value of ***k*** (so using different ***k*** values are likely to produce different classification results) [22].

2.4 Ensemble of Classifiers

Ensemble methods have been increasingly applied to bioinformatics problems in dealing with small sample size, high-dimensionality, and complexity in data structures [28]. We can build many classifiers by combining them to produce a better classifier, so many classifiers are built and the final classification decision for each test instance is made based on some forms of voting of the committee of classifiers. Therefore, ensemble learning methods are used for training example classifiers on different datasets by using a resampling process for a common training set such as bagging and boosting methods[21].

The boosting ensemble (***AdaBoost* -** Adaptive Boosting) classification method manipulates training examples and produces multiple classifiers to improve classification accuracy [28, 29]. The ***AdaBoosting*** method constructs a good classifier by using repeated calls of weak learning procedures. It was initially developed as a method for constructing good classifiers by repeated calls to "weak" learning procedures [21, 29]. In general, the idea of a boosting classifier depends on a rule (classifier or base learner). AdaBoosting apply this base learner algorithm with a different distribution(threshold) and assign equal weight to each observation. Each time base learning algorithm is applied, it generates a new weak prediction rule. This is an iterative process. After many iterations, the boosting algorithm combines these weak rules into a single strong prediction rule (with smallest error) [26, 29].

Initially, the ***AdaBoost*** ensemble learning method constructs new training examples and gives them a weight. Then ***AdaBoost*** classifier invokes the base learner (rule) on the re-weighting training dataset and obtains a new classifier (for example **ft**). Afterwards the process of re-weighting is iterated. Thus, the algorithm builds a sequence of **k** classifiers and the **k** is usually defined by the user. The following is a popular pseudo code for an AdaBoosting algorithm [30].

```
function adaboost(dataset d, lable y, base learner :decision stump , k)
begin
%initialize the weights
initialize d1(wi)=1/n for all i ;
for t=1 to k do
% build a new classifier ft
ft=base learner(Dt);
% now compute the error of ft
et=$\sum_{i:f(Dt(Xi))<>Yi} Dt(wi)$ ;
if et>0.5      % the error is too large
%remove the iteration and exit
k=k-1 ;
exit loop;
else
βt=et/(1-et) ;
%update the weights
$D_{t+1}$(wi)=  Dt(wi) * $\begin{cases} \beta t & if\ ft(Dt(Xi)) = Yi \\ 1 & other\ wise \end{cases}$
% now normalize the weight
$D_{t+1}$(wi)=$\frac{\mathrm{Dt+1\ (wi\ )}}{\sum_{i=1}^{n} Dt+1\ (wi)}$
End if
End for
$F_{final}$ (x)=argmax y∊Y   $\sum_{t:ft(x)=y} \log \frac{1}{\beta t}$
End
```

From the previous algorithm, we can divide this algorithm into a **training and testing** phase. In the **training** phase, each classifier is dependent on the previous one and focuses on the previous one's errors. Training examples that are incorrectly classified by the previous classifiers are given higher weights. Each iteration builds a new classifier ***ft*** . The error of ***ft*** is calculated ***(et)***. If ***et*** it is too large (greater than 50%), delete the iteration and exit. Then we use the updated and normalized weights for building the next classifier (AdaBoost assigns a weight to each training example). *Testing* each stage, each classifier is combined to determine the final class of the test case ($F_{\mathbf{final}}$ (x)). Therefore, the AdaBoost ensemble method can be considered a meta-algorithm which can be used in combination with many other learning algorithms to improve their performance [30].

2.5 Bayesian Network (BN)

The Bayesian Network plays an increasingly important role in designing models and data analysis in the machine learning field because it is used to solve a problem of uncertainty and complexity in learning models [31]. BN is a graphical model based on probability (joint probability) and graph theory (directed acyclic graph [DAG]) which differs from the naïve Bayesian classifier by allowing the representation of **dependencies (conditional independencies)** among subsets of and attribute [22]. The idea of a BNs graph model (DAG) is used to show the collection of events and their influence on each other (graphical representation of [or conditional] independence relationships in a joint distribution). Each node in the DAG represents a random variable, where each arc represents a probabilistic dependence. However, probability theory is used to provide a way for a learning model to inference the new data by giving a description of how the variables are related to each other[22, 32].

In this context, this leads to techniques for learning causal relationships from data, for example, suppose there is an arc from node A to node B (A ⇒ B), indicating A (or a hidden variable) causes B, which means a causal relationships between A and B , and this helps the feature subset selection methods solve a prediction problem based on training data [32]. In the Figure (5) example, the arrows correspond with causal links between variables (i.e. smoking status or family history – causes Leukemia cancer). So, Bayesian networks are particularly fit for representing domains where there are causal relationships to predicate a consequence of giving actions. It is useful to think of causal relationships when we try to build a Bayesian network that represent a problem. However, in causal structure learning, we are interested in graphically representing conditional dependencies found in the data.

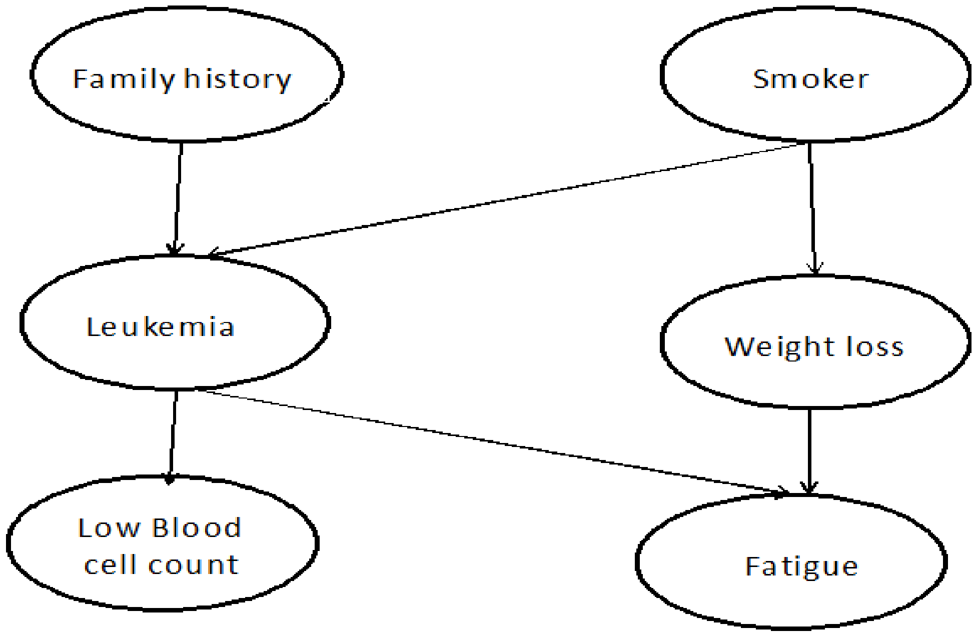


Figure 5 : Causal model, represented by a Directed A cyclic Graph (DAG)

2.5.1 Joint probability and conditional independence

A **BN** uses a conditional independence table for each random variable. Suppose A and B are conditionally independent given a set of random variables C, denoted as A ⊥ B| C, if P(A,B| C) = P(A| C)P(B| C), for all assignments of values to A, B, and C. If C is the empty set, then A and B are independent, denoted as A ⊥ B [23]. We can say the simplest conditional independence relationship is used in a **BN,** which can be stated as follows:

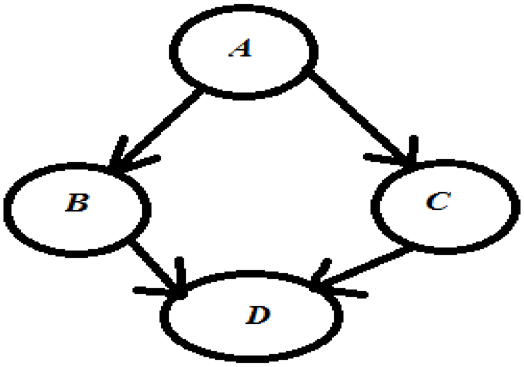


Figure 6: DAG for BN conditional independinces

Based on DAG in Figure 6, any node is considered independent of its ancestors given its parents, where the ancestor/parent relationship is with respect to some fixed topological ordering of the nodes. Therefore, the joint probability (chain rule of probability) of all the nodes in the graph is

$$\mathbf{P(A, B, C, D) = P(A) * P(B|A) * P(C|A,B) * P(D|A,B,C)}$$

We can write this equation in its general form as

$\mathbf{P(X1,\ldots.,Xn) = \prod_{i=1}^{n} p(xi \mid parents(Xi))}$, **This form is called general factorization.**

By using conditional independence relationships, we can rewrite this as

$$\mathbf{P(A,B,C,D) = P(A)* P(B|A) * P(C| A) * P(D|B,C)}$$

To simplify conditional independence, let us take part of the probabilistic graphical models from Figure 6-**(A, B,C)** and assume the hypothesis of independence $\mathbf{H : B \perp\!\!\!\perp C|A}$ is true, and it means variable *B* has a conditional independence with C given A. Then the conditional independencies for this part of the graph model is

$$\mathbf{P(B,C \mid A) = P(B \mid A)\, P(C \mid A)}$$

We can see that the conditional independence relationships allow us to represent the joint probability more compactly. The BN depends on the graphical model to define the probabilistic independence relationship (casual induce) among the variables and represents the joint probability distribution factorized in terms of the graph model [31, 33].

2.5.2 V-structure

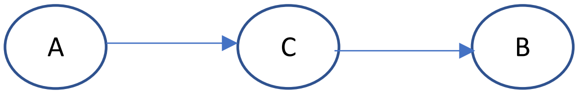
The most important concept in DAGs is the V-structure, which denotes a variable having two parents which are *not connected* by an edge. In Figure 3, for example, nodes B, C, and D implement the V-structure.

2.5.3 D-separation

In a DAG, independence is encoded by the relation d-separation, and we can define it as

$$A \perp B \mid C \;\Leftrightarrow\; A \text{ d-separated from } B \text{ by } C$$

D-separation means that knowing the value of A is d-separated from B by C if all the paths between sets A and B are blocked by elements of C, and vice versa [34, 35].



Chapter 3

Feature selection methods

3.1 Goals of feature selection

Machine learning and data mining techniques deal with extremely massive datasets. These data can suffer from high dimensionality (many features and instances), which affects the performance (usually the accuracy) of classification due to noisy irrelevant and redundant features. In this case, when the dataset is very large, many learning algorithms are simply intractable and time consuming. Moreover, this causes the classification algorithm to overfit the training dataset which confuses the learning process. Consequently, the demand of an efficient algorithm for feature subset selection techniques is increased to get the optimal (minimal) features subset selection. These optimal features selections can be fed into the classifier to help reduce the induction time, thereby increasing predicative accuracy and reducing the learning process' complexity [1, 11]. Furthermore, feature subset selection methods can be used as data understanding to identify factors relevant to the target [32].

3.2 Feature selection categories

Feature subset selection methods typically fall into two approaches, *feature ranking* and *subset evaluation method*. Feature ranking is a method of ranking all features depending on their importance in a set of different samples of different class labels. Features which do not get an adequate score and have similar rankings can be removed and considered irrelevant and redundant features. Specifically, this approach depends on selecting the most relevant features where usually relevance does not imply optimality[11, 15, 31]. The method of subset evaluation method is performed by searching in the space of possible feature subsets for an optimal subset based on some criterion (objective function) and goodness measure. This method can remove irrelevant features as well as redundant ones[19, 31].

Essentially, Feature subset selection methods can be classified into two methods based on whether the feature subset selection method is using learning algorithm or not (based on some statistical measurements). So we can divided the feature subset selection method into **filters and wrappers** methods[11, 32].:

Filter method: this method does not have an induction algorithm, which means feature selection is done independently of the learning algorithm (filters work independent of the chosen classifier). Filter method use statistical properties to define the scores of feature relevance, and those of high-scoring features are selected as inputs to the classifier after removing low-scoring ones. This method can be used as a preprocessing step, and is considered computationally more efficient to scale a high-dimensional dataset. However, the drawbacks of this method is an optimal subset of variables will be dependent on the learning algorithm's representational biases used to build the classifier. This means it contributes less information and estimates each feature's relevance independently (separately) from others, which may lead to a lower classification performance when compared to other types of feature selection techniques [11, 14, 17].

Wrapper method: this method uses an induction(classification) algorithm to evaluate the score of a possible features subset regarding to their predictive power. This method evaluates each subset feature through a specific classification learner algorithm to measure the goodness of feature subset in determining an optimal one. In general, the wrapper method consists in using the prediction performance of a given learning machine to evaluate the relative goodness of subsets of features. The evaluation process in wrapper method is obtained by training and testing a specific classification model. Therefore, the wrapper method evaluates and selects subset of features based on accuracy level which are estimated by the target classification algorithm. Using a certain classification algorithm, wrapper method basically searches the feature space by omitting some subset of features and testing the impact of subset of feature omission on the classification algorithm performance. The subset of features that make significant difference in learning process implies it does matter and should be considered as good subset features. So, in the search space, the search algorithm is defined to be "wrapped" around the classification model [14, 15, 17, 36]. Many studies have found that the wrapper method provides a better solution than the filter method because it uses a classification algorithm for evaluation in the subset feature selection process. As a result of wrapper method can be more computationally intensive because training model and cross-validation must be repeated over each feature subset, and the outcome is assessed to a particular model. Accordingly, sometimes become so costly as to be impractical without pre-reduction of the search space with a filter method[37].

The wrapper method depends on different types of search techniques like a greedy search for forward feature selection (or backward feature elimination) or stochastic search like genetic algorithms (GA). In the forward feature selection example, it is an iterative method in which we start with an empty feature subset in the model and in each iteration, we keep adding the feature which best improves our model until an addition of a new variable does not improve the performance of the model. However, in the backward elimination example, we start with all the features and remove the least significant feature at each iteration which improves the model's performance. We repeat this until no improvement is observed after each removal of features. The wrapper method involves high computational overhead to define and select a candidate feature, but in most cases, provides better results than filter methods [14, 15, 17]. In this method, we can use genetic algorithms (GA) as search optimization tools to offer and effective approach for solving large scale problems. GA can be used by feature selection methods as an optimization tool to find an optimal feature subset as we will see in Section 3.7 [38].

3.3 Feature relevance and redundancy

Fundamentally, the problem is finding the feature subset of a minimum subset that preserves the information contained in the whole set of features with respect to target class. We can solve this problem by finding the relevant features and discarding redundant (or irrelevant) features [1]. In this context, many strategies and algorithms can be defined to solve these problems. *Filtering* algorithms concentrate on removing irrelevant variables. Another strategy can be used by the *ranking method,* where the concentration is defining and obtaining the relative relevance of features for all input features with respect to the target one. Therefore, we might be interested in a compact, effective model, where the goal is to identify the smallest subset of independent features with the most predictive power, although a few alternative groups might be reasonable[38, 39]. In this part, we depend on *Koller and Sahami*, 1996 [1] and *Kohavi and John*, 1997 [15] to review the different definitions of relevance and redundancy found in the literature.

3.3.1 Feature relevance

In 1997, *Kohavi and John* [15], showed the classification of input variables ***F*** with respect to their relevance to the target ***C*** in terms of conditional independence. They used a probabilistic framework to define three levels of relevance: strongly relevant, weakly relevant, and irrelevant features. Let **F** be a full set of features, **F_i** a feature, and **S_i =F-(Fi)**. Then, these categories of relevance can be formalized as follows:

- **Definition 1 - Strong relevance*:*** *A variable **F_i** is strongly relevant to the target **C** if*

$$\mathbf{P(C|F_i,S_i) \neq P(C|S_i)}$$

- **Definition 2 – Weak relevance**: *A variable $\boldsymbol{F_i}$ is weakly relevant to the target **C** if it is not strongly relevant*

$$\mathbf{P(C \mid Fi, Si) = P(C \mid Si),} \text{ and}$$

$$\mathbf{P(\, F_i = f_i, \acute{S}_i = \acute{s}_i) > 0} \text{ , and } \exists\ \acute{S}_i \ \square S_i \text{ , such that } \mathbf{,\ P(C \mid F_i, \acute{S}_i) \neq P(C \mid \acute{S}_i),}$$

- **Corollary 1 (Irrelevance):** *A feature $\boldsymbol{F_i}$ is irrelevant if*

$$\forall\ \acute{S}i \ \square Si,\ \ \mathbf{P(C \mid Fi, \acute{S}i) = P(C \mid \acute{S}i),}$$

A feature is considered irrelevant if it provides no information on the target class at all.

From previous definitions, the **strongly relevant features** provide unique information about C, which means they cannot be replaced (or removed) by other features without affecting the original conditional class distribution. **Weakly relevant features** provide information about C, but they can be replaced by other features without losing information about C. **Irrelevant features** do not provide information about C, and they can be discarded without losing information. The disadvantages of the probabilistic approach are that for each feature subset we need to test conditional independence and define the probability density functions [11, 17, 34]. We can use a framework of ***mutual information*** and ***entropy*** to solve these drawbacks in the probabilistic approach to feature subsets as we will see in the next section [15, 17] .

3.3.2 Feature redundancy

It is clear from definitions of feature relevance that an optimal subset should return all strongly relevant features and a subset of weakly relevant features. However, it is not given in the definitions which weakly relevant features should be selected and which should be removed. Therefore, it is necessary to define feature redundancy among relevant features [11].

In the next section, we explain our goal of feature subset *redundancy elimination* which is focusing on important cases with redundant features and obtain at least one of Markov blanket (*MB*) in weakly relevant features. *Koller and Sahami* in 1996 [1] and Lei *Yu and Huan Liu* in 2004 [11] described the solution to feature subset selection by obtaining a minimal Markov blanket to identify and eliminate redundant features. However, the MB is not a unique method to deal with redundant features, sometimes the redundant features can be defined in terms of features correlation [11, 39].

3.4 Introduction to Markov Blanket

Unfortunately, BN learning is extremely computationally expensive. This is because the network structure must have prior knowledge of each node (variable). Furthermore, Bayesian networks tend to perform poorly on highly dimensional data, which leads to finding the optimal subset of features intractable. Finally, Bayesian network models can be hard to interpret, and require separating effects between different parts of the network [11, 34]. From this point, we can use the Markov blanket for feature subset selection to avoid a lot of computations by selecting the most minimal set of relevant subset features, which lead to a strong performance of the classification measure[1, 34].

From a theoretical and practical perspective, many feature selection methods are heuristic (forward selection or backward elimination) in nature because they are working without knowing what consists an optimal feature selection solution independently of the class of models fitted, and under which conditions an algorithm will output such an optimal solution. So we can use the

Markov blanket(*MB*) to detect a minimum size subset of features that might be considered a feature selection solution and use them to maximize predictive (classification) performance [40].

Let us define a set of features. $\mathbf{F =(F_1,….,F_n)}$,**f** is a set of values **f=(f1,…,fn),** and asset of class labels $\mathbf{C =(c_1….,c_r\ )}$. We can use the probability distribution for each value **f** to **F** as **P(C|F=f)**. Let **G** a subset of **F**, for example **F=(A, B,C,D)** an**d G=(A)**. A feature vector **f=(a,b,c,d),** so **f(A)=a**. The conditional probability distribution can be denoted as ***P(C|F=f)***. In the reduced feature space, the same instance induces the $\mathbf{P(C|G=F_G)}$.

We can depend on probabilistic reasoning[41] to define the set of features that cause a small increase in ***Δ*** as those that give us the least additional information beyond what we would obtain from other features in G. We use the definition of Markov blanket as mentioned in the work of *Koller and Sahami* [1] and *Yu and Liu* [11] as the following. Initially, we define a form of probabilistic reasoning. We have a set of conditionally independent variables **A, B**, and **X** if for any assignment of values ***a,b*** and **x** respectively **P(A=a| X=x, B=b) = P(A=a| X=x)**. This means **B** gives no information about **A** beyond what is already in **X**. Specifically, if we remove a feature *Fi* that is conditionally independent of class label *C* without effect on a distance from the desired distribution. While it is also impractical to find conditional independencies for all remaining features. We can utilize from Markov blanket concepts within a set of features which is consider stronger than conditional independencies to remove unnecessary features [1]. So, at any phase, if we find a Markov blanket for *Fi* within the current *G*, *Fi* is removed from *G* [11].

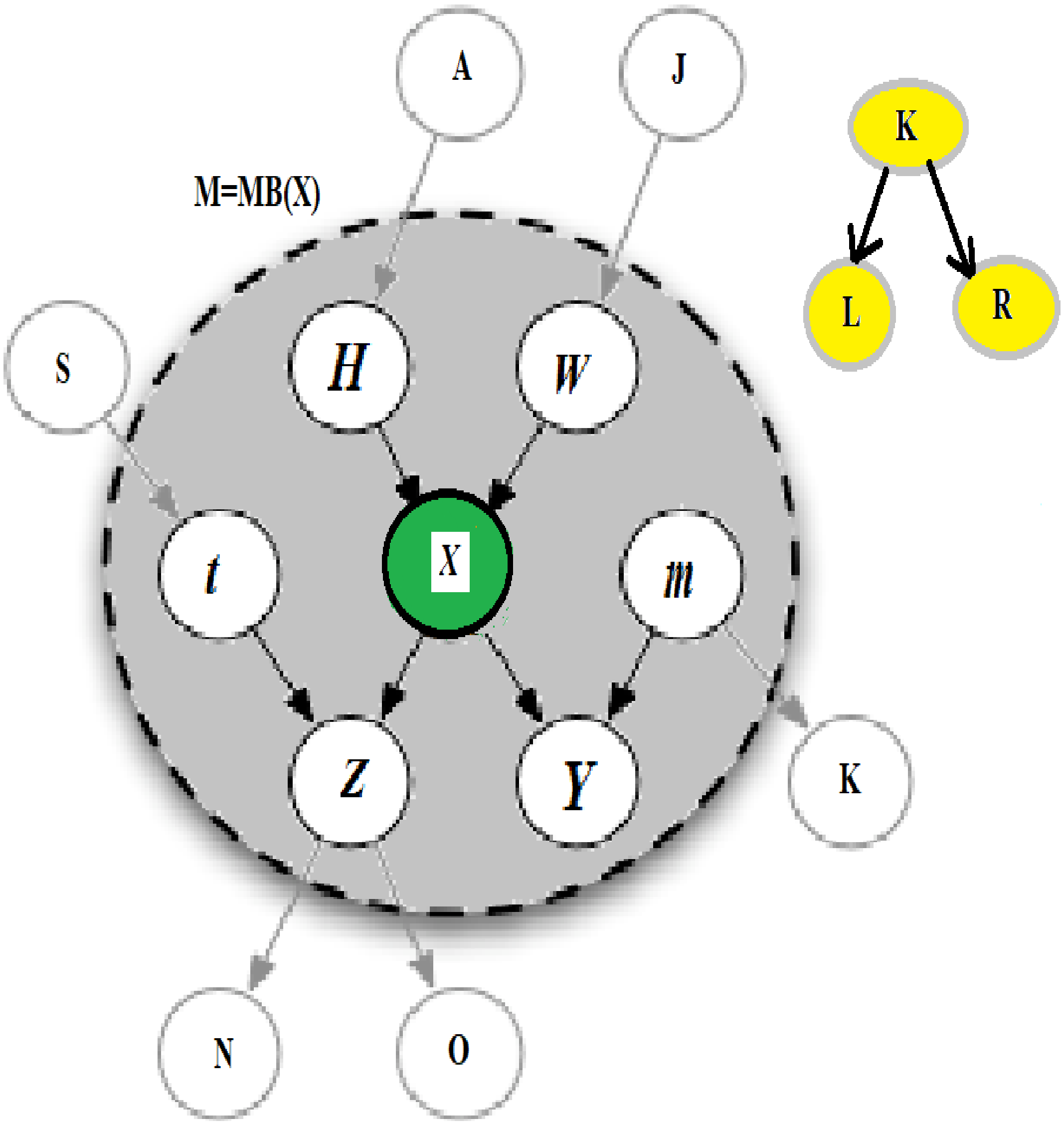


Figure 7: Implement Bayesian Network based on Markov Blanket(MB) for the variable X (Target) : Consist of all nodes that makes X is conditionally independences in the model

One of the most common methods to a model and induce causal relations is by causal structure. Learning Bayesian networks which aims to build a directed acyclic graph (DAG) showing direct causal relationships among the variables of interest of a given system. Based on Bayesian network, Markov blanket and DAG, it is possible to determine and explore causal relationships which is the central in probabilistic reasoning and decision making. The goal of determining causal relationships is predicting the consequences of given actions or manipulations. The goal of causal discovery and feature selection is specifically to uncover causal relationships between variables for one of several purposes (understanding data) [32]. A Bayesian network and Markov blanket can be considered an essential graph, where the directed edges in the graph represent the causal relations on which all equivalent networks agree upon their directionality and all the remaining edges are undirected. The range of datasets typical algorithms can deal with is restricted, meaning, no probability distribution can be *faithfully* represented by a *DAG*. *Faithfulness* of the distribution is a well-defined condition: it guarantees the existence of a *DAG*, called a *perfect map*, where there is a one-to-one mapping between the graphical criterion of *d*-separation and conditional independence in the data [34].

In Figure 7, we can see the relationship between causal structure and predictivity in faithful distributions. The ***X*** variable is a member of Markov blanket ***M***. They are depicted inside the black dotted circle (i.e. variables with have and undirected path to target ***X*** and are predictive of ***X*** given the remaining variables which makes them strongly relevant). Markov blanket variables include direct causes of ***T (H,W)***, direct effects (***Y,Z***), and "spouses" of ***X*** (i.e. direct causes of the direct effects of ***X***) (***m,t***). The variables outside the dotted circle are non-members of Markov blanket of ***X*** that have an undirected path to ***X***. They are not predictive of ***X*** given the remaining variables, but they are predictive given a subset of the remaining variables (which makes them weakly relevant). The Cyan variables are variables without an undirected path to ***X***. They are not predictive of ***X*** given any subset of the remaining variables, thus they are irrelevant.

Based on the definition of feature subset selection and causal structure learning, we can identify common concepts for those terms. The *Markov blanket* of a variable *X* is the smallest set **Mb**(*X*) containing all variables carrying information about *X* that cannot be obtained from any other variable (see Figure 7). Based on the perspective feature subset selection, this leads to selecting the set of *strongly relevant* features carrying information about the target feature that cannot be obtained from any other feature. However, based on the causal graph, this approach leads to defining the set of all parents, children, and spouses of *X*. Both tasks of feature subset selection and causal graph construction can be specified to some extent as Markov blanket identification tasks [34]. We believe analysis of observational data using the Markov blanket for feature selection can help guide obtaining more accurate results and designing proper experiments.

Definition 3: Let **M** be some set of features where **F_i** does not belong to ***M***; we can say that ***M*** is a Markov blanket for **F_i** if **F_i** is conditionally independent of **F-M-$\{F_i\}$** given ***M*** (see figure 7). We say a set of features ***M*** that does not contain **F_i** is a Markov blanket for **F_i** if **F_i** is conditionally independent of everything not in ***M***, given ***M***. We also can say the class ***C*** is conditionally independent of the feature **F_i** given ***M*** [11] .

$$P(F\text{-}M_i – \{ F_i \},C \mid F_i,M_i) = P(F\text{-}M_i - \{ F_i \}, C \mid M_i)$$

From the previous paragraph, we can explain the reasons for using a Markov blanket in feature subset selection techniques. The class ***C*** becomes conditionally independent of **F_i** given ***M***; this means **F_i** gives no additional information about the class when ***M*** is known, and we can remove it. On the other hand, it can be also proved that removing a feature cannot render previously removed features relevant again.

Definition 4- redundant feature: let ***G*** be the current set of features; a feature is redundant and hence should be removed from ***G*** if it is weakly relevant and has a Markov blanket **M_i** within ***G.***

Therefore, in this paper, we focus on how to eliminate the redundant (or irrelevant) features by defining the Markov blanket based on the Bayesian Network. In the field of machine learning, the problems are defining the Markov blanket exactly and identifying how we can measure feature relevance, which is considered a difficult process due to a limited sample size, high time complexity, and noise in the data [41]. Therefore, we will use the Markov blanket criterion that helps us remove unnecessary features (namely, irrelevant and redundant features) to obtain optimal classification [32]. For a Markov blanket example, see Figure 7. More formally, the MB of **X** (target) insists of all nodes that makes **X** conditionally independent of all other nodes in the model (Parent **[H,W]**, children **[Z,Y]**, and spouse **[m,t])**. Each feature in the Markov blanket measures the coverage of its blanket (scores), then the feature with the lowest score (highest coverage) will be considered redundant and then removed from the dataset. This procedure will be iterated until criterion stopping occurs (no longer possible create Markov blanket) [11]. Moreover, we can demonstrate that the classification-level when using the Markov blanket of a target variable in a Bayesian network has significant properties. We can obtain a statistically efficient prediction result of a feature's probability distribution from the smallest subset of features selection containing all the information about the target feature. We can obtain a greater accuracy level through avoiding overfitting due to redundant variables, and it provides a classifier of the target variable from a reduced set of predictors [11, 16].

Definition 5- (Faithfulness) A Bayesian network with a DAG ***S*** and a joint distribution ***P*** with a set of vertex ***V*** are faithful to one another if there are no conditional independence relations in ***P*** other than those entailed by satisfying the Markov condition for ***S***. Given the faithfulness assumption, the set of distributions represented by ***S*** is the set of distributions satisfying the Markov condition for ***S***. If p is faithful to the graph ***S***, then given a Bayesian Network ***(S, p)***, there is a unique Markov Blanket for ***Y*** consisting of the set of parents ***(Y)***, the set of children ***(Y)***, and the set of parents of children ***(Y)[42].***

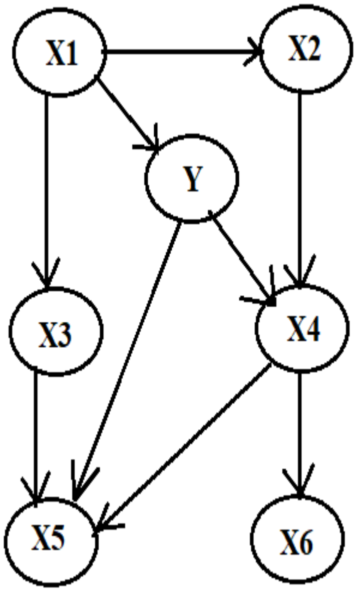


Figure 8.1 BN(S,P)

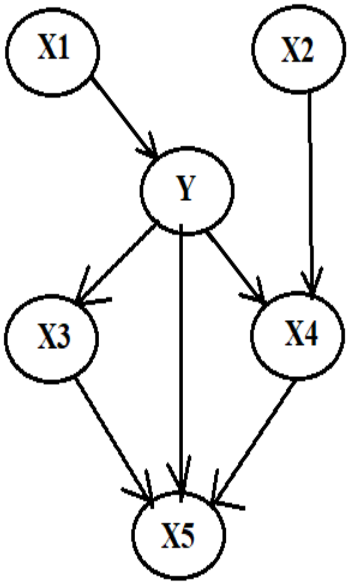


Figure 8.2 Unique MB(Y)

Figure 8 : *If* **p** *is faithful to the graph* **S***, then given a Bayesian Network (***S***,* **p***), there is a unique Markov Blanket for* **Y** *consisting of the set of parents (***Y***) ; the set of children (***Y***) ; the set of parents of children (***Y***) .*

From the example in Figure 8, consider the two DAGs in Figure ***8.1*** and ***8.2***, above. The factorization of ***P*** entailed by the Bayesian Network ***(S, P)*** is

$$P(Y,X1, ...,X6) = P(Y\,|X1) \cdot P(X4|X2, Y\,) \cdot P(X5|X3,X4, Y\,) \cdot P(X2|X1) \cdot P(X3|X1) \cdot P(X6|X4) \cdot p(X1)$$

The factorization of the conditional probability ***P(Y |X1, ...,X6)*** entailed by the Markov Blanket for ***Y*** corresponds to the product of those (local) factors in the previous equation containing the term ***Y*** .

$$P(Y\,|X1, ...,X6) = C' \cdot P(Y\,|X1) \cdot P(X4|X2, Y\,) \cdot P(X5|X3,X4, Y\,)\,;$$

where ***C'*** is a normalizing constant independent of the value of ***Y[43]***.

More formally, the Bayesian network with DAG ***S*** has to be faithful, if all dependencies (or conditional dependencies) are represented and independencies entailed by the distribution, such a graph is called a ***perfect map*** of the distribution if there is a one-to-one mapping between the conditional-independence relationship defined by variables and the *d-separation criterion* defined by the graphical nodes [34].

3.5 A Correlation Based Method

Feature selection based on a correlational method can be used to evaluate the usefulness of individual features and the goodness of feature subsets through using a heuristic searching algorithm for predicting the class label based on intercorrelation between those subset features. Based on the definition of a Markov blanket, we can deal with *folk-theorem* (some features with direct influence are stronger than indirect influence [conditional independencies]) to measure the influences between the Markov blanket (*Mi*) and asset of features (*Fi*). Therefore, those influences can be measured by using a correlational method [1].

3.6 Using entropy and mutual information

Entropy is used to measure the uncertainty of a random variable, while uncertainty is a measure of the probability occurrence of an event. the entropy of a discrete random variable $\boldsymbol{x}$, with mass probability

$\mathbf{P(x(i)) = P\{x = x(i)\}, x(i) \epsilon x}$ is defined as:

$\square\mathbf{(x)}= -\sum_{i=1}^{n} P(xi)\ \mathbf{log_2}(P(xi))$ **;** needed to identify

The expected information (**□(x))** is needed to classify the class label of a tuple in ***x.*** (**□(x))** is also known as the *entropy* of ***x***. Now let ***x*** and ***y*** be two random discrete variables. The joint entropy of ***x*** and ***y***, with joint probability ***P(x(i), y(j)),*** is the sum of the uncertainty contained by the two variables.

$$\square\mathbf{(x,y)}= -\sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{P(x(i),y(j))} \ \mathbf{log_2}(\boldsymbol{P\big(x(i),y(j)\big)})$$

The maximum value of □***(x,y***) happened when the ***x*** and ***y*** is completely independent. The minimum value of □***(x,y)*** happened when the ***x*** and ***y*** is completely dependent (equal to zero). Conditional entropy measures the remaining uncertainty of the random variable ***x*** when the value of the random variable ***y*** is known. Formally, conditional entropy is:

$$\square\mathbf{(x|y)}=\sum_{j=1}^{n}\boldsymbol{P\big(y(j)\big)}.\ \ \square\mathbf{(x|y=y(j))}\ ,\ \text{ where }\ \mathbf{0} < \square\mathbf{(x|y)} < \square\mathbf{(x)}\ .$$

Mutual information (MI) is a measure of the amount of information that one random variable has about another variable. In the information theory field, it is widely used to measure the mutual independency of two subset features. Intuitively, it measures how much information a random feature tells about the other [22]. Mutual information(MI) can be given by

$$\mathbf{MI(X,Y)}=\sum_{i=1}^{n}\ \sum_{j=1}^{n}\boldsymbol{P\big(x(i),y(j)\big)}.\,\mathbf{log}\Big(\ \frac{\boldsymbol{P(x(i),y(j))}}{\boldsymbol{P\big(x(i)\big).\ P(y(j))}}\Big)$$

From this definition, it detects the relevance of a feature subset with respect to the response vector ***C*** [11, 17, 41]. The amount by which the entropy of *X* minimized the information needed about *X* provided by *Y* and is called *information gain (IG) [11]*.

$$\mathbf{IG(X|Y)}=\ \square\mathbf{(x)}\ -\ \square\mathbf{(x,y)}$$

3.7 Evolutionary and Memetic Algorithm for feature selection

Evolutionary algorithms are stochastic optimization methods that attempt to mimic in some way the inner workings of evolution, as we understand it. One component common to all these algorithms is the generation of random perturbations, or mutations, and the presence of a fitness function used to assess the quality of a given point and filter out mutations that are not useful. In this sense, random descent methods and even simulated annealing can be viewed as special cases of evolutionary algorithms. One of the broadest subclasses of evolutionary algorithms is genetic algorithms (***GA***) [9]. *GA* can be used as an attractive approach for feature subset selection depending on some criterion optimization function (e.g., accuracy of classification or number of selected features) because it is very effective when used for global searching and solving large-scale problems [14, 20].

GA often takes a long time to locate the local optimum in a region of convergence. Convergence is a phenomenon lead to restart the population, which happens when all individuals of a population are very similar to each other. In this case, the GA will expand most of the time to do the resampling process with a limited region of the search space, with subsequent waste of computational efforts. Consequently, to find a solution to a convergence problem and find the local optimum, we can use hybridized *GA* with some local search approaches which is called a memetic algorithm (MA) [9, 33].

Memetic algorithms (*MA*) utilize knowledge available (exploitation) about the *NP* optimization problem and try to find an approximate solution. Therefore, we can consider *MA* a search strategy which is a population-based metaheuristic composed of a Genetic Algorithm (wrapper method) and a set of local search algorithms (filter method) which are activated within the generation cycle of the external framework [44, 45]. We can say MA is synergistic with two different types of incorporated search algorithm approaches (population search, local search) to solve a NP hard optimization problem. Therefore, *MA* has the ability to run different types of search operators in a search space and make a collection of candidate solutions (chromosomes) to improve the solution of a *NP* hard optimization problem more precisely and efficiently [9, 20].

Pablo Moscato gave an early definition of *MA[45]*. *MAs* were a modification of Genetic Algorithms (*GAs*) also employing a local search operator for solving the Min Euclidean Traveling Salesman problem. While the role of optimization is employing a hybrid genetic algorithm (*GAs*) and simulated annealing (*SA*) [45]. In the previous definitions, it is clear that *MA* is a developing process of *GA* and gives rise to the notions of neighborhood-based local search and population-based search. The *GA* (or *MA*) process depends on fitness function and some operators to evaluate the candidate subset of feature by its ability to obtain a good classification. To get the fitness of an individual, we use a *KNN* classifier is trained with this representation and its classification accuracy is estimated by 10-fold cross validation and 30-Bootstrapping methods that allows assigning measures of accuracy. After that, the *GA* will be repeated many times until get a satisfactory solution to the problem is found or some other termination criteria are met (number of generations reaches maximum generation) [38]. Practically, we can show the *MA's* goal is effectively utilizing useful information from different feature selection methods to select a better subset feature with a smaller size and a better classification performance than the individual feature selection algorithms. Moreover, *MA* helps find feature subsets which are considered the most appropriate for a target learning algorithm. The framework is applied to several types of gene selection examples. The experiments applied to different types of cancer microarray data show the proposed *MA* feature selection method is capable of achieving the highest goal [10, 38].

3.7.1 GA history and definition

Genetic algorithms (GA) were introduced by Holland, and mimic nature's evolutionary method of adapting to a changing environment. He introduced the population search algorithm based on crossover, and mutation operators was a major innovation [46]. The adaptive idea or natural selection (based on crossover and mutation operators) was inspired by many science fields (e.g., biological, engineering, artificial intelligence) to develop high quality solutions for solving complex problems and use these solutions in the face of a changing environment [33, 46]. The parameters and operators of Genetic algorithms (GA) can be modified to get suitable data and obtain the best performance or the best search results. This heuristic search is routinely used to generate useful solutions to optimization and search problems by using a genetic algorithm [47]. GAs use a ***fitness function*** to evaluate the candidate solutions and find an optimal or high-quality solution among these candidate solutions. Many authors have used GAs as optimization tools which depend on population heuristic search techniques to utilize in a feature subset of selection methods [15, 38]. Therefore, using the GAs to evaluate a weight for each feature and getting a population of candidate solutions which are combined by using crossover, and varying solutions by mutation operators will increase the performance of classification algorithms [38].

3.7.2 GA operators

The simplest form of genetic algorithm involves three types of operators: selection, crossover (single point), and mutation [46].

- **Selection** This operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce.

- **Crossover (binary recombination)** exchanges information contained in two parents (or more), then combines the two chromosomes to create two **offspring**. For example, the strings 10000100 and 11111111 could be crossed over after the third locus in each to produce two offspring: 10011111 and 11100100. The crossover operator roughly mimics biological recombination between two single-chromosome (haploid) organisms. We can distinguish three types of crossover which are single point crossover, double point crossover, and multi point cross over.
- **Mutation** This operator randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string but usually with individual that has small probability.
- **Local offspring enhancement:** The goal of local modification is obtaining another offspring solution depending on a local neighborhood search. Candidate solutions undergo refinement which corresponds with the life-time learning of the individuals in the original metaphor of *MAs*. This is a very powerful metaphor that increase the performance of fitness land space by obtaining a local optimum and is considered a guiding function value which is better than value of all it neighbors.
- **Update of the population:** a new solution is added to the population where the existing old solution in the population should be replaced. Often, these decisions are made according to criteria related to both quality and diversity. The goal of diversity in the population of solution is avoiding premature convergence in the locality search (i.e. convergence that is too rapid towards a suboptimal region of the search space), and to help the algorithm continuously discover new promising search areas.

Chapter 4

Methodology

In this work, our methodology shows two approaches. The first one depends on *MA*, while the second one depends on *MRMR* for the feature subset selection method. Both approaches depend on hybridized GA with different types of local search operators. In both approaches, we compare the performance of each with a Bayesian network based Markov blanket for the feature selection method.

4.1 Memetic algorithm (MA) representation and operators

We depend on the ***MAFS*** (memetic algorithm for feature selection) software program to compare two types of hybridized *GA* with a local search [45]. The first approach of *MA* depends on the wrapper method called wrapper filter feature for selection algorithm (*WFFSA*) and the second MA approach is the Markov Blanket embedded genetic algorithm (*MBEGA*). We use different types of microarray cancer datasets [48, 49] and the *MAFS* software runs memetic algorithms to remove redundant (or irrelevant) features or add relevant features. Then, the performance between these two approaches is compared.

In this study, we depend on the algorithm used in [38] to outline the steps of how the *MA* process works. The MA for feature subset selection pseudo code can be implemented in algorithm 1. At the beginning, we randomly initialize the population *(P)* solution of gene selection. Each solution is considered a candidate chromosome in gene subset selection. Depending on fitness function, it selects the best individual from population *(P)*. At each generation, a new population is replaced with a previous one depending on some *MA* operators (crossover, local search, and mutation). This process is iterated until the maximum number of generations is reached.

Algorithm1 : Pseudo code for a memetic algorithm procedure

```
Procedure MA
begin
generate random population of P solutions (chromosomes);
for each individual i∈P: calculate fitness(i);
for j=1 to #generations
for each individual i∈P: do i=Local-Search(i);
for crossover
select two parents ia, ib∈P randomly;
generate offspring ic=Crossover(ia, ib);
ic=Local-Search(ic);
add individual ic to P;
end for
for mutation
select an individual i∈P randomly;
generate offspring ic=Mutate(i);
ic=Local-Search(ic);
add individual ic to P;
end for
P=select(P);
j=j+1;
end for
end
```

4.1.1Parameter Settings

Our experiments used the following parameter settings. Population size: 200, number of generations: 200, minimum number of attributes: 1, maximum number of attributes: 50, crossover type: a uniform crossover, probability of crossover: 0.6, and probability of mutation: 0.03.

4.1.2 Objective Function and Fitness evaluation:

In this study, we use a k-nearest-neighbor (KNN) classifier to train the example and define the classification accuracy that is estimated by **10-*fold cross validation*** and **30-*runs Bootstrap validation*** tests. KNN was chosen because of its flexibility metric, simplicity, and competitive performance, compared with another classification algorithm [22]. We use K=3 to define the number of neighbors. The fitness function can be written as

$$\textbf{Fitness}_{(MA)} = \mathbf{J_s} \textbf{ (k-NN classifier, define the accuracy)}$$

where J_s denotes the fitness function used to obtain the gene subset selection, provides a measure on the *k-NN* classification error for the given gene subset **s.** For a given subset the evaluation is done by the feature selection criterion function J_s. In this paper, J_s is specified as the accuracy rate for the selected feature subset.

4.1.3 Local improvement offspring function

The goal of using a local search operator is improving the quality of an offspring gene selection (chromosome) as much as possible. Therefore, the local search procedure takes the current solution’s selected offspring and then iteratively replaces this current solution by another better solution taken from a given neighborhood. This process is used to get the local optimum. For this, we use two types of local search methods. The first one is *WFFSA* and the second one is *MBEGA.* Algorithm ***2*** shows the pseudo code for a local search procedure. The local improvement of offspring used to improve the performance of classification during feature subset selection. The move operator, which defines the neighbor, is called (add/remove) operator which defines the relevance of genes for classification. This is typically denoted by $\mathbf{S_1}$=$\mathbf{S_0}$ $\oplus$ mv [9].

Algorithm 2: Pseudo code for a local search procedure

```
Procedure local search engine(current)
Begin
Repeat
Input : an new individual (current )
Output : new improvement (current)
If (output is better than input)
Input =mv $\oplus$ output
End if
Until (stop criterion )
Return  input as output result
```

To simplify these algorithms (1,2) to get optimal solutions for different types of feature subset selection methods, we can show the flowchart for the proposed algorithms as in figure 9 .

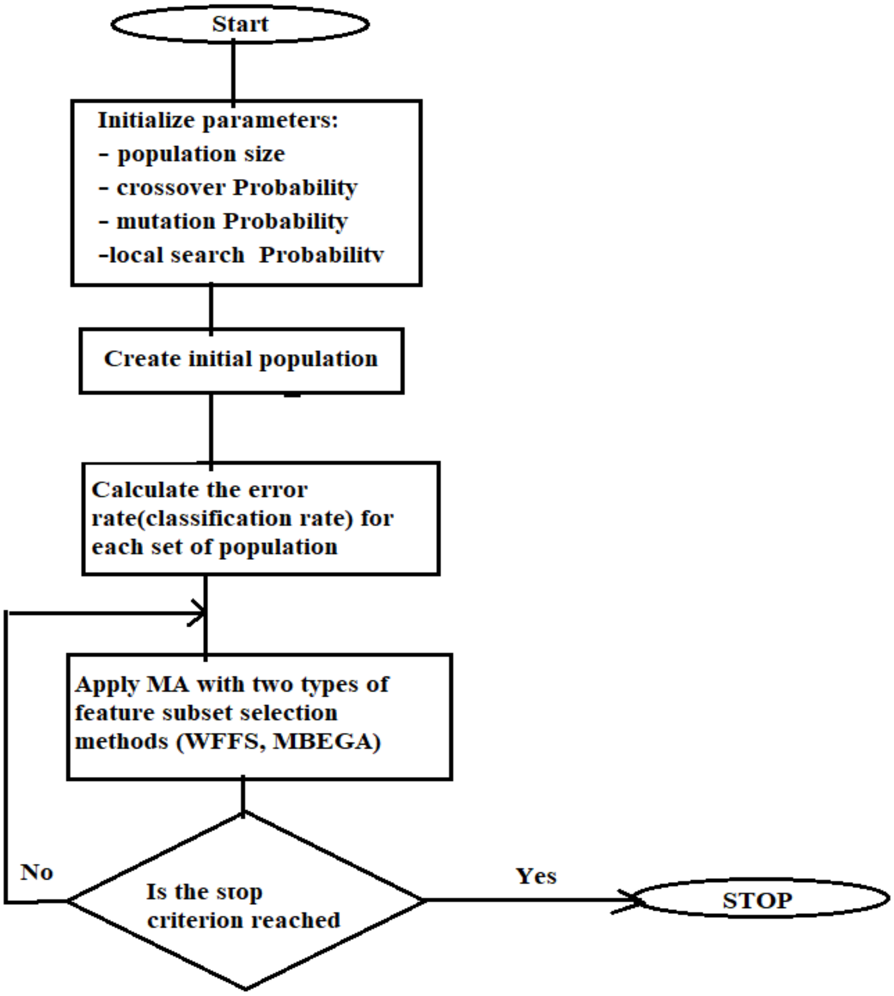

Start
Initialize parameters:
- population size
- crossover Probability
- mutation Probability
-local search Probability
Create initial population
Calculate the error rate(classification rate) for each set of population
Apply MA with two types of feature subset selection methods (WFFS, MBEGA)
Is the stop criterion reached
No
Yes
STOP

Fig.9 Flow chart for feature selection using the memetic algorithm

By using the above procedures, we can get optimal solutions in terms of the maximum accuracy rate with a predefined number of selected features, and the selection of features is determined by the MA.

4.2 Minimum redundancy-maximum relevance (MRMR) feature selection

We use a minimum redundancy-maximum relevance (***MRMR***) feature subset selection framework. We depend on Weka for the machine learning software product to implement this type of feature subset selection method[50]. When we run ***MRMR*** for feature subset selection, the ***MRMR*** method ranks features according to their maximum relevance to the class sample. Therefore, MRMR feature subset selection can be used to get an optimal features set based on the importance of ranked features. This method is used to maximize the joint dependency of top ranking variables on the target variable, the redundancy among them must be reduced. So, we select maximally relevant variables and avoiding the redundant ones. First, mutual information (***MI***) between the candidate variable and the target variable is calculated (relevance term). Then average ***MI*** between the candidate variable and the variables that are already selected is computed (redundancy term). The entropy-based ***MRMR*** score is obtained by subtracting the redundancy from relevance [51, 52].

$$\text{D} = 1/(|S|) \sum_{Xi\epsilon S} I(Xi, Y)$$

$$\text{R}= 1/(|S|)^2 \sum_{Xj,\, Xi\epsilon S} I(Xi, Xj)$$

$$\text{MRMR} = \text{MAX } \{D - R\ )\}$$

One of this paper’s main goals is highlighting the importance of using ***MRMR*** feature subset selection in the machine learning field. One innovative point we used based on ***MRMR*** is to reduce redundancy in feature selection by using different classification algorithms. We use four common classical classification algorithms, which are (Lazy-nearest neighbor, linear support vector machine, Naïve bays, and AdaBoost ensemble). All these learning algorithms run under a feature subset selection method as an attribute evaluator. For the search methods in *MRMR* for feature subset selection, we use *GA* for this search strategy to select a candidate (ranked) feature. In our experiment, we show that features selected in this way lead to higher accuracy than features selected through traditional approaches [19]. We use features obtained from the *MRMR* method and run classical learning classification algorithms, and then compare performance accuracy with a Bayesian Network based Markov blanket.

Chapter 5

Datasets and Evaluation Process

In the previous section, we discussed several approaches for classification and feature selection techniques. In this section, we examine their performance on experimental data sets.

5.1 Datasets:

To validate the performance of our gene selection methods, the experiments are conducted on several real-world datasets. These datasets are used to demonstrate the feature selection methods' classification power. The datasets consist of small samples with high dimensionality, such as colon, prostate, brain tumor, diffuse large b-cell lymphomas (DLBCL), and leukemia.

The Colon cancer diagnosis data set is introduced in [48]. The other microarray cancer datasets are presented in the gene expression model selector [49]. These cancer datasets are described in Table 1.

Table 1: Description of experimental datasets

Data set	**# samples**	**# features**	**#classes**
Colon	62	2000	2
Brain tumor2	50	10367	4
Leukemia	47	2000	2
Lymphomas	77	5470	2
Prostate	102	1500	2

5.2 Estimating prediction errors

The following subsection reviews some of the methods used to evaluate the learning classifiers' performance.

5.2.1 Cross validation

In ***k-fold*** cross-validation, usually called ***k-fold CV***, the training set is split into **k** smaller sets. The ***k-fold*** cross validation uses part of the available data to fit the model, and a different part to test it. The following procedure is followed for each of the ***k "folds"***:

- A model is trained using ***k-1*** of the folds as training data;

- the resulting model is validated on the remaining part of the data (i.e. it is used as a test set to compute a performance measure such as accuracy).

The cross-validation process' performance measure is then repeated ***k*** times (the *folds*), with each of the ***k*** subsamples used exactly once as the validation data. The ***k*** results from the folds then can be averaged to produce a single estimation [26, 53].

5.2.2 Bootstrap

The bootstrap is usually called ***0.632 bootstrap*** and is used as a powerful tool in several contexts, most commonly used is to provide a measure of accuracy of a parameter estimator with a given statistical learning method to analyze and quantify the uncertainty for learning algorithm. So, we can state that the goal of bootstrap method is to assess sampling variation and use the measurement to assess population[53]. The ***bootstrap*** generates new random samples with size ***N*** by drawing instances from the training data with replacements and calculate the estimating prediction errors for each sample. Estimating the prediction errors depends on a partitioning or resampling the observed data to implement the learning model and test set. There are different types of bootstrap evaluation methods, but the most common one is 0.632 bootstrap. The 0.632 bootstrap works as follows. Suppose we have a dataset of *d* tuples. The dataset is resampled *d* times with replacement, the bootstrap finds the predication error of training set of *d* samples. The data tuples that was not used during the training phase, at end forms the test set. This process is repeated several times. As it turns out, an average of the original data tuples ends up in the bootstrap, and the remaining 36.8% forms the test set. The overall accuracy of the model is then estimated. At the end of iteration, there are a set of predication rules will be existing, so the voting strategies are used to get the predictions for a given unknown tuple[2, 22].

5.3 Evaluated methods

For classification, especially for two-class problems, a variety of measures has been proposed. Let us define the first important measure which is a confusion matrix; it is used for evaluating the classifiers according to their performance. There are four possible cases, as shown in Table 2.

Table 2: Confusion Matrix

	True class		
Predicted class	Positive	Negative	Total
Positive	True positive (tp)	False positive(fp)	P
Negative	False negative (fn)	True Negative(tn)	n
Total	P'	n'	N

A confusion matrix presents information regarding predicted and actual classifications performed following a classification system. There are four possible outcomes of a classifier to predict instances for class labels:

- If the instance is positive and it is classified as positive, then this is a true positive.
- If the instance is positive and it is classified as negative, then this is a false negative.
- If the instance is negative and it is classified as positive, then this is a false positive.
- If the instance is negative and it is classified as negative, then this is a true negative.

From the confusion matrix, we can derive several characteristics of classification performance. We can summarize some performance measures depending on Table 3 as follows.

Table 3: Performance measures based on confusion matrix (Table2)

Name	Formula
Error rate **Accuracy**	(fp+fn)/N (tp+tn)/N ((or) 1- Error-rate)
tp-rate **tn-rate**	tp/p' tn/n'
Precision (or positive predictive value) **Recall =tp-rate**	tp/ P tp/p'
Sensitivity = tp-rate **Specificity**	tp/p' tn/n' (or (1- fp-rate)

Chapter 6

Results and Discussion

6.1 Results

In this paper, the results are divided into two parts. In the first part, the output depends on memetic algorithm (*MAFS* software product) for five microarray cancer datasets that we mentioned previously. Table 4 shows the results. The local learning takes a long time for computation to obtain a high performance for ***MAs***. To be more accurate, we used two types of learning classification predictions (***10-kfold cross validation*** and ***30-run Bootstrap***). In the second part, the output depends on the result of ***MRMR*** for feature selection methods. We measure the selected features by using a set of classical classification algorithms and the Bayesian network based on Markov blanket for the five microarray cancer datasets. Tables 5 and 6 show the results.

6.2 Discussion

In this paper, we describe two different approaches to feature subset selection depending on a hybridized genetic algorithm. As we have seen, the difference between these approaches depends on how we use the GA for search mechanism. We can see from previous results that the feature subset selection methods achieved satisfactory results in reducing the number of features in five microarray cancer datasets.

Table 4: MAFS-output (memetic algorithm for feature selection)

Memetic Methods **Microarrays**	**MB(10-CV)**	**Wrapper (10-CV)**	**MB (30 Bootstrap)**	**Wrapper (30 Bootstrap)**
1-Colon cancer data set				
Average -selected genes	14.5	**10.2**	9.6	**9.4**
Running time	1109165	**1078326**	**4006795**	4391285
Average test error	**13%**	19%	**16%**	18%
2-Prostate cancer data set				
Average -selected genes	***13.3***	19.9	***13***	16.3
Running time	***2835186***	3725789	***10493308***	11987329
Average test error	21%	**17%**	**17%**	20%
3-Leukemia cancer data set				
Average -selected genes	***3.3***	5.6	**3.2**	3.8
Running time	***1002343***	1054632	**3300494**	3713434
Average test error	14%	**10%**	**9%**	12%
4-Lymphomas cancer data set				
Average -selected genes	**4**	8	***5***	7.9
Running time	**1330868**	1586993	***5046821***	8956452
Average test error	**16%**	19%	**11%**	12%
5- Brain Tumor 2 cancer data set				
Average -selected genes	19.3	**18.7**	**14.4**	18.7
Running time	1318257	**1271379**	4216340	**1271379**
Average test error	44%	**40%**	**33%**	40%

Table 5: Classification based on accuracy for classical classification algorithms and Markov Blanket(MB) for all microarray cancer datasets (without using MRMR for feature subset selection method)

Classifier **Microarrays**	**Naïve Bayes 10- CV**	**SVM 10-CV**	**IBK3(10-CV)**	**Ada-BoostM1 (10-CV)**	**MB(10-CV)**
1-Colon cancer data set					
%correctly Classified	53%	**82%**	75%	74%	75.8%
2-Prostate cancer data set					
%correctly Classified	63%	**86%**	79%	70%	74.5%
3-Leukemia cancer data set					
%correctly Classified	95.7%	80.8%	87%	89%	**97.8%**
4-Lymphomas cancer data set					
%correctly Classified	83%	75%	**91%**	84%	85.7
5- Brain Tumor cancer data set					
%correctly Classified	70 %	72%	70%	74%	**76%**

Table 6 : Classification after using MRMR for feature subset selection. The comparison based on accuracy for classical classification algorithms and Markov Blanket(MB) ,

Classifier Microarrays datasets	**Naïve Bayes 10- CV**	**MB(10-CV)**	**SVM (10-CV)**	**MB(10-CV)**	**IBK_3(10-CV)**	**MB(10-CV)**	**AdaBoostM1 (10-CV)**	**MB(10-CV)**
1-Colon Cancer								
No. of selected genes	42	42	36	36	37	37	25	25
%Correctly Classified	87%	**90%**	80%	**90%**	84%	**89%**	80%	**88%**
2-Prostate Cancer								
No. of selected genes	31	31	28	28	44	44	43	43
%Correctly Classified	82%	**83%**	82%	**83%**	81%	**82%**	77%	**81%**
3-Leukemia Cancer								
No. of selected genes	11	11	7	7	18	18	7	7
%Correctly Classified	**97.8%**	**97.8%**	93.6	**97.8**	98%	**100%**	95.7%	**100%**
4-Lymphomas								
No. of selected genes	27	27	25	25	28	28	22	22
%Correctly Classified	93.5%	**96%**	92%	**93.5%**	92%	**93.5%**	91%	**95%**
5- Brain Tumor 2								
No. of selected genes	40	40	31	31	47	47	29	29
%correctly Classified	84%	**86%**	76%	**88%**	**90%**	86%	76%	**82%**

In the results of feature subset selection in Table 4, we compared two methods of memetic algorithms, which are the MA-based wrapper method (*WFFSA*) and the *MA* based on the Markov Blanket method (*MBEGA*). The tabulated result in Table 4 showed that in the bootstrap (30 runs using the 0.632 bootstrap) learning model, the Markov blanket is better than the Wrapper method for average testing errors in all five microarray cancer datasets. This implies that the Markov blanket has improved classification accuracy more than the wrapper method. For the average number of selected subset features (genes) and running-time space available for the bootstrap learning model, we observe that both Markov blanket and wrapper methods outperform each other on five cancer datasets. However, 10-kfold cross-validation learning model's (*10-K-fold-CV*) results showed that *WFFSA* and *MBEGA* are competitive with each other regarding classification accuracy (average testing error), average number of selected features, and running time space available (seen in Table 2 in bold font).

The results in Table 5 presents the performance accuracy of different learning classification algorithms for all microarray cancer datasets (without using *MRMR* for feature selection methods). The different learning classification algorithms' accuracy results are competitive. The Bayesian Net based on Markov blanket gets the highest accuracy level in two cancer datasets (leukemia and brain tumor datasets), while the linear support vector machine is most accurate in the other two cancer datasets (colon and prostate datasets). The *Lazy-IBK3* classifier gets the highest accuracy level in only one cancer dataset (Lymphomas datasets).

The results displayed in Table 6 reveal accuracy measures improved after using *MRMR* for feature subset selection methods in all microarray cancer datasets for all learning classification algorithms without exception. In this table, we show the comparison of the performance measure for the classical classification algorithms and the Bayesian network based on Markov blanket depending on the *10 K-fold-cross-validation* learning model. Also, Table 6 shows that knowing Bayesian network based on the Markov blanket classification algorithm has the highest accuracy level in almost all five-cancer feature subsets selected by *MRMR* methods. The exception of this performance measure is on the brain tumor feature subset selection where the *IBK3* classification algorithm gets a higher accuracy (90%) than the Bayesian Network based on Markov blanket (86%).

Chapter 7

Conclusions

In this paper, we discussed different methods of feature (gene) subset selection methods. These methods aim to reduce the number of features especially in high dimensionality datasets. Additionally, it improves the classification performance and efficiency of microarray cancer datasets. We attempted to identify the importance of a memetic algorithm in the feature subset selection field and how a memetic algorithm uses several types of search methods (*GA* for population search hybridized with a local search engine) to solve a NP hard optimization problem. Consequently, we compared two types of memetic algorithms which are *MBEGA* and *WFFSA*. We observed that *MBEGA* is often better than *WFFSA* in classification accuracy for some cases of microarray cancer datasets.

Regarding the minimum redundancy-maximum relevance (*MRMR)* feature selection method, we noticed how it helped reduce the features in microarray datasets and improved classification accuracy performance after using the *MRMR* method. We attempted to make comparisons regarding classification accuracy for several types of classical classification algorithms after using the *MRMR* feature subset selection method. We can see its accuracy outperformed the Bayesian network based on Markov blanket classification algorithm and produced higher classification accuracy with a small number of feature subsets for five microarray cancer datasets.

References

[1] D. Koller and M. Sahami, "Toward optimal feature selection," in "toward optimal feature selection," Stanford InfoLab1996.

[2] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[3] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[4] M. W. Pleil, "Plug and Play Microsystems (MEMS) Technology into an Engineering and Technology Program."

[5] A. C. Tan and D. Gilbert, "Ensemble machine learning on gene expression data for cancer classification," 2003.

[6] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: A survey," *IEEE Transactions on knowledge and data engineering,* vol. 16, no. 11, pp. 1370-1386, 2004.

[7] M. M. Babu, "Introduction to microarray data analysis," *Computational genomics: Theory and application*, vol. 17, no. 6, pp. 225-49, 2004.

[8] A. Brazma and J. Vilo, "Gene expression data analysis," *FEBS letters,* Gene expression data analysis

Authors

Alvis Brazma,

Jaak Vilo vol. 480, no. 1, pp. 17-24, web site reference 2000.

[9] Z. Zhu and Y.-S. Ong, "Memetic algorithms for feature selection on microarray data," *Advances in Neural Networks–ISNN 2007*, pp. 1327-1335, 2007.

[10] T. Jirapech-Umpai and S. Aitken, "Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes," *BMC bioinformatics,* vol. 6, no. 1, p. 148, 2005.

[11] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *Journal of machine learning research,* vol. 5, no. Oct, pp. 1205-1224, 2004.

[12] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 856-863.

[13] Y. Zheng and C. K. Kwoh, "A feature subset selection method based on high-dimensional mutual information," *Entropy,* vol. 13, no. 4, pp. 860-901, 2011.

[14] L. Oliveira, N. Benahmed, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Feature subset selection using genetic algorithms for handwritten digit recognition," in *Computer Graphics and Image Processing, 2001 Proceedings of XIV Brazilian Symposium on*, 2001, pp. 362-369: IEEE.

[15] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence,* vol. 97, no. 1-2, pp. 273-324, 1997.

[16] Y. Tan and Z. Liu, "Feature selection and prediction with a Markov blanket structure learning algorithm," *BMC Bioinformatics,* vol. 14, no. 17, p. A3, 2013.

[17] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural computing and applications,* vol. 24, no. 1, pp. 175-186, 2014.

[18] N. J. Nilsson, "Artificial intelligence: A modern approach: Stuart Russell and Peter Norvig,(Prentice Hall, Englewood Cliffs, NJ, 1995); xxviii+ 932 pages," ed: Elsevier, 1996.

[19] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research,* vol. 3, no. Mar, pp. 1157-1182, 2003.

[20] V. Singh and S. Pathak, "Feature Selection Using Classifier in High Dimensional Data," *arXiv preprint arXiv:1401.0898,* 2014.

[21] O. Okun, "Feature Selection and Ensemble Methods for Bioinformatics," *Hershey, PA: Medical Information Science Reference,* 2011.

[22] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[23] Z. John Lu, "The elements of statistical learning: data mining, inference, and prediction," *Journal of the Royal Statistical Society: Series A (Statistics in Society),* vol. 173, no. 3, pp. 693-694, 2010.

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

[25] A. Statnikov, C. F. Aliferis, D. P. Hardin, and I. Guyon, *A Gentle Introduction To Support Vector Machines In Biomedicine: Volume 1: Theory and Methods*. World Scientific Publishing Co Inc, 2011.

[26] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini, "Tissue classification with gene expression profiles," *Journal of computational biology,* vol. 7, no. 3-4, pp. 559-583, 2000.

[27] M. E. Syed, "Attribute weighting in k-nearest neighbor classification," 2014.

[28] P. Yang, Y. Hwa Yang, B. B Zhou, and A. Y Zomaya, "A review of ensemble methods in bioinformatics," *Current Bioinformatics,* vol. 5, no. 4, pp. 296-308, 2010.

[29] B. Liu, *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.

[30] S. Dudoit and J. Fridlyand, "Classification in microarray experiments," *Statistical analysis of gene expression microarray data,* vol. 1, pp. 93-158, 2003.

[31] S. Fu and M. C. Desmarais, "Markov blanket based feature selection: a review of past decade," in *Proceedings of the world congress on engineering*, 2010, vol. 1, pp. 321-328.

[32] I. Guyon, C. Aliferis, and A. Elisseeff, "Causal feature selection," *Computational methods of feature selection,* pp. 63-82, 2007.

[33] P. Baldi and S. Brunak, *Bioinformatics: the machine learning approach*. MIT press, 2001.

[34] J.-P. Pellet and A. Elisseeff, "Using Markov blankets for causal structure learning," *Journal of Machine Learning Research,* vol. 9, no. Jul, pp. 1295-1342, 2008.

[35] K. P. Murphy, a. T. E. o. S. L. b. H. "Machine Learning: A Probabilistic Perspective" by K. Murphy, Tibshirani and Friedman, Ed. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[36] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics,* vol. 23, no. 19, pp. 2507-2517, 2007.

[37] P. Shi, S. Ray, Q. Zhu, and M. A. Kon, "Top scoring pairs for feature selection in machine learning and applications to cancer outcome prediction," *Bmc Bioinformatics,* vol. 12, no. 1, p. 375, 2011.

[38] D. Kumar, S. Kumar, and C. Rai, "Feature selection for face recognition: a memetic algorithmic approach," *Journal of Zhejiang University-Science A,* vol. 10, no. 8, pp. 1140-1152, 2009.

[39] M. A. Hall, "Correlation-based feature selection of discrete and numeric class machine learning," 2000.

[40] C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. D. Koutsoukos, "Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation," *Journal of Machine Learning Research,* vol. 11, no. Jan, pp. 171-234, 2010.

[41] Y. Zhang, Z. Zhang, K. Liu, and G. Qian, "An Improved IAMB Algorithm for Markov Blanket Discovery," *JCP,* vol. 5, no. 11, pp. 1755-1761, 2010.

[42] K. Yu, X. Wu, Z. Zhang, Y. Mu, H. Wang, and W. Ding, "Markov blanket feature selection with non-faithful data distributions," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, 2013, pp. 857-866: IEEE.

[43] X. Bai and R. Padman, "Tabu search enhanced markov blanket classifier for high dimensional data sets," in *The Next Wave in Computing, Optimization, and Decision Technologies*: Springer, 2005, pp. 337-354.

[44] E. Tuv, A. Borisov, G. Runger, and K. Torkkola, "Feature selection with ensembles, artificial variables, and redundancy elimination," *Journal of Machine Learning Research,* vol. 10, no. Jul, pp. 1341-1366, 2009.

[45] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms," in *Handbook of metaheuristics*: Springer, 2003, pp. 105-144.

[46] E. Volna, "Introduction to soft computing," ed, 2013.

[47] M. Melanie, "An introduction to genetic algorithms," *Cambridge, Massachusetts London, England, Fifth printing,* vol. 3, pp. 62-75, 1999.

[48] U. Alon *et al.*, "colon dataset

Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences,* vol. 96, no. 12, pp. 6745-6750, 1999.

[49] A. Statnikov, I. Tsamardinos, Y. Dosbayev, and C. F. Aliferis, "dataset ref:34

GEMS: a system for automated cancer diagnosis and biomarker discovery from microarray gene expression data," *International journal of medical informatics,* vol. 74, no. 7, pp. 491-503, 2005.

[50] http://www.cs.waikato.ac.nz/ml/weka/index_downloading.html, "Machine Learning Group at the University of Waikato," vol. Weka 3.8 is the latest stable version of Weka. This branch of Weka receives bug fixes only, although new features may become available in packages., no. java -jar weka.jar.

[51] B.-Q. Li, L.-L. Hu, L. Chen, K.-Y. Feng, Y.-D. Cai, and K.-C. Chou, "Prediction of protein domain with mRMR feature selection and analysis," *PLoS One,* vol. 7, no. 6, p. e39308, 2012.

[52] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of bioinformatics and computational biology,* vol. 3, no. 02, pp. 185-205, 2005.

[53] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013.