Table of Contents

Chapter 1 Introduction

1.1 Feature Selection

Feature Selection is a technique, which finds its application in reducing data to small size so that reduced data, can be used for processing and analysis. Feature selection technique is not only used to reduce the number of attributes for building a model but also help us to choose attributes (removing noisy and redundant attributes) based on their usefulness. This can be done either by analyst or any modeling tool. Generally the data set which are available have large data then required to build a model. But unneeded columns in data sets should be removed as they will only increase the CPU cycles and more memory is required for both training and storing the final model.

Other Reasons which invokes the need for removing extra columns from datasets are:

- Extra columns only degrade the quality of discovered patterns
- Some columns in data sets are noisy or redundant which hinders the finding of meaningful patterns from data sets.
- Sometimes, data mining algorithms requires small training set for finding quality patterns.

Feature selection technique is used to solve two kind of problems, first the large data which have less value and small data which have very high value. There are two reasons which motivates to apply feature selection on large datasets, first is to enhance the classifier performance by removing redundant, noisy and irrelevant features from the datasets and second is to reduce the number of features when classification algorithms could not scale up to the size of feature set either in time or space.

Feature selection generally consists of two steps first is to search for subset and second is to evaluate the subset produced. Search strategy that is employed can be either approximate or exhaustive. While exhaustive search strategy evaluates all probabilities of the feature subset, approximate search strategy only generates high quality solutions with no guarantee of finding a global optimal solution. Exhaustive search guarantees optimal solution but this method is not practical for even a medium-

sized dataset as finding the optimal subset of features. Since exhaustive search is not practical, research effort and focus on search strategies have since shifted to metaheuristic algorithms, which are considered as a subclass of approximate methods. The literature has shown that the metaheuristic algorithms are capable of handling large- size problem instances with satisfactory solutions within a reasonable time [25]. Once the feature subset is selected, each feature in subset is evaluated on the basis of some predefined criteria. There are three categories of feature subset evaluations depending on how the searching strategy is being associated with the classification model, whether as filter, wrapper, or embedded methods. These three categories will be explained in more detail in subsequent chapter. The metaheuristic algorithm used is Bacterial foraging algorithm. Bacterial Foraging Algorithm (BFA) is one of the strong nature-inspired optimization algorithms, which emulate the behavior of *E. Coli* bacteria. In BFA group of bacteria forage in that direction where they finds more of nutrients medium. In this algorithm, an objective function is calculated in terms of cost or efforts of bacteria applied for searching the food in medium. The literature had shown that there are number of approaches employed for this and their comparison is done on the basis of number of features selected and classification accuracy [26].

1.2 Research Objective

With the view explained in the previous section, the objective of our research work can be identified as:

- To design a new hybrid algorithm that exploits a Naive Bayes algorithm to guide a BFA.
- To evaluate the performance of the proposed hybrid algorithm against other well-known feature selection algorithms
- To test the effect of the resulting features in terms of classification accuracy using three different classifiers.

1.3 Thesis Organization

We start this dissertation with introduction in chapter 1. A detailed description of background is presented in chapter 2 which includes reviews on searching and evaluating algorithms in feature selection, literature review of Bacterial foraging algorithm and Naive Bayes guided classification. Chapter 3 describes the mechanics of bacterial foraging algorithm and details out the Naive Bayes classification which are used in our proposed work. Chapter 4 explains in detail about our proposed algorithm that is Naive Bayes guided Bacterial foraging algorithm for feature selection. We evaluate the performance of the proposed technique with the existing algorithm in chapter 5. We conclude about the work done and future work in chapt

Chapter 2 Literature Review

In this chapter, a brief review on feature selection, bacterial foraging algorithm and Naive Bayes classification is given.

2.1 Feature Selection

Feature selection is an activity which finds its application in data preprocessing, classification, regression ,time series prediction, mining and machine learning community. Feature selection main objective is to improve the accuracy and efficiency of classier by minimizing the prediction errors. Feature selection also known as variable selection mainly used for those research applications where data sets consists of thousands of variables.

Feature selection algorithms are majorly divided into three categories filter, wrapper and embedded [1]. If the feature selection algorithm involves learning algorithm then the approach is classified as wrapper [2]. The wrapper approach performs selection by considering classifier as a black box and ranking the subset of features by their predictive power. The filter approach [3] selects the features by using a preprocessing step that does not involve learning algorithm. The main disadvantage of this procedure is that it ignores the effect of the subset of features in the learning algorithm. The primary advantage of the filter-based approach over the wrapper-based approach is that it is computationally faster. However, if computational complexity was not a factor, then a wrapper-based approach was the best overall feature selection scheme in terms of accuracy. We can perform exhaustive search under these two approaches, if the number of variables is less. However, the problem is known to be NP-hard [4] and the search quickly becomes computationally intractable. The computational cost increases as the number of variable increases. Embedded methods, in contrast to wrapper approaches, select features while taking into account the classifier design.

2.1.1 Wrappers approach

A wrappers model consists of two phases [5]

Phase 1 – feature subset selection, which selects the best subset using the accuracy of the classifier (on the training data) as a criterion.

Phase 2 – learning and testing, where a classifier is learned from the training data with the best feature subset, and is tested on the test data.
The wrappers approach consists of using the prediction performance of a given learning machine is used to assess the relative usefulness of subsets of variables. It performs selection taking into account the classifier as a black box and ranking the subset of features as per their predictive power. Although the wrappers approach is often criticized to be a ‘‘brute force” method which involves massive computation time, some researchers have different opinions. In this regard, [11] indicated that coarse search strategies may alleviate the problem of over-fitting.

Full search have 2N different evaluations, In order to overcome this ‘‘forward” selection or ‘‘backward” elimination methods are used. These methods are also referred to as ‘‘sequential” methods. These are the most commonly used methods for performing feature selection. Sequential Forward Selection, SFS [6] starts with the empty set and variables are progressively added into larger and larger subsets, whereas in Sequential Backward Selection, SBS [7], one starts with the set of all the variables and progressively eliminates the least promising ones. Both methods suffer from the problem called “nesting effect”. This means that once the features are selected in forward selection can not be discarded later and once the features are eliminated in backward selection, they can’t be reselected.

2.1.2 Sequential Forward Floating Selection

In order to prevent nesting effect, SFS and SBS can be combined together. The approach proposed was called “plus-l-take-away-r”. In this method first SFS was applied l times followed by r steps of SBS, this is repeated until the required number of features is reached. In this case, features that have been previously added can be

removed in posterior steps thus avoiding the nesting effect. This method allows ‘‘fixed backtracking” defined by the values of l or r depending on whether the search is top-down or bottom-up. Although this procedures solves nesting effect but another problem arises: there is no theoretical guide to determine the appropriate value of l and r to obtain good enough solutions with a moderate amount of computation. Pudil et al. (1994a)[8] introduced the concept of ‘‘floating feature search” and two ‘‘floating” selection methods, SFFS (Sequential Forward Floating Selection) and SBFS (Sequential Backward Floating Selection). The floating feature search methods are similar to plus-l-take-away-r algorithm, but in latter the number of forward and backtracking steps is dynamically controlled instead of being fixed beforehand. The SFFS and SBFS methods are characterized by the changing number of features included or eliminated at different stages of the procedure and consider to be best feature selection techniques. In forward search, start with empty set and in each step new feature is added which satisfies some criterion function. At the same time algorithm also verifies the possibility of improvement of the criterion if some feature is excluded. Therefore, the SFFS proceeds by dynamically increasing and decreasing the number of features until the desired number of features is reached. SFBS works similarly, but starts with the full feature set and performs the search until the desired dimension is reached, using SBS and SFS steps. The particular advantage of the floating search methods over the plus-l-take-away-r methods is that they can make more than one sweep through feature set subsets to achieve good performance. In practice, searching with dynamic backtracking has very robust performance and if one had to choose between feature set search methods then the floating search procedures would be the first choice [9]. In addition, the floating methods are at least as good as the best sequential methods. Although the floating methods are considered to be more intelligent, they are still suboptimal. The drawback of these sequential floating methods is that they are still likely to become trapped in a local optimal solution even if the criterion function is monotonic and the scale of the problem is quite small

2.1.3 The filter approach

The filters approach is based on the intrinsic properties of the data without having any dependency on any classifier. The essence of filters is to seek the relevant features and eliminate the irrelevant ones. A filters model of feature selection also consists of

two phases [10].

Phase 1 – feature selection using measures such as information, distance, dependence, or consistency.

Phase 2 – this phase is same as in the wrappers model, where a classifier is learned on the training data with the selected features, and tested on the test data.

In addition to being based on the intrinsic properties of the data, the filters approach has other characteristics also, as follows:

1. Measuring information gains, distance, dependence, or consistency is usually cheaper than measuring classifier accuracy, so a filters method can produce a subset faster, all things being equal.

2. Filter method can handle larger amount of data as compared to classifier because of its simplicity of the measures. so in the case where a classifier cannot directly be learned from the large amount of data, it can be used to reduce data dimensionality so that the classifier can be learned from such data.

2.1.4 Meta-heuristic and other algorithms:

There is numbers of algorithms that are used in the literature for the feature selection problem include the following [11]:

The Branch-and-Bound (BB) feature selection algorithm

This algorithm was proposed by Narendra and Fukunaga [12], can be used to find the optimal subset of features under the monotonicity assumption. The monotonicity assumption states that the addition of features can only increase the value of the objective function, this is Branch and Bound starts from the full set and removes features using a depth-first strategy. Nodes whose objective function are lower than the current best are not explored since the monotonicity assumption ensures that their children will not contain a better solution. One drawback is that this method requires the feature selection criterion function to be monotonic. This means that the addition of new features to a feature subset can never decrease the value of the criterion function. Unfortunately, the monotonic condition is seldom satisfied. Nevertheless, its computational cost is prohibitive for large feature spaces: in the worst case, it

performs an exhaustive search and its time complexity is exponential according to the dimension of the feature space. The BB method is still impractical for problems with very large feature sets.

The Max–Min (MM) method

This algorithm was proposed by Backer and Shipper [13] is a computationally efficient method. It only evaluates the individual and pairwise merits of features. The results achieved with this method are invariably rather unsatisfactory. The experiments performed by many researchers show that this method gives the poorest results [14].

Genetic algorithm

Siedlecki and Sklansky [15] introduced the use of GA for feature selection. Genetic algorithms (GAs), is a kind of inductive learning strategy, are adaptive search techniques which have shown mark improvement over a variety of random and local search methods. They have ability to exploit accumulating information about an initially unknown search space in order to bias subsequent search into promising subspaces. Since GAs are basically a domain independent search technique, they are ideal for applications where domain knowledge and theory is difficult or impossible to provide. The main issues in applying GAs to any problem are selecting an appropriate representation and an adequate evaluation function.

In a GA approach, a given feature subset is represented as a binary string (a ‘‘chromosome”) of length n, with a zero or one in position i denoting the absence or presence of feature i in the set where n is the total number of available features. Each chromosome is evaluated to determine its ‘‘fitness,” which determines how likely the chromosome is to survive and breed into the next generation. New chromosomes are created from old chromosomes by the following processes:

(1) **Crossover**, where parts of two different parent chromosomes are mixed
To create offspring

(2) **Mutation**, where the bits of a single parent are randomly changed to create a child.

Ant Colony optimization

In order to solve an optimization problem, a number of artificial ants are used to iteratively construct solutions. In each iteration, an ant would deposit a certain amount of pheromone proportional to the quality of the solution. At each step, every ant computes a set of feasible expansions to its current partial solution and selects one of these depending on two factors: local heuristics and prior knowledge [16,17]. Training dataset is prepared according to binary class classification problem. From the training dataset, features are extracted, after that the best subset of features is selected by algorithm and then, by using the selected best features, classification is done.

Particle Swarm Optimization (PSO) algorithm

This simulates the social behavior of natural creatures such as bird flocking and fish schooling to discover a place with adequate food. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lbest*. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest*.

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations.

For feature selection it represent the particle's position as binary bit strings of length N, where N is the total number of attributes. Every bit represents an attribute, the value '1' means the corresponding attribute is selected while '0' not selected.

2.2 Bacterial Foraging Algorithm

Bacteria Foraging Optimization (BFO) is applied for feature selection for face recognition by Jakhar and Kaur 18]. The algorithm is applied to coefficients extracted by discrete cosine transforms. Evolution is driven by a fitness function defined in terms of maximizing the class separation (scatter index). If the algorithm is compared with PSO-based feature selection, the average recognition rate of the proposed algorithm is better than that of PSO-based feature selection. The number of selected features by proposed algorithm is comparable to those selected by PSO-based feature selection

Another application of BFO is used in predicting ground water without drilling holes by only knowing its ground features. It was proposed by Kumar [19]. The techniques is compared with linear regression and multiple regression. It was found that BFO is best among the other existing techniques in term of accuracy in predicting ground water.

Dr Pattnaik [20], developed a novel technique by hybridizing Bacterial Foraging Optimization technique in conjunction with Particle Swarm Optimization to calculate accurately the desired resonant frequency of rectangular micro strip antenna of any dimension and of any substrate thickness. After his experiment, the result shows promising improvement in the accuracy, while achieving drastic reduction in computational time .

Jae Hoon Cho proposed a method which consists of two parts: a wrapper part by bacterial foraging optimization and a filter part by mutual information. In order to select the best feature subset to achieve the best performance of the classifiers. Experimental results show that this method can achieve better performance for pattern recognition problems other than other conventional ones [21].

Mannat [22] presented an extensive review of data classification methods over the diabetes data sets using different mining classification technique such as neural

network structures, support vector machine. All these methods have some shortcomings. So he had proposed to use a novel Bacterial Forging Optimization applied to data classification to overcome these shortcomings. This method can be combined with medical software's to assist physicians so that they can take make more accurate decisions about Diabetes disease.

2.3 Naives Bayes Classification

Naive Bayes (NB) classification is one of the popular classification methods which is also used for stream mining as it is an incremental classification method whose model can be easily updated as new data arrives. It has been observed in the literature that the performance of the NB classifier improves when irrelevant features are eliminated from the modeling process. Lutu [23] conduct the studies to identify efficient computational methods for selecting relevant features for NB classification based on the sliding window method of stream mining and also experimental results demonstrates that continuous feature selection for NB stream mining provides high levels of predictive performance.

Fleuret [24] proposed a very fast feature selection technique based on conditional mutual information. It select those features which maximize their mutual information with the class and also it ensures the selection of features which are both individually informative and two-by-two weakly dependent. This feature selection method outperforms other classical algorithms, and that a Naive Bayesian classifier built with features selected that way achieves error rates similar to those of state-of-the-art methods such as boosting or SVMs. The implementation selects 50 features among 40,000, based on a training set of 500 examples in a tenth of a second on a standard 1Ghz PC.

Chapter 3 Methodology Used

In this chapter we will discuss in detail about the Bacterial foraging algorithm and Naive Bayes classification.

3.1 The Bacteria Foraging Optimization Algorithm

3.1.1 Basic Idea

When real bacteria forage towards the nutrient medium, it uses tensile flagella to achieve locomotion. During foraging the bacteria either swim or tumble using its flagella. Swim and tumble are basic operations performed during foraging. The bacteria when rotates the flagella in clockwise direction it pulls on the cell resulting in moving of flagella independently and there was less number of tumbling. When the bacteria is in noxious medium it tumbles frequently to avoid that medium and try to find nutrient medium. In order to swim at high pace bacteria rotates its flagella in counter clockwise direction. Chemotaxis is the term which means that bacteria move in that direction where it will find more nutrient gradient. Figure 1 explains how bacteria rotates its flagella in clockwise and counter clockwise direction in nutrient medium.

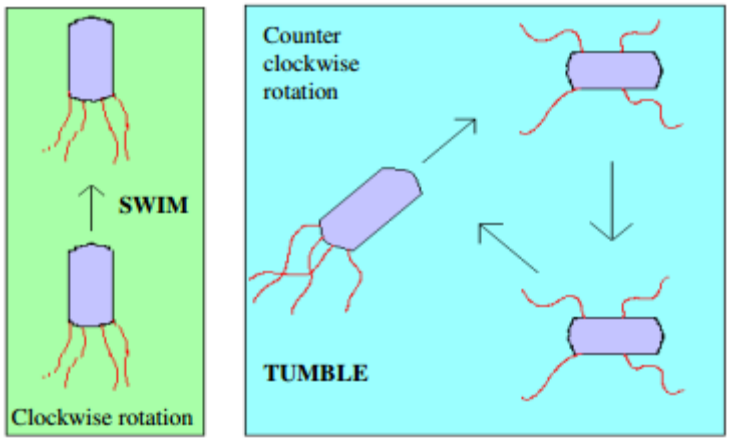


Figure 1 The clockwise and anti clockwise movements of a bacterium

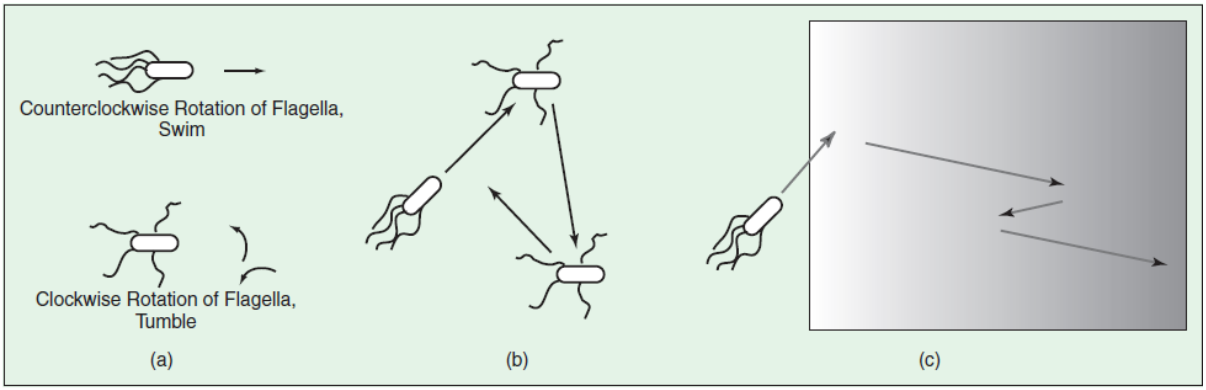


Figure 2 Swimming, Tumbling and chemotactic behavior of E. coli

Bacteria when get adequate food they arise in the size and also when temperature conditions are favorable they split to form exact replica of itself. This phenomenon inspired Passino and he added reproduction step in BFA. These bacteria some times moves to other places due to occurrence of sudden environmental change and it also hampers their chemotactic behavior. This is called elimination dispersal, where all either all bacteria population died or group of bacteria move to new place. Now suppose that we want to find the minimum of $J(\theta)$ where $\theta \in \Re^p$ (i.e. θ is a p-dimensional vector of real numbers), and we do not have measurements or an analytical description of the gradient $\nabla J(\theta)$. BFOA imitates the four major principal mechanisms of a real bacterial system:

- chemotaxis
- swarming
- reproduction
- elimination-dispersal

to solve this optimization problem. A virtual bacterium is actually one trial solution (also called as a search-agent) that relocate on the functional surface to locate the global optimum.

3.2 Constituting Steps

Let us define a chemotactic step to be a tumble followed by a tumble or a tumble followed by a run.

Let j be the index for the chemotactic step.

Let k be the index for the reproduction step.

Let i be the index of the elimination-dispersal event.

Also let

$$p: Dimension\ of\ the\ search\ space,$$

$$S: Total\ number\ of\ bacteria\ in\ the\ population,$$

$$N_c: The\ number\ of\ chemotactic\ steps,$$

$$N_s: The\ swimming\ leng\square h.$$

$$N_{re}: The\ number\ of\ reproduction\ steps,$$

$$N_{ed}: The\ number\ of\ elimination-dispersal\ events,$$

$$P_{ed}: Elimination-dispersal\ probability,$$

$$C\ (i): The\ size\ of\ the\ step\ taken\ in\ the\ random\ direction\ specified\ by\ the\ tumble.$$

Let P(j,k,l)= {Θ^i (j,k,l) i= 1,2 ,...,S} shows the position of each member in the population of the S bacteria at the j-th chemotactic step, k-th reproduction step, and l-th elimination-dispersal event step.

Here, let J(i,j,k,l) denote the cost or effort at the location of the i-th bacterium Θ^i(j,k,l). Note that we will interchangeably use J as being a "cost" or being a nutrient surface. For actual bacterial populations, S can be very large (e.g., S =109), but p = 3. In our computer simulations, we will use much smaller population sizes and will keep the population size fixed. BFOA, however, allows p > 3 so that we can apply the method to higher dimensional optimization problems. Primary four steps of BFA are described below.

3.2.1 Chemotaxis

Chemotaxis imitates the motion of E.coli cell via swimming and tumbling using flagella. E.coli bacteria biologically can move in two different manners. One is moving in same direction for a interval of time or tumble. Second is an alternate movement between these two modes. Suppose Θ^i (j, k, l) represents i-th bacterium at jth chemotactic, k-th reproductive and l-th elimination-dispersal step. C (i) represents the tumble (run length unit), it is the size of the step taken in the random direction. The movement of bacteria is represented by equation 1 where Δ reprsents a vector in the random direction whose value lie in [-1, 1].

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \qquad \text{--(1)}$$

3.2.2 Swimming

Ecoli bacteria shows an complex but interesting behavior of forming patterns or swarms in semi solid nutrient medium. These bacteria form a kind of ring structure when placed in semisolid matrix with a single nutrient chemo-effecter. The cells when vitalize by a high level of succinate, they release an attractant aspartate which aggregate them into the groups and this makes them to move in swarm. The cell-to-cell signaling in E. Coli swarm is given by the following function.

$$J_{cc}(\theta, \left(P(j,k,l)\right) = \sum_{i=1}^{s} J_{cc}\,(\theta, (\theta^i(j,k,l) \qquad \text{--(2)}$$

$$= \sum_{i=1}^{s} [-d_{attractant} \exp(-w_{attractant \sum_{m=1}^{p} (\theta_m - \theta_m^i)^2})]$$

$$+ \sum_{i=1}^{s} [h_{repellent}\ \exp(-w_{repellent}) \sum_{m=1}^{p} (\theta_m - \theta_m^i)^2]$$

Where $J_{cc}(\Theta, \left(P(j,k,l)\right))$ is value to be added to the actual objective function to be minimized to represent a time varying objective function, S is the total number of bacteria, p is the dimension of the problem or number of variables to be optimized, which are present in each bacterium and $\Theta = [\Theta_1, \Theta_2 \ldots\ldots \Theta_p]^T$ is a point in the p-dimensional search domain. $d_{attractant}$, $w_{attractant}$,$w_{repellent}$, $h_{repellent}$ are different coefficients that should be chosen properly.

3.2.3 Reproduction

In reproduction step, the healthy bacteria (one with lower objective function value) splits into two bacteria and get placed themselves to the same location and least healthy bacteria eventually dies.

3.2.4 Elimination Dispersal

Generally the environment where bacteria lives changes gradually or suddenly due to some influence. Due to these changes either all bacteria dies in a region or
group of bacteria moved from one place to another. Also water and animals facilates the movement of bacteria. Such events have spread bacteria in every part of environment from hot springs to our intestine. Although elimination and dispersal

hampers chemotactic progress but they also assist chemotaxis, since dispersal may place bacteria near good food sources.

3.3 Guidelines for Algorithm Parameter Choices

The bacterial foraging optimization algorithm requires specification of a variety of parameters. First, we can pick the size of the population, *S*. Clearly, increasing the size of *S* can significantly increase the computational complexity of the algorithm. However, for larger values of *S*, if you choose to randomly distribute the initial population, it is more likely that you will start at least some bacteria near an optimum point, and over time, it is then more likely that many bacterium will be in that region, due to either chemotaxis or reproduction.

For the values of the $C(i)$, (i =1,2,K,*S)* we can choose a biologically motivated value; however, such values may not be the best for an engineering application. If the $C(i)$ values are too large, then if the optimum value lies in a valley with steep edges, the search will tend to jump out of the valley, or it may simply miss possible local minima by swimming through them without stopping. On the other hand, if the $C(i)$ values are too small, convergence can be slow, but if the search finds a local minimum it will typically not deviate too far from it. We should think of the $C(i)$ as a type of "step size" for the optimization algorithm. The size of the values of the parameters that define the cell-to-cell attractant functions J_{cc} will define the characteristics of swarming. If the attractant width is high and very deep, the cells will have a strong tendency to swarm (they may even avoid going after nutrients and favour swarming). On the other hand, if the attractant width is small and the depth shallow, there will be little tendency to swarm and each cell will search on its own. Social versus independent foraging is then dictated by the balance between the strengths of the cell-to-cell attractant signals and nutrient concentrations. Next, large values for N_c result in many chemotactic steps, and hopefully more optimization progress, but of course more computational complexity. If the size of N_c is chosen to be too short, the algorithm will generally rely more on luck and reproduction, and in some cases, it could more easily get trapped in a local minimum (premature convergence). We should think of *Ns* as creating a bias in the random walk (which would not occur if N_s = 0), with large values tending to bias the walk more in the direction of climbing down the hill. If N_c is large enough, the value of N_{re} affects how

the algorithm ignores bad regions and focuses on good ones, since bacteria in relatively nutrient-poor regions die (this models, with a fixed population size, the characteristic where bacteria will tend to reproduce at higher rates in favourable environments). If N_{re} is too small, the algorithm may converge prematurely; however, larger values of N_{re} clearly increase computational complexity.

3.4 The BFA

[Step 1] Initialize parameters *p, S, Nc, Ns, Nre, Ned, Ped, C(i)(i=1,2…S)*.

[Step 2] Elimination-dispersal loop: $l=l+1$

[Step 3] Reproduction loop: $k=k+1$

[Step 4] Chemotaxis loop: $j=j+1$

[a] For i =1,2…S take a chemotactic step for bacterium i as follows.

[b] Compute fitness function, $J(i, j, k, l)$.

Let, $J(i, j, k, l) = J(i, j, k, l) + J_{cc}(\Theta^i(j, k, l), P(j, k, l))$ (i.e. add on the cell-to-cell attractant–repellent profile to simulate the swarming behaviour) where, J_{cc} is defined in (2).

[c] Let $J_{last}=J(i, j, k, l)$ to save this value since we may find a better cost via a run.

[d] Tumble: generate a random vector $\Delta(i) \in R^p$ with each element $\Delta_m(i)$, m =1,2,..., p, a random number on [-1, 1].

[e] Move:

Let $$\Theta^i(j+1, k, l) = \Theta^i(j, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad \text{-- (3)}$$

This results in a step of size $C(i)$ in the direction of the tumble for bacterium i.

[f] Compute $J(i, j+1, k, l)$ and let,

$$J(i, j+1, k, l) = J(i, j, k, l) + J_{cc}(\theta^i(j+1, k, l), P(j+1, k, l)) \quad \text{-- (4)}$$

[g] Swim

i) Let m=0 (counter for swim length).

ii) While $m < N_s$ (if have not climbed down too long).

• Let $m=m+1$.

• If $J(i, j+1, k, l) < J_{last}$ (if doing better), let $J_{last} = J(i, j+1, k, l)$ and let

$$\Theta^i(j+1, k, l) = \Theta^i(j, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

And use this $\Theta^i(j+1, j, k)$ to compute the new $J(i, j+1, k, l)$ as we did in [f] Else, let $m = N_s$. This is the end of the while statement.

[h] Go to next bacterium (i+1) if i !=S (i.e., go to [b] to process the next bacterium).

[Step 5] If $j < N_c$, go to step 4. In this case continue chemotaxis since the life of the bacteria is not over.

[Step 6] Reproduction:

[a] For the given k and l, and for each i=1,2,..., S , let

$J^i_{health} = \sum_{j=1}^{N_{c+1}} j(i,j,k,l)$

be the health of the bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances)

sortbacteria and chemotactic parameters $C(i)$ in order of ascending cost J_{health} higher cost means lower health).

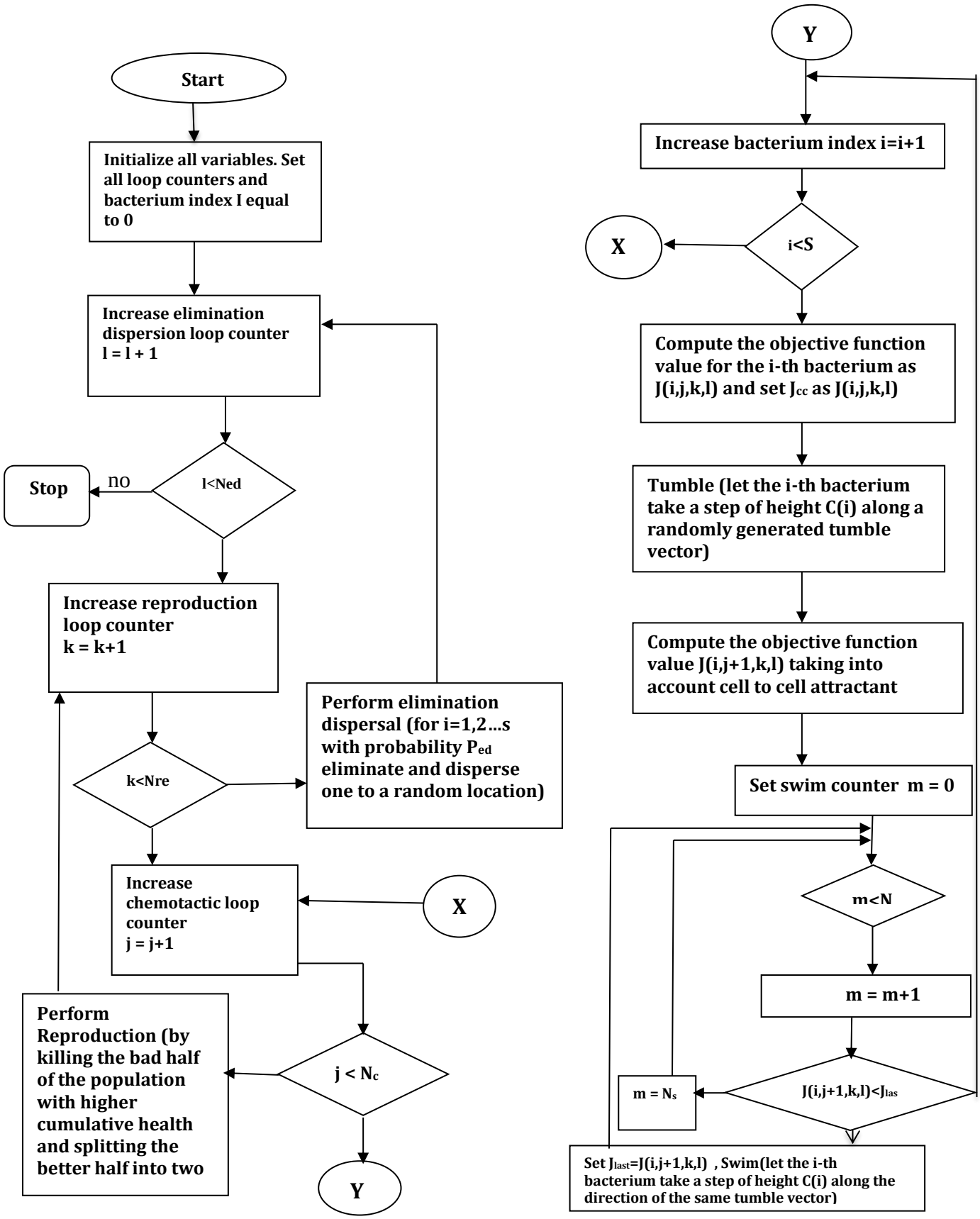
[b] The S_r bacteria with the highest J_{health} values die and the remaining S_r bacteria with the best values split (this process is performed by the copies that are made are placed at the same location as their parent).

[Step 7] If $K < N_{re}$, go to step 3. In this case, we have not reached the number of specified reproduction steps, so we start the next generation of the chemotactic loop.

[Step 8] Elimination-dispersal: For i=1,2..., S with probability P_{ed} , eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if a bacterium is eliminated, simply disperse another one to a random location on the optimization domain. If $l < N_{ed}$, then go to step 2; otherwise end.

3.5 Flowchart

Figure 3: Flowchart showing the complete working of the algorithm



3.6 Naive Bayes algorithm

Naive Bayes algorithm belongs to the wrapper approach and is one of the most effective and efficient inductive learning algorithms for data mining along with machine learning. NB is a classifier based on a statistical theory "Bayes theorem." The Bayesian algorithm is called "naive" because it is founded on Bayes Rule, which states that the features are conditionally independent from each other with regard to the class [27]. In the literature, the NB algorithm has in various applications such as text classification, improving search engine quality, image processing, fault prediction and medical diagnoses.

Naive Bayes classifier works as follows: let □ be a vector of random variables for the observed attribute values in the training set □ = [□1, □2, . . . , □□] to certain class label □ in the training set. The probability of each class given the vector of observed values for the predictive attributes can be computed using the following formula:

$$P(Yj|X) = \frac{P(Yj|)\,P(X|Y_J)}{\sum_i^c P(\mathrm{Y_i})\,P(X|Y_i)} \qquad \text{j = 1, 2 …..c.} \qquad \text{--(5)}$$

where □(□□) is the prior probability of class □□ and □(□□ | □) is the class conditional probability density functions. Basically the conditional independence assumption assumes that each variable in the dataset is conditionally independent of the other. This is simple to compute for test cases and to estimate from training data as follows:

$$P\left(X\middle|Y_j\right) = \prod_{i-1}^{n} P(X_i|Y_j) \qquad \text{j = 1, 2 …..c.} \qquad \text{--(6)}$$

where □□ is the value of the □th attribute in □ and □ is the number of attributes. Let □ be the number of classes and □□ is the □th class the probability distribution over the set of features is calculated using the following equation:

$$P(x) = \prod_{i-1}^{k} P(c_i)P(X|c_i) \qquad \text{--(7)}$$

Naive Bayes algorithm is popular in classification and because of the following reasons:

- Since it is considered as linear time complexity classifier it have high computational efficiency as compared to other wrapper methods.
- Due to less searching it has low variance.
- It is characterized by incremental learning as NB functions work from approximation of low-order probabilities that are deduced from the training data. Hence, these can be quickly updated as new training data are obtained.
- It have high capability to handle noise in the dataset.
- It also have high capability to handle missing values in the dataset.

Furthermore, NB implementation has no required adjusting parameters or domain knowledge. The main drawback of NB is that it assumes features independence. Despite this, NB often delivers competitive classification accuracy and is widely applied in practice especially as benchmark results.

Chapter 4 Proposed Work

This chapter gives you the understanding of proposed work "Naive Bayes guided bacterial foraging algorithm for features selection". The work can be divided into two parts. In part 1 we basically find the feature subset using bacterial foraging algorithm and in next part we analyze the obtained feature subset using Naive Bayes classification. The algorithm is made to run for the number of generations until it gives the optimal feature subsets. Later, the obtained subset is used to find classification accuracy using three different classifiers by applying on number of datasets and it can be shown that our proposed algorithm gives better accuracy and selects lesser number of features as compared to other known algorithm used for feature selection.

4.1 Basic Idea

From figure 4, we can observe a simple scheme of how features are extracted from the initial dataset by using BFO with binary encoding and in second phase, how the resulted subset is evaluated by means of Naive Bayes classifier and 10 Fold cross validation to obtain the fitness value of such bacteria.

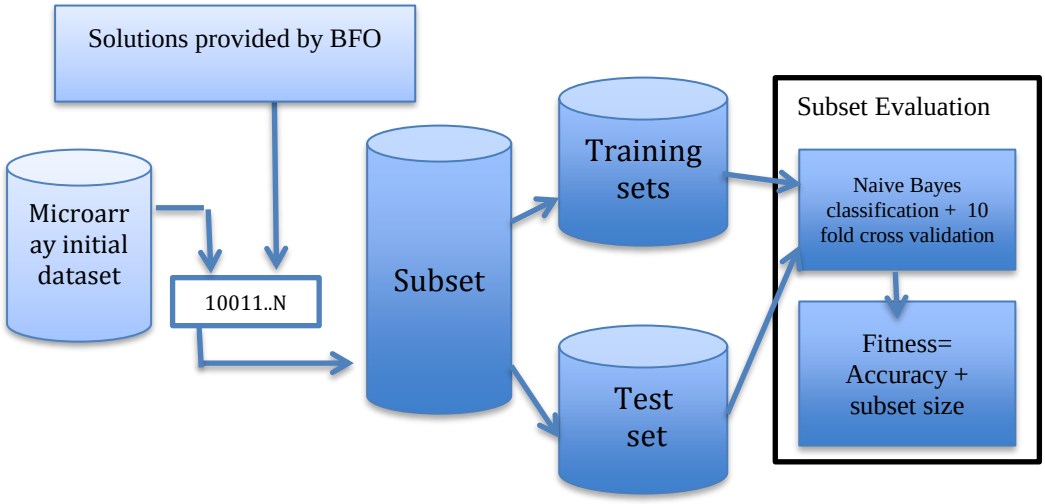


Figure 4: Basic flow of proposed work

4.2 Naive bayes guided BFO

In our proposed work, bacterial foraging technique is used as the basis of finding feature subset from the large data sets. Initially few rows of datasets are selected to server as initial bacteria population to start with. These bacteria are binary encoded bacteria which means if a bit is 1, it means that this feature is kept in the subset and 0 indicates that the feature is not included in the subset. Therefore, the bacteria length is equal to the number of features in the initial dataset. Then chemotaxis step of swimming and tumbling are performed. The bacteria foraging is driven by nutrient distribution. The bacteria move in that where they find more nutrients.

Steps of the proposed approach

4.2.1 Binary encoding of bacteria

For feature selection problem the initial population of bacteria is first encoded into binary form using sigmoid function.

$$y = \frac{1}{1+e^{-x}} = \begin{cases} 1, & y > random[0,1] \\ 0, & y < random[0,1] \end{cases} \quad \text{--(7)}$$

Here the bit 1 represents that corresponding feature of dataset is selected and 0 else.

4.2.2 Initialization of parameter and search space

a. Initialization and search space identification

The bacteria are first placed randomly all over the space which would then undergo foraging in search for food.

b. Initialize parameters

Let p be the dimension of the search space, S be the number of bacteria placed on the image, N_c be the Number of chemotactic steps, N_{re} be the number of reproduction steps, N_{ed} be the number of elimination-dispersal steps, P be the elimination-dispersal with probability, N_s be the Swim length and r be run-length of each bacterium.

4.2.3 Chemotactic step

a. Compute fitness function

$J(i,j,k,l)$ Fitness function of a bacterium i in its j^{th} chemotactic, k^{th} reproductive, and l^{th} elimination dispersal step at position $(1...p)$ is denoted as

$$\textbf{Fitness Value} = \delta\,\mathrm{CA} + \varphi\,\frac{\mathrm{TF}-\mathrm{SF}}{\mathrm{TF}} \qquad \text{--(8)}$$

Where CA is the classification accuracy, TF is the total number of all features, and SF is the number of selected features. □ and □ are two parameters corresponding to the weight of classification accuracy and subset length, where □ ∈ [0, 1] and □ = 1 − □. From (8), we can see that the importance of classification accuracy and how subset size is weighted differently. We have given more weight to classification accuracy than the size of the subset.

The classification accuracy CA is calculated from the Naive Bayes classifier which is applied on the datasets having as number of features as per the given bacteria using 10 cross fold validation technique.

10 Fold cross validation is a technique which involves portioning a sample of data into10 subsets, performing the analysis on one subset (called the *training set*), and validating on the other subsets (called the *validation set* or *testing set*). This process is repeat 10 times and the validation results are averaged over the rounds.

b. Compute cell-to-cell attractant repellant profile to simulate the swarming behavior:

This is calculated using the Eq. 4.

Tumble: A bacterium starts the process by tumbling. That is, it first moves in a random direction in search for food. Calculating the fitness value in that direction does the decision for this direction. If the value come out to be less than that of global fitness value then bacteria will move in that direction (as there is more nutrient value). Else it will not move in that direction.

Swim: The bacterium is moved to the new location using the Eq. (3). Since the bacteria aim at maximizing the energy per unit time, so they are always in search for area with higher nutrient concentration and lesser noxious substances. The bacteria avoids negative gradient region and prefers to move in positive nutrient gradient.

Thus, the bacterium follows the following criterion: Wherever it gets a positive nutrient gradient, it swims in that direction. And in neutral medium, it tumbles but if it reaches its maximum number of chemotactic steps and still not finds a positive nutrient gradient medium then it dies and it completely avoids negative nutrient gradient.

4.2.4 Reproduction Step

For given k and l, each bacterium i =1,2….S.

1) Sum

$$J_{health}^{i} = \sum_{j=1}^{N_{c+1}} j(i,j,k,l)$$

2) **Sort** the bacteria in order of ascending cost $J_{health.}$

3) **Split and eliminate**

The S_r bacteria with the highest $J_{health.}$ values die and the remaining S_r bacteria with the best values split .

4.2.5 Elimination Step

In the elimination-dispersal step those bacteria, which are eliminated, are dispersed to a random location. Each bacterium is subjected to elimination-dispersal with probability $P_{ed.}$.

4.3 Modified Naive Bayes guided BFO algorithm

[Step 1] Initialize parameters *p, S, Nc, Ns, Nre, Ned, Ped, C(i)(i=1,2…S)*.

[Step 2] Represent the bacteria population in binary encoded form using sigmoid function $y = \frac{1}{1+e^{-x}} = \begin{cases} 1, & y > random[0,1] \\ 0, & y < random[0,1] \end{cases}$

[Step 2] Elimination-dispersal loop: *l*=*l*+1

[Step 3] Reproduction loop: *k*=*k*+1

[Step 4] Chemotaxis loop: *j*=*j*+1

[a] For *i* =1,2…S take a chemotactic step for bacterium *i* as follows.

[b] Compute fitness function, *J (i, j, k, l).*

$$\textbf{Fitness Value} = \delta\,\mathrm{CA} + \varphi\,\frac{\mathrm{TF}-\mathrm{SF}}{\mathrm{TF}}$$

The classification accuracy CA is calculated from the Naive Bayes classifier which is applied on the datasets having as number of features as suggested the given bacteria using 10 cross fold validation technique. Final fitness value is calculated as

$J(i, j, k, l) = J(i, j, k, l) + J_{cc}(\Theta^i(j, k, l), P(j, k, l))$

[c] Let $J_{last}=J(i, j, k, l)$ to save this value since we may find a better cost via a run.

[d] Tumble: generate a random vector $\Delta(i) \in R^p$ with each element $\Delta_m(i)$, m =1,2,..., p, a random number on [-1, 1].

[e] Move:
Let $\Theta^i(j+1, k, l) = \Theta^i(j, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$
This results in a step of size $C(i)$ in the direction of the tumble for bacterium i.

[f] Compute $J(i, j+1, k, l)$ and let,
$J(i, j+1, k, l) = J(i, j, k, l) + J_{cc}(\theta^i(j+1, k, l), P(j+1, k, l)$

[g] Swim

i) Let m=0 (counter for swim length).
ii) While $m < N_s$ (if have not climbed down too long).
• Let $m=m+1$.
• If $J(i, j+1, k, l) < J_{last}$ (if doing better), let $J_{last} = J(i, j+1, k, l)$ and let
$\Theta^i(j+1, k, l) = \Theta^i(j, k, l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$
And use this $\Theta^i(j+1, j, k)$ to compute the new $J(i, j+1, k, l)$ as we did in [f] Else, let $m = N_s$. This is the end of the while statement.

[h] Go to next bacterium (i+1) if i !=S (i.e., go to [b] to process the next bacterium).

[Step 5] If $j < N_c$, go to step 4. In this case continue chemotaxis since the life of the bacteria is not over.

[Step 6] Reproduction:

[a] For the given k and l, and for each i=1,2,..., S , let

$J_{health}^{i} = \sum_{j=1}^{N_{c+1}} j(i,j,k,l)$

be the health of the bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances)

sortbacteria and chemotactic parameters $C(i)$ in order of ascending cost J_{health} higher cost means lower health).

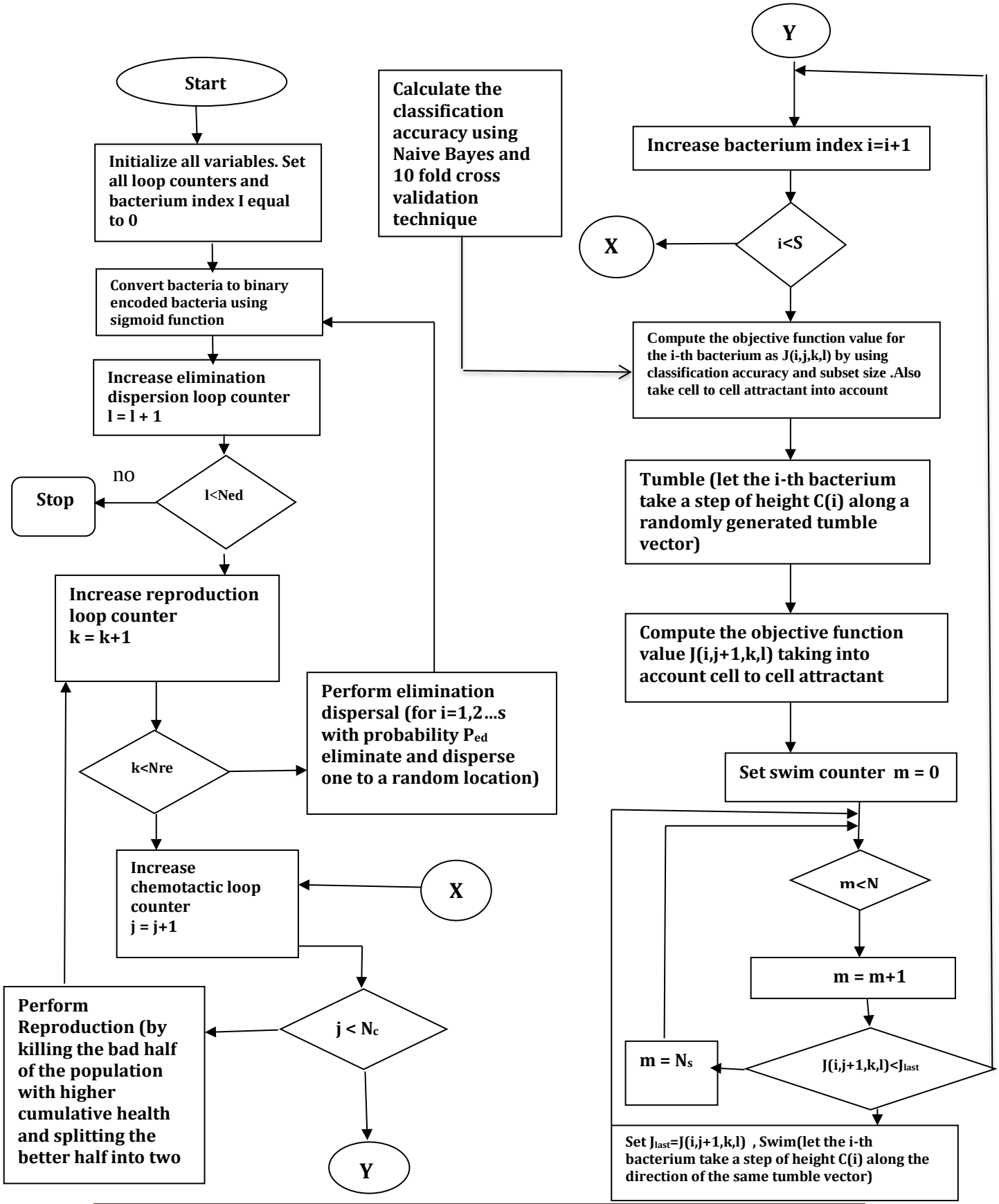
[b] The S_r bacteria with the highest J_{health} values die and the remaining S_r bacteria with the best values split (this process is performed by the copies that are made are placed at the same location as their parent).

[Step 7] If $K < N_{re}$, go to step 3. In this case, we have not reached the number of specified reproduction steps, so we start the next generation of the chemotactic loop.

[Step 8] Elimination-dispersal: For i=1,2..., S with probability P_{ed} , eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if a bacterium is eliminated, simply disperse another one to a random location on the optimization domain. If $l < N_{ed}$, then go to step 2; otherwise end.

4.4 Flow chart for the proposed work

Figure 5: Flow Chart of proposed work



Chapter 5 Result And Analysis

Problem statement is to minimize the number of features in datasets and to obtain higher classification accuracy. To achieve this objective, we compared the number of features and classification accuracies of our proposed algorithm with several well-known algorithms, which are Genetic Algorithms (GA) [28], Particle Swarm Optimization (PSO) [5], and Geometric Particle Swarm Optimization (GPSO) [29]. Similar to the our algorithm, we used Naive Bayes classifier for all other comparative algorithms as the attribute evaluator. However, the parameters for the algorithms had the same settings as those used by the original authors. This chapter will detail about the experimental setup, result and analysis of result obtained.

5.1 Experimental Setup

The experiment was conducted to evaluate our algorithm performance with other algorithms and the performance is evaluated using three different type of classifiers that we will discuss later in this chapter.

Furthermore, various parameters of the proposed approach are set as follows:

Table 1 Parameter and their values

Parameter	Value
S	25
$d_{attractant}$	0
$W_{attractant}$	10
$d_{repellant}$	0
$W_{repellant}$	1
N_s	4
P_{ed}	0.25
N_{ed}	2
Nre	2
Nc	10

The experiment was conducted on Macintosh operating system with version 10.10.2 with 2.5 GHz RAM and i5 Processor. The code is written in JAVA and run on Netbeans tool. After obtaining the resulting bacteria the evaluation is done using three different classifiers name as JRip, PART and J48[30] on WEKA tool.

5.2 Description of data sets

For the experiments, six datasets were considered which cover both cases of binary and multiclass data. Datasets were taken from the UCI data repository [31]. The datasets are Vote, Credit, Derm, Lung, Heart and Mushroom. Vote is widely used as a binary classification dataset. The dataset represents votes for each of the U.S. House of Representatives congressmen with the class label democrat and republican. Credit dataset is a binary classification data that is concerned with credit card applications. Derm represent real data in dermatology concerning differential diagnosis of erythematosquamous diseases. The class labels contain six values, which refer to six different diseases. Lung dataset is the pathological types of Lung cancer that aims to demonstrate the power of the optimal discriminant plane even in ill-posed settings. Heart is a binary class data that has been used to predict heart diseases, whereby the class label of zero and one refers to the absence or existence of heart disease in the patient. Finally, Mushroom is a binary class dataset that includes characterization of hypothetical samples identical to 23 types of gilled mushrooms in the Agaricus and Lepiota family. Table 2 shows the characteristics of the datasets

Table 2 Dataset and its characteristics

Datasets	No. of features	No. of Samples
Heart	13	294
Vote	16	300
Credit	20	1000
Mushroom	22	8124
Derm	34	366
Lung	56	32

5.3 Results of experiment

In this experiment, we compared our prosed algorithm against GA [28], PSO [5], and GPSO [29] in terms of the number of features selected from the original dataset and classification accuracy. Table 3, 4,5 and 6 provides the comparison results. The number of features obtained from the comparative algorithms is shown in Table 3 and the best results are highlighted in bold. The second aim of the experiment is to compare the classification accuracy of our algorithm with other three algorithms over 10 runs using 10 fold cross validation method. Three well-known classifiers were

employed for the purpose of evaluating the resulting subsets among different classifiers, which were JRip PART and J48 Tables 4, 5, and 6 show the average classification accuracy comparison respectively.

Table 3 Comparison on number of features

Dataset	Our algorithm	GA	GPSO	PSO
Heart	**6**	6	5.7	6.25
Vote	3	3	3.1	3.55
Credit	**8**	10	12	12
Mushroom	6	6	6	6
Derm	**16**	18.1	17.1	21.5
Lung	20	8.6	7.9	8.7

Table 4 Average of classification accuracy for Jrip.

Dataset	Our algorithm	GA	GPSO	PSO
Heart	81.4	81.97	81.66	82.47
Vote	**94.4**	93.66	93.92	94.42
Credit	70.20	70.70	70.75	70.48
Mushroom	**99.9**	100	99.4	99.30
Derm	**93.0**	89.0	90.18	91.0
Lung	71.4	84.6	83.7	84.5

Table 5 Average of classification accuracy for PART.

Dataset	Our algorithm	GA	GPSO	PSO
Heart	**80.3**	80	80.0	79.0
Vote	94.4	94.33	94.42	94.42
Credit	71.20	72.9	72.24	71.81
Mushroom	**99.9**	100	99.3	99.30
Derm	93.9	94.73	95.65	95.62
Lung	71.1	79.37	80.62	82.18

Table 6 Average of classification accuracy for J48.

Dataset	Our algorithm	GA	GPSO	PSO
Heart	**84**	79.27	79.57	79.79
Vote	94.4	94.0	94.13	94.39
Credit	**73.20**	72.2	72.21	72.08
Mushroom	**99.9**	100	99.4	99.3
Derm	94.8	94.73	95.65	95.62
Lung	72.2	79.37	86.62	86.6

In terms of number of feature, Table 3 shows that the proposed algorithm obtained the smallest or equal number of features across all datasets except for lung dataset. In

evaluating the feature subset, if we take into consideration the interaction between classification accuracy and number of features selected by the proposed algorithm as compared to other algorithms, we can categorize the results into three cases. In the first case, a reduced number of features deliver the same classification accuracy. This is shown in the datasets of heart and credit that produced similar classification accuracy in both JRip and PART classifiers.

In the second case, the proposed algorithm reduced the number of features while at the same time increased the classification accuracy. For example, our algorithm selected less number of features in case of Derm dataset for J48 classifier. In the third case, smaller feature subset that is selected delivers a slightly lower classification accuracy, such as in Heart and Mushroom dataset.

Finally, it can be noted from Tables 4, 5, and 6 that the classification accuracies achieved by the proposed algorithm is less as compared to other three different classifiers. This can be noted obviously from the experimental results using lungs dataset.

Comparison between the four algorithms is also represented in graphs form. Following figure 6, 7, 8, 9, 10 and 11 shows the comparison on the number of features and classification accuracy for heart, vote, credit, mushroom, Derm and Lung dataset respectively using JRip classifier.

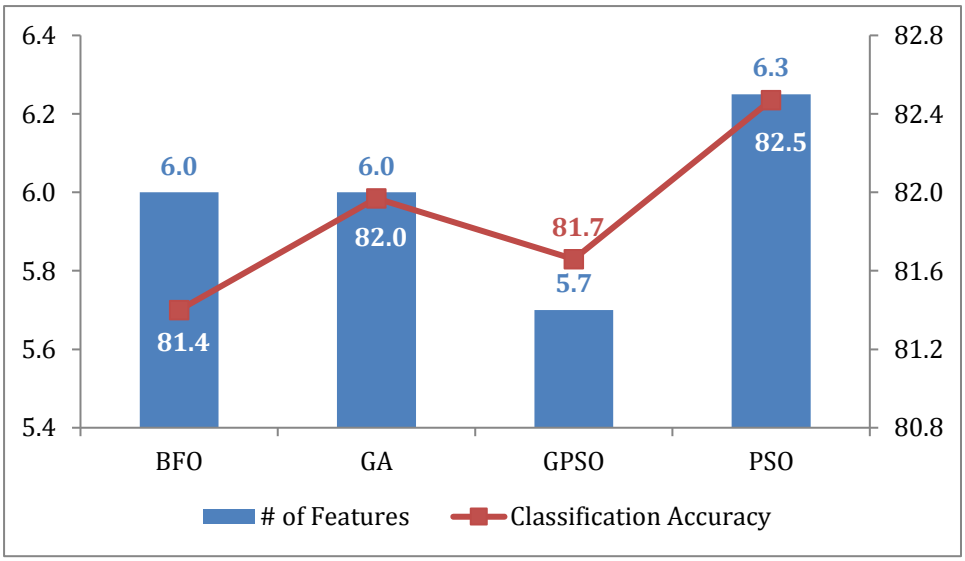


Figure 6: Comparison on Heart dataset using JRip classifier

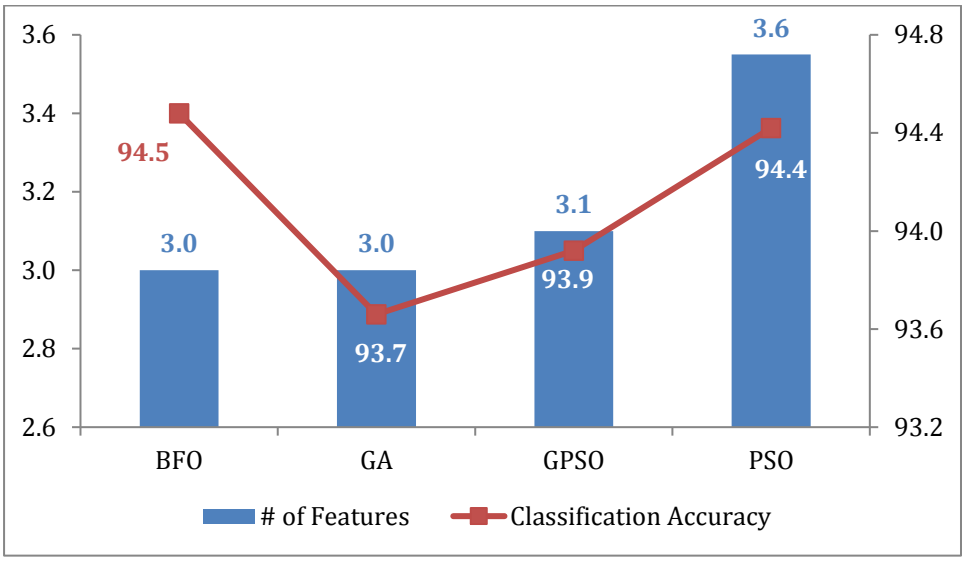


Figure 7: Comparison on Vote dataset using JRip classifier

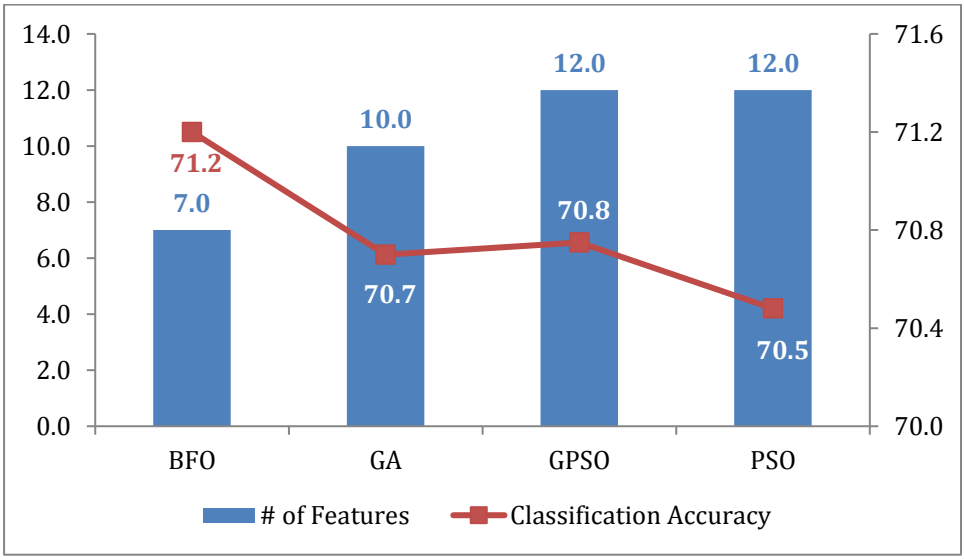


Figure 8: Comparison on Credit dataset using JRip classifier

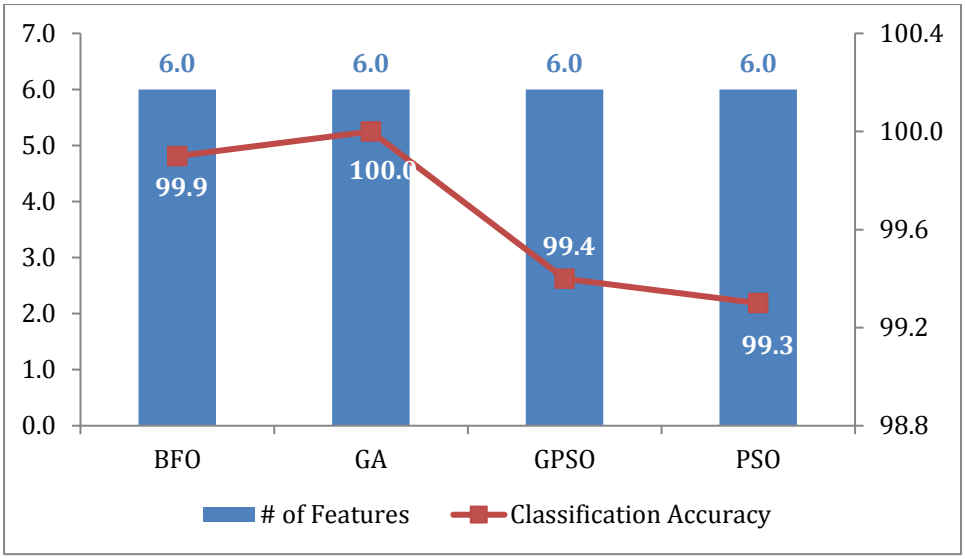


Figure 9: Comparison on Mushroom dataset using JRip classifier

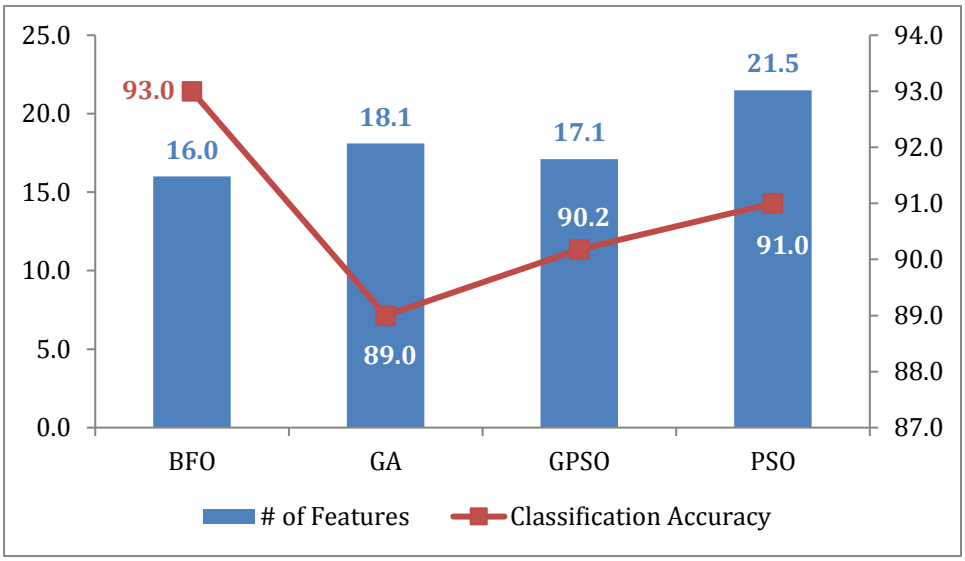


Figure 10: Comparison on Derm dataset using JRip classifier

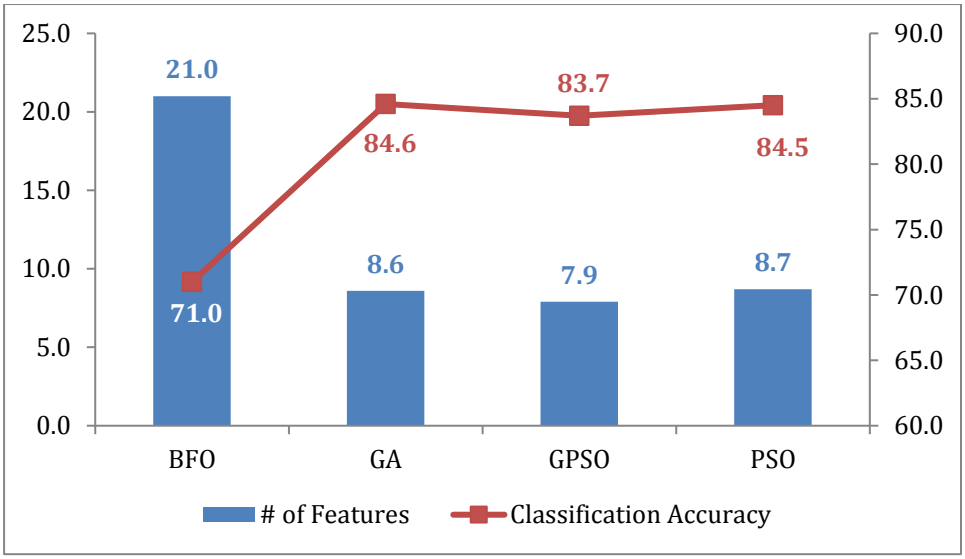


Figure 11: Comparison on Lung dataset using JRip classifier

Chapter 6 Conclusion And Future Work

A new hybrid feature selection algorithm has been presented. The BFA Algorithm employed Naive Bayes Algorithm to intelligently select the most convenient feature that could maximize the classification accuracy while ignoring redundant and noisy features. We compared our proposed algorithm with three other algorithms using six well-known UCI datasets. The performance was evaluated from two perspectives, which are the number of features and classification accuracy. From the experiments, we can conclude that the proposed Naive Bayes guided BFA Algorithm outperformed other metaheuristic algorithms with a selection of feature subsets that are smaller with a less number of features. In terms of classification accuracy, BFA has proven to achieve higher or equal accuracy as compared to other algorithms.

For future work, further investigations are required to observe the behavior of the proposed algorithm on very high-dimensional datasets. Moreover improvised BFA can be used for feature selection problem. Experiment with other known classifier

References

[1] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. J. Mach. Learn Res. 3 , 1157–1182.

[2] Devijver, P., & Kittler, J. (1982). Pattern Recognition: A Statistical Approach. Prentice Hall.

[3] Kira, K., & Rendell, L. (1992). A practical approach to feature selection. Internat. Conf. on Machine Learning, (pp. 249–256).

[4] Kohavi, R. Wrappers for Performance Enhancement and Oblivious Decision Graphs. Stanford University, Computer Science Department.

[5] Liu, H., & Motoda, H. (1998). Feature Selection for Knowledge Discovery and Data Mining .

[6] Whitney, , A. (1971). A direct method of nonparametric measurement selection. IEEE Trans. Comput., (pp. 1100-1103).

[7] Green, , D., & Marill, T. (1963). On the effectiveness of receptors in recognition system. IEEE Trans. Inform. Theory 9, (pp. 11-17).

[8] Pudil, P., Novovicova, J., & Kittler, J. (1994). Floating search methods in feature selection. Pattern Recognition Lett. 15 , 1119–1125.

[9] Pudil, P. F. (1994). Floating search methods for feature selection with nonmonotonic criterion functions. 12th IEEE Conf. on Pattern Recognition (pp. 279–283). Jerusalem, Israel.

[10] Huwan, L. (1998). Feature Selection for Knowledge Discovery and Data Mining. Kluwer Publishers Norwell, MA, USA.

[11] Reunanen, J. (2003). Overfitting in making comparisons between variable selection methods. J. Mach. Learn Res. 3 , 1371–1382.

[12] Narendra, P., & Fukunaga, K. A Branch and Bound Algorithm for Feature Subset Selection. IEEE Trans. Comput. 26 (9), (pp. 917–922).

[13] Backer, E., & Shipper, J. (1977). On the max–min approach for feature ordering and selection. Seminar on Pattern Recognition Liege.

[14] Kittler, J. (1978). Feature set search algorithms. In Sijthoff, & C. Chen (Ed.), Pattern Recognition and Signal Processing (pp. 41-60).

[15] Siedlecki, W., & Sklansky, J. (1989). A note on genetic algorithms for large-scale feature selection. . Pattern Recognition Lett. 10 , 335–347.

[16] Jensen, R., & Shen, Q. (2003). Finding rough set reducts with Ant colony optimization. (pp. 15-22). UK Workshop on Computational Intelligence.

[17] Sivagaminathan, R., & Ramakrishnan, S. (2007). A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert Systems with Applications , 33, 49–60.

[18] Jakhar, R., & Kaur, N. (2011). Face Recognition Using Bacteria Foraging Optimization Based Selected Features. International Journal of Advanced Computer Science and Applications, Special Issue on Artificial Intelligence .

[19] Kumar, D. P. (2014). Ground Water Prediction Using Bacterial Foraging Optimization Technique. American Journal of Computing Research Repository , pp 44-48.

[20] Gollapudi, S. V., Pattnaik, S. S., & Bajpai , O. (2008). Bacterial foraging optimization technique to calculate resonant frequency of rectangular microstrip antenna. 18 (4).

[21] Cho, J. H., & Kim , D. H. (2011). Intelligent Feature Selection by Bacterial Foraging Algorithm and Information Theory. In Advanced Communication and Networking (Vol. 199, pp. pp 238-244).

[22] Mangat, V., & Rani, D. (n.d.). Design of an efficient data classification method for disease diagnosis. International Journal of Engineering, Business and Enterprise Applications .

[23] Lutu, P. E. (2013). Fast Feature Selection for Naive Bayes Classification in Data Stream Mining. World Congress on Engineering & Computer Science.

[24] Fleuret, F. (2004). Fast Binary Feature Selection with Conditional Mutual Information. Journal of Machine Learning Research , 5.

[25] L.Ke, Z.Feng, & Z.Ren. (2008). An efficient ant colony optimization approach to attribute reduction in rough set theory. Pattern Recognition Letters , 29 (9), 1351-1357.

[26] Wang, X., Yang, J., Teng, X., Xia, W., & Jensen, R. (2007). Feature selection based on rough sets and particle swarm optimization. Pattern Recognition Letters , 28, 459–471.

[27] Webb, G. (2010). Naive Bayes. In Encyclopedia of MachineLearning (pp. 713-714). Springer.

[28] D.E.Goldberg. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning.

.

[29] Moraglio, A., & DiChio, C. (2007). Geometric particle swarm optimisation. 10th European Conference, (pp. 125–136).

[30] Duan, Q., Miao, D., Zhang, H., & Zheng, J. (2007). Personalized web retrieval based on rough-fuzzy method. Journal of Computational Information Systems , 3, 1067–1074.

[31] Blake, C., & Merz, C. (1998). UCI Repository of Machine Learning Databases. University of California . Irvine, Calif, USA.