Journal of Computer Science & Systems Biology - Open Access
Research Article JCSB/Vol.2 May-June 2009

Combination of Ant Colony Optimization and Bayesian Classification for Feature Selection in a Bioinformatics Dataset

Mehdi Hosseinzadeh Aghdam[1,*], Jafar Tanha[2], Ahmad Reza Naghsh-Nilchi[3] and Mohammad Ehsan Basiri[3]

[1]Computer Engineering Department, Technical & Engineering Faculty of Bonab, University of Tabriz, Tabriz, Iran
[2] Computer Engineering Department & IT, Payame Noor University, Lashgarak, MiniCity, Tehran, Iran
[3]Computer Engineering Department, Faculty of Engineering, University of Isfahan, Hezar Jerib Avenue, Isfahan, Iran

*Corresponding author: Mehdi Hosseinzadeh Aghdam, Computer Engineering Department, Technical & Engineering Faculty of Bonab, University of Tabriz, Tabriz, Iran, Tel: +98 311 7932671; Fax: +98 311 7932670; E-mail: hosseinzadeh@comp.ui.ac.ir, mhaghdam@yahoo.com

Received March 31, 2009; Accepted June 14, 2009; Published June 15, 2009

Citation: Aghdam MH, Tanha J, Naghsh-Nilchi AR, Basiri ME (2009) Combination of Ant Colony Optimization and Bayesian Classification for Feature Selection in a Bioinformatics Dataset. J Comput Sci Syst Biol 2: 186-199. doi:10.4172/jcsb.1000031

Abstract

Feature selection is widely used as the first stage of classification task to reduce the dimension of problem, decrease noise, improve speed and relieve memory constraints by the elimination of irrelevant or redundant features. One approach in the feature selection area is employing population-based optimization algorithms such as particle swarm optimization (PSO)-based method and ant colony optimization (ACO)-based method. Ant colony optimization algorithm is inspired by observation on real ants in their search for the shortest paths to food sources. Protein function prediction is an important problem in functional genomics. Typically, protein sequences are represented by feature vectors. A major problem of protein datasets that increase the complexity of classification models is their large number of features. This paper empowers the ant colony optimization algorithm by enabling the ACO to select features for a Bayesian classification method. The naive Bayesian classifier is a straightforward and frequently used method for supervised learning. It provides a flexible way for dealing with any number of features or classes, and is based on probability theory. This paper then compares the performance of the proposed ACO algorithm against the performance of a standard binary particle swarm optimization algorithm on the task of selecting features on Postsynaptic dataset. The criteria used for this comparison are maximizing predictive accuracy and finding the smallest subset of features. Simulation results on Postsynaptic dataset show that proposed method simplifies features effectively and obtains a higher classification accuracy compared to other feature selection methods.

Keywords: Feature selection; Ant colony optimization; Particle swarm optimization; Bayesian classification; Bioinformatics

Introduction

The feature selection problem can be viewed as a particular case of a more general subset selection problem in which the goal is to find a subset maximizing some adopted criterion. Feature selection methods search through the subsets of features and try to find the best subset among the competing 2^N-1 candidate subsets according to some evaluation measure, where N denotes the total number of features. Feature selection is used in many application areas as a tool to remove irrelevant and redundant features. The objective of feature selection is to simplify a dataset by reducing its dimensionality and identifying relevant underlying features without sacrificing predictive accuracy (Jensen, 2005). In many applications, the size of a dataset is so large that learning might not work as well before removing these

unwanted features. Reducing the number of irrelevant/ redundant features drastically reduces the running time of a learning algorithm and yields a more general concept. This helps in getting a better insight into the underlying concept of a real world classification problem (Dash and Liu, 1997).

Protein function prediction is an important problem in functional genomics. Proteins are large molecules that perform nearly all of the functions of a cell in a living organism (Alberts et al., 2002). The primary sequence of a protein consists of a long sequence of amino acids. Proteins are the most essential and versatile macromolecules of life, and the knowledge of their functions is a crucial link in the development of new drugs, better crops, and even the development of synthetic biochemical such as biofuels.

Over the past few decades, major advances in the field of molecular biology, coupled with advances in genomic technologies, have led to an explosive growth in the biological information generated by the scientific community. Although the number of proteins with known sequence has grown exponentially in the last few years, due to rapid advances in genome sequencing technology, the number of proteins with known structure and function has grown at a substantially lower rate (Freitas and de Carvalho, 2007).

Searching for similar sequences in protein databases is a common approach used in the prediction of a protein function. The objective of this search is to find a similar protein whose function is known and assigning its function to the new protein. Despite the simplicity and usefulness this method in a large number of situations, it has also some limitations (Freitas and de Carvalho, 2007). For instance, two proteins might have very similar sequences and perform different functions, or have very different sequences and perform a similar function. Additionally, the proteins being compared may be similar in regions of the sequence that are not determinants of their function.

Another approach that may be used alternatively or in complement to the similarity-based approach is to build a model for predictive classification. The goal of such a model is to classify data instances into one of a predefined set of classes or categories. In this approach a feature vector represents each protein, a learning algorithm captures the most important relationships between the features, and the classes present in the dataset. A major problem in protein datasets is the high dimensionality of the feature space. Most of these dimensions are not relative to protein function; even some noise data hurt the performance of the classifier. Hence, we need to select some representative features from the original feature space to reduce the dimensionality of feature space and improve the efficiency and performance of classifier.

Among too many methods which are proposed for feature selection, population-based optimization algorithms such as particle swarm optimization (PSO)-based method (Wang, 2007) and ant colony optimization (ACO)-based method (Aghdam et al., 2008) have attracted a lot of attention. These methods attempt to achieve better solutions by application of knowledge from previous iterations.

Particle swarm optimization comprises a set of search techniques, inspired by the behavior of natural swarms, for solving optimization problems (Kennedy and Eberhart, 2001). PSO is a global optimization algorithm for dealing with problems in which a point or surface in an *n*-dimensional space best represents a solution. Potential solutions are plotted in this space and seeded with an initial velocity. Particles move through the solution space and certain fitness criteria evaluate them. After a while particles accelerate toward those with better fitness values.

Meta-heuristic optimization algorithm based on ant's behavior was represented in the early 1990s by M. Dorigo and colleagues (Dorigo and Caro, 1999). Ant colony optimization is a branch of newly developed form of artificial intelligence called swarm intelligence. Swarm intelligence is a field which studies "the emergent collective intelligence of groups of simple agents" (Bonabeau et al., 1999). In groups of insects which live in colonies, such as ants and bees, an individual can only do simple task on its own, while the colony's cooperative work is the main reason determining the intelligent behavior it shows (Liu et al., 2004).

ACO algorithm is inspired by ant's social behavior. Ants have no sight and are capable of finding the shortest route between a food source and their nest by chemical materials called pheromone that they leave when moving (Bonabeau et al.,1999). ACO algorithm was firstly used for solving traveling salesman problem (TSP) (Dorigo et al., 1996) and then has been successfully applied to a large number of difficult problems like the quadratic assignment problem (QAP) (Maniezzo and Colorni, 1999), routing in telecommunication networks, graph coloring problems, scheduling, etc. This method is particularly attractive for feature selection as there seems to be no heuristic that can guide search to the optimal minimal subset every time (Jensen, 2005). On the other hand, if features are represented as a graph, ants will discover best feature combinations as they traverse the graph.

This paper proposed an ACO-based algorithm for the feature selection task in bioinformatics datasets. This paper extended by taking advantage of naive Bayes classifier and

enable ACO to select features for a Bayesian classification method, which is more sophisticated than the nearest neighbor classifier. Bayesian classifiers are statistical classifiers that can predict class membership probabilities, such as the probability that a given sample belongs to a particular class. Bayesian classification is based on the Bayes theorem (Feller, 1971).

The naive Bayes (NB) classifier uses a probabilistic approach to assign each record of the dataset to a possible class. A naive Bayes classifier makes significant use of the assumption that all features are conditionally independent of one another given the class. This assumption is called class conditional independence (Mitchell, 1996). It is made to simplify the computations involved and, in this sense, is considered naive. In practice, dependencies can exist between variables; however, when the assumption holds true, then the Naive Bayes classifier is the most accurate in comparison with other classifiers (Han and Kamber, 2001).

For testing the proposed ACO algorithm, it is applied to the problem of predicting whether or not a protein has a post-synaptic activity, based on features of protein's primary sequence and finally, the classifier performance and the length of selected feature subset are considered for performance evaluation.

The rest of this paper is organized as follows. Section 2 presents a brief overview of feature selection methods. Sections 3 and 4 briefly address *K*-nearest neighbor and naive Bayes classifiers. Ant colony optimization is described in sections 5. Section 6 explains the particle swarm optimization. Section 7 reports computational experiments. It also includes a brief discussion of the results obtained and finally the conclusion is offered in the last section.

Feature Selection Approaches

Feature selection is a process that selects a subset of original features. The optimality of a feature subset is measured by an evaluation criterion. As the dimensionality of a domain expands, the number of features increases. Finding an optimal feature subset is usually intractable (Kohavi and John, 1997) and many problems related to feature selection have been shown to be NP-hard. A typical feature selection process consists of four basic steps, namely, subset generation, subset evaluation, stopping criterion, and result validation (Dash and Liu, 1997). Subset generation is a search procedure that produces candidate feature subsets for evaluation based on a certain *search strategy*. Each candidate subset is evaluated and compared with the previous best one according to a certain *evaluation criterion*. If the new subset turns out to be better, it replaces the previous best subset. The process of subset generation and evaluation is repeated until a given stopping criterion is satisfied. Then the selected best subset usually needs to be validated by prior knowledge or different tests via synthetic and/or real world datasets.

The generation procedure implements a search method (Siedlecki and Sklansky, 1988) that generates subsets of features for evaluation. It may start with no features, all features, a selected feature set or some random feature subset. Those methods that start with an initial subset usually select these features heuristically beforehand. Features are added (*forward selection*) or removed (*backward elimination*) iteratively in the first two cases (Dash and Liu, 1997). In the last case, features are either iteratively added or removed or produced randomly thereafter (Jensen, 2005). The disadvantage of forward selection and backward elimination methods is that the features that were once selected/eliminated cannot be later discarded/re-selected. To overcome this problem, Pudil et al. proposed a method to flexibly add and remove features (Pudil et al., 1994). This method has been called floating search method.

According to the literature, the approaches to feature subset selection can be divided into filters and wrappers approaches (Guyon and Elisseeff, 2003). The filter model separates feature selection from classifier learning and selects feature subsets that are independent of any learning algorithm. It relies on various measures of the general characteristics of the training data such as distance, information, dependency, and consistency (Liu and Motoda, 1998). In the wrapper approach feature subset is selected using the evaluation function based on the same learning algorithm that will be used later for learning. In this approach the evaluation function calculates the suitability of a feature subset produced by the generation procedure and it also compares that with the previous best candidate, replacing it if found to be better. A stopping criterion is tested in each of iterations to determine whether or not the feature selection process should continue. Although, wrappers may produce better results, they are expensive to run and can break down with very large numbers of features. This is due to the use of learning algorithms in the evaluation of subsets, some of which can encounter problems while dealing with large datasets (Forman, 2003; Jensen, 2005).

Literature Review

John, Kohavi and Pfleger addressed the problem of irrelevant features and the subset selection problem. They presented definitions for irrelevance and for two degrees of relevance (weak and strong). They also state that features selected should depend not only on the features and the

target concept, but also on the induction algorithm (John et al., 1994).

Pudil, Novovicova and Kittler presented "floating" search methods in feature selection. These are sequential search methods characterized by a dynamically changing number of features included or eliminated at each step. They were shown to give very good results and to be computationally more effective than the branch and bound method (Pudil et al., 1994).

Dash and Liu gave a survey of feature selection methods for classification (Dash and Liu, 1997).

Kohavi and John introduced wrappers for feature subset selection. Their approach searches for an optimal feature subset tailored to a particular learning algorithm and a particular training set (Kohavi and John, 1997).

Yang and Honavar used a genetic algorithm for feature subset selection (Yang and Honavar, 1998).

Liu and Motoda wrote their book on feature selection which offers an overview of the methods developed since the 1970s and provides a general framework in order to examine these methods and categorize them (Liu and Motoda, 1998).

Forman presented an empirical comparison of twelve feature selection methods. Results revealed the surprising performance of a new feature selection metric, 'Bi-Normal Separation' (BNS) (Forman, 2003).

Guyon and Elisseeff gave an introduction to variable and feature selection. They recommend using a linear predictor of your choice (e.g. a linear SVM) and select variables in two alternate ways: (1) with a variable ranking method using correlation coefficient or mutual information; (2) with a nested subset selection method performing forward or backward selection or with multiplicative updates (Guyon and Elisseeff, 2003).

K-Nearest Neighbor Classifier

The *K*-nearest neighbor (KNN) algorithm is amongst the simplest of all machine learning algorithms. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *K* nearest neighbors. *K* is a positive integer, typically small. If $K = 1$, then the object is simply assigned to the class of its nearest neighbor. In binary (two class) classification problems, it is helpful to choose *K* to be an odd number as this avoids tied votes.

The same method can be used for regression, by simply assigning the property value for the object to be the average of the values of its *K* nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

The neighbors are taken from a set of objects for which the correct classification (or, in the case of regression, the value of the property) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. In order to identify neighbors, the objects are represented by position vectors in a multidimensional feature space. It is usual to use the Euclidean distance, though other distance measures, such as the Manhattan distance could in principle be used instead. The *K*-nearest neighbor algorithm is sensitive to the local structure of the data.

The performance of a KNN classifier is primarily determined by the choice of *K* as well as the distance metric applied (Latourrette, 2000). However, it has been shown in (Domeniconi et al., 2002) that when the points are not uniformly distributed, predetermining the value of *K* becomes difficult. Generally, larger values of *K* are more immune to the noise presented, and make boundaries smoother between classes. As a result, choosing the same (optimal) *K* becomes almost impossible for different applications.

Naive Bayes Classifier

One highly practical Bayesian learning method is the naive Bayes leaner, often called the naive Bayes (NB) classifier. In some domains its performance has been shown to be comparable to that of neural network and decision tree learning. This section introduces the naive Bayes classifier.

The naive Bayes classifier applies to classification tasks where each instance x is described by a conjunction of feature values and where the target function $f(x)$ can take on any value from some finite set V. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of feature values $<a_1,a_2...a_n>$. The classifier is asked to predict the target value, or classification, for this new instance (Mitchell, 1996).

The Bayesian approach to classifying the new instance is to assign the most probable target value, v_{MAP}, given the feature values $<a_1,a_2...a_n>$ that describe in instance.

$$v_{MAP} = \arg\max_{vj \in V} P(v_j \mid a_1, a_2 ... a_n) \quad (1)$$

We can use Bayes theorem to rewrite this expression as

$$v_{MAP} = \arg\max_{v_j \in V} \frac{P(a_1, a_2 ... a_n \mid v_j) P(v_j)}{P(a_1, a_2 ... a_n)}$$
$$= \arg\max_{v_j \in V} P(a_1, a_2 ... a_n \mid v_j) P(v_j) \quad (2)$$

We could attempt to estimate the two terms in equation (2) based on the training data. It is easy to estimate each of the $P(v_i)$ simply by counting the frequency with which each target value v_i occurs in the training data. However, estimating the different $P(a_1, a_2 ... a_n \mid v_j)$ terms in this fashion is not feasible unless we have a very, very large set of training data. The problem is that the number of these terms is equal to the number of possible instances times the number of possible target values. Therefore, we need to see every instance in the instance space many times in order to obtain reliable estimates.

The naive Bayes classifier is based on the simplifying assumption that the feature values are conditionally independent given the target value. In other words, the assumption is that given the target value of the instance, the probability of observing the conjunction $a_1, a_2 ... a_n$ is just the product of the probabilities for the individual features: $P(a_1, a_2 ... a_n \mid v_j) = P(v_j) \Pi_i P(a_i \mid v_j)$. Substituting this into equation (2), we have the approach used by the naive Bayes classifier.

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j) \quad (3)$$

where v_{NB} denotes the target value output by the naive Bayes classifier (Mitchell, 1996).

One interesting difference between the naive Bayes classification method and other classification methods we have considered is that there is no explicit search through the space of possible hypotheses (in this case, the space of possible hypotheses is the space of possible values that can be assigned to the various $P(v_i)$ and $P(a_i \mid v_i)$ terms). Instead, the hypothesis is formed without searching, simply by counting the frequency of various data combinations within the training examples.

To evaluate the performance of the Bayesan classification, we use a 10-fold cross-validation. We divide the data set into 10 equally sized folds. For all class levels each fold maintains roughly the same proportion of classes present in the whole data set before division (called stratified cross-validation). Eight of the ten folds are used to compute the probabilities for the Bayesian classification. The ninth fold is used as validation set and the tenth fold as test set. During the search for the solution only the validation set is used to compute predictive accuracy. The performance of the candidate solutions is given by the predictive accuracy of the classification in the validation set. The solution that shows the highest predictive accuracy on the validation set is then used to compute the predictive accuracy on the test set. Once the solution is selected, the nine folds are merged and this merged dataset is used to compute the probabilities for the Bayesian classification. The predictive accuracy (reported as the final result) is then computed on the previously untouched test set fold. Every fold will be once used as validation set and once used as test set. A similar process is adopted for the computation of the predictive accuracy using the nearest neighbor classifier, which will be described in more details in subsection 7.2.

Ant Colony Optimization

Ant colony optimization was introduced by Marco Dorigo (Dorigo, 1992) and his colleagues in the early 1990s. The first computational paradigm appeared under the name ant system (AS). It is another approach to stochastic combinatorial optimization. The search activities are distributed over: "ants" – agents with very simple basic capabilities that mimic the behavior of real ants. The main aim was not to simulate ant colonies, but to use artificial ant colonies as an optimization tool. Therefore the system exhibits several differences in comparison to the real (natural) ant colony: artificial ants have some memory; they are not completely blind; they live in an environment with discrete time. In ACO algorithms, artificial ants construct solutions from scratch by probabilistically making a sequence of local decisions. At each construction step an ant chooses exactly one of possibly several ways of extending the current partial solution. The rules that define the solution construction mechanism in ACO implicitly map the search space of the considered problem (including the partial solutions) onto a search tree.

The paradigm is based on the observation made by ethologists about the medium used by ants to communicate information regarding shortest paths to food by means of pheromone trails. A moving ant lays some pheromone on the ground, thus making a path by a trail of this substance. While an isolated ant moves practically at random (exploration), an ant encountering a previously laid trail can detect it and decide with high probability to follow it and consequently reinforce the trail with its own pheromone (exploitation). What emerges is a form of autocatalytic process through which the more the ants follow a trail, the more attractive that trail becomes to be followed. The process is thus characterized by a positive feedback loop, during which the probability of choosing a path increases with the number of ants that previously chose the same path. The mechanism above is the inspiration for the algorithms of the ACO family

(Engelbrecht, 2005).

Proposed ACO Algorithm for Feature Selection

Feature selection is one of the applications of subset problems. Given an feature set of size n, the feature selection problem is to find a minimal feature subset of size s ($s < n$) while retaining a suitably high accuracy in representing the original features. Therefore, there is no concept of path. A partial solution does not define any ordering among the components of the solution, and the next component to be selected is not necessarily influenced by the last component added to the partial solution (Aghdam et al., 2008). Furthermore, solutions to a feature selection problem are not necessarily of the same size. To apply an ACO algorithm to solve a feature selection problem, these aspects need to be addressed. The first problem is addressed by redefining the way that the representation graph is used.

The feature selection task may be reformulated into an ACO-suitable problem. Ant colony optimization requires a problem to be represented as a graph. Here nodes represent features, with the edges between them denoting the choice of the next feature. The search for the optimal feature subset is then an ant traversal through the graph where a minimum number of nodes are visited that satisfies the traversal stopping criterion. Figure 1 illustrates this setup. The ant is currently at node a and has a choice of which feature to add next to its path (dotted lines). It chooses feature b next based on the transition rule, then c and then d. Upon arrival at d, the current subset $\{a, b, c, d\}$ is determined to satisfy the traversal stopping criterion (e.g. suitably high classification accuracy has been achieved with this subset). The ant terminates its traversal and outputs this feature subset as a candidate for data reduction (Aghdam et al., 2008).

A suitable heuristic desirability of traversing between features could be any subset evaluation function for example, an entropy-based measure (Jensen, 2005) or rough set dependency measure (Pawlak, 1991). The heuristic desirability of traversal and edge pheromone levels are combined to form the so-called probabilistic transition rule, denoting the probability that ant k will include feature i in its solution at time step t:

$$P_i^k(t) = \begin{cases} \dfrac{[\tau_i(t)]^\alpha . [\eta_i]^\beta}{\sum_{l \in J^k} [\tau_l(t)]^\alpha . [\eta_l]^\beta} & if \ \ i \in J^k \\ 0 & otherwise \end{cases} \quad (4)$$

where η_j is the heuristic desirability of choosing feature i (η_j is optional but often needed for achieving a high algorithm performance), J^k is the set of feasible features that can be added to the partial solution. $\alpha > 0$, $\beta > 0$ are two parameters that determine the relative importance of the pheromone value and heuristic information (the choice of α, β is determined experimentally) and $\tau_j(t)$ is the amount of virtual pheromone on feature i.

The pheromone on each feature is updated according to the following formula:

$$\tau_i(t+1) = (1-\rho).\tau_i(t) + \sum_{k=1}^{m} \Delta_i^k(t) \quad (5)$$

where

$$\Delta_i^k(t) = \begin{cases} \psi(S^k(t))/|S^k(t)| & if \ \ i \in S^k(t) \\ 0 & otherwise \end{cases} \quad (6)$$

The value $0 = \rho = 1$ is decay constant used to simulate the evaporation of the pheromone, $S^k(t)$ is the feature subset found by ant k at iteration t, and $|S^k(t)|$ is its length. The pheromone is updated according to both the measure of the "goodness" of the ant's feature subset (ψ) and the size of the subset itself. By this definition, all ants can update the pheromone.

The overall process of ACO feature selection can be seen in Figure 2. The process begins by generating a number of ants, m, which are then placed randomly on the graph i.e. each ant starts with one random feature. Alternatively, the number of ants to place on the graph may be set equal to the number of features within the data; each ant starts path construction at a different feature. From these initial positions, they traverse edges probabilistically until a traversal stopping criterion is satisfied. The resulting subsets are gathered and then evaluated. If an optimal subset has been found or the algorithm has executed a certain number of times, then the process halts and outputs the best feature subset encountered. If none of these conditions hold, then the pheromone is updated, a new set of ants are created and the process iterates once more.

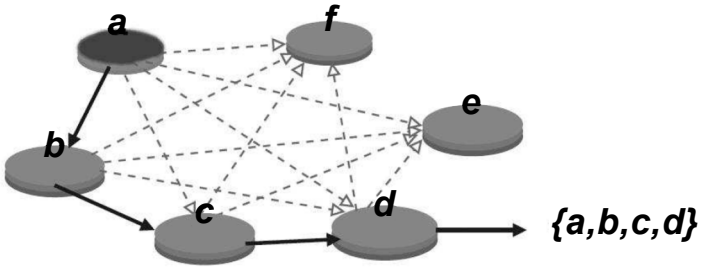

Figure 1: ACO problem representation for feature selection.

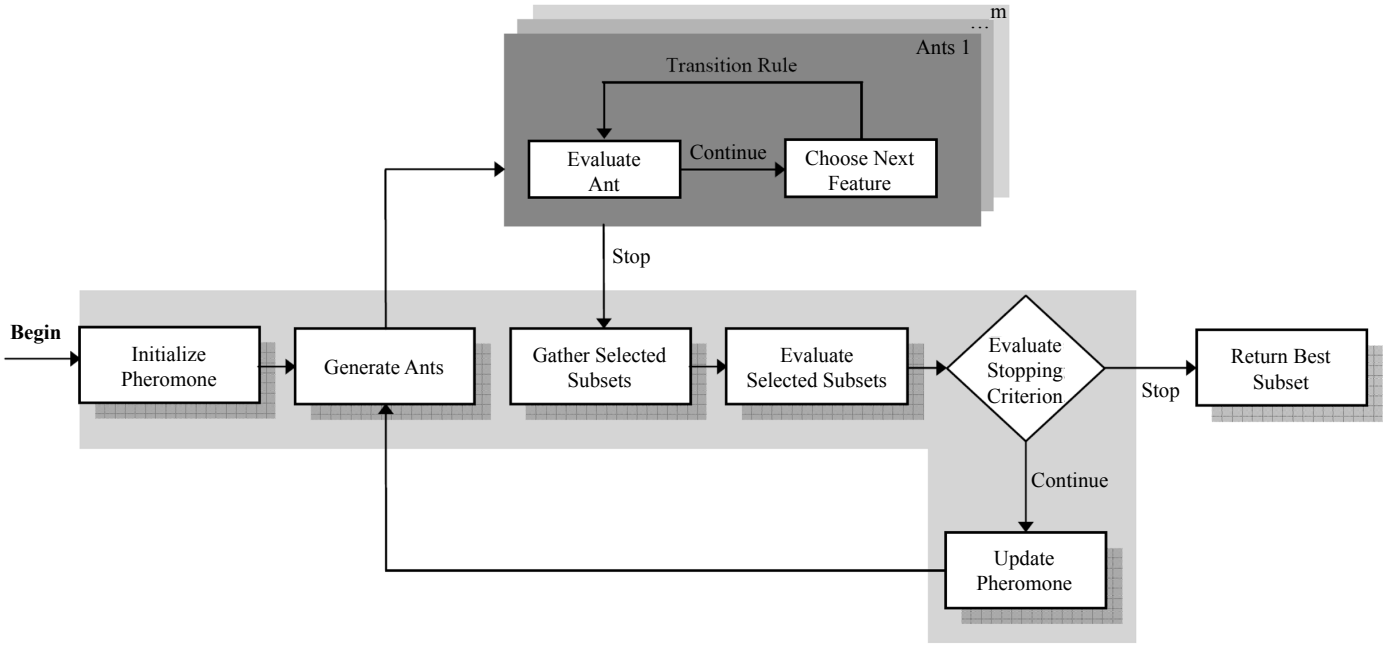

Figure 2: ACO-based feature selection algorithm.

The main steps of proposed feature selection algorithm are as follows:

1. Generation of ants and pheromone initialization.
   - Determine the population of ants.
   - Set the intensity of pheromone trial associated with any feature.
   - Determine the maximum of allowed iterations.
2. Ant foraging and evaluation.
   - Any ant randomly is assigned to one feature and it should visit all features and build solutions completely.
   - The evaluation criterion is mean square error (MSE) of the classifier. If an ant is not able to decrease the MSE of the classifier in five successive steps, it will finish its work and exit.
3. Evaluation of the selected subsets.
   - Sort selected subsets according to classifier performance and their length. Then, select the best subset.
4. Check the stop criterion.
   - Exit, if the number of iterations is more than the maximum allowed iteration, otherwise continue.
5. Pheromone updating.
   - Decrease pheromone concentrations of nodes then, all ants deposit the quantity of pheromone on graph. Finally, allow the best ant to deposit additional pheromone on nodes.
6. Generation of new ants.
   - In this step previous ants are removed and new ants are generated.
7. Go to 2 and continue.

The time complexity of proposed algorithm is $O(Imn)$, where I is the number of iterations, m the number of ants, and n the number of original features. This can be seen from Figure 2. In the worst case, each ant selects all the features. As the heuristic is evaluated after each feature is added to the candidate subset, this will result in n evaluations per ant. After the first iteration in this algorithm, mn evaluations will have been performed. After I iterations, the heuristic will be evaluated Imn times.

Particle Swarm Optimization

Particle swarm optimization is an evolutionary computation technique developed by Kennedy and Eberhart (Kennedy and Eberhart, 2001). The original intent was to graphically simulate the graceful but unpredictable movements of a flock of birds. Initial simulations were modified to form the original version of PSO. Later, Shi introduced inertia weight into the particle swarm optimizer to produce the standard PSO (Kennedy and Eberhart, 2001).

PSO is initialized with a population of random solutions, called "particles". Each particle is treated as a point in an n-dimensional space. The ith particle is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. The best previous position (*pbest*, the position giving the best fitness value) of any particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{in})$. The

index of the best particle among all the particles in the population is represented by the symbol "*gbest*". The rate of the position change (velocity) for particle *i* is represented as $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$. The particles are manipulated according to the following equation:

$$V_i(t+1) = w.V_i(t) + c_1.r_1(t).[P_i(t) - X_i(t)] + c_2.r_2(t).[P_g(t) - X_i(t)] \quad (7)$$

where

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (8)$$

Where *w* is the inertia weight, it is a positive linear function of time changing according to the generation iteration. Suitable selection of the inertia weight provides a balance between global and local exploration and results in less iteration on average to find a sufficiently optimal solution. The acceleration constants c_1 and c_2 in equation (7) represent the weighting of the stochastic acceleration terms that pull each particle toward *pbest* and *gbest* positions. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement toward, or past, target regions. $r_{1d}(t)$ and $r_{2d}(t) \sim U(0,1)$ are random values in the range [0,1], sampled from a uniform distribution.

Particles' velocities on each dimension are limited to a maximum velocity, V_{max}. It determines how large steps through the solution space each particle is allowed to take. If V_{max} is too small, particles may not explore sufficiently beyond locally good regions. They could become trapped in local optima. On the other hand, if V_{max} is too high particles might fly past good solutions.

The first part of equation (7) provides the ''flying particles'' with a degree of memory capability allowing the exploration of new search space areas. The second part is the ''cognition'' part, which represents the private thinking of the particle itself. The third part is the ''social'' part, which represents the collaboration among the particles. Equation (7) is used to calculate the particle's new velocity according to its previous velocity and the distances of its current position from its own best experience (position) and the group's best experience. Then the particle flies toward a new position according to equation (8). The performance of each particle is measured according to a pre-defined fitness function.

PSO for Feature Selection

The idea of PSO can be used for the optimal feature selection problem. Consider a large feature space full of feature subsets. Each feature subset can be seen as a point or position in such a space. If there are *n* total features, then there will be 2^n kinds of subsets, different from each other in the length and features contained in each subset. The optimal position is the subset with the least length and the highest classification accuracy. A particle swarm is put into this feature space, each particle takes one position. The particles fly in this space, their goal is to fly to the best position. After a while, they change their position, communicate with each other, and search around the local best and global best position. Eventually, they should converge on good, possibly optimal, positions. It is the exploration ability of particle swarms that equips them for performing feature selection and discovering optimal subsets.

Experimental Results

In this section, we report and discuss computational experiments and compare ACO feature selection algorithm with PSO-based approach. The quality of a candidate solution is computed by the naive Bayes classifier which described in section 2. Finally the classifier performance and the length of selected feature subset are considered for evaluating the proposed algorithm.

A series of experiments was conducted to show the utility of proposed feature selection algorithm. All experiments have been run on a machine with 3.0GHz CPU and 1024 MB of RAM. We implement ACO algorithm and PSO-based algorithm in Java and Weka 3.5.5. The operating system was Windows XP Professional. For experimental studies we have considered Postsynaptic dataset. The following sections describe Postsynaptic dataset and implementation results.

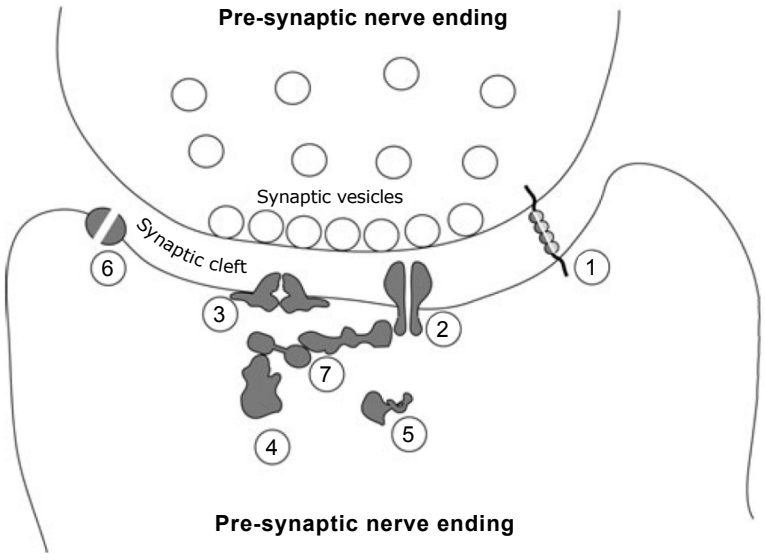

Figure 3: Main elements involved in pre-synaptic and post-synaptic activity.

Postsynaptic Dataset

This section presents Postsynaptic dataset used in the present work for feature selection. A synapse is a connection between two neurons: pre-synaptic and post-synaptic. The first is usually the sender of some signals such as the release of chemicals, while the second is the receiver. A post-synaptic receptor is a sensor on the surface of a neuron. It captures messenger molecules from the nervous system, neurotransmitters, and thereby functions in transmitting information from one neuron to another (Pappa et al., 2005).

The main elements found in synapses are shown in Figure 3. The cells are held together by cell adhesion molecules (1). In the cell where the signal is coming from (the pre-synaptic cell) neurotransmitters are stored in bags called synaptic vesicles. When signals are to be transmitted from the pre-synaptic cell to the post-synaptic cell, synaptic vesicles fuse with the pre-synaptic membrane and release their contents into the synaptic cleft between the cells. The transmitters then diffuse within the cleft, and some of them meet a post-synaptic receptor (2), which recognizes them as a signal. This activates the receptor, which then transmits the signal on to other signalling components such as voltage-gated ion channels (3), protein kinases (4) and phosphatases (5). To ensure that the signal has terminated, transporters (6) remove neurotransmitters from the cleft. Within the post-synaptic cell, the signalling apparatus is organized by various scaffolding proteins (7).

Postsynaptic dataset has been recently created and mined in (Basiri et al., 2008; Correa et al., 2006; Pappa et al., 2005). The dataset contains 4303 records of proteins. These proteins belong to either positive or negative classes. Proteins that belong to the positive class have post-synaptic activity while negative ones don't show such activity. From the 4303 proteins on the dataset, 260 belong to the positive class and 4043 to the negative class. This dataset has many features which makes the feature selection task challenging. More precisely, each protein has 443 PROSITE patterns, or features. PROSITE is a database of protein families and domains. It is based on the observation that, while there are a huge number of different proteins, most of them can be grouped, on the basis of similarities in their sequences, into a limited number of families (a protein consists of a sequence of amino acids). PROSITE patterns are small regions within a protein that present a high sequence similarity when compared to other proteins. In our dataset the absence of a given PROSITE pattern is indicated by a value of 0 for the feature corresponding to that PROSITE pattern which its presence is indicated by a value of 1 for that same feature (Correa et al., 2006).

Experimental Methodology

The computational experiments involved a ten-fold cross-validation method (Witten & Frank, 2005). First, the 4303 records in the Postsynaptic dataset were divided into 10 almost equally sized folds. There are three folds containing 431 records each one and seven folds containing 430 records each one. The folds were randomly generated but under the following regulation. The proportion of positive and negative classes in every single fold must be similar to the one found in the original dataset containing all the 4303 records. This is known as stratified cross-validation. Each of the 10 folds is used once as test set and the remaining of the dataset is used as training set. Out of the 9 folds in the training set, one is reserved to be used as a validation set.

In each of the 10 iterations of the cross-validation procedure, the predictive accuracy of the classification is assessed by 3 different methods:

- **Using all the 443 original features:** all possible features are used by the nearest neighbor classifier and the naive Bayes classifier.
- **Standard binary PSO algorithm:** only the features selected by the best particle found by the binary PSO algorithm are used by the nearest neighbor classifier and the naive Bayes classifier.
- **Proposed ACO algorithm:** only the features selected by the best ant found by the ACO algorithm are used by the nearest neighbor classifier and the naive Bayes classifier.

As the standard binary PSO and the ACO algorithm are stochastic algorithms, 20 independent runs for each algorithm were performed for every iteration of the cross-validation procedure. The obtained results, averaged over 20 runs, are reported in Table 2. The average number of fea-

Method	Population	Iteration	Initial pheromone	c_1	c_2	w	α	β	ρ
BPSO	30	50	-	2	2	0.8	-	-	-
ACO	30	50	1	-	-	-	1	0.1	0.2

Table 1: Binary PSO and ACO parameter settings.

tures selected by the feature selection algorithms has always been rounded to the nearest integer.

Various values were tested for the parameters of ACO algorithm. The results show that the highest performance is achieved by setting the parameters to values shown in Table 1.

Where w is inertia weight and c_1 and c_2 are acceleration constants of standard binary PSO algorithm. The choice of the value of this parameter was based on the work presented in (Shi, & Eberhart, 1998). For ant colony optimization, parameter values were empirically determined in our preliminary experiments for leading to better convergence; but we make no claim that these are optimal values. Parameter optimization is a topic for future research.

Performance Measure

The measurement of the predictive accuracy rate of a model should be a reliable estimate of how well that model classifies the test examples (unseen during the training phase) on the target problem. In data mining, typically, the following equation is used to assess the accuracy rate of a classifier:

$$\text{Standard accuracy rate} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \quad (9)$$

where *TP* (true positives) is the number of records correctly classified as positive class and *FP* (false positives) is the number of records incorrectly classified as positive class. *TN* (true negatives) is the number of records correctly classified as negative class and *FN* (false negatives) is the number of records incorrectly classified as negative class.

Nevertheless, if the class distribution is highly unbalanced, which is the case with the Postsynaptic dataset, equation (9) is an ineffective way of measuring the accuracy rate of a model. For instance, on a dataset in which 10% of the examples belong to the positive class and 90% to the negative class, it would be easy to maximize equation (9) by simply predicting always the majority class. Therefore, on our experiments we use a more demanding measurement for the accuracy rate of a classification model. It has also been used before in (Basiri et al., 2008; Pappa et al., 2005). This measurement is given by the equation:

$$\text{Predictive accuracy rate} = TPR \times TNR \quad (10)$$

where *TPR* and *TNR* are defined as follows:

$$TPR = \frac{TP}{TP + FN} \quad (11)$$

$$TNR = \frac{TN}{TN + FP}. \quad (12)$$

Results

Table 2 gives the optimal selected features for each method. As discussed earlier the experiments involved 200 runs of ACO and standard binary PSO, 10 cross- validation folds times 20 runs with different random seeds. Presumably, those 200 runs selected different subsets of features. So, the features which have been listed in Table 2 are the ones most often selected by ACO and standard binary PSO across all the 20 runs. Both ACO-based and PSO-based methods significantly reduce the number of original features, however; ACO-based method chooses fewer features.

Another trend observed in the results was found at each run of the ACO algorithm. We recorded the best ant by the ACO algorithm on each of the 20 runs and for every one of the 10 folds. We then computed the frequency of the features selected on the 10 × 20 = 200 best ants found by the ACO algorithm. The following 3 features have been selected, by ACO algorithm, in more than 80% of its respective 200 best ants found: F_{342}, F_{352} and F_{353}. The names of

Method	Selected Features	Number of Selected Features
BPSO	134, 162, 186, 320, 321, 333, 342, 351, 352, 353	10
ACO	352, 381, 419, 353, 342	5

Table 2: Selected features of standard binary PSO and ACO algorithms.

Feature / PROSITE pattern ID	Name
F_{342}/ps00410	Dynamin
F_{352}/ps00236	Neurotransmitter-gated ion-channel
F_{353}/ps00237	Rhodopsin-like GPCR superfamily

Table 3: PROSITE patterns selected in more than 80% of the runs performed by the ACO algorithm.

	Using all the 443 original feature			Standard Binary PSO algorithm				Proposed ACO Algorithm			
Fold	TPR	TNR	TPR × TNR	TPR	TNR	TPR × TNR	No. of selected features	TPR	TNR	TPR × TNR	No. of selected features
1	1.00	1.00	1.00	1.00	1.00	1.00	16	1.00	1.00	1.00	6
2	1.00	1.00	1.00	1.00	1.00	1.00	19	1.00	1.00	1.00	2
3	1.00	1.00	1.00	1.00	1.00	1.00	21	1.00	1.00	1.00	4
4	0.73	1.00	0.73	0.76	1.00	0.76	14	0.73	1.00	0.73	3
5	0.00	1.00	0.00	0.69	0.92	0.63	17	0.73	0.96	0.70	5
6	0.00	1.00	0.00	0.65	0.96	0.62	18	0.88	0.99	0.87	5
7	0.92	1.00	0.92	0.88	1.00	0.88	11	0.92	1.00	0.92	9
8	1.00	1.00	1.00	0.69	1.00	0.69	15	1.00	1.00	1.00	6
9	0.73	1.00	0.73	0.73	1.00	0.73	9	0.73	1.00	0.73	3
10	0.42	1.00	0.42	0.42	1.00	0.42	14	0.42	1.00	0.42	8
AVG	**0.68**	**1.00**	**0.68**	**0.78**	**0.99**	**0.77**	**15.4**	**0.84**	**0.99**	**0.83**	**5.1**

Table 4: Comparison of obtained results for ACO and standard binary PSO using nearest neighbor.

	Using all the 443 original feature			Standard Binary PSO algorithm				Proposed ACO Algorithm			
Fold	TPR	TNR	TPR × TNR	TPR	TNR	TPR × TNR	No. of selected features	TPR	TNR	TPR × TNR	No. of selected features
1	0.92	1.00	0.92	0.88	1.00	0.88	12	0.92	1.00	0.92	6
2	0.69	1.00	0.69	0.69	1.00	0.69	13	1.00	1.00	1.00	4
3	0.73	1.00	0.73	0.73	1.00	0.73	11	0.92	1.00	0.92	4
4	0.00	1.00	0.00	0.88	0.96	0.84	9	0.88	0.96	0.84	5
5	1.00	1.00	1.00	1.00	1.00	1.00	11	1.00	1.00	1.00	4
6	0.69	0.96	0.66	0.69	0.96	0.66	13	0.73	0.96	0.70	5
7	0.42	1.00	0.42	0.42	1.00	0.42	8	0.42	1.00	0.42	5
8	1.00	1.00	1.00	1.00	1.00	1.00	7	1.00	1.00	1.00	2
9	1.00	1.00	1.00	1.00	1.00	1.00	15	1.00	1.00	1.00	3
10	0.76	1.00	0.76	0.76	1.00	0.76	10	0.76	1.00	0.76	4
AVG	**0.72**	**0.99**	**0.71**	**0.80**	**0.99**	**0.79**	**10.9**	**0.86**	**0.99**	**0.85**	**4.2**

Table 5: Comparison of obtained results for ACO and standard binary PSO using naïve Bayes classifier.

the PROSITE patterns that correspond to these features are shown in Table 3.

The information was obtained from the web site of the European Bioinformatics Institute, UniProtKB/Swiss-Prot (http://www.ebi.ac.uk/swissprot/).

Also, the results of both algorithms for all of the 10 folds are summarized in Tables 4 and 5. The classification quality and feature subset length are two criteria which are considered to assess the performance of algorithms. Comparing these criteria, we noted that ACO and standard binary PSO algorithms did very better than the Baseline algorithm (using all features). Furthermore, for all of the 10 folds the ACO algorithm selected a smaller subset of features than the standard binary PSO algorithm.

As we can see in Table 5, the average number of selected features for standard binary PSO algorithm was equal to 10.9 with the average predictive accuracy of 0.79 and the average number of selected features for ACO algorithm was equal to 4.2 with the average predictive accuracy of 0.85. Furthermore, in (Correa et al., 2006) a new discrete PSO algorithm, called DPSO, has been introduced for feature selection. DPSO has been applied to Postsynaptic dataset and the average number of features selected by that was 12.70 with the average predictive accuracy of 0.74. Comparison of these three algorithms shows that ACO tends to select a smaller subset of features than the standard binary PSO algorithm and DPSO. Also, the average predictive accuracy of ACO is higher than that of the standard binary PSO algorithm and DPSO. Predictive accuracy and number of selected features for ACO and stan-

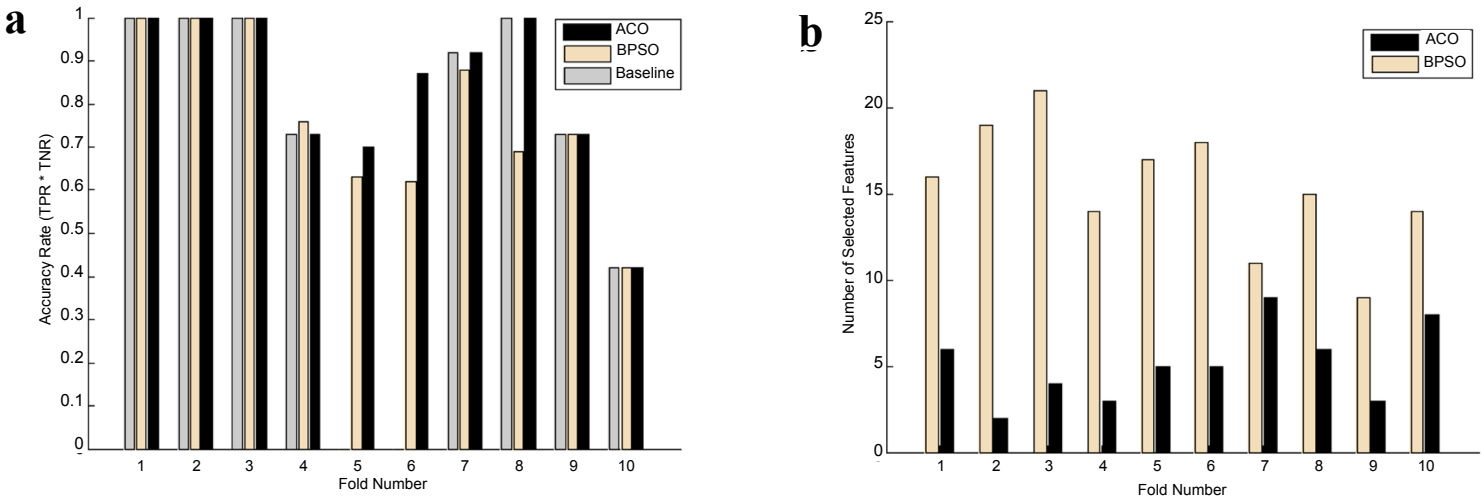

Figure 4: (a) Accuracy rate of feature subsets obtained using three methods. (b) Number selected features. (Using nearest neighbor classifier)

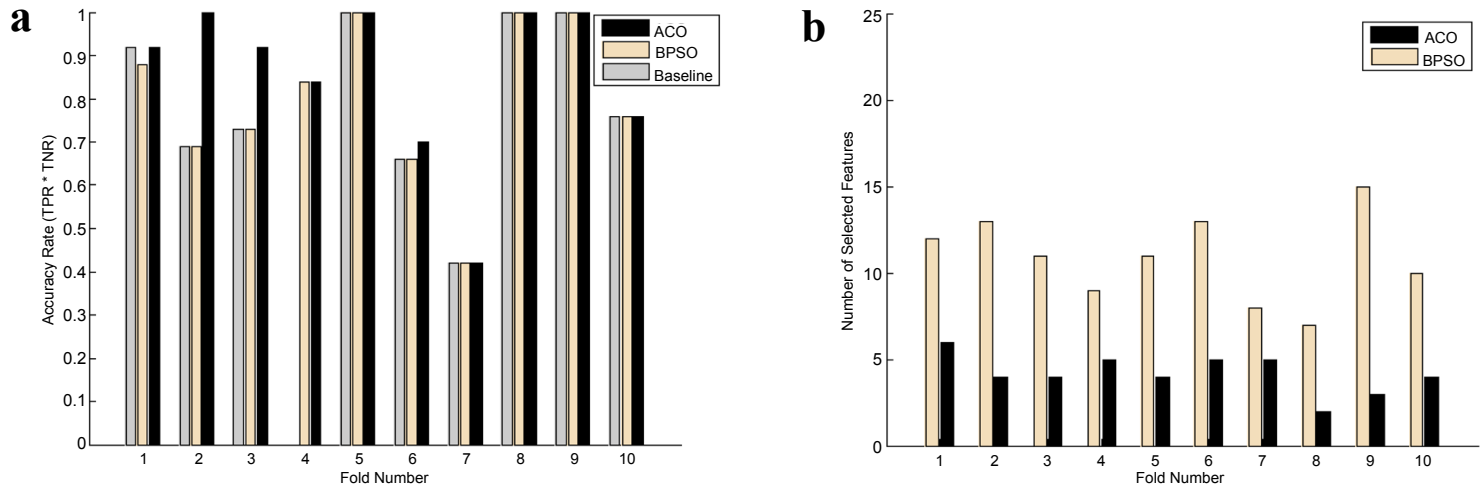

Figure 5: (a) Accuracy rate of feature subsets obtained using three methods. (b) Number selected features. (Using naive Bayes classifier)

dard binary PSO algorithm are shown in Figures 4 and 5.

Discussion

Experimental results show that the use of unnecessary features hurt classification accuracy and FS is used to reduce redundancy in the information provided by the selected features. Using only a small subset of selected features, the ACO and the PSO algorithms obtained better classification accuracy than the baseline algorithm using all features.

ACO shares many similarities with evolutionary computation (EC) techniques in general and GAs in particular. These techniques begin with a group of a randomly generated population and utilize a fitness value to evaluate the population. They all update the population and search for the optimum with random techniques.

Both ACO and PSO are stochastic population-based search approaches that depend on information sharing among their population members to enhance their search processes using a combination of deterministic and probabilistic rules. They are efficient, adaptive and robust search processes, producing near optimal solutions, and have a large degree of implicit parallelism. The main difference between the ACO compared to PSO, is that ACO does not have PSO operators such as inertia weight and acceleration constants. Ants update themselves with the pheromone update rule; they also have a memory that is important to the algorithm.

Compared to PSO, the ACO has a much more intelligent background and can be implemented more easily. The computation time used in ACO is less than in PSO. The param-

eters used in ACO are also fewer. However, if the proper parameter values are set, the results can easily be optimized. The decision on the parameters of the ant colony affects the exploration–exploitation tradeoff and is highly dependent on the form of the objective function. Successful feature selection was obtained even using conservative values for the ACO basic parameters.

Conclusion

Experimental results show that the use of unnecessary features decrease classifiers' performance and hurt classification accuracy and feature selection is used to reduces redundancy in the information provided by the selected features. Using only a small subset of selected features, the binary PSO and the ACO algorithms obtained better predictive accuracy than the Baseline algorithm using all features. ACO has the ability to converge quickly; it has a strong search capability in the problem space and can efficiently find minimal feature subset. Experimental results demonstrate competitive performance.

The ACO clearly enhances computational efficiency of the classifier by selecting fewer features than the standard binary PSO algorithm. Therefore, when the difference in predictive accuracy is insignificant, ACO is still preferable. In the proposed ACO algorithm, the classifier performance and the length of selected feature subset are adopted as heuristic information. So, we can select the optimal feature subset without the prior knowledge of features.

Also as we expected, computational results show the clear difference in performance between nearest neighbor and naive Bayes classifiers. To summarize, the naive Bayes classification method involves a classification step in which the various $P(v_i)$ and $P(a_i \mid v_i)$ terms are estimated, based on their frequencies over the training data. The set of these estimates corresponds to the learned hypothesis. This hypothesis is then used to classify each new instance by applying the rule in equation (3). Whenever the naive Bayes assumption of conditional independence is satisfied, this naive Bayes classification v_{NB} is identical to the maximum a posterior classification. The naive Bayes approach outperformed the nearest neighbor approach in all experiments.

Acknowledgments

The authors wish to thank the Office of Graduate studies of the University of Isfahan for their support and also wish to offer their special thanks to Dr. Alex A. Freitas for providing Postsynaptic dataset.

References

1. Aghdam MH, Ghasem-Aghaee N, Basiri ME (2008) Application of Ant Colony Optimization for Feature Selection in Text Categorization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008), pp. 2872-2878.

2. Alberts B, Bruce A, Johnson A, Lewis J, Raff M, et al. (2002) The molecular biology of the cell (4th ed.). Garland Press.

3. Basiri ME, Ghasem-Aghaee N, Aghdam MH (2008) Using ant colony optimization-based selected features for predicting post-synaptic activity in proteins. Proceedings of 6th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics LNCS 4973: pp12-23. » CrossRef» Google Scholar

4. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York, 1999. » CrossRef » Google Scholar

5. Correa ES, Freitas AA, Johnson CG (2006) A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics dataset. Proceedings of the Genetic and Evolutionary Computation Conference pp 35-42. » CrossRef » Google Scholar

6. Dash M, Liu H (1997) Feature Selection for Classification. Intelligent Data Analysis 3: pp131-156. » CrossRef » Google Scholar

7. Domeniconi C, Peng J, Gunopulos D (2002) Locally adaptive metric nearest-neighbor classification. IEEE Transaction Pattern Analysis and Machine Intelligence. 24: 1281-1285.» CrossRef » Google Scholar

8. Dorigo M (1992) Optimization, learning and natural algorithms. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy.

9. Dorigo M, Caro GD (1999) Ant Colony Optimization: A New Meta-heuristic. Proceedings of IEEE Congress on Evolutionary Computing, 1999.» CrossRef » Google Scholar

10. Dorigo M, Maniezzo V, Colorni A (1996) The Ant System: Optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern 26: 29-41. » CrossRef » Pubmed » Google Scholar

11. Engelbrecht AP (2005) Fundamentals of Computational Swarm Intelligence. Wiley, London, 2005. » CrossRef » Google Scholar

12. Feller W (1971) An introduction to probability theory and its applications. John Wiley and Sons, Inc., New York, NY, USA, 1971.

13. Forman G (2003) An extensive empirical study of feature selection metrics for text classification. J Mach Learn Res 1289-1305. » CrossRef » Google Scholar

14. Freitas AA, de Carvalho ACPLF (2007) A tutorial on hierarchical classification with applications in bioinformatics. Research and Trends in Data Mining Technologies and Applications 99:175-208. » CrossRef » Google Scholar

15. Guyon I, Elisseeff A (2003) An Introduction to Variable and Feature Selection. J Mach Learn Res 3: 1157-1182. » CrossRef » Google Scholar

16. Han J, Kamber M (2001) Data mining: concepts and techniques. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2001. » CrossRef » Google Scholar

17. Jensen R (2005) Combining rough and fuzzy sets for feature selection. Ph.D. dissertation, School of Information, Edinburgh University, 2005.» CrossRef » Google Scholar

18. John GH, Kohavi R, Pfleger K (1994) Irrelevant Features and the Subset Selection Problem. Proceedings of the 11th International Conference on Machine Learning ICML 94: 121-129.» CrossRef » Google Scholar

19. Kennedy J, Eberhart RC (2001) Swarm Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, 2001. » CrossRef » Google Scholar

20. Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97: 273-324. » CrossRef » Google Scholar

21. Latourrette M (2000) Toward an explanatory similarity measure for nearest-neighbor classification. In: ECML '00: Proceedings of the 11th European Conference on Machine Learning, London UK 238-245. » CrossRef » Google Scholar

22. Liu B, Abbass HA, McKay B (2004) Classification Rule Discovery with Ant Colony Optimization. IEEE Computational Intelligence Bulletin 3: 31-35.

23. Liu H, Motoda H (1998) Feature Selection for Knowledge Discovery and Data Mining. Boston: Kluwer Academic Publishers, 1998. ISBN 0-7923-8198-X.» CrossRef » Google Scholar

24. Maniezzo V, Colorni A (1999) The Ant System Applied to the Quadratic Assignment Problem. IEEE Trans Knowl Data Engin 11: pp769-778. » CrossRef » Google Scholar

25. Mitchell T (1996) Machine Learning. McCraw Hill, 1996.

26. Pappa GL, Baines AJ, Freitas AA (2005) Predicting post-synaptic activity in proteins with data mining. Bioinformatics 21: pp19-25.» CrossRef » Pubmed » Google Scholar

27. Pawlak Z (1991) Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishing, Dordrecht, 1991. » CrossRef » Google Scholar

28. Pudil P, Novovicova J, Kittler J (1994) Floating search methods in feature selection. Pattern Recognit Lett 15: pp1119-1125.» CrossRef » Google Scholar

29. Shi Y, Eberhart RC (1998) Parameter selection in particle swarm optimization. Proceedings of the 7th International Conference on Evolutionary Programming, 591-600. » CrossRef » Google Scholar

30. Siedlecki W, Sklansky J (1988) On Automatic Feature Selection. Intern J Pattern Recognit Artif Intell 2: pp197-220.

31. Wang X, Yang J, Teng X, Xia W, Jensen R (2007) Feature selection based on rough sets and particle swarm optimization. Pattern Recognit Lett 28: pp459-471. » CrossRef » Google Scholar

32. Witten IH, Frank E (2005) Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco, 2005.

33. Yang J, Honavar V (1998) Feature Subset Selection Using a Genetic Algorithm. IEEE Intell Syst 13: pp44-49. » CrossRef » Google Scholar