Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Feature subset selection in large dimensionality domains

Iffat A. Gheyas*, Leslie S. Smith

Department of Computing Science and Mathematics, University of Stirling, Stirling, FK9 4LA, Scotland, UK

ARTICLE INFO

Article history:
Received 30 January 2009
Received in revised form 31 May 2009
Accepted 17 June 2009

Keywords:
Curse of dimensionality
Feature subset selection
High dimensionality
Dimensionality reduction

ABSTRACT

Searching for an optimal feature subset from a high dimensional feature space is known to be an NP-complete problem. We present a hybrid algorithm, SAGA, for this task. SAGA combines the ability to avoid being trapped in a local minimum of simulated annealing with the very high rate of convergence of the crossover operator of genetic algorithms, the strong local search ability of greedy algorithms and the high computational efficiency of generalized regression neural networks. We compare the performance over time of SAGA and well-known algorithms on synthetic and real datasets. The results show that SAGA outperforms existing algorithms.

1. Introduction

The purpose of data mining is knowledge discovery: to generate new knowledge about events and phenomena from existing data sets for classification or forecasting future events. Data sets consist of a number of vectors, each corresponding to some occurrence of an event: each vector consists of a large number of features (or explanatory variables). In general, which features matter is not known. As a result, all sorts of information about events of interest are often gathered. Improvements in data acquisition capacity, falling costs of data storage, and development of database and data warehousing technology, have led to more and more high dimensional datasets (with tens or hundreds of thousands of features) emerging [1]. Many of these features are irrelevant or redundant. Unnecessary features increase the size of the search space and make generalization more difficult. This *curse of dimensionality* (each feature is a separate dimension) is a major obstacle in machine learning and data mining. Hence feature selection is an active area of research in pattern recognition [2], machine learning [3], data mining [4] and statistics [5]. In particular, the prediction performance of any learning algorithm depends on how efficiently the algorithm learns patterns in the data. Irrelevant and redundant features increase the search space size, making patterns more difficult to detect and making it more difficult to capture rules necessary for forecasting or classification, whether by machine or by hand. In addition, the more the features, the higher the risk of overfitting. The probability that some features will coincidentally fit the data increases, unless the sample size grows exponentially with the number of features. Furthermore, in most practical applications, we want to know the collection of core variables that are most critical in explaining an event.

Feature subset selection entails choosing the feature subset that maximizes the prediction or classification accuracy. The feature subset selection approach is based on the *principle of parsimony* (or Occam's razor) [6]. This says that, we prefer the model with the smallest possible number of parameters that adequately represents the data. Einstein is quoted in Parzen [7, p. 68] as remarking that "everything should be made as simple as possible, but not simpler". However, this principle is difficult to apply in feature selection problems. Selecting the best feature subset is proven to be an NP-complete problem [8]. The task is challenging because, first, features which do not appear relevant singly may become highly relevant when taken with others. There can be two-way, three-way or complex multi-way interactions among features. As a result a feature that is weakly associated with prediction (or classification) can improve prediction accuracy if it is complementary to other features. Second, relevant features may be redundant so that the omission of some of them will remove unnecessary complexity (and noise) from the forecasting problem. There can be many levels of multi-way redundancy in the feature space. Third, high feature correlation does not imply absence of feature complementarity. Fourth, high levels of multicollinearity increase the probability that a good predictor of the output signal will be found non-significant and rejected from the model.

An exhaustive search of all possible subsets of features will guarantee that the best subset of features is found. Unfortunately this is computationally impractical for even a medium sized database (for

Abbreviations: ACO, ant colony optimization; GA, genetic algorithm; GRNN, generalized regression neural networks; PSO, particle swarm optimization; SA, simulated annealing; SBS, sequential backward selection; SFBS, sequential floating backward selection; SFFS, sequential floating forward selection; SFS, sequential forward selection

* Corresponding author. Tel.: +44 01786 467430; fax: +44 01786 464551.
E-mail addresses: iag@cs.stir.ac.uk (I.A. Gheyas), lss@cs.stir.ac.uk (L.S. Smith).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.06.009

n features, the number of possible feature subsets is 2^n, too large to be evaluated even for modest n). A major thrust of current research work is focused on the determination of an optimal feature subset. The choice is a trade-off between computational time and quality of the generated feature subset solutions. In this paper, we present and test a novel hybrid algorithm for selection of optimal feature subsets. The proposed algorithm consistently generates better feature subsets compared to existing search algorithms within a predefined time limit and keeps improving the quality of selected subsets as the algorithm runs.

The rest of the paper is organized as follows: a brief review of previous work in Section 2, the new algorithm in Section 3, (with the procedure for estimating fitness of a feature subset in Section 3.1, an overview of generalized regression neural networks (GRNN) in Section 3.2 and details of the constituent search algorithms in Section 3.3), and comparative performance measurement in Section 4, results and discussions in Section 5, followed by summary and conclusions in Section 6.

2. Review of existing techniques

Two broad categories of optimal feature subset selection have been proposed: filter and wrapper. In filter approaches, features are scored and ranked based on certain statistical criteria and the features with highest ranking values are selected. Frequently used filter methods include t-test [9], chi-square test [10], Wilcoxon Mann–Whitney test [11], mutual information [12], Pearson correlation coefficients [13] and principal component analysis [14]. Filter methods are fast but lack robustness against interactions among features and feature redundancy. In addition, it is not clear how to determine the cut-off point for rankings to select only truly important features and exclude noise.

In the wrapper approach, feature selection is "wrapped" in a learning algorithm. The learning algorithm is applied to subsets of features and tested on a hold-out set, and prediction accuracy is used to determine the feature set quality. Generally, wrapper methods are more effective than filter methods. Since exhaustive search is not computationally feasible, wrapper methods must employ a search algorithm to search for an optimal subset of features. Wrapper methods can broadly be classified into two categories based on search strategy: (i) greedy and (ii) randomized/stochastic.

Greedy wrapper methods use less computer time than other wrapper approaches. Sequential backward selection (SBS) (also known as backward stepwise elimination) [15] and Sequential forward selection (SFS) (also known as forward stepwise selection) [16] are the two most commonly used wrapper methods that use a greedy hill-climbing search strategy. SBS starts with the set of all features and progressively eliminates the least promising ones. SBS stops if the performance of learning algorithms drops below a given threshold due to removal of any remaining features. SBS relies heavily on the monotonicity assumption [17]. This states that prediction accuracy never decreases as the number of features increases. This assumption is dubious because of the difficulties associated with search space dimensionality and overfitting. In reality, the predictive ability of a learning algorithm may decrease as the feature subspace dimensionality increases after a maximum point due to a decreasing number of samples for each feature combination. When faced with high-dimensional data, SBS often finds difficulties in identifying the separate effect of each explanatory variable on the target variable. Because of this, good predictors can be removed early on in the algorithm (in SBS, once a feature is removed, it is removed permanently). By contrast, SFS starts with an empty set of features and iteratively selects one feature at a time—starting with the most promising feature—until no improvement in classification accuracy can be achieved. In SFS, once a feature is added, it is never removed. SBS is robust to interaction problems but sensitive to multicollinearity. On the other hand, SFS is robust to multicollinearity problems but sensitive to feature interaction. As a result, both SBS and SFS can easily be trapped into local minima. The problem with SFS and SBS is their single-track search. Hence, Pudil et al. [18] suggest floating search methods (SFFS, SFBS) that performs greedy search with provision for backtracking. However, recent empirical studies demonstrate that sequential floating forward selection (SFFS) is not superior to SFS [19] and sequential floating backward selection (SFBS) is not feasible for feature sets of more than about 100 features [20]. The problem with sequentially adding or removing features is that the utility of an individual feature is often not apparent on its own, but only in combinations including just the right other features.

Stochastic algorithms developed for solving large scale combinatorial problems such as ant colony optimization (ACO), genetic algorithm (GA), particle swarm optimization (PSO) and simulated annealing (SA) are at the forefront of research in feature subset selection [17,21–23]. These algorithms efficiently capture feature redundancy and interaction and do not require the restrictive monotonicity assumption. However, these algorithms are computationally expensive (though far less so than exhaustive search).

Recently, several authors proposed hybrid approaches taking advantages of both filter and wrapper methods. Examples of hybrid algorithms include t-statistics and a GA [24], a correlation-based feature selection algorithm and a genetic algorithm [25], principal component analysis and an ACO algorithm [26], chi-square approach and a multi-objective optimization algorithm [27], mutual information and a GA [28,29]. The idea behind the hybrid method is that filter methods are first applied to select a feature pool and then the wrapper method is applied to find the optimal subset of features from the selected feature pool. This makes feature selection faster since the filter method rapidly reduces the effective number of features under consideration. Advocates of hybrid methods argue that the risk of eliminating good predictors by filter methods is minimized if the filter cut-off point for a ranked list of features is set low. However, hybrids of filter and wrapper methods may suffer in terms of accuracy because a relevant feature in isolation may appear no more discriminating than an irrelevant one in the presence of feature interactions.

Wrapper methods use a learning algorithm to assess the accuracy of potential subsets in predicting the target. Currently, the most popular learning algorithm used in wrapper schemes is the support vector machine (SVM) [30]. However, the accuracy of an SVM is dependent on the choice of kernel function and the parameters (e.g. cost parameter, slack variables, margin of the hyper plane, etc.). Failure to find the optimal parameters for an SVM model affects its prediction accuracy [31]. Another drawback of the SVM is its computational cost [32]. Wrapper methods are computationally more demanding than filter methods because they evaluate the candidate feature subsets using a learning algorithm, and these are usually iterative methods. This can increase the computational cost. To accelerate the wrapper approach in feature subset search, it is vital to employ a fast learning algorithm. Furthermore, empirical evidence suggests that SVMs are very sensitive to noisy training data, which can degrade their performance [33]. They are also prone to overfitting and poor generalization [34].

Development of a highly accurate and fast search algorithm for the selection of optimal feature subset is an open issue.

3. Proposed algorithm

A good search algorithm should provide: (1) good global search capability that allows for the exploration of new regions of the solution space without getting stuck in local minima, (2) rapid convergence to a near optimal solution, (3) good local search ability, and (4) high computational efficiency.

We present a hybrid algorithm (SAGA), named after two major underlying search algorithms (SA and GA), for selecting optimal feature subsets efficiently. This algorithm is based on a simulated annealing, a genetic algorithm, a generalized regression neural networks and a greedy search algorithm. SAGA combines the ability to avoid being trapped in a local minimum of SA with a very high rate of convergence of the crossover operator of GA, the strong local search ability of the greedy algorithm and high computational efficiency of GRNN. Our hybrid approach solves the feature selection problem without including filter steps. Hence, unlike existing hybrid algorithms, SAGA does not compromise accuracy for speed.

The SA algorithm here is a mutation-based search approach. Mutation represents a long jump in the search space. The strength of SA is good global search ability. The major disadvantage of SA is its slow convergence speed. On the other hand, GA implements both crossover and mutation operations. The strength of GA is its rapid convergence, but the combination of crossover and a low fixed mutation rate often traps the search in a local minimum. In addition, the local search capability of SA and GA is weak. By contrast, greedy algorithms have good local search ability, but lack global search ability.

SAGA organizes a search in three stages.

Stage 1: SAGA employs a SA to guide the global search in a solution space. As long as the temperature is very high, SA accepts every new solution, thus yielding a near random search through the search space. On the other hand, as the temperature becomes close to zero, only improvements are accepted. The SA is run for approximately 50% of the total time available.

Stage 2: SAGA employs a GA to perform optimization. The GA population was set at 100. The initial population consists of the best solutions detected by SA. The main purpose of crossover in GA is to exchange information between pairs of good solutions to form new (and hopefully better) solutions. The crossover operator thereby assists in rapid convergence to a good solution. The mutation operator in GA introduces new genes into the population and retains genetic diversity. The GA runs for about 30% of total time spent by SAGA to find the optimal feature subset solution.

Stage 3: SAGA applies a hill-climbing feature selection algorithm. The greedy algorithm performs a local search on the k-best solutions (elite) given by two global optimization algorithms (SA and GA) and selects the best neighbours (in our context neighbours are defined in terms of the Euclidean distance between a pair of feature subsets). The hill-climbing algorithm is run in the remaining execution time.

Computational efficiency is essential for exploring a huge search space. To enhance it, the following measures were taken. First, SAGA employs a robust and fast learning algorithm (GRNN) for assessing candidate solutions. GRNN, based on fuzzy means clustering, is a 'one-pass' algorithm. GRNN has just one parameter (smoothing factor) that needs to be chosen, but our empirical research reveals that the prediction accuracy of GRNN is not very sensitive to the parameter setting. Hence in GRNN we need not to develop and validate many predictive models. Furthermore, GRNN requires no training time other than the time required to pre-process and store the entire training set. Another major reason why we choose GRNN is that it suffers relatively more from the curse of dimensionality than other algorithms [35]. This is an advantage when trying to eliminate unnecessary variables because GRNN does not have the luxury of producing good results when there are irrelevant and redundant features. Other advantages of GRNN are its nonparametric nature and its robustness against local minima, overfitting and outliers [36–38]. Second, Cooper and Hinde (2003) report that evolutionary algorithms spend approximately a third of the time testing on already tested candidate solutions [39]. SAGA stores information about the candidate solutions evaluated so far in a database and never evaluates a possible solution more than once.

As a result, SAGA has all the four qualities mentioned above.

3.1. Computing fitness of feature subsets

Our proposed algorithm (SAGA) employs GRNN classifiers to evaluate candidate feature subset solutions. Before the evaluation of feature subsets, each feature was normalized by scaling it between 0 and 1. We perform 10-fold cross validation to estimate the testing accuracy of the GRNN classifier. The higher the accuracy, the fitter the solution. If the accuracies of two solutions are the same, then the solution using the smaller number of features wins.

3.2. Generalized regression neural networks learning algorithm

GRNN is an instance-based algorithm. In GRNN [40] each observation in the training set forms its own cluster. When a new input pattern x is presented to the GRNN for the prediction of the output value, each training pattern (prototype pattern) y_i assigns a membership value h_i to x based on the Euclidean distance d as in Eq. (2). The formula for the Euclidean distance between x and y_i is

$$d = d(x, y_i) = \sqrt{\sum_{j=1}^{n} (x_j - y_{ij})^2} \tag{1}$$

where $x = (x_1, \ldots, x_n)$ is the presented pattern and $y_i = (y_{i1}, \ldots, y_{in})$ is the i-th prototype pattern. n is the total number of features in the study. x_j is the value of the j-th feature of the presented pattern (features can be multivalued or not). y_{ij} is the value of the j-th feature of the i-th prototype pattern

$$h_i = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{d^2}{2\sigma^2}\right) \tag{2}$$

where σ is the smoothing function parameter (we specify a default value: $\sigma = 0.5$).

Finally, GRNN calculates the output value z of the pattern x as in Eq. (3). The predicted output is the weighted average of the outputs of all prototype patterns. GRNN can handle continuous output variables and categorical output variables with two categories: event of interest (coded as '1') or not (coded as '0'):

$$z = \frac{\sum_i (h_i \times \text{output of } y_i)}{\sum_i h_i} \tag{3}$$

If the output variable is binary, then GRNN calculates the probability of the event of interest. If the output variable is continuous, then it estimates the value of the variable.

3.3. Implementation of the underlying search algorithms of the SAGA

We encode possible feature subset solutions in ordered, fixed-length binary strings where '1' indicates the presence of the feature and '0' its absence. One of the objectives was realizing exactly how time consuming the feature selection task can be. Hence, a predefined time limit instead of the maximum number of total iterations was chosen as the stopping criterion which has inevitably made our algorithm rather complicated (in any practical application, one should therefore use a standard stochastic algorithm with imposing a maximum number of iterations as stopping criterion). A search algorithm spends almost 99% of its running time evaluating the fitness of solutions. The computation time required to evaluate a feature subset depends on the number of features present in the subset and the number of instances in the dataset. Hence, we empirically find out the time $t(=t(1:10{,}000))$ required to estimate the fitness scores of feature subsets with various dimensionalities from 1 to 10,000 using GRNN and store the information (dimensionalities of subsets and time required to assess their fitness) in database.

3.3.1. Pseudocode of simulated annealing

```
Step 1: Set the initial temperature ($T_i$): $T_i$ is the total run time for SA.
Step 2: Set the current temperature ($T_c$): $T_c = T_i$.
Step 3: Initialize population: Randomly select 100 individuals $I\,(=I(1:100))$ from the pool of individuals for initial population.
Step 4: Evaluate the fitness of each individual: Based on each individual $I$, extract a new dataset $D_{new}$ from the (normalized) original dataset $D$ with the features that are present in the solution of the individual. Evaluate the fitness scores $E_o\,(=E_o(1:100))$ of feature subsets using GRNN and store the information (feature subset solutions with fitness scores).
Step 5: Update the effective temperature ($T$): Based on the dimensionality of each individual evaluated in the previous step, retrieve the time elapsed in evaluating the individual. Calculate the total time spent $T_{spent}$ on evaluating individuals of population by adding the time spent for each individual. Finally update the effective temperature: $T_c = T_c - T_{spent}$.
Step 6: For all current feature subset vectors $I\,(=I(1:100))$ change the bits of vectors with probability $p_{mutation}$: $p_{mutation} = 0.5 - 0.5\exp(T_c/\lambda)$ where $\lambda = T_i/\log_2(0.5)$.
Step 7: Evaluate the fitness $E_n\,(=E_n(1:100))$ of the new candidate solutions if not already evaluated.
Step 8: Determine if this new solution is kept or rejected and update the database.
    • If $E_n \geq E_o$, the new solution is accepted. The new solution replaces the old solution and $E_o$ is set to $E_n$.
    • If $E_n < E_o$, calculate the Boltzmann acceptance probability $P_{accept}$. If the acceptance probability is greater than or equal to a random number between 0 and 1, the new solution is accepted and it replaces the old one and $E_o$. If the acceptance probability is less than the random number, the new solution is rejected and the old solution stays the same: $P_{accept} = \exp(-(E_o - E_n)/T_c)$.
Step 9: Update the effective temperature. If the effective temperature is greater than or equal to zero, return to step 6. Otherwise, the run is finished.
```

3.3.2. Pseudo code of GA (Genetic Algorithm)

```
Step 1: Construct a chromosome pool of size 100 with the 100 fittest chromosomes from the list of feature subset solutions evaluated so far by the SA.
Step 2: Select 50 pairs of chromosomes using rank-based selection strategy.
Step 3: Perform crossover between the chromosomes using the half uniform crossover scheme (HUX). In HUX, half of the non-matching parents' genes are swapped.
Step 4: Kill the parent solutions.
Step 5: Mutate offspring with probability 0.001.
Step 6: Evaluate the fitness of the offspring provided if it has not already been evaluated and if sufficient time is available. Update the database and estimate the time left.
Step 7: Go back to step 2 if the time is not up.
```

3.3.3. Pseudocode of hill-climbing algorithm

```
Step 1: Select the best-to-date solution.
Step 2: Create 10,000 new candidate solutions from the selected solution by changing only one bit (feature) at a time.
Step 3: Evaluate the new solutions if they are not evaluated before and update the database. Replace the previous solution by the new solution(s) if they are better than the previous solution.
Step 4: Go back to step 2 and perform the hill climbing on each of the accepted new solutions. Repeatedly apply the process from steps 2 to 3 on selected solutions as long as the process is successful in finding improved solutions in every repetition and as long as the time is available.
Step 5: Update the database and update the time available.
Step 6: Select the next best-to-date solution from the database and go back to step 2 if time is still available.
```

4. Comparative performance analysis

We compare our algorithms with the following benchmark algorithms: four commonly used greedy search algorithms (sequential backward selection [15], sequential forward selection [16], sequential floating forward selection [18], and sequential floating backward selection [18]) and four popular stochastic search algorithms (ant colony optimization [21], genetic algorithm [17], particle swarm optimization [22] and simulated annealing [23]). We also compare our algorithm against a hybrid of filter and wrapper approaches—filter-wrapper (FW). Many hybrid algorithms have been proposed for feature subset selection with encouraging results [24–29]. It was not possible to implement all the methods and empirically assess them. Instead, based on the experience of other authors, we develop a representative hybrid algorithm FW. This consists of a number of popular filter methods and a stochastic algorithm. FW is a three stage algorithm.

Stage 1: Since, it is hard to decide which filter method is best for a dataset because the performance of a filter method varies with different datasets [24], we use a number of popular filter methods to filter out irrelevant features. FW eliminates a feature when all of these filter methods—*t*-test, symmetric uncertainty [38], and Pearson's correlation coefficients—dismiss the feature as irrelevant at the 0.05 level.

Stage 2: FW uses PCA [14] to filter out redundant features.

Stage 3: FW uses SA [23] to find an optimal solution since our empirical results suggests that SA is better than other stochastic algorithms.

The proposed and benchmark algorithms were tested on 30 datasets (descriptions of datasets are provided in Section 4.2).

4.1. Test strategy for a standardized comparison of search algorithms

There are a number of strategies employed to ensure fair comparison of search algorithms.

- All algorithms were run on a 3.40 GHz Intel® Pentium® D CPU with 2 GB RAM.
- The values of each feature were normalized in a 0–1 range before the experiment.
- All algorithms use GRNN classifiers to evaluate each of the resulting subsets using 10-fold cross validation.
- No algorithm evaluates the same solution more than once.
- Each stochastic search algorithm (ACO, FW, GA, PSO, SA and SAGA) was run 10 times on each dataset, each time with different initial populations of 100 individuals. The final performance of each algorithm was calculated by averaging over all 10 simulations.
- Algorithms were ranked based on their performance. Their performance is measured in terms of classification accuracy with the best solution found during the entire run. Two different solutions having the same accuracy level are assessed in terms of the number of features present in the feature subset solutions. We assign rank 1 to the best algorithm and rank $m\,(m \leq 10)$ to the worst

algorithm. The Friedman test is used to test the null hypothesis that the performance is the same for all algorithms. After applying the Friedman test and noting that it is significant, a pairwise comparison test (comparison of groups or conditions with a control [41, p. 181]) was used in order to test the (null) hypothesis that there is no significant difference between any pair of the 10 algorithms.

4.2. Descriptions of datasets

We use 11 synthetic datasets, 18 real-world benchmark datasets and one new real-world dataset to perform experiments. All of these datasets are high dimensional.

4.2.1. Synthetic datasets

Feature interactions and feature redundancy are two major problems often encountered when reducing the dimensionality of feature space. The principal motivation behind generating synthetic datasets was to recreate these problems on large scale and perform experiments on controlled datasets.

Each dataset consists of 10,000 instances each of 10,000 features. Approximately one third of these features were completely irrelevant. Among these 10,000 features only 10 informative features were included in the model. One third of them were actually the exact copy of the set of these 10 relevant features. The remaining features are correlated to varying degrees with the relevant features. All features are continuous-valued. They are highly correlated and they interact with one another. The response variable is a binary variable. The following steps were taken to generate these datasets.

Step 1: Specify different mean vectors and different covariance matrices for all the features for the 11 different datasets. Since mean vectors and covariance matrices of no two datasets are the same, the joint distribution of features is different in each dataset.

Step 2: Generate 10,000 combinations of feature values for each dataset from its unique mean vector and covariance matrix.

Step 3: The probability of the event of interest for each instance was estimated by the following model (we specified different sets of model parameters for different datasets). Only 10 features among 10,000 features were included in the model. To simulate interactions between features, we included three interaction terms. Interaction terms are formed by the multiplication of two or more explanatory variables. We included one two-way interaction term ($-\beta_5 X_{66} X_{5789}$), one three-way interaction term ($-\beta_9 X_{420} X_{1103} X_{8652}$) and one multi-way interaction term ($+\beta_6 X_{420} X_{6166} X_{6999} X_{7200}$).

$$P(Y) = 1/(1 + \exp(-Z)) \tag{4}$$

$$\begin{aligned} Z = {} & \beta_0 + \beta_1 X_{66} + \beta_2 X_{1103} + \beta_3 X_{4447} + \beta_4 X_{5789} - \beta_5 X_{66} X_{5789} \\ & + \beta_6 X_{420} X_{6166} X_{6999} X_{7200} + \beta_7 X_{8652} + \beta_8 X_{9995} \\ & - \beta_9 X_{420} X_{1103} X_{8652} \end{aligned}$$

where $P(Y)$ is the probability of the event of interest; $(X_1, X_2, \ldots, X_{10,000})$ represent different features; $(\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9)$ are the model parameters.

We used Eq. (4) to generate all of the synthetic datasets. All the features in the model were arranged in the random order in all datasets. The differences between the datasets are mainly due to different combinations of feature values and different values of model parameters.

Step 4: Generate a uniformly distributed random number in the range (0, 1) for each observation. If the random number is greater than the probability of the event of interest, the value of the response variable is 1, otherwise 0.

4.2.2. Benchmark datasets (modified)

In addition to 11 synthetic datasets, we tested these algorithms on 18 benchmark datasets. Benchmark datasets were taken from UCI machine learning repository. The benchmark datasets are real-world datasets. The benchmark datasets on which the algorithms were tested are: (1) Adult dataset, (2) Annealing dataset, (3) Breast Cancer Wisconsin (Diagnostic) dataset, (4) Breast Cancer Wisconsin (Prognostic) dataset, (5) Chess—King-Rook vs. King-Pawn, (6) Congressional Voting Records dataset, (7) Dermatology—Psoriasis, (8) Dermatology—Seboreic Dermatitis, (9) Dermatology—Lichen Planus, (10) Dermatology—Pityriasis Rosea, (11) Dermatology—Cronic Dermatitis, (12) Dermatitis—Pityriasis Rubra, (13) Hepatitis, (14) Mushroom, (15) Spambase, (16) Wine, (17) Yeast, and (18) Zoo. The descriptions of the original benchmark datasets are available in [42]. These datasets contain varying number of features and instances, but have fewer than 10,000 features. Hence, we add a series of randomly generated features to each dataset to make a total of 10,000 features. We added completely irrelevant features because we did not want to destroy the original properties of the benchmark datasets. We did not change the number of observations of the benchmark datasets.

4.2.3. New real-world dataset (smoking dataset)

We received a three stage cross sectional survey data on the smoking habits of teenagers from the Centre for Tobacco Control Research at the University of Stirling and Open University. The data were collected from Scotland, England, Northern Ireland and Wales in three survey stages: stage 1 in 1999, stage 2 in 2002 and stage 3 in 2004. The response variable is a binary variable ($1 = $ smoker, $0 = $ non-smoker). Explanatory variables include socio-demographic characteristics of respondents, their knowledge and attitudes towards tobacco promotion of all sorts and their smoking knowledge, attitudes and behaviour. This smoking dataset contains 285 features, 3321 instances but has a large number of missing values. Among the respondents, an overall proportion of 11% (355 respondents) are smokers. We applied multiple imputation with MCMC (Monte Carlo Markov Chain) algorithm to replace missing values [43]. We did not add artificial features to this dataset.

5. Results and discussion

We compare our proposed algorithm (SAGA) with the conventional search algorithms (ACO, FW, GA, PSO, SA, SBS, SFBS, SFFS and SFS) on 30 high dimensional datasets. The algorithms were

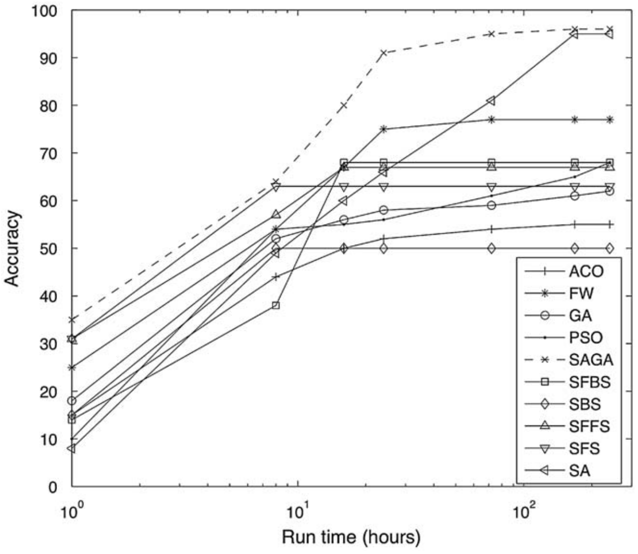

Fig. 1. Performance of the different algorithms (accuracy only).

Table 1
Summary results.

Search method	Time (h)	Number of significantly outperformed algorithms	Accuracy (%)	Features	Search method	Time (h)	Number of outperformed algorithms	Accuracy (%)	Features
ACO	1	2	15 (8)	4241 (1231)	SAGA	1	8	35 (182)	71 (26)
	8	1	44 (17)	2375 (1296)		8	8	64 (12)	27 (24)
	16	2	50 (19)	1595 (4388)		16	9	80 (10)	22 (14)
	24	1	52 (19)	836 (546)		24	9	91 (6)	10 (6)
	72	1	54 (18)	599 (808)		72	9	95 (4)	17 (9)
	168	1	55 (18)	524 (1792)		168	9	96 (3)	17 (7)
	240	1	55 (18)	442 (617)		240	9	96 (3)	12 (8)
FW	1	7	25 (129)	5797 (2487)	SBS	1	2	15 (79)	8750 (87)
	8	5	54 (14)	436 (283)		8	2	50 (5)	35 (38)
	16	5	67 (11)	218 (129)		16	0	(converged)	(converged)
	24	7	75 (12)	169 (134)		24	0		
	72	7	77 (10)	256 (249)		72	0		
	168	7	77 (10)	226 (107)		168	0		
	240	7	77 (10)	200 (91)		240	0		
GA	1	4	18 (93)	6984 (18)	SFBS	1	2	14 (75)	9040 (96)
	8	2	52 (15)	553 (409)		8	0	38 (8)	57 (29)
	16	1	56 (14)	446 (379)		16	6	68 (6)	45 (27)
	24	1	58 (14)	553 (2119)		24	5	(converged)	(converged)
	72	1	59 (13)	513 (204)		72	5		
	168	2	61 (11)	388 (320)		168	4		
	240	2	62 (10)	482 (206)		240	4		
PSO	1	0	10 (53)	5598 (3046)	SFFS	1	2	31 (161)	1 (0.4)
	8	3	54 (11)	852 (1230)		8	5	57 (9)	41 (10)
	16	2	55 (11)	242 (166)		16	6	67 (5)	34 (11)
	24	1	56 (11)	222 (305)		24	5	(converged)	(converged)
	72	2	61 (7)	240 (87)		72	5		
	168	3	65 (5)	307 (100)		168	4		
	240	4	68 (4)	181 (76)		240	4		
SA	1	0	8 (42)	6647 (3451)	SFS	1	8	31 (162)	3 (2)
	8	1	49 (12)	708 (745)		8	8	63 (11)	37 (42)
	16	4	60 (10)	290 (137)		16	5	(converged)	(converged)
	24	4	66 (11)	142 (208)		24	4		
	72	7	81 (6)	229 (153)		72	2		
	168	8	95 (3)	323 (187)		168	2		
	240	8	95 (3)	157 (69)		240	2		

The values in parenthesis represents the standard deviation. The lower the standard deviation, the more consistent the algorithm.

evaluated based on the fitness of the best feature subset solutions generated by the algorithms within the allowed time limits. The Friedman test reveals significant differences ($p < 0.05$) in the performance of the 10 search algorithms at all time limits. Table A1 shows statistical test results for pairwise comparisons of algorithms. We assign rank 1 to the best algorithm, rank 2 to the next best algorithm and so on.

Fig. 1 displays the results for the mean accuracy of the reduced datasets at run times of 1, 8, 16, 24, 72, 168 and 240 h: SAGA outperforms the others at all run times, though not always significantly. Table 1 summarizes the results in more detail, providing the mean accuracy, average number of features selected over all 30 datasets both with standard deviations. The average accuracy represents the overall performance. However, if two solutions are equally accurate (as is almost the case with SAGA and SA at 168 and 240 h), then the one with fewer features is fitter. We note that SAGA achieves the same accuracy with far fewer features than SA at 168 and 240 h. Table 1 also reports the relative performance of search algorithms in terms of the number of algorithms that are significantly worse than the control (based on pairwise comparison tests).

The key findings of this work are:

- SAGA holds the first position at all seven durations of running where the best performance is significantly better than the next best at the significance level of 0.05.
- The rate of improvement in search algorithms decreases as the time passes. However, as the running time increases, the improvement in the performance of SAGA declines slowly, relative to the others, with the exception of SA. Extensive experiments illustrate that after 8 h of searching; SAGA had a 64% mean accuracy with a 12% standard deviation. The mean accuracy rate of SAGA improved by 32% (from 64% to 96%) over the last 232 h of running for which the standard deviation was reduced by about 9% (from 12% to 3%), while the overall accuracy rate of SA increased from 49% to 95% (46%) and the standard deviation dropped from 12% to 3% (a 9% drop). The other algorithms had significantly lower rates of increase in accuracy with increasing running time. The accuracy improvements of other algorithms over the last 232 h of running are as follows: using the ACO the mean accuracy increased by about 10% and the standard deviation increased by about 1%, using the

FW the accuracy increased by about 23% while the standard deviation dropped by about 4%, using the GA the accuracy increased by about 10% while the standard deviation dropped by about 5%, and using the PSO the accuracy increased by about 14% while the standard deviation dropped by about 7%.

- Throughout the running period, SAGA selected a much smaller number of features than other algorithms. For this reason, the performance of SAGA remained significantly better than SA even when the accuracy rates of both algorithms were almost the same (at 168 and 240 h).

In addition, we note that if the search space is too large while the time available to conduct a search through the search space is very brief, SFS is a good choice among other conventional search algorithms. One and 8 h later, SFS finished joint first with SAGA, outperforming eight algorithms.

Further, if sufficient time is available, then SA should be considered among the conventional search algorithms. After the first 1 h of running, the performance of SA was the worst among all search algorithms. After 8 h, SA outperformed two algorithms. After 24 h, it outperformed four algorithms. After 72 h of searching, it became the second best algorithm, outperforming seven algorithms. After 168 h, it outperformed eight algorithms.

We also noted that when a greedy algorithm reaches the local minima it cannot climb ‘out’. Both SBS and SFS rapidly converged within approximately 8 h. SFFS and SFBS offer only slightly more resistance to local minima. They converged within 16 h. We found an interesting pattern in the relative performance of the algorithms (in terms of the number of significantly outperformed algorithms) over time. The relative performance of ACO, GA, SBS and SFS deteriorates, while the relative performance of PSO and SA improves as the time elapses. The relative performance of SFBS and SFFS roughly follows the Gaussian distribution over time. In addition, FW offers the most consistency (after SAGA) over time in terms of relative performance. Although FW applies a SA, it did not generate a dramatic performance improvement over time, like SA did. We suspect that a number of key features were already removed from the feature pool by filter methods, before SA began its search.

6. Summary and conclusions

We have presented a hybrid algorithm SAGA that combines the strengths of a number of existing algorithms to select the optimal feature subsets from a large feature space. SAGA is a hybrid of a number of wrapper methods—a SA, a GA, a GRNN and a greedy search algorithm. We compare our proposed algorithm against the following benchmark algorithms: ACO, FW, GA, PSO, SA, SBS, SFBS, SFFS and SFS on both synthetic and real-world datasets. Among these datasets, one dataset has 285 features and the remaining 29 datasets have 10,000 features each. We study the performance of these algorithms at different time intervals: after 8, 16, 24, 72, 168 and 240 h of running. The performance of our algorithm is highly encouraging. SAGA shows the best performance over every interval. We conclude that no existing algorithm is entirely satisfactory in isolation, but that a carefully designed combination can overcome the weaknesses of each.

Acknowledgements

We would like to thank our anonymous reviewers for their insight and suggestions for improvement.

Appendix A

Table A1 shows statistical test results for pairwise comparisons of algorithms.

Table A1
Pairwise comparisons between search algorithms.

Rank	Algorithm (s)	Significantly outperformed algorithms
After 1 h		
1	SAGA, SFS	(1) FW, (2) GA, (3) SFFS, (4) ACO, (5) SBS, (6) SFBS, (7) PSO, (8) SA
2	FW	(1) GA, (2) SFFS, (3) ACO, (4) SBS, (5) SFBS, (6) PSO, (7) SA
3	GA	(1) SBS, (2) SFBS, (3) PSO, (4) SA
4	SFFS, ACO, SBS, SFBS	(1) PSO, (2) SA
5	PSO, SA	–
After 8 h		
1	SAGA, SFS	(1) SFFS, (2) FW, (3) PSO, (4) GA, (5) SBS, (6) SA, (7) ACO, (8) SFBS
2	SFFS, FW	(1) GA, (2) SBS, (3) SA, (4) ACO, (5) SFBS
3	PSO	(1) SA, (2) ACO, (3) SFBS
4	GA, SBS	(1) ACO, (2) SFBS
5	SA, ACO	(1) SFBS
6	SFBS	–
After 16 h		
1	SAGA	(1) SFBS, (2) SFFS, (3) FW, (4) SFS, (5) SA, (6) PSO, (7) ACO, (8) GA, (9) SBS
2	SFBS, SFFS	(1) SFS, (2) SA, (3) PSO, (4) ACO, (5) GA, (6) SBS
3	FW	(1) SA, (2) PSO, (3) ACO, (4) GA, (5) SBS
4	SFS, SA	(1) PSO, (2) ACO, (3) GA, (4) SBS
5	PSO	(1) GA, (2) SBS
6	ACO, GA	(1) SBS
7	SBS	

Table A1 (*Continued*)

Rank	Algorithm (s)	Significantly outperformed algorithms
After 24 h		
1	SAGA	(1) FW, (2) SFBS, (3) SFFS, (4) SA, (5) SFS, (6) GA, (7) PSO, (8) ACO, (9) SBS
2	FW	(1) SFFS, (2) SA, (3) SFS, (4) GA, (5) PSO, (6) ACO, (7) SBS
3	SFBS, SFFS	(1) SFS,(2) GA, (3) PSO, (4) ACO, (5) SBS
4	SA, SFS	(1) GA, (2) PSO, (3) ACO, (4) SBS
5	GA, PSO, ACO	(1) SBS
6	SBS	–
After 72 h		
1	SAGA	(1) SA, (2) FW, (3) SFBS,(4) SFFS, (5) SFS, (6) PSO, (7) GA, (8) ACO, (9) SBS
2	SA, FW	(1) SFBS, (2) SFFS, (3) SFS, (4) PSO, (5) GA, (6) ACO, (7) SBS
3	SFBS, SFFS	(1) SFS, (2) PSO, (3) GA, (4) ACO, (5) SBS
4	SFS, PSO	(1) ACO, (2) SBS
5	GA, ACO	(1) SBS
6	SBS	–
After 168 h		
1	SAGA	(1) SA, (2) FW, (3) SFBS, (4) SFFS, (5) PSO, (6) SFS, (7) GA, (8) ACO, (9) SBS
2	SA	(1) FW, (2) SFBS, (3) SFFS, (4) PSO, (5) SFS, (6) GA, (7) ACO, (8) SBS
3	FW	(1) SFBS, (2) SFFS, (3) PSO, (4) SFS, (5) GA, (6) ACO, (7) SBS
4	SFBS, SFFS	(1) SFS, (2) GA, (3) ACO, (4) SBS
5	PSO	(1) GA, (2) ACO, (3) SBS
6	SFS, GA	(1) ACO, (2) SBS
7	ACO	(1) SBS
8	SBS	–
After 240 h		
1	SAGA	(1) SA, (2) FW, (3) PSO, (4) SFBS, (5) SFFS, (6) SFS, (7) GA, (8) ACO, (9) SBS
2	SA	(1) FW, (2) PSO, (3) SFBS, (4) SFFS, (5) SFS, (6) GA, (7) ACO, (8) SBS,
3	FW	(1) PSO, (2) SFBS, (3) SFFS, (4) SFS, (5) GA, ((6) ACO, (7) SBS
4	PSO, SFBS, SFFS	(1) SFS, (2) GA, (3) ACO, (4) SBS
5	SFS, GA	(1) ACO, (2) SBS
6	ACO	(1) SBS
7	SBS	–

References

[1] I. Guyan, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157–1182.

[2] P. Mitra, C.A. Murthy, S.K. Pal, Unsupervised feature selection using feature similarity, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 301–312.

[3] M. Robnik-Sikonja, I. Kononenko, Theoretical and empirical analysis of Relief and ReliefF, Machine Learning 53 (2003) 23–69.

[4] M. Dash, K. Choi, P. Scheuermann, H. Liu, Feature selection for clustering—a filter solution, in: Proceedings of the Second IEEE International Conference on Data Mining (ICDM'02), IEEE Computer Society Washington, DC, USA, 2002, pp. 115–122.

[5] T. Hastie, R. Tibshirani, J.H. Friedman, The Elements of Statistical Learning, second ed., Springer, Berlin, 2001.

[6] D.A. Bell, H. Wang, A formalism for relevance and its application in feature subset selection, Machine Learning 41 (2004) 175–195.

[7] E. Parzen, ARARMA models for time series analysis and forecasting, Journal of Forecasting 1 (1982) 67–87.

[8] A.A. Albrecht, Stochastic local search for the feature set problem, with applications to microarray data, Applied Mathematics and Computation 183 (2006) 1148–1164.

[9] J. Hua, W. Tembe, E.R. Dougherty, Feature selection in the classification of high-dimension data, in: IEEE International Workshop on Genomic Signal Processing and Statistics, 2008, pp. 1–2.

[10] X. Jin, A. Xu, R. Bie, P. Guo, Machine learning techniques and chi-square feature selection for cancer classification using SAGE gene expression profiles, Lecture Notes in Computer Science 3916 (2006) 106–115.

[11] C. Liao, S. Li, Z. Luo, Gene selection using Wilcoxon rank sum test and support vector machine for cancer, Lecture Notes in Computer Science 4456 (2007) 57–66.

[12] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 1226–1238.

[13] J. Biesiada, W. Duch, Feature selection for high-dimensional data—a Pearson redundancy based filter, Advances in Soft Computing 45 (2008) 242–249.

[14] L. Rocchi, L. Chiari, A. Cappello, Feature selection of stabilometric parameters based on principal component analysis, Medical and Biological Engineering and Computing 42 (2004) 71–79.

[15] S.F. Cotter, K. Kreutz-Delgado, B.D. Rao, Backward sequential elimination for sparse vector selection, Signal Processing 81 (2001) 1849–1864.

[16] S. Colak, C. Isik, Feature subset selection for blood pressure classification using orthogonal forward selection, in: Proceedings of 2003 IEEE 29th Annual Bioengineering Conference, 22–23 March 2003, pp. 122–123.

[17] J. Yang, V. Honavar, Feature subset selection using a genetic algorithm, IEEE Intelligent Systems and their Applications 13 (1998) 44–49.

[18] P. Pudil, J. Novovicov, J. Kittler, Floating search methods in feature selection, Pattern Recognition Letters 15 (11) (1994) 1119–1125.

[19] M. Bensch, M. Schroder, M. Bogdan, W. Rosenstiel, P. Czerner, R. Montino, G. Soberger, P. Linke, R. Schmidt, Feature selection for high-dimensional industrial data ESANN 2005, Brugge, 27–29 April 2005.

[20] H.T. Ng, W.B. Goh, K.L. Low, Feature selection, perceptron learning, and a susability case study for text categorization, in: 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Philadelphia, 27–31 July 1997, pp. 67–73.

[21] S.M. Vieira, M.C. Sousa, T.A. Runkler, Ant colony optimization applied to feature selection in fuzzy classifiers, Lecture notes in computer science 4529 (2007) 778–788.

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

[23] M. Ronen, Z. Jacob, Using simulated annealing to optimize feature selection problem in marketing applications, European Journal of Operational Research 171 (2006) 842–858.

[24] F. Tan, X. Fu, H. Wang, Y. Zhang, A. Bourgeois, A hybrid feature selection approach for microarray gene expression data, Lecture Notes in Computer Science 3992 (2006) 678–685.

[25] K.M. Shazzad, J.S. Park, Optimization of intrusion detection through fast hybrid feature selection, in: Proceedings of the Sixth International Conference on Parallel and Distributed Computing, IEEE Computer Society, Washington, DC, USA, 2005, pp. 264–267.

[26] Z. Yan, C. Yuan, Ant colony optimization for feature selection in face recognition, Lecture notes in Computer Science 3072 (2004) 221–226.

[27] K.M. Osei-Bryson, K. Giles, B. Kositanurit, Exploration of a hybrid feature selection algorithm, Journal of the Operational Research Society 54 (2003) 790–797.

[28] M. Fatourechi, G. Birch, R.K. Ward, Application of a hybrid wavelet feature selection method in the design of a self-paced brain interface system, Journal of Neuroengineering and Rehabilitation 4 (2007).

[29] J. Huang, Y. Cai, X. Xu, A wrapper for feature selection based on mutual information, in: 18th International Conference on Pattern Recognition, vol. 2, 2006, pp. 618–621.

[30] K.Z. Mao, Feature subset selection for support vector machines through discriminative pruning analysis, IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics 34 (2004) 60–67.

[31] C. Campbell, N. Cristianini, Simple learning algorithms for training support vector machines, CiteSeerXbeta, 1998.

[32] C. Zhang, P. Li, A. Rajendran, Y. Deng, D. Chen, Parallelization of multicategory support vector machines (PMC-SVM) for classifying microarray data, BMC Bioinformatics 7 (2006).

[33] Z. Gao, G. Lu, M. Liu, M. Cui, A novel risk assessment system for port state control inspection, in: IEEE International Conference on Intelligence and Security Informatics, 17–20 June 2008, pp. 242–244.

[34] L. Bo, L. Wang, L. Jiao, Training hard margin support vector machines using greedy stepwise algorithm, Lecture Notes in Computer Science 3518 (2005) 632–638.

[35] D. Tormandl, A. Schober, A modified general regression neural network (MGRNN) with new, efficient training algorithm as a robust 'black box'-tool for data analysis, Neural Networks 14 (2001) 1023–1034.

[36] N. Currit, Inductive regression: overcoming OLS limitations with the general regression neural network, Computers, Environment and Urban Systems 26 (2002) 335–353.

[37] I. Bialobrzewski, Neural modelling of relative air humidity, Computers and Electronic in Agriculture 60 (2008) 1–7.

[38] O. Yagci, D.E. Mercan, H.K. Cigizoglu, M.S. Kabdasli, Artificial intelligence methods in breakwater damage ratio estimation, Ocean Engineering 32 (2005) 2016–2088.

[39] J. Cooper, C. Hinde, Improving genetic algorithms' efficiency using intelligent fitness functions, Lecture Notes in Computer Science 2718 (2003) 1–58.

[40] D.F. Specht, A general regression neural network, IEEE Transactions on Neural Networks 20 (1991) 568–576.

[41] S. Singel, N.J. Castellan Jr., Nonparametric Statistics: for the Behavioural Sciences, McGraw-Hill, New York, 1988.

[42] UCI Irvine Machine Learning Repository, available online: ⟨http://archive.ics.uci.edu/ml/⟩.

[43] H.T. Ng, W.B. Goh, K.L. Low, Feature selection, perceptron learning, and a susability case study for text categorization, in: 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Philadelphia, 27–31 July 1997, pp. 67–73.

About the Author—IFFAT A. GHEYAS is a Ph.D. student in the Department of Computing Science and Mathematics at the University of Stirling. Her primary research interest is artificial intelligence. Currently, she is working on machine learning algorithms for data mining tasks.

About the Author—LESLIE S. SMITH (B.Sc. (Glasgow) 1973, Ph.D. (Glasgow) 1981) is a Professor of Computing Science at Stirling University. He is particularly interested in early auditory processing, neuroinformatics, neuromorphic systems. He is SMIEEE and a member of Acoustical Society of America and the Society of Neuroscience.