Tournament Searching Method to Feature Selection Problem

Grzegorz Dudek

Department of Electrical Engineering, Czestochowa University of Technology,
Al. Armii Krajowej 17, 42-200 Czestochowa, Poland
dudek@el.pcz.czest.pl

Abstract. A new search method to the feature selection problem – the tournament searching – is proposed and compared with other popular feature selection methods. The tournament feature selection method is a simple stochastic searching method with only one parameter controlling the global-local searching properties of the algorithm. It is less complicated and easier to use than other stochastic methods, e.g. the simulated annealing or genetic algorithm. The algorithm was tested on several tasks of the feature selection in the supervised learning. For comparison the simulated annealing, genetic algorithm, random search and two deterministic methods were tested as well. The experiments showed the best results for the tournament feature selection method in relation to other tested methods.

Keywords: feature selection, tournament feature selection, tournament searching, stochastic combinatorial optimization.

1 Introduction

The feature selection (FS) is an essential problem in the modelling of objects, processes and phenomenon, explored especially in pattern recognition, machine learning, data mining and artificial intelligence. The aim of FS is to reduce the dimension of the input vector by the feature (variable) subset selection which describes object in the best manner and ensures the best quality of the learning model. In this process the irrelevant, redundant and unpredictive features are omitted. For example, only the features, which contain the most information about the membership of the objects to the classes and which allow to minimize the risk of misclassification, are selected in pattern recognition. The other way of reduction in dimension of modelling problem relies on an application of the linear or nonlinear transformation, mapping the space of original features on the space with smaller dimension (the principal component analysis is the popular example). A creation of the new features on the base of the original features is called the feature extraction.

FS is applied due to several reasons [1], [2], [3]:

1. Model simplification. Consideration of the large number of features increases complexity of the model, what does not go with improvement in its quality. Some features can be redundant (e.g. the correlated features) or irrelevant (features not

carrying information about model output). The elimination of these features simplifies the model, improves its comprehensibility, stays in contrary to curse of dimensionality and clarifies the relationships between features.
2. Improvement in model quality. A removal of the unpredictive features does not often worsen the model quality, but improves it. For example a removal of features with random values, interferenced by huge noise, usually allows to improve the model operation.
3. Improvement in generalization. The simpler models with smaller number of parameters have greater generalization abilities (elimination of the noisy features reduces the model overfitting).
4. Reduction of computation time. The model created with the use of lower number of the features learns and works faster.
5. Saving of computer memory is proportional to the number of eliminated features.
6. Reduction in cost of data collection, communication, maintenance and technical realization of the model.
7. Data visualization when the number of selected features is low (1, 2 or 3).

The methods of the feature selection can be generally divided into filter and wrapper ones [4]. Filter methods do not require application of learning model to select relevant features. They select features as a preprocessing step, independent on the choice of the predictor. They also use information included in the dataset, e.g. the correlation between variables and discriminatory abilities of the individual features, to create the most promising feature subset before commencement of learning. The main disadvantage of the filter approach is the fact that it totally ignores the effect of the selected feature subset on the performance of the learning model [4]. Many approaches to the feature relevance measurements are presented in [5].

The wrapper approach operates in the context of the learning model – it uses feature selection algorithm as a wrapper around the learning algorithm and has usually better predictive accuracy than the filter approach. The wrapper approach using the learning model as a black box is remarkably universal. However, this approach can be very slow because the learning algorithm is called repeatedly. The comparative experiments between the wrapper and filter models confirmed that it is inappropriate to evaluate the usefulness of an input variable without taking into consideration the algorithms that built the classification or regression model [4]. The filter approach can be used as a preprocessing step for the wrapper approach in the analysis of huge datasets.

Some learning models have internal build-in mechanisms of the FS. For example decision trees (CART, ID3, C4.5) which incorporate FS routine as a subroutine and heuristically search the space of feature subsets along tree structure during the learning process. The approaches in which FS is an essential part of the training process are so-called the embedded methods.

The described above FS methods attempt to select the most relevant feature subset. Another approach is to apply a weighting function to features and in effect to attribute degrees of relevance to them. The well-known feature weighting method is the perceptron updating rule, which adds or subtracts weights on a linear threshold unit in response to training errors.

Many approaches were used to the FS problem: branch and bound algorithm [6], simulated annealing [7], genetic algorithms [7], [8], rough sets [9], adaptive agents [10], tabu search method [11], support vector machines [12] and ensembles [13]. The popular suboptimal fast strategies are the sequential forward selection and sequential backward selection algorithms [14], [3] based on simple greedy deterministic heuristics. The expansion of these strategies is plus l-take away r method [15] and floating search method [16]. The review of the FS subject matter can be found in [1], [2], [4], [17] and [18].

A goal of this work is to propose a simple, easy to use and control wrapper method of FS.

2 Tournament Feature Selection Method

In the FS problem the decision variables are binary and denote whether the feature is selected (1) or not (0). The solution is a binary vector composed of bits representing all m features: $\mathbf{x} = [x_1, x_2, \dots, x_m]$.

The tournament method to feature selection (TFS) consists of exploring the solution space starting from a randomly selected solution and generating the new ones by perturbing it. When the set of new l candidate solutions $\Omega = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_l\}$ is generated (l is called the tournament size), their costs (e.g. classification errors) are calculated using learning model. The best candidate solution (the tournament winner), with the lowest value of the cost function $C(\mathbf{x})$, is selected and it replaces the parent solution, even in case it is worse than the parent solution.

TFS searches the solution space by exploring neighborhood of the current solution by means of the move operator. The neighborhood is defined as $\Phi = \{\mathbf{x} \mid d_H(\mathbf{x}, \mathbf{x}^*) = 1\}$, where $\mathbf{x}^*$ is the parent solution and d_H is the Hamming distance. The move operator produces a candidate solution by switching the value of the randomly chosen bit (different for each candidate solution) of the parent solution.

If the tournament size is equal to 1, this procedure comes down to the random search method. On the other hand, when $l = m$ this method becomes the hill climbing method. $l = 1, 2, \dots, m$ is the only parameter here. Its value decides about the exploration/exploitation properties of the algorithm, and can be fixed or can increase with the iteration number.

The TFS algorithm can be summarized in the following steps:

1. Generation of the initial solution (first parent solution) by random.
2. Generation of the set of l candidate solutions from the parent solution using the move operator.
3. Evaluation of the candidate solutions using learning model.
4. Selection of the best solution among the candidate solutions.
5. Replacement of the parent solution by the tournament winner.
6. Repeat steps 2-6 until the stop criterion is reached.

The flowchart of the TFS is shown in Fig. 1, where the flowcharts of popular stochastic FS methods, the simulated annealing (SA) and genetic algorithm (GA), are shown as well. The GA and TFS have parallel structure, the solution space is searched by a population of points. In SA solutions are generated one by one.

The evaluation procedures and move or mutation operators are similar in these algorithms, but the selection procedures are different. In the SA method the new solution **x** is either accepted or rejected according to an acceptance probability based on the Boltzmann distribution:

$$p(\mathbf{x}) = \min\left[1, \exp\left(\frac{-\Delta C}{T}\right)\right], \tag{1}$$

where T is a global time-varying parameter called the temperature, $\Delta C = C(\mathbf{x}) - C(\mathbf{x}^*)$.

The cooling schedule specifies an initial value of the temperature T_0, a temperature update function $T_{k+1} = f(T_k)$ and the inner-loop and outer-loop criterions (not shown in Fig. 1). The temperature parameter T controls the probability of acceptance. Initially the high value of temperature is selected. Then it decreases slowly according to search advance in order to provoke the convergence of the algorithm to the global optimum. In the pioneering work [19] the temperature was reduced according to the geometric cooling schedule:

$$T_{k+1} = \alpha T_k, \tag{2}$$

where $0<\alpha<1$ is a constant factor.

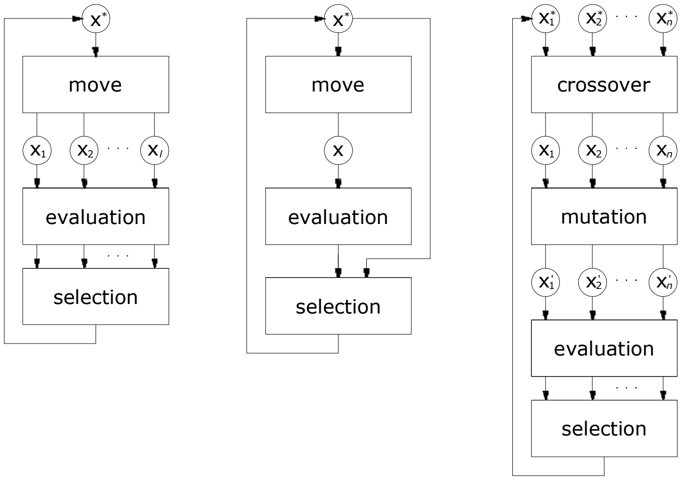

Fig. 1. Simple flowcharts of the tournament feature selection method (left), simulated annealing (middle) and genetic algorithm (right)

The GA is the more complicated method than TFS and SA. It requires many parameters to be set, i.e.: population size, probability of crossover and mutation, selection procedure parameters.

There are many methods of selection in GA, e.g.: roulette wheel selection, tournament selection, rank selection and some others. The main idea of these methods is to select the relatively good solutions from the population – the better the solution is, the more chances to be selected it has. The tournament selection procedure is similar to that used in TFS, but instead of one, many tournaments of the size l are performed. The winners form the new population. Parameter l controls the selective pressure.

All these FS methods have some built-in mechanism of escaping from the local minima/maxima. In TFS method this mechanism is very simple – the parent solution is replaced by the tournament winner, even if it represents a worse solution.

3 Experimental Comparison of Search Algorithms

The TFS method was verified on several test problems of data classification. Benchmark datasets, described in Table 1, were taken from the UCI repository [20]. The features in datasets were standardized.

Table 1. Description of data used in experiments

Dataset	Size	Features	Classes	Optimal k value
Ionosphere	351	34	2	3
Cancer	569	30	2	4
Heart	270	13	2	7
Wine	178	13	3	4
Glass	214	9	6	5
Diabetes	768	8	2	14

where: Cancer – the Wisconsin diagnostic breast cancer dataset; Heart – the Statlog heart dataset.

K-nearest neighbor method (k-NN) was used as a classifier, with k determined a priori for all features (optimal k values are shown in Table 1). The classification accuracy was determined in the leave-one-out procedure. For comparison, sequential forward selection (SFS) and sequential backward selection (SBS) algorithms [3], [14] were used to FS as well as the genetic algorithm, simulated annealing and random search method. For each dataset the feature space was optimized running algorithms 30-times and starting from the same initial solution (except for GA which operates on a population of solutions). The number of solutions generated in the searching process was the same for all algorithms and equal $40 \cdot \text{round}(m/2)^2$. The set of tested algorithms and the values of their parameters, which were adjusted experimentally, are listed below.

- TFS: $l = \text{round}(m/3)$.
- SA schedule proposed by Kirkpatrick et al. in [19]. The initial temperature was calculated iteratively from (1), where $p_0 = 0.99$ and ΔC was the average of the absolute value of the cost variation obtained for an initial sequence of random transitions. The value of temperature reduction coefficient α was determined in such a way that the acceptance probability of the trial solution, which is 0.01% worse from the parent solution, is equal 0.1, when the annealing process is advanced in 80%:

$$\alpha = \left(\frac{0.0043}{T_0}\right)^{\frac{1}{0.8L_{out}-1}}, \qquad (3)$$

where L_{out} is the number of the outer-loop iterations, $L_{out} = 20\cdot\text{round}(m/2)$. The number of inner-loop iterations $L_{inn} = 2\cdot\text{round}(m/2)$.

- GA: selection method – binary tournament, crossover method – uniform crossover with 0.5 probability of swapping, probability of crossover – 0.9, mutation method – classical, expected number of the chromosome mutation – 1, number of generations – 20·round(m/2), population size – 2·round(m/2), elite strategy – copying the best chromosome of population t-1 into the population t.
- RS – random search algorithm in which the new solution was generated by changing one randomly chosen bit of the old solution.

The results – accuracies of the classifier using selected features (minimal, maximal and mean in 30 runs) and their standard deviations – are presented in Tab. 2. The ranking of FS methods is shown in Fig. 2, where horizontal axis represents the mean difference between the mean accuracies of the best FS method and the given FS method.

In general the effectiveness of the search process depends on the algorithm parameter values. Usually the faster convergence leads to the freezing of the searching process in the local optimum. In most cases the tuning of the algorithm is a hard job requiring many experiments. Here, the parameter values of the algorithms were set in such a way in order to ensure enough time to the broad exploration of the feature space.

Unexpectedly in all cases TFS method gave the best results. The TFS method turned out to be more resistant to the local minima traps than others stochastic search algorithms tested in this work. The main advantage of TFS is only one parameter to adjust – the tournament size l.

The second best method was the genetic algorithm – the most complicated among tested methods. All stochastic methods except for random search gave better results on average than the deterministic approaches SFS and SBS.

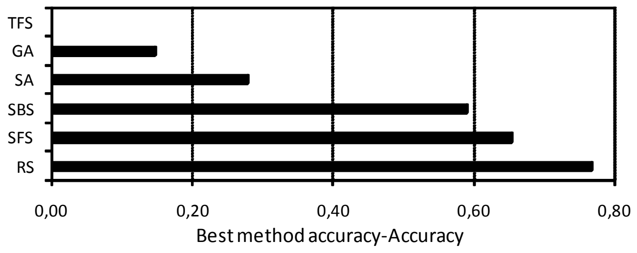

Fig. 2. Ranking of FS methods

Table 2. Percentage accuracies of the classifier: mean Acc_{mean}, minimal Acc_{min}, maximal Acc_{max}, and their standard deviations σ_{Acc}

Dataset		TFS	SA	GA	RS	SFS	SBS	Without FS
Ionosphere	Acc_{mean}	94.78	94.00	94.40	92.17	94.02	94.30	84.33
	Acc_{min}	94.59	92.88	93.73	91.45			
	Acc_{max}	95.44	94.59	95.44	93.16			
	σ_{Acc}	0.26	0.32	0.26	0.47			
Cancer	Acc_{mean}	98.25	97.97	98.11	97.57	97.72	97.54	96.61
	Acc_{min}	97.89	97.54	97.72	97.36			
	Acc_{max}	98.42	98.42	98.42	97.89			
	σ_{Acc}	0.18	0.19	0.22	0.12			
Heart	Acc_{mean}	86.25	85.92	86.07	85.67	85.93	85.93	82.22
	Acc_{min}	85.93	85.19	85.56	84.44			
	Acc_{max}	86.30	86.30	86.30	86.30			
	σ_{Acc}	0.13	0.29	0.21	0.43			
Wine	Acc_{mean}	98.88	98.62	98.69	98.35	98.88	97.75	96.07
	Acc_{min}	98.88	97.75	98.31	97.75			
	Acc_{max}	98.88	98.88	98.88	98.88			
	σ_{Acc}	0.00	0.29	0.27	0.25			
Glass	Acc_{mean}	74.77	74.75	74.77	74.67	73.36	74.77	65.89
	Acc_{min}	74.77	73.36	74.77	73.36			
	Acc_{max}	74.77	74.77	74.77	74.77			
	σ_{Acc}	0.00	0.14	0.00	0.36			
Diabetes	Acc_{mean}	77.21	77.20	77.21	77.09	76.30	76.30	73.96
	Acc_{min}	77.21	76.56	77.21	76.69			
	Acc_{max}	77.21	77.21	77.21	77.21			
	σ_{Acc}	0.00	0.08	0.00	0.19			

4 Conclusions

A practitioner would like to know which algorithm is the best choice to solve a given problem. Usually it is very hard to select one method looking on “no free lunch” theorem. There are no problem-independent reasons to favor one optimization method over another. The simulated annealing, genetic algorithm and proposed tournament searching method belong to the stochastic generic search algorithms. Their application to the optimization problem requires preliminary tests in order to tune the parameters. This is a time-consuming procedure depending on the numbers of parameters. There is only one parameter in the tournament searching, controlling the global-local search properties, which makes this algorithm easy to use.

The empirical comparison between all of the presented algorithms showed that the tournament searching method produced the best results (the accuracies of the k-NN classifier) for all datasets.

The tournament searching method, similarly to the genetic algorithm, has a parallel structure – several candidate solutions can be generated and evaluated at the same time. This results in the runtime decreasing.

In order to evaluate the tournament searching method more tests on different combinatorial optimization problems are necessary. This will be the subject of the future work.

References

1. Guyon, I., Elisseeff, A.: An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3, 1157–1182 (2003)
2. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A. (eds.): Feature Extraction. Foundations and Application. Springer, Berlin (2006)
3. Theodoridis, S., Koutroumbas, K.: Pattern Recognition. Elsevier Academic Press, Amsterdam (2003)
4. Kohavi, R., John, G.H.: Wrappers for Feature Subset Selection. Artificial Intelligence 1-2, 273–324 (1997)
5. Duch, W.: Filter Methods. In: [2]
6. Somol, P., Pudil, P., Kittler, J.: Fast Branch and Bound Algorithm in Feature Selection. IEEE Transaction on Pattern Analysis and Machine Intelligence 26, 900–912 (2004)
7. Siedlecki, W., Sklansky, J.: On Automatic Feature Selection. International Journal on Pattern Recognition and Artificial Intelligence 2(2), 197–220 (1988)
8. Raymer, M.L., Punch, W.F., Goodman, E.D., Kuhn, L.A., Jain, A.K.: Dimensionality Reduction using Genetic Algorithms. IEEE Trans. Evol. Comput. 4(2), 164–171 (2000)
9. Modrzejewski, M.: Feature Selection using Gough Sets Theory. In: Brazdil, P.B. (ed.) ECML 1993. LNCS, vol. 667, pp. 213–226. Springer, Heidelberg (1993)
10. Menczer, F., Degeratu, M., Street, W.N.: Efficient and Scalable Pareto Optimization by Evolutionary Local Selection Algorithms. Evolutionary Computation 8(2), 223–247 (2000)
11. Oduntan, I.O., Toulouse, M., Baumgartner, R., Bowman, C., Somorjai, R., Crainic, T.G.: A Multilevel Tabu Search Algorithm for the Feature Selection Problem in Biomedical Data. Computers and Mathematics with Applications 55, 1019–1033 (2008)
12. Bi, J., Bennett, K.P., Embrechts, M., Breneman, C.M., Song, M.: Dimensionality Reduction via Sparse Support Vector Machines. Journal of Machine Learning Research 3, 1229–1243 (2003)
13. Tuv, E., Borisov, A., Runger, G., Torkkola, K.: Feature Selection with Ensembles, Artificial Variables and Redundancy Elimination. Journal of Machine Learning Research 10, 1341–1366 (2009)
14. Devijver, P.A., Kittler, J.: Pattern Recognition: A Statistical Approach. Prentice-Hall, London (1982)
15. Stearns, S.: On Selecting Features for Pattern Classifiers. In: Proc. of 3rd International Joint Conf. on Pattern Recognition, pp. 71–75 (1976)
16. Pudil, P., Novovicova, J., Kittler, J.: Floating Search Methods in Feature Selection. Pattern Recognition Letters 15, 1119–1125 (1994)
17. Blum, A.L., Langley, P.: Selection of Relevant Features and Examples in Machine Learning. Artificial Intelligence 97, 245–271 (1997)
18. Kim, Y.: Feature Selection in Supervised and Unsupervised Learning via Evolutionary Search. Ph.D. Dissertation, University of Iowa (2001)
19. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220, 671–677 (1983)
20. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`