Tournament Feature Selection with Directed Mutations

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 tournament searching method with new mutation operators for a problem of the feature subset selection is presented. The probability of the bit mutation in a classical approach is fixed. In the proposed methods this probability is dependent on the history of the searching process. Bit position whose mutation from 0 to 1 (from 1 to 0) improved the evaluation of the solution in early iterations, are mutated more frequently from 0 to 1 (from 1 to 0). The roulette wheel method and the tournament method are used to select the bits for the mutation according to the adaptive probability. The algorithms were tested on several tasks of the feature selection in the supervised learning. The experiments showed the faster convergence of the algorithm with directed mutations in relation to the classical mutation.

Keywords: feature selection, tournament feature selection, directed mutation, roulette wheel mutation, tournament mutation, supervised learning, *k*-nearest neighbour method.

1 Introduction

Feature selection (FS) is an important stage in the design of classification and approximation systems, as well as in modelling the phenomena, processes and physical objects in general. The aim of FS is to reduce the dimension of the input vectors by the feature (variable) subset selection which describes object in the best manner and ensures the best quality of the model. In this process the irrelevant, redundant and unpredictive features are omitted.

The methods of FS can be generally divided into filter and wrapper ones [1]. 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 or 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.

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

L. Rutkowski et al. (Eds.): SIDE 2012 and EC 2012, LNCS 7269, pp. 190–198, 2012.

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 [1].

Some learning models have internal build-in mechanisms of 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 or artificial immune system proposed in [2] which includes the local feature selection mechanism. This approach, inspired by the binding of an antibody to an antigen, which occurs between amino acid residues forming an epitope and a paratope, allows the detection of many relevant feature sets (a separate relevant feature set is created for each learning point and its neighborhood).

In [3] the wrapper method of FS called tournament feature selection (TFS) was proposed. The solution strings processed by TFS are vectors composed of bits representing all m features: $\mathbf{x} = [x_1, x_2, \ldots, x_m]$. Ones and zeros in these vectors indicate whether the feature is selected or not. TFS is a simple stochastic search mechanism which explores the solution space starting from an initial solution and generating new ones by perturbing it using a mutation operator. This operator switches the value of one randomly chosen bit (but different for each candidate solution) of the parent solution. When the set of new l candidate solutions is generated (l represents the tournament size), their evaluations are calculated. The best candidate solution (the tournament winner), with the highest value of the criterion function, is selected and it replaces the parent solution, even if it is worse than the parent solution. This allows us to escape from the local maxima of the criterion function. If l is equal to 1, this procedure comes down to a random search process. On the other hand, when l is equal to the total number of features this method becomes a hill climbing method where there is no escape from the local maxima.

The TFS turned out to be very promising in the feature selection problem, better than a genetic algorithm and simulated annealing, as well as deterministic sequential forward and backward selection algorithms [3]. The TFS 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. The main advantage of TFS is only one parameter to adjust – the tournament size l.

This paper presents TFS with specialized binary search operators: roulette wheel and tournament mutations. These operators use information gained during the searching process about the effect of mutations at different bit positions on the solution quality. Bit position whose mutation from 0 to 1 (or 1 to 0) improved the evaluation of the solution in earlier iterations are more frequently mutated from 0 to 1 (or 1 to 0). This mechanism should speed up the convergence of the algorithm.

The biological inspiration for the proposed directed mutations is a hypothesis of directed mutagenesis proposing that organisms can respond to environmental stresses through directing mutations to certain genes or areas of the genome [4].

2 Roulette Wheel Mutation

In the roulette wheel method the mutation intensity is determined individually for each bit position $i = 1, 2, \ldots, m$. The indexes of mutation intensity from 0 to 1 $w_{0\text{-}1}(i)$ and from 1 to 0 $w_{1\text{-}0}(i)$ are introduced and initialized with zeros for each position i. The index values are updated after each algorithm iteration according to the following scheme:

- if the solution evaluation after mutation increases, the index values for mutated positions increase, i.e. $w_{0\text{-}1}(i)$ or $w_{1\text{-}0}(i)$ (respectively to the direction of mutation (from 0 to 1 or from 1 to 0) and the mutated bit position) is incremented by u.
- if the solution evaluation after mutation decreases, the index values for mutated positions decrease, i.e. $w_{0\text{-}1}(i)$ or $w_{1\text{-}0}(i)$ is decremented by u.
- if the solution evaluation after mutation does not change, the index values remain unchanged.

This can be expressed by formulas:

$$w_{0-1}(i) = \begin{cases} w_{0-1}(i) + u, & \text{if } i \in M \wedge F(\mathbf{x}') > F(\mathbf{x}) \wedge x_i = 0 \\ w_{0-1}(i) - u, & \text{if } i \in M \wedge F(\mathbf{x}') < F(\mathbf{x}) \wedge x_i = 0 \\ w_{0-1}(i), & \text{otherwise} \end{cases}, \tag{1}$$

$$w_{1-0}(i) = \begin{cases} w_{1-0}(i) + u, & \text{if } i \in M \wedge F(\mathbf{x}') > F(\mathbf{x}) \wedge x_i = 1 \\ w_{1-0}(i) - u, & \text{if } i \in M \wedge F(\mathbf{x}') < F(\mathbf{x}) \wedge x_i = 1 \\ w_{1-0}(i), & \text{otherwise} \end{cases}, \tag{2}$$

where: $u > 0$ – the incrementation/decrementation constant, $\mathbf{x}$, $\mathbf{x}'$ – the solution before and after mutation, M – the set of mutated position in the solution $\mathbf{x}$, $F(\mathbf{x})$, $F(\mathbf{x}')$ – the evaluation values of the solution $\mathbf{x}$ before and after mutation, respectively.

The high value of the index $w_{0\text{-}1}(i)$ ($w_{1\text{-}0}(i)$) informs that in the current realization of the searching process the change of the i-th bit value from 0 to 1 (from 1 to 0) caused improvement of the evaluation in the most cases. Thus the probability of the mutation of this bit from 0 to 1 (from 1 to 0) in the next iterations should be adequately high, depending on the value of $w_{0\text{-}1}(i)$ ($w_{1\text{-}0}(i)$) index. The mutation probability is calculated according to the formula:

$$p(i) = \frac{f\left(w_{d(i)}(i)\right)}{\sum_{j \in \Omega_0(i)} f\left(w_{0-1}(j)\right) + \sum_{k \in \Omega_1(i)} f\left(w_{1-0}(k)\right)}, \tag{3}$$

where: $w_{d(i)} = w_{0\text{-}1}(i)$ if the i-th bit value in the mutated solution is equal to 0 and $w_{d(i)} = w_{1\text{-}0}(i)$ otherwise, $\Omega_0(i)$ is a set of positions of zeros and $\Omega_1(i)$ is a set of positions of ones in the mutated solution, $f(.)$ is a logistic function of the form:

$$f(z) = \frac{1}{1+e^{-z}}. \tag{4}$$

The task of the logistic function having an "S" shape is to reduce the mutation probability of positions with large values of the mutation indexes and transform the negative values of indexes to the positive ones. This is illustrated in Fig. 1. Index values after transformation using (5) are in the range from 0 to 1. The positive values of indexes are necessary for the proper operation of the roulette wheel method. The reduction of mutation probability for positions with large index values eliminates the premature convergence to the superindividuals. The mutation probability is proportional to the value of $f(w_{d(i)}(i))$ now, and not to the value of $w_{d(i)}(i)$, which may increase/decrease to +/- infinity during the searching process and which may affect the mutation probabilities in an undesirable way. The mutation probability of z_2 in Fig. 1 is only about 11% larger than the mutation probability of z_1, although z_2 is two times larger than z_1.

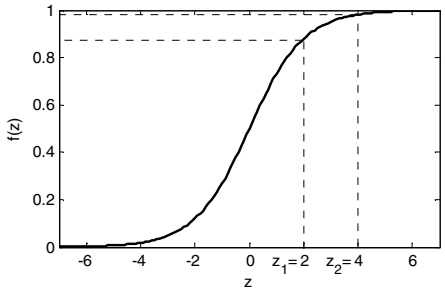

Fig. 1. The logistic function transforming the mutation index values

The mutation positions are chosen according to the probabilities $p(i)$ using the roulette wheel. The roulette wheel is composed of m sectors which sizes depend on $p(i)$, and their boundaries are determined as follows:

$$S(i) = \begin{cases} (0,\ p(1)] & \text{for } i = 1 \\ \left(\sum_{j=1}^{i-1} p(j),\ \sum_{j=1}^{i} p(j) \right] & \text{for } i = 2,3,...,m \end{cases}. \tag{5}$$

The circuit of the roulette wheel equals 1.

The scheme of the roulette wheel mutation for TFS is as follows:

1. For each position i = 1, 2, …, m, according to the values of $w_{0\text{-}1}(i)$ and $w_{1\text{-}0}(i)$ indexes, the values of probability $p(i)$ are calculated from equation (3).
2. For each candidate solution the roulette wheel is constructed taking into account sector boundaries (5).

3. For each candidate solution the uniformly distributed random number r from the range (0, 1] is drawn. The sector including number r determines the position and direction of the mutation.

In the original TFS algorithm, each candidate solution is mutated in a different position. If we want to introduce such a requirement in the roulette wheel mutation, before calculating probabilities (3) for the mutated candidate solution, we assume $p(i) = 0$ for all previously mutated positions in the current iteration and remove these positions as forbidden from the sets $\Omega_0(i)$ and $\Omega_1(i)$. For illustration in Fig. 2 the roulette wheels are shown for three 8-bit candidate solutions, where the parent solution is $\mathbf{x}$ = [0 1 1 0 0 0 1 0], the mutation indexes are: $\mathbf{w}_{0\text{-}1}$ = [-0.1, 0.5, 0.8, -0.4, -0.6, -0.2, 0.9, 0.0], $\mathbf{w}_{1\text{-}0}$ = [0.4, -0.3, -0.8, 0.5, 0.9, 0.6, -0.9, 0.1], and the random numbers are r = 0.76, 0.28 and 0.59.

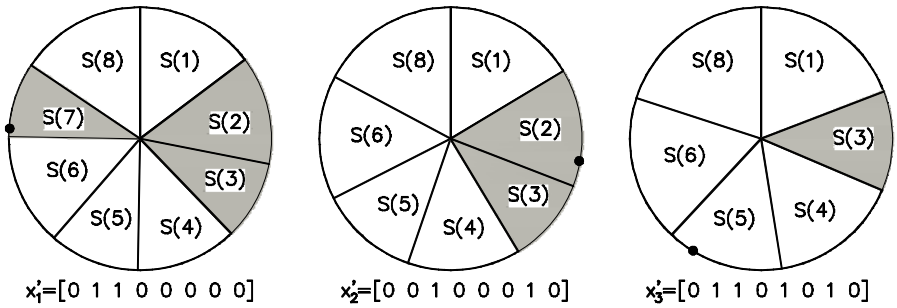

Fig. 2. The roulette wheels for the three successive candidate solutions generated from the parent solution $\mathbf{x}$ = [0 1 1 0 0 0 1 0]. (Dark sectors correspond to ones in the parent solution, and white sectors correspond to zeros.)

The mutation parameter u allows to control the selection pressure. The higher u value increases the selection pressure. Since the function (4) does not reach zero there is nonzero probability of mutation of each bit in both directions.

3 Tournament Mutation

Analogically to the roulette wheel mutation, in the tournament mutation the mutation intensity indexes $w_{0\text{-}1}(i)$ and $w_{1\text{-}0}(i)$ are defined. For each candidate solution h bit positions are sampled uniformly at random with replacement, where h = 1, 2, … is the tournament mutation size. Among h positions the one with the highest value of $w_{d(i)}(i)$ is chosen, and the value of this position is changed to the opposite one.

Note that here we do not need to calculate the probability of mutation and the sizes of sectors such as in the case of the roulette wheel mutation, because in the tournament selection procedure there is not important what is the difference between the mutation index values $w_{d(i)}(i)$ corresponding to the bit positions competing in the tournament.

The tournament size h controls the selection pressure. If $h = 1$, the tournament mutation is reduced to the classical random mutation.

Restriction used in TFS, that every candidate solution is mutated in a different position, is implemented here in such a way that the positions mutated in the previously considered candidate solutions do not participate in the tournament for the current candidate solution.

The roulette wheel and tournament mutations should bring good results in the tasks where the i-th bit value influences the value of objective function in the same way, independently of the bit context (values of the remaining bits in the solution).

4 Application Examples

The proposed TFS method with roulette wheel and tournament mutations was verified on several test problems of data classification. Benchmark datasets, described in Table 1, were taken from the UCI Machine Learning Repository. The features in the datasets were standardized to zero-mean and unit-variance.

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 Statlog	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.

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 each dataset the feature space was optimized running algorithms 30-times. The number of solutions generated in the searching process was the same for all mutation variants: $40 \cdot \text{round}(m/2)^2$. The parameter values are listed below:

- tournament size in TFS: $l = \text{round}(m/3)$,
- incrementation/decrementation constant $u = 0.1$,
- tournament mutation size: $h = 2$ or 4.

Experiments were carried out using TFS with standard mutation (SM), roulette wheel mutation with replacement (the same mutations for different candidate solutions are possible, RWM1), roulette wheel mutation without replacement (the same mutations for different candidate solutions are not possible, RWM2), tournament mutation with replacement for $h = 2$ (TM1), tournament mutation without replacement for $h = 2$ (TM2) and tournament mutation with replacement for $h = 4$ (TM3).

The results are presented in Table 2, where Acc_{mean}, Acc_{min}, Acc_{max} are accuracies of the classifier using selected features (mean, minimal and maximal accuracies returned in 30 runs) and σ_{Acc} is the standard deviation of accuracy.

The convergence curves averaged from 30 runs for SM, RWM2 and TM2 are shown in Fig. 3. Characteristically, the convergence curve for SM is the lowest. This indicates the large variance of the searching process (we observe high variability of the process). Directed mutations RWM and TM reduce the variance leading the searching process to the promising regions of the solution space, which have been identified in an earlier stage of searching and stored in the mutation indexes. But from Table 2 it can be seen that TFS with the simple standard mutation usually leads to no worse results than the directed mutations.

Table 2. Results of classification using *k*-NN and TFS

Dataset		SM	RWM1	RWM2	TM1	TM2	TM3	Without FS
Ionosphere	Acc_{mean}	94.78	94.09	94.12	94.68	94.62	94.23	84.33
	Acc_{min}	94.59	92.88	92.88	94.30	94.02	92.88	
	Acc_{max}	95.44	94.87	94.87	95.44	95.44	94.87	
	σ_{Acc}	0.26	0.45	0.47	0.35	0.26	0.40	
Cancer	Acc_{mean}	98.25	97.87	97.89	98.05	98.01	97.91	96.61
	Acc_{min}	97.89	97.72	97.72	97.72	97.54	97.36	
	Acc_{max}	98.42	98.24	98.42	98.42	98.42	98.42	
	σ_{Acc}	0.18	0.15	0.17	0.20	0.23	0.22	
Heart Statlog	Acc_{mean}	86.16	85.84	85.84	86.06	86.04	85.83	82.22
	Acc_{min}	85.93	84.81	84.81	85.93	85.93	85.19	
	Acc_{max}	86.30	86.30	86.30	86.30	86.30	86.30	
	σ_{Acc}	0.18	0.30	0.32	0.18	0.17	0.27	
Wine	Acc_{mean}	98.88	98.86	98.86	98.88	98.88	98.67	96.07
	Acc_{min}	98.88	98.31	98.31	98.88	98.88	98.31	
	Acc_{max}	98.88	98.88	98.88	98.88	98.88	98.88	
	σ_{Acc}	0.00	0.10	0.10	0.00	0.00	0.28	
Glass	Acc_{mean}	74.77	74.77	74.77	74.77	74.77	74.77	65.89
	Acc_{min}	74.77	74.77	74.77	74.77	74.77	74.77	
	Acc_{max}	74.77	74.77	74.77	74.77	74.77	74.77	
	σ_{Acc}	0.00	0.00	0.00	0.00	0.00	0.00	
Diabetes	Acc_{mean}	77.21	77.21	77.20	77.21	77.21	76.94	73.96
	Acc_{min}	77.21	77.21	76.69	77.21	77.21	76.30	
	Acc_{max}	77.21	77.21	77.21	77.21	77.21	77.21	
	σ_{Acc}	0.00	0.00	0.10	0.00	0.00	0.33	

In order to confirm the faster convergence of TFS with directed mutation, we check whether the difference d between the average evaluation of the parent solutions in all iterations and runs of the algorithm in the case of TFS with standard mutation and TSF with directed mutation is statistically significant. Because we cannot assume a normal distribution of accuracies we use for this purpose two nonparametric tests: the sign test for the null hypothesis that the difference d has zero median and

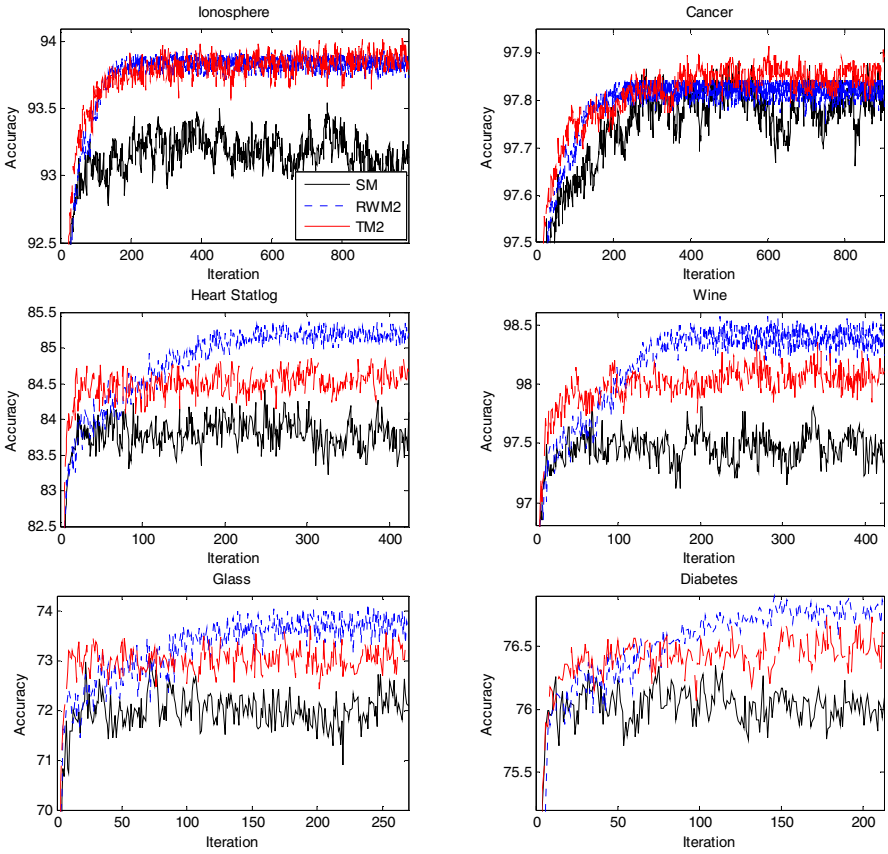

Fig. 3. The mean convergence curves

Wilcoxon rank sum test for equality of medians of two distributions. The 5% significance level is applied in this study. The test results confirmed that in all cases TFS with the directed mutation (RWM1, RWM2, TM1, TM2 and TM3) converges faster than TFS with standard mutation.

Fig. 4 demonstrates how for the Diabetes dataset the transformed values of the mutation indexes changed during the searching process. Decreasing values of $w_{0\text{-}1}$ and in the same time increasing values of $w_{1\text{-}0}$ for features 1, 3, 4 and 5 inform, that these features are irrelevant, because switching bits corresponding to them from 0 to 1 most frequently resulted in the deterioration of the classifier accuracy, and switching these bits from 1 to 0 resulted in increased accuracy. The $w_{1\text{-}0}(2)$ decreases very rapidly which means that the mutation of the second bit from 1 to 0 is unfavourable. As a result, this bit often takes value 1, so the mutation from 0 to 1 does not occur and $w_{0\text{-}1}(2)$ cannot adapt its value (straight line for $w_{0\text{-}1}(2)$ in Fig. 4(a)). A similar but opposite situation is for the 4-th feature.

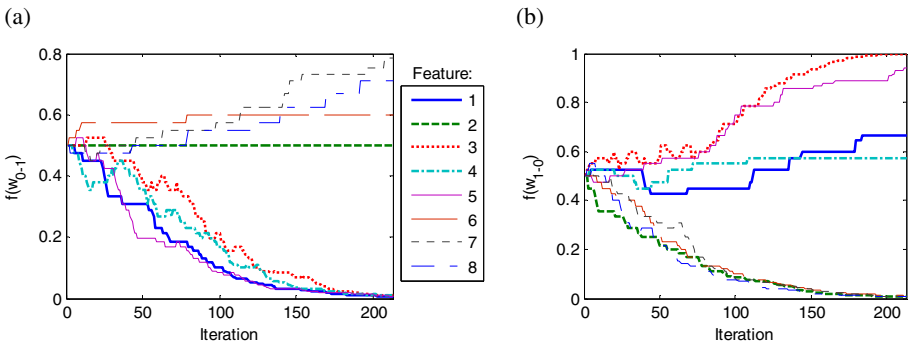

Fig. 4. The mutation index values transformed using logistic function (4) during the searching process for Diabetes data and RWM2

5 Conclusion

The article describes an attempt to improve the performance of the tournament feature selection method by introducing new methods of mutation, in which the probability of the bit mutation depends on the effectiveness of mutation of this bit in an earlier stage of the searching process.

In the early iterations of the algorithm the probability of mutation of all bits are equal. This ensures the thorough search of the solution space in the whole range. In the course of the search process information about whether the change of the specific bit from 0 to 1 and vice versa improves or deteriorates the solutions are stored. This information is used in subsequent iterations to adapt mutation probabilities of individual bits: bits that mutation from 0 to 1 (1 to 0) increased the solutions are mutated in this direction more often. As a result, the algorithm exploitation capabilities are enhanced: the neighborhood of the best solution is searched more intensively. This mechanism is effective when the value of the bit affects the evaluation of solutions in the same way, regardless of the context.

The results of this investigation have shown that the convergence of the algorithm was improved through the use of the roulette wheel and tournament mutations, but better classifier accuracy than using tournament feature selection with the standard mutation operator were not achieved.

References

1. Kohavi, R., John, G.H.: Wrappers for Feature Subset Selection. Artificial Intelligence 1-2, 273–324 (1997)
2. Dudek, G.: Artificial Immune System for Classification with Local Feature Selection. IEEE Transactions on Evolutionary Computation (in print)
3. Dudek, G.: Tournament Searching Method to Feature Selection Problem. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010. LNCS (LNAI), vol. 6114, pp. 437–444. Springer, Heidelberg (2010)
4. Cairns, J., Overbaugh, J., Miller, S.: The Origin of Mutants. Nature 335, 142–145 (1988)