UDC 681.3

FEATURE SELECTION FOR TIME-SERIES PREDICTION IN CASE OF NONDETERMINED ESTIMATION

S. V. Khmylovy, Y. A. Skobtsov

Donetsk National Technical University

Abstract

This paper discusses the problems of feature selection when a feature set estimation is not stochastic. In this case two sets of features can be compared just with some probability, so the existing methods should be modified. For this purpose we propose the use of Compact Genetic Algorithms (CGA) and present a scheme of feature selection. The step of genetic algorithm learning is modified for the case of stochastic estimation of a feature set. The results for Internet-traffic forecasting are obtained.

Keywords: data mining, evolutionary computations, forecasting, time series

1. Feature Selection Problem

Feature selection problem refers to *Data Preprocessing task* in *Data Mining* and *Knowledge Discovery.* In [21] the *feature selection* and the *feature extraction tasks* are singled out in *Data Preprocessing*. Blum and Langley [5] divide the methods of feature selection into *embedded approach*, *filtering approach* and *wrapper approach*.

Traditionally, for the task of feature selection a mathematical apparatus of correlation analysis, linear dependences, etc. were used. Baestans in [2] shows that in the case of insignificant input factors dropping the rate of correlation between this factor and other input factors may be high, and the rate of correlation between this input variable and the output variable can be low. Hattingh, Kruger [13] use *mixed integer linear programming* to remove insignificant factors and to filter the data that cause the most noise in the prediction. The significance of factors is defined with the help of the linear model. Ahmad, Dey [1] use a probability based method to extract the significant attributes. F. Moerchen [18] offers a DWT and DFT based modified algorithm for reducing a set of data.

During the input factors interaction analysis Ezhov [7] uses *Principal Component Analysis* to reduce input dimensions. It is based on dropping the inputs that have minor values of the covariance matrix, which takes into account the linear interaction only. The same work [7] describes the application of neural networks (NN) in nonlinear principal components method, which allows the interactions of high order. To consider the interaction between the input factors and the output variable the author propose to use *Box-counting algorithms* based on calculating the numbers of boxes occupation (the space of variables is divided into boxes) by the examples of the training set. The author also suggests a suboptimal algorithm of successive addition of the most significant inputs. At each stage this algorithm adds another factor which will be the most important in the set that had already been selected.

The use of genetic algorithms (GA) for feature selection is justified and can compete with other methods in its efficiency. Such approaches refer to wrapper-methods. Freitas [8], Vafaie, De Jong [23] suggest the classical approach to coding of individuals and to GA operators. The use of the classical approach is also shown by Guerra-Salcedo, Whitley [12], Vafaie, De Jong [23]. Minaei-Bidgoli, Punch [17] propose to use GA not only in factors selection, but also in defining their significance. Hsu et al. [14] use GA not only in *attribute selection*, but also in *attribute partitioning* (creating new attributes based on the groups of old one). Oh et al. [19] develop the idea of successive addition/removal of the most/ the least significant inputs (suggested by Ezhov [7]), having made this addition/removal a part of a hybrid genetic algorithm.

The idea of using GA as a factors set selector together with other methods (as classifiers) is also well known. Bala et al. [4] use GA as a “filter+wrapper” for a *decision-tree learning algorithm*, which carries out the classification. Bala et al. [4] and Raimer et al. [21] apply GA together

with k-nearest neighbor algorithm. Bala et al. [3] use GA for factor selection and the *decision tree* serves as a *predictor.*

The combination "GA+NN" can be found, for instance, in [11]. The authors use GA for creating a grammar tree that yields both architecture and weights, specifying a particular neural network for solving specific Boolean functions. The neural networks for classification are used byYang, Honawar [25]. The selection of significant factors for a neural network is carried out by a genetic algorithm, which estimates a *multi-criteria task* (prediction accuracy and the cost of obtaining factor values).

But there are some conditions whish complicate the use of these methods, and the results obtained are not optimal. In case when the quality estimation of a particular data set is stochastic, the use of most methods is limited. For the purpose of qualitative assessment each subset of attributes should be repeatedly estimated so the algorithm must be modified. Both, neural networks and genetic algorithms, which carry out the prediction, can act as the environment giving the stochastic quality assessment of a factor set. While selecting a particular set of influencing factors it is reasonable to use genetic algorithms.

The rest of this paper is organized as follows: Section 2 describes factor selection problem formalization. Section 3 discusses the scheme of factor selection. Section 4 describes compact genetic algorithm application for factor selection. Finally, section 5 presents the practical results of the presented system.

2. Factor Selection Problem Formalization

The problem of factors selection consists in choosing a d size subset from an attribute set with total number D on the basis of a given optimization criterion. We will designate the total input data set (the maximum possible number of attributes) as $U = \{1, 2, \ldots, D\}$.

On the other hand, the subset of the selected factors is designated as $X = \{1, 2, \ldots, d\}$ while Y stands for the set of remaining (removed) factors. Thus, $U=X\cup Y$.

When information is being transformed and reduced we consider (instead of U) a certain range of values $f(U)$, which is a functional transformation from U. Respectively, X and Y are the sets of selected and dropped attributes of this transformed set. Thus, $X\cup Y= f(U)$.

For example, for the case of time series the data can be represented both in time and in frequency domain. Operating with time domain we reduce a set of factors, and operating with frequency domain we can also drop harmonics with low energy. The transition from one domain to the other is carried out by means of Fourier transformation that serves as $f(U)$. Besides, there is, for example, a factor analysis applied for obtaining a certain set of factors (describing the given situation) through functional transformation of time-series data.

The quality estimation criterion X is designated as $J(X)$. This criterion allows evaluating both the accuracy of a certain qualifier on a certain set of the data ("wrapper" approach) and a universal statistic unit ("filtering" approach). Anyway, the choice of J depends on a particular task.

The factor selection procedure is aimed at finding such a set of X which would satisfy the following condition:

$$\begin{cases} J(X) \to \max \, with \\ \quad |X| \to \min \end{cases}, \tag{1}$$

where $|X|$ is the number of the attributes contained in X.

Thus, we have a multi-criteria problem. Its extremum can not be defined beforehand and does not depend on $J(X)$ and on the minimum threshold for $|X|$, set by the user.

3. Scheme of Factor Selection

There are many problems for which the result is not an exact estimation, but is obtained with some probability. One of such problems is forecasting using artificial intelligence methods. For example, the prediction which employs neural networks is nondetermined and contains some uncertainty. Very often evolutionary algorithms can only find a suboptimal solution, so we may state that the obtained solution is stochastic. For example, different program launches (including learning) can give different results.

As it has been stated above, if the result of estimating a certain attribute subset is stochastic then, in order to find the optimal set, we need to involve the algorithms which would estimate repeatedly each of such sets. In particular, genetic algorithms are to be used (Goldberg [9]). Consequently the scheme of factor selection (Figure 1) will be as follows

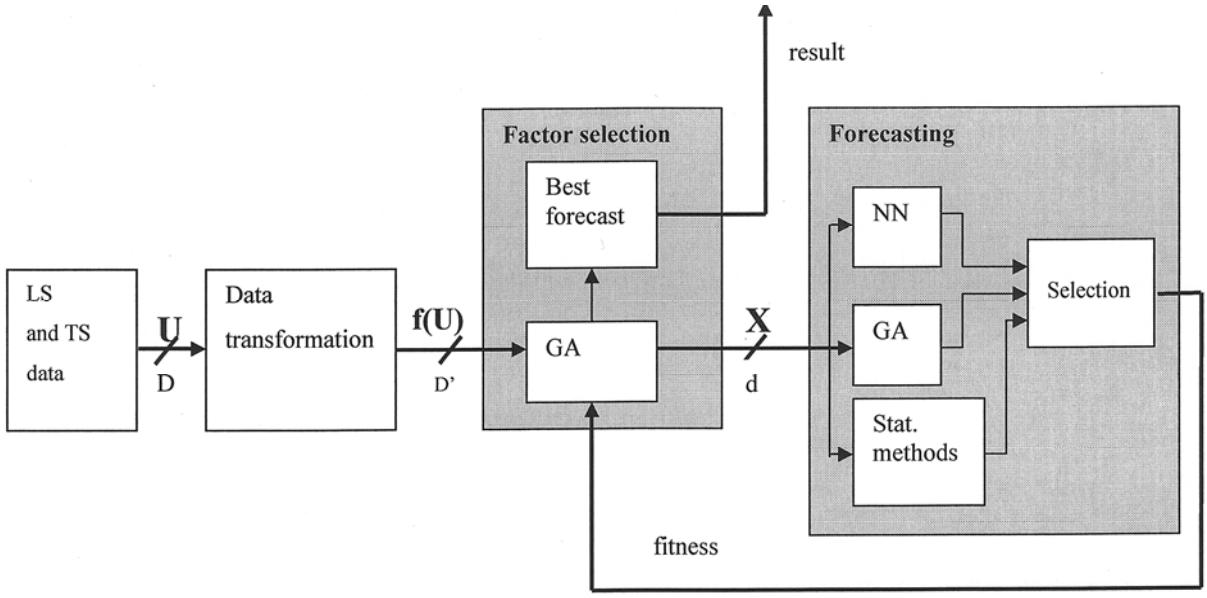

Figure 1. Scheme of factors selection

In general the set of data U (which includes learning set (LS) and testing set (TS)) with total number of factors D is subjected to transformation (feature extraction task). The transformed set of data $f(U)$ with $D`$ number of factors is reduced using genetic algorithms. The reduced set of data X with $d = |X|$ is used in forecasting. In this case one can use various methods, both statistical and those of artificial intelligence (neural networks or genetic algorithms). The choice of the method is made on the basis of the obtained prediction or manually by the user. However, it's worth mentioning that if prediction is performed by means of statistic methods its results are not stochastic, so the meaning of the given scheme is lost. The forecast value obtained is used as a fitness-function for the factor selection genetic algorithm. The best forecast value and the set of factors corresponding to this forecast should be stored; they are the result of factor selection system.

4. Compact Genetic Algorithm Application for Factors Selection

In Figure1 genetic algorithm application for factor selection is offered. In this case a particular kind of genetic algorithm should be selected. The algorithm should be simple and brief as the value of fitness-function is recalculated for each chromosome in each epoch.

Haric et al. [10] suggest a CGA (Compact Genetic Algorithm) and study its efficiency in solving optimization problems. We should mention that the given approach presupposes an extremely simple soft- and hardware implementation; the results being comparable to classical genetic algorithms. Compact genetic algorithm scheme is represented in Fig. 2.

In CGA the population of binary individuals is replaced by a probability vector; for each bit of chromosomes the probability of its zero (unit) value is provided. It allows compact representation of the initial population of binary chromosomes. For example, see table1. The population (the first four lines) can be represented in CGA by the following vector of probabilities (Pcga, the last line).

Table 1. Vector of probabilities in CGA

X1	X2	X3	X4	X5
1	0	1	1	0
1	0	0	1	0
1	0	1	1	0
0	0	0	1	1
0.75	**0**	**0.5**	**1**	**0.25**

The essence of the compact genetic algorithm is that in each epoch a new micro-population is created. It is generated on the basis of the above mentioned probability vector. Then tournament selection takes place when the fitness-functions of all micro-population individuals are compared. The probability vector is corrected after each comparison. The correction is done as follows: if there are different values (0 and 1) in the alleles of the winner and the loser the probability of further generation of the number in the winner's allele increases. In the original (Haric et al. [10]) the step of change is $1/n$, where n is the full size of algorithm population. After the tournament selection the individuals are exterminated and created again in the next epoch. When the transition of all probabilities of the probability vector to stationary statuses (0 or 1) takes place the algorithm stops its work.

In case of applying genetic algorithms in factor selection the individual is represented as a characteristic vector where 1 in i^{th} position corresponds to the input of the i^{th} factor into LS and 0 means its absence. The length of an individual is equal to the total number of all significant factors.

In the given case the fitness-function is the neural network prediction error for the sample made up of the factors coded by the given individual. The process of obtaining fitness-functions is the following: learning and testing sets (LS and TS) are made which include the factors encoded by the given individual. Then LS is used for neural network learning. After NN learning we involve TS input parameters. Then we define the prediction error by means of comparing NN parameters obtained at the output and standard TS values. This error is used as a fitness-function. The peculiarity of this CGA application is that the prediction error is not stable. It is different for the same LS and TS at each particular session of NN learning. Thus, the fitness-function of an individual in CGA is a random variable. So comparing two fitness-functions we may speak about the superiority of one of the functions just with some probability. When calculating the fitness-function of and individual (and consequently prediction error) it is possible to minimize this drawback each epoch.

```
1.  probability vector initialization
    for i:=1 to l do p[i]:=0.5;

2.  Generation of m individuals based on a probability vector
    for i:=1 to m do
    M[i]:=generate(p);

3.  Circular tournament
    for j:=1 to m-1 do
       for k:=j+1 to m do
       begin
         victor, vanquished:=evaluate(M[j],M[k])
4.           Updating the probability vector
             for l:=1 to length(M) do
                if winner[l]<>loser[l] then
                if winner[l]=1
                then Pcga[l]:=Pcga[l]+chag
                else Pcga[l]:=Pcga[l]-chag
             end
       end
    end

5.  Check vector convergence
    for i:=1 to l do
      if p[i]>0 and p[i]<1 then
       return to step 2;

6.  p represents a final solution

CompactGA parameters:
n: population size
l: length of a chromosome
```

Figure 2. The scheme of a compact genetic algorithm

There is a certain parameter *n* in the algorithm. It is the initial population size "inherited" from the standard genetic algorithms, so it is not based on the structure of CGA itself. The step of probability vector change is based on this parameter. Generally this parameter can be taken absolutely arbitrary.

Suppose the value of fitness-function is calculated correctly. Let us define the step for achieving the optimum in 1 epoch. Each initial probability of each element of the probability vector is $p[i] = 0.5$, and the size of micro-population is m. Then we have a set of equiprobable states (0 or 1). The probability of choosing $k1$ zeros and $(k-k1)$ unities from a set containing $m1$ zeros and $(m-m1)$ unities is equal [6])

$$P = \frac{C_{m1}^{k1} * C_{m-m1}^{k-k1}}{C_m^k}. \tag{2}$$

For the case of two values comparison the probability of comparing different values 0 and 1 (otherwise probability vector modification for CGA is impossible) is equal to

$$P_h = \frac{C_{n/2}^1 * C_{n/2}^1}{C_n^2}. \tag{3}$$

where n is the size of the set.

The total number of comparisons is C_n^2. So in order to achieve the optimum in one CGA epoch the step should be equal to

$$h = \frac{1}{C_n^2 * P_h} = \frac{1}{(C_{n/2}^1)^2}. \quad (4)$$

This formula is true for the even number of chromosome alleles and that is a severe constraint.

In case of incorrectness the fitness-function is described as a random variable with expected value μ and deviation σ. Then in the process of tournament the expected values of random variables μ_1 and μ_2 are compared. When $\mu_2 > \mu_1$, the probability of an error [6] is

$$P_{err} = P(\mu_1 - \mu_2 < 0) = 1 - \Phi(\frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}), \quad (5)$$

where $\Phi(z) = \frac{1}{2\pi}\int_0^z e^{-\frac{z^2}{2}} dz$ is Laplace function.

Thus the size of the step must be corrected taking into account the probability of wrong error comparison. The size of the step is to be multiplied by P=1-2*$Perr$.

Then the formula of the step of probability vector change in CGA looks as follows:

$$h_m = h * (1 - 2 * P(\mu_1 - \mu_2 < 0)). \quad (6)$$

While the algorithm is working the average number of zeros and unities varies and is determined by the probability vector Pcga. The average number of unities in a chromosome is equal to

$$SumPcga = \sum_{i=1}^{length(P_{CGA})=l} P_{CGA}[i]. \quad (7)$$

The probability of comparing various values is reduced:

$$P_h = \frac{C_{SumPcga}^1 * C_{l-SumPcga}^1}{C_l^2}. \quad (8)$$

Accordingly, the step of probability vector change must increase:

$$h = \frac{1}{C_n^2 * P_h} = \frac{C_l^2}{C_n^2 * C_{SumPcga}^1 * C_{l-SumPcga}^1} = \frac{C_l^2}{C_n^2 * SumPcga * (l - SumPcga)}. \quad (9)$$

The final modified step looks as follows:

$$h_m = \frac{C_l^2}{C_n^2 * SumPcga * (l - SumPcga)} * (1 - 2 * P(\mu_1 - \mu_2 < 0)). \quad (10)$$

So we have obtained an algorithm for defining the optimal subset of attributes in the case when the quality estimate of this subset is stochastic. The value at the NN output is taken as an estimation measure. CGA (Compact Genetic Algorithms) are used for adjusting the sets of attributes. The step of probability vector change is modified taking into account estimation stochasticity.

5. Testing and Practical Results

The results of CGA step modification were tested on benchmarks from Proben1 [20], UCI Library [22] and in actual prediction of the data flows from in- and outcoming channels of Internet Service Provider, ISP.

5.1. Testing with Standard Benchmarks

In each test problem the data distribution must be uniform because it increases the accuracy of testing. Classification results were obtained by NN on usual non-transformed data (*Untransform*), on transformed data (*Uniform*), on data with a factor set reduced by classical CGA and with reduced *ModCGA*. In the testing set Glass 1 [20] NN learned on LS, was tested on TS and on checking sample. The other sets (Hepatitis and Ionosphere) [22] involved 3-fold cross-validation: three sets were created, TS occupied one third of the sampling being taken from the beginning, the middle and the end of a data file. MSE results were obtained on LS (*Train Error*) and TS (*Test Error*) together with classification errors (*Train Class Err*). For TS we estimated classification accuracy *Test Class* instead of estimating classification error. The obtained results were averaged; the mean value and standard deviation of classification accuracy (*Dev*) are provided. Besides, for CGA the number of epochs of algorithm realization (*Ep*) and the number of factors in a set (*Fact*) were introduced. Classification accuracy (on LS) served as CGA fitness function. The components responsible for the set reduction were not used. The results were compared with those obtained by NN on (*PCA+NN*) sample (transformed by means of principal component analysis) and with results provided in [16], [19], [25]. In [19] and [25] the combination GA + NN was also used, while Kwedlo and Kretowsky [16] applied C4.5 and EDRL.

Table 2. Results obtained on Glass1 benchmark

Glass1 9+6, 214rec	Train error	Test error	Train Class Err	Test Class	Ep	Fact
Untransform	2.83 (2.51)	12.26 (10.31)	2.88 (3.2)	**88.13 (11.15)**		
Uniform	4.59 (4.51)	11.01 (9.4)	5.50 (6.17)	**88.6 (10.69)**		
CGA	5.7 (4.13)	10.74 (10.47)	8.16 (6.11)	**90.32 (8.29)**	79	3
CGA Valid	5.71 (4.16)	10.59 (11.01)	8.15 (6.17)	**90.21 (8.13)**	79	3
ModCGA	4.22 (3.79)	10.04 (7.85)	5.39 (5.14)	**90.30 (7.35)**	78	4
ModCGA Valid	4.20 (3.75)	10.39 (9.35)	5.36 (5.05)	**90.13 (7.89)**	78	4
PCA+NN	0.77 (0.65)	17.47 (10.12)	0.34 (0.61)	**84.78 (10.8)**		6
Proben1 linear	8.83 (0.01)	9.98 (0.1)		**53.96 (2.21)**		
Proben1 mult 8+0+l			9.184	**67.92**		
Proben1 pivot	7.68 (0.79)		9.75 (0.41)	**61.97 (8.14)**		
Yang, Honavar (1998)				**70.5 (8.5)**		
Yang, Honavar GA. (1998)				**80.8 (5.0)**		
Oh (2004)				**100**		
Kwedlo, Kretowsky (1998) C4.5				**67.5 (0.8)**		
Kwedlo, Kretowsky (1998) EDRL				**66.7 (1.0)**		

Table 3. Results obtained on Ionosphere benchmark

Ionosphere 34+2, 351 rec	Train error	Dev	Test error	Dev	Train Class Err	Dev	Test Class	Dev	Ep	Fact
Uniform	0.52 (0.38)	0.38	3.84	1.51	0.00	0.04	97.89	2.28		
CGA	0.30	1.09	1.40	3.17	0.35	1.52	99.39	2.49	168 (49.38)	8.66 (6.26)
ModCGA	0.46	0.89	2.54	3.66	0.00	0.20	98.82	3.22	70.93 (51.99	8.73 (3.96)
PCA+NN	0.37	0.12	8.08	2.72	0.03	0.11	94.43	1.6		34
Yang, Honavar (1998)							94.30	5.00		
Yang, Honavar (1998)							98.60	2.40		
Oh (2004)							91.45			

Table 4. Results obtained on Hepatitis benchmark

Hepatitis 19+1, 155rec	Train error	Dev	Test error	Dev	Train Class Err	Dev	Test Class	Dev	Ep	Fact
Uniform	1.21	2.32	20.00	8.57	0.31	1.37	79.96	7.50		
CGA	4.50	1.56	16.62	13.86	4.90	3.89	82.53	9.83	160.67 (61.12	8.67 (1.32)
ModCGA	3.49	12.14	18.27	23.07	3.40	10.02	82.04	12.30	15.33 (6.08)	10.50 (1.64)
PCA+NN	0.03	0.07	40.94	12.56	0.02	0.12	64.09	6.15		19
Yang, Honavar (1998)							84.70	9.50		
Yang, Honavar (1998)							97.1	4.30		
Kwedlo, Kretowsky (1998) C4.5							79.6	0.60		
Kwedlo, Kretowsky (1998) EDRL							81.2	1.80		

As we can see, the results obtained by CGA+NN combination on uniform data actually exceed all the results of similar works. Introduction of modified step does not reduce the accuracy, but saves the time of search (by 90,6 per cent).

5.2. ISP Data Forecasting Description

Every provider has incoming and outcoming channels of data communication. Depending on the topology of data flow commutation on ISP the same flow can be incoming for one site and outcoming for another one. Besides, it can be multiplexed, i.e. it becomes a part of a larger data flow which combines several logical flows in a physical one.

Figure 3 shows the time dependences of two resulting data flows which are the main numerical expressions of the loading of provider's channels. The In- and Out-channels have already

included all incoming and outcoming ISP data flows and unite several logical flows into a physical one. This information (as well as some other data) is stored in ISP database.

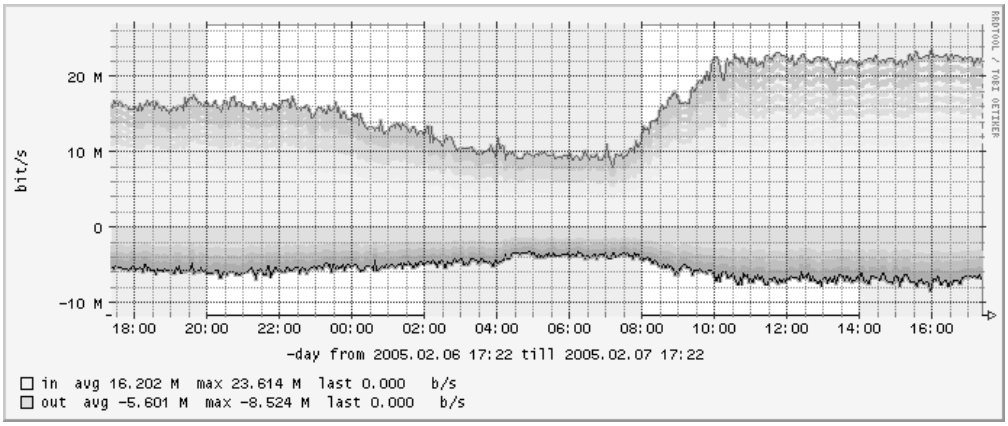

Figure 3. General ISP in- and out-channels

From time to time during ISP functioning the so-called critical situations occur. They are the spontaneous jumps and falls in the data volume (while the data are passing through On- or Out-channel). Sometimes they are caused by ISP itself, and sometimes the causes do not depend upon the provider. In such cases maintenance staff should be involved to improve the situation. ISP must detect such situations in proper time and predict the channels loading (the volume of information passing through channels per one time unit) in next time intervals.

A more detailed description of this problem and the selection of influencing factors are presented in [15]. Having analyzed the problem we singled out a redundant set of influencing factors. Redundancy is also confirmed by the fact that some influencing factors reduced overall prediction accuracy. Further optimum reduction of the influencing factors set is one of tasks of this work.

5.3. Selecting the Minimal Number of Significant Factors

We have compared the following results:

1. The accuracy of the prediction made by NN on a full redundant set of factors.
2. The prediction made on a factor set with the time series reduced up to 1 lag (see [2]).
3. The prediction made on a factor set reduced by means of correlation analysis, standard CGA with n=50 and ModCGA.

Simple reduction of the time series length resulted in removing 4 significant factors of 20. The result of forecasting got improved by 5% compared with the full set of factors.

In the process of forecasting by means of correlation analysis we used 10 factors. The results got improved by 11% compared with the results obtained on the full learning set. Correlation analysis showed:

- Great interconnection between many variables. For each input factor there are 2-3 factors with correlation coefficients more than 0.5.
- Insignificant influence of the factors on the predicted values. For 40% of influencing factors the coefficient of correlation with predicted values did not exceed 0.05.
- Different degree of connection of different influencing factors with different predicted values. Thus, in order to obtain the optimal prediction results we need three different sets of influencing factors for three predicted values.

The use of standard CGA with population size *n=50* for factor set minimization has decreased the forecasting error by 16% compared with the result on the full learning set. The use of ModCGA reduced the time of solution search by 51%. The forecasting error (if compared with standard CGA results) did not change considerably. The results obtained are presented in Table 5.

Table 5. Results obtained on Internet traffic Data

Compared to: Modification	Complete set of factors	CGA with n=50
Small series length	Error reduced by 5%	
Correlation analysis	Error reduced by 11%	
CGA with n=50	Error reduced by 16%	
ModCGA	Error reduced by 16%	Calculation time decrease by 51%
Separation of Classification and forecasting tasks	Error reduced by 4%	

5.4. Separation of Classification and Forecasting Tasks

Two problems were initially considered: classification of critical situations and time series forecasting. It was proved that different sets of factors are relevant for these problems. So we attempted to separate these tasks by means of two-stage data processing. At the first stage we determined and deleted critical situations and at the second stage we carried out time series prediction. As a result prediction error decreased by 4% as compared to the initial one. So it is obvious that the problem of critical situations classification and that of time series prediction should be solved separately with individual search of an optimal factor set for each problem.

Conclusion

1. There are certain conditions (in particular, the stochasticity of factor set estimation) when the classical methods of factor selection are not helpful and require modification. In this case the most effective methods are those including the repeated procedure of factor estimation. Genetic algorithms are most suitable for this purpose.

2. We have specified the conditions under which the obtained estimate of attribute subsets is stochastic. Such estimation is possible at the output of NN or GA used in time series prediction

3. The factors should be selected according to a particular scheme. Such scheme allows preliminary data transformation based on factor analysis or Fourier transformation. Then a genetic algorithm is applied for factor selection. The prediction is made on the basis of the obtained reduced set of factors. Prediction results are used as the fitness-function for factor selection GA.

4. Compact GA is the most promising genetic algorithm for such kind of tasks. It is distinguished by simplicity of software implementation and high performance in solving such problems.

5. The step of CGA probability vector change has been modified for the case of stochastic estimation. For this purpose we should define the dependence of probability vector change upon the reliability of comparing two values (measures of attribute subsets estimation).

References:

1. Ahmad Amir, Dey Lipika, 2005. A feature selection technique for classificatory analysis. Sciencedirect. Pattern Recognition Letters, 26, pp 43-56.
2. Dirk-Emma Baestaens, Douglas Wood, Max Van De Bergh 1994. Neural Network Solutions For Trading In Financial Markets. Financi Times/ Prentice Hall.
3. Bala, J., K. DeJong, J. Huang, H. Vafaie, and H. Wechsler, 1995. Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern Classification, 14th Int. Joint Conf. on Artificial Intelligence (IJCAI), Montreal, Canada.
4. Bala J, De Jong K, Huang J, Vafaie H, and Wechsler H., 1997. Using learning to facilitate the evolution of features for recognizing visual concepts. Evolutionary Computation 4(3) - Special Issue on Evolution, Learning, and Instinct: 100 years of the Baldwin Effect.
5. Blum, L.A., Langley, P., 1997. Selection of relevant features and examples in machine learning. Artifical Intelligence; 97; pp.245-271.
6. Chernova N.I., 1999. Theory of chances. http://text.marsu.ru/books_edu/11/lec.html
7. Ezhov A.A., Shumskiy S.A., 1998. Neurocomputing and its using in economy and business. Moskow.

8. Freitas Alex, 2002. A survey of evolutionary algorithms for data mining and knowledge discovery. Springer-Verlag. In A. Ghosh and S. Tsutsui *editors*, Advances in Evolutionary Computation, chapter 33, pages 819-845.
9. Goldberg D.E., 1989. Genetic Algorithms in search, optimization and machine learning; Addison-Wesley.
10. Georges R. Harik, Fernando G. Lobo, David E. Goldberg, 1998. The compact Genetic Algorithm //http://citeseer.ist.psu.edu/harik98compact.html
11. Gruau F., and D. Whitley, 1993. Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect, Evolutionary Computation, Vol.1, No.3, pp. 213-234.
12. Guerra-Salcedo C. and Whitley D., 1999. Feature selection mechanisms for ensemble creation: a genetic search perspective. Data Mining with Evolutionary Algorithms: Research Directions, pp. 13-17, AAAI Press, 18 July.
13. Hattingh, J.M. & Kruger, H.A. 2001. Robust linear models by discarding data and regressors simultaneously, Research Report No FABWI-N-RKW:2001-56.
14. Hsu William H. and Pottenger William M., Weige Michael, Wu Jie. and Yang Ting-Hao, 1999. Genetic algorithms for selection and partitioning of attributes in large-scale data mining problems. Data Mining with Evolutionary Algorithms: Research Directions, pp. 1-6, AAAI Press, 18 July.
15. Khmilovyy S.V., 2005. Internet-traffic forecasting with neural approach. Ukraine: Donetsk; Naukovi Pratsi DONNTU; DONNTU Publishing House.
16. Kwedlo Wojciech and Kretowski Marek, 1998. Discovery of Decision Rules from Databases: An Evolutionary Approach. Proc. 2nd European Symp. on principles of Data Myning and Knowledge Discovery (PKDD-98). Lecture Notes in Artificial Intelligence 1510, 371-378. Springer-Verlag.
17. Minaei-Bidgoli Behrouz, Punch William F. III, 2003. Using Genetic Algorithms for Data Mining Optimization in an Educational Web-based System. GECCO 2003 Genetic and Evolutionary Computation Conference, Springer-Verlag 2252-2263, Chicago, IL.
18. Moerchen Fabian, 2003. Time Series Feature extraction for data mining using DWT and DFT. In Technical Report No. 33, Departement of Mathematics and Computer Science Philipps-University Marburg.
19. Oh II-Seek, Lee Jin-Seon, And Moon Byung-Ro, 2004. Hybrid Genetic Algorithms for Feature Selection. IEEE Transactions on pattern analysis and machine intelligence, vol 26, no.11.
20. PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules, 1994. Lutz Prechelt, Fakultaet fur Informatic, Universitaet Karlsruhe.– Karlsruhe, Germany. ftp.ira.uka.de/pub/neuron.
21. Raimer Michael L., Punch William F., Goodman Erik D., Kuhn Leslie A. and Jain Anil K., 2000. Dimensionality Reduction Using Genetic Algorithms. IEEE Transactions on Evolutionary Computation, ISSN 1089-778X CODEN ITEVF5 , vol. 4, no 2, pp. 164-171 (42 ref.)
22. UCI Machine Learning Data: - University of California, Irvine . – http://mlearn.ics.uci.edu/databases
23. Vafaie, H., and De Jong, K., 1992. Genetic Algorithms as a Tool for Feature Selection in Machine Learning. Proceeding of the 4th International Conference on Tools with Artificial Intelligence, Arlington, VA, P. 200-204. http://citeseer.ist.psu.edu/vafaie92genetic.html
24. Vafaie H and De Jong K. Robust feature selection algorithms. Proc. 1993. IEEE Int. Conf on Tools with AI, 356-363. Boston, Mass., USA. Nov. - 1993.
25. Yang J.H., Honawar V., Feature Subset Selection Using a Genetic Algorithms. IEEE Intelligent Systems.- 1998. - vol. 13, NO. 2. P. 44-49.

Received on 29.01.2010