International Journal of Science, Environment and Technology, Vol. 3, No 2, 2014, 714 – 722
ISSN 2278-3687 (O)

PHONEME RECOGNITION BY APPLICATION OF GENETIC PROGRAMMING

Stanislav Foltán[1] and Juraj Smieško[2]
Žilinská univerzita v Žiline, Fakulta riadenia a Informatiky,
Univerzitná8215/1, 01026 Žilina, Slovakia
Emails: [1]stanislav.foltan@kis.fri.uniza.sk, [2]juraj.smiesko@kis.fri.uniza.sk

Abstract: This paper presents our approach to voice recognition, more precisely the recognition of phonemes extracted from continuous speech. We provide analysis of input data demonstrating a complexity of the solved problem. The complexity rises as we decided to use fuzzy logic for later possibility of implementation with memristors. The solution demonstrated in this paper shows application of genetic programming for searching structures of fuzzy networks performing the recognition.
Keywords: phoneme recognition, genetic programming, PCA, fuzzy systems.

1. INTRODUCTION

The objective of our work is to create a network of fuzzy flip-flops used for speech recognition [1]. This network should consist of several hierarchical layers: input layer with spectrum of speech, input-fuzzyfication layer, logical layer responsible for speech recognition and layer(s) with fuzzy flip-flops simulating neural activity in a brain (memory, learning). This article is aimed on logical layer and searching structures (by genetic programming) representing functions which could be able to recognize input speech.

[2] presents circuits that are suitable as fuzzy classifiers and suggests memristor circuits for emulation of complex biological neural systems. We believe that memristors could be the key elements of our fuzzy logic network. The decision on their later utilization indicates a restriction for design of the function that we are searching for. If we want to use memristors, the function has to contain only operations minimum, maximum, average and negation.

Later in the article we present input dataset that we are currently using for testing our functions and the way we are searching for the functions by application of genetic programming [3].

1 INPUT DATA ANALYZIS

Our input data is spectra of speech. For initial tests we used dataset for "The Elements of Statistical Learning" [4]. The data were extracted from the TIMIT database (TIMIT Acoustic-

*Received Mar 20, 2014 * Published April 2, 2014 * www.ijset.net*

Phonetic Continuous Speech Corpus, NTIS, US Dept of Commerce) which is a widely used resource for research in speech recognition. A dataset was formed by selecting five phonemes for classification based on digitized speech from this database. The phonemes are transcribed as follows: "sh" as in "she" (872 samples), "dcl" as in "dark" (757 samples), "iy" as the vowel in "she" (1163 samples), "aa" as the vowel in "dark" (695 samples), and "ao" as the first vowel in "water" (1022 samples). From continuous speech of 50 male speakers, 4509 speech frames of 32ms duration were selected, approximately 2 examples of each phoneme from each speaker. Each speech frame is represented by 512 samples at a 16kHz sampling rate, and each frame represents one of the above five phonemes. From each speech frame, a log-periodogram was computed. Thus the final data consist of log-periodograms of length 256.

At data analysis, we can consider our 256-dimensional data as one point in 256-dimensional space. In total, we are dealing with 4509 points in 256-dimensional space. All data are summarized in the matrix$\mathbf{X}_{4509x256}$. This matrix can be understood as 4509 realizations of random vector $\boldsymbol{x}(\omega)$ with coordinates representing random variables $X_i(\omega)$.Our aim is to rotate the space to see the greatest dispersions in data, in other words, we are searching for mutually independent, linear combinations of random variables $X_i(\omega)$. We denote these combinations $C_i(\omega)$and name them main components. In terms of linear algebra, we are trying to define new basis for our 256-dimensional space.

The matrix of our new basis is denoted **U**, the projection of the random vector **x**(ω) into a new basis **c**(ω), **x**(ω) = **c**(ω)**U**.

$$\mathbf{x}(\omega) = \sum_{k=1}^{n} X_k(\omega)\mathbf{e}_k = \sum_{k=1}^{n} C_k(\omega)\mathbf{u}_k = \left(C_1(\omega), \dots, C_k(\omega)\right) \cdot \begin{pmatrix} \mathbf{u}_k \\ \vdots \\ \mathbf{u}_k \end{pmatrix} = \mathbf{c}(\omega)\mathbf{U}$$

Random variables $C_i(\omega)$ are mutually independent, covariance matrix of the vector **x**(ω) is:$\mathbf{R} = \mathbf{U}^T\mathbf{\Lambda}\mathbf{U}$:

$$\mathbf{R} = \mathrm{E}[\mathbf{x}(\omega)^T\mathbf{x}(\omega)] = \mathrm{E}[(\mathbf{c}(\omega)\mathbf{U})^T(\mathbf{c}(\omega)\mathbf{U})] = \mathrm{E}[\mathbf{U}^T\mathbf{c}(\omega)^T\mathbf{c}(\omega)\mathbf{U}] =$$
$$= \mathbf{U}^T\mathrm{E}[\mathbf{c}(\omega)^T\mathbf{c}(\omega)]\mathbf{U} = \mathbf{U}^T\mathbf{\Lambda}\mathbf{U}$$

Covariance matrix **Λ** is diagonal matrix of variables' dispersions $\lambda_k=DC_k(\omega)$. We know from the Principal Component Analysis (PCA), that the searched basis is the basis of eigenvectors of the covariance matrix **R** forming orthonormal basis of the space. Therefore, the basis matrix $\mathbf{U}^{-1} = \mathbf{U}^T$.

For random vectors we obtain the relation

$$\mathbf{x}(\omega) = \mathbf{c}(\omega)\mathbf{U} \implies \mathbf{c}(\omega) = \mathbf{x}(\omega)\mathbf{U}^T$$

Due to orthonormality of the basis of eigenvectors is also true

$$\mathbf{R} = \mathbf{U}^{\mathrm{T}}\mathbf{\Lambda}\mathbf{U} \implies \mathbf{R}\mathbf{U}^{\mathrm{T}} = \mathbf{U}^{\mathrm{T}}\mathbf{\Lambda}\mathbf{U}\mathbf{U}^{\mathrm{T}} \implies \mathbf{R}\mathbf{U}^{\mathrm{T}} = \mathbf{U}^{\mathrm{T}}\mathbf{\Lambda} \implies \forall \mathrm{k}; \mathbf{R}\mathbf{u}_{\mathrm{k}}^{\mathrm{T}} = \lambda_{\mathrm{k}}\mathbf{u}_{\mathrm{k}}^{\mathrm{T}}$$

Dispersions λ_k are eigenvalues corresponding to eigenvectors $\mathbf{u}_k$. When we sort eigenvalues descending according to value $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{256}$, the first principal component is linear combination of variables $\mathbf{x}(\omega)\mathbf{u}^{\mathrm{T}}{}_1$

$$\mathbf{x}(\omega)\mathbf{u}_1^{\mathrm{T}} = \mathrm{u}_{1,1}\mathrm{X}_1(\omega) + \cdots + \mathrm{u}_{1,\mathrm{n}}\mathrm{X}_\mathrm{n}(\omega)$$

This is the component with the highest dispersion λ_1:

$$\mathrm{D}[\mathbf{x}(\omega)\mathbf{u}_1^{\mathrm{T}}] = \mathrm{D}\left[\mathrm{C}_1(\omega)\mathbf{u}_1\mathbf{u}_1^{\mathrm{T}}\right] = \mathrm{DC}_1(\omega) = \lambda_1$$

Due to orthonormality of the basis of eigenvectors of the covariance matrix, the principal components are mutually perpendicular and they continuously define directions of the most important dispersions in data. We calculated the eigenvectors and eigenvalues of covariance matrix $\mathbf{R}_{256\mathrm{x}256}$:

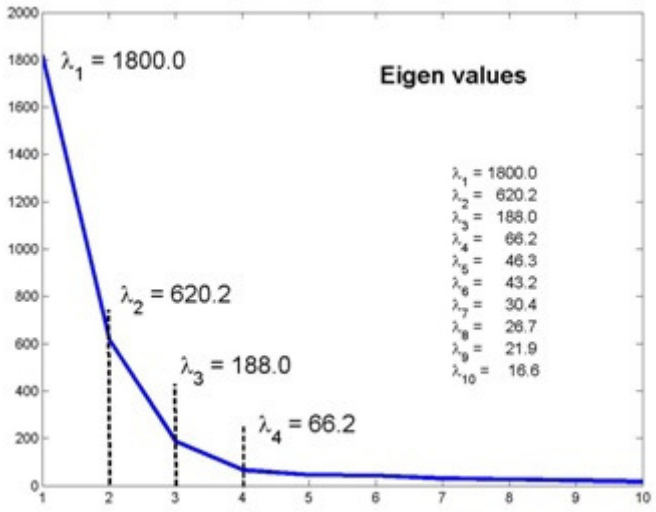

Figure 1: Eigenvalues of covariance matrix $\mathbf{R}_{256\mathrm{x}256}$.

We can see that the first three eigenvalues are significantly different in comparison with following values. The amount of information preserved in the projection of the spectra in 3-dimensional space generated by the basis of the first three eigenvalues are:

$$\frac{\lambda_1 + \lambda_2 + \lambda_3}{\lambda_1 + \ldots + \lambda_{256}} = 76.43\%$$

For better illustration in the article, we will use the projections to 2-dimensional planes, which are formed by ($\boldsymbol{u}_1$,$\boldsymbol{u}_2$), ($\boldsymbol{u}_1$,$\boldsymbol{u}_3$), and ($\boldsymbol{u}_2$,$\boldsymbol{u}_3$). The amount of information preserved in each plane is:

$$(\boldsymbol{u}_1,\boldsymbol{u}_2): \frac{\lambda_1 + \lambda_2}{\lambda_1 + \ldots + \lambda_{256}} = 71.0\% \qquad (\boldsymbol{u}_1,\boldsymbol{u}_3): \frac{\lambda_1 + \lambda_3}{\lambda_1 + \ldots + \lambda_{256}} = 58.4\%$$

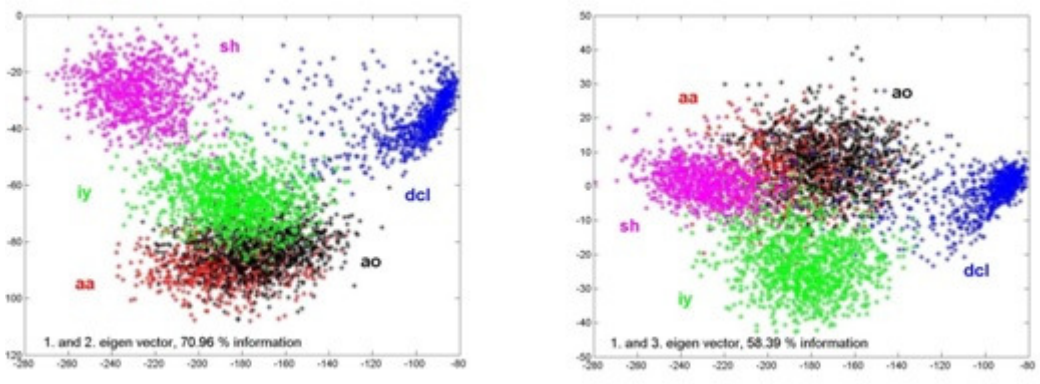

Figure 2: Projection of input data into planes formed by vectors ($\boldsymbol{u}_1$,$\boldsymbol{u}_2$) and ($\boldsymbol{u}_1$,$\boldsymbol{u}_3$)

Projections to the planes formed by vectors ($\boldsymbol{u}_1$,$\boldsymbol{u}_2$) and ($\boldsymbol{u}_1$,$\boldsymbol{u}_3$) show that it is relatively easy to distinguish between the phonemes "sh", "iy" and "dcl".

Possibility of successful recognition between phonemes can be expressed by means of Mahalanobis distance. The distance between given sample **v** and a centre of measured data **s** is defined as

$$d(\mathbf{v},\boldsymbol{s}) = (\mathbf{v}-\mathbf{s})\mathbf{R}^{-1}(\mathbf{v}-\mathbf{s})^{\mathrm{T}}$$

where the matrix **R** is covariance matrix of measured data. In order to identify significant areas for the individual clusters of phonemes we have to calculate eigenvalues and eigenvectors for each phoneme. According to Mahalanobis distance, searched areas will form rotational ellipsoids with semi axes in directions of eigenvectors and length corresponding to multiples of eigenvalues of each cluster.

To determine the size of Mahalanobis areas of individual phonemes we used 2.3 multiple of the size of semi-axes which determine the direction of dispersion in data. At this option only 7.9% of "dcl" samples lies outside of its ellipse. This number is even smaller at the other phonemes. Our experiments confirmed that we can distinguish approximately 92% of"sh", "iy" and "dcl" samples.

Recognition between "aa" and "ao" is not that simple. According to position of ellipses for these two phonemes it is clear that PCA is not able to distinguish between them. This is the reason why in the following text, we chose these two phonemes for demonstration of recognition possibilities of functions found by genetic programming.

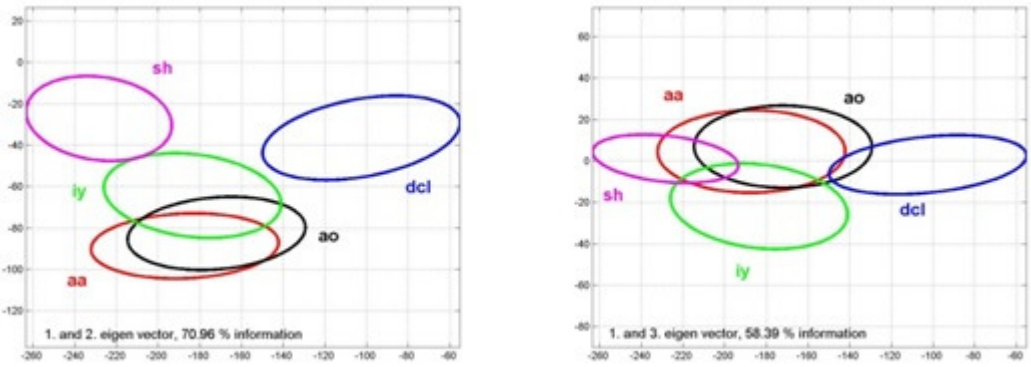

Figure 3: Mahalanobis ellipsoids drawn in planes formed by vectors ($\boldsymbol{u}_1,\boldsymbol{u}_2$) and ($\boldsymbol{u}_1,\boldsymbol{u}_3$).

2. APPLICATION OF GENETIC PROGRAMMING

Genetic programming (GP) is often discussed area; therefore we will focus on our implementation instead of describing it deeply. GP is a branch of Evolutionary strategies (ES). Unlike Genetic algorithms (GA), what we can say is parametric optimalization, or searching for optimal parameters of system of defined structure, at GP, this structure is not defined. That means that we are going to search not only for parameters, but also for the structure of a system.

Algorithm of genetic programming consists of few simple steps:

1. generation of initial population,
2. fitness calculation for each individual,
3. creating a new generation by means of genetic operations over population (selection, mutation, recombination),
4. repeat steps 2 and 3 while stopping criteria is not met.

It is desired to repeat this algorithm several times for getting more objective results.

2.1 Representation of structures

The structure is defined as a tree structure with one or two child nodes. Leaf nodes contain parameters (terminal set - in our case it is index of value from spectrum of a speech (from 0 to 255)), and all other nodes holds one of four functions used for building a structure (function set - *minimum, maximum, average, negation*). One child node is the case when the node represents negation.

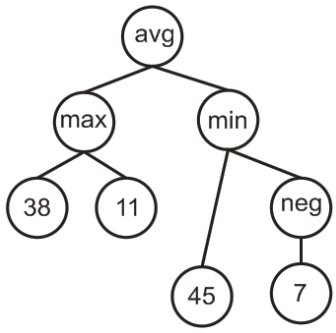

Figure 4: The tree representation of a function: f = avg(max(spectrum[38], spectrum[11]), min(spectrum[45], neg(spectrum[7])))

2.2 Generation of initial population

In our experiments we tested different sizes of initial population. The smallest population consisted out of 10 tree structures, the largest out of 500. Tree structures were generated by 'Ramped half-and-half' method. Half of trees were generated as trees with all leaf nodes of same depth (full binary tree if we had not operation negation, which is unary operation) and other half as random trees where each branch can be terminated in different depth. Initial depth was selected by random between 2 and 7.

2.3 Fitness function(s)

To calculate the success of individuals in generations we used two types of fitness functions that we were trying to maximize. There is expected strong correlation between them, but behavior of program differs slightly depending on which one was used. The first fitness function is:

$$F_A = \frac{(\mu 1 - \mu 2)^2}{\sigma_1^2 + \sigma_2^2}$$

where $\mu_i$is expected value and $\sigma_i^2$is variance of Gaussian curve fitted on results (response of a function) for i-th type of input vector (spectrum), where i=1 for phoneme "aa" and i=2 for phoneme "oo". The second used fitness function $F_B$was directly number of correctly recognized phonemes. In both cases we used 70% of input data as a training set and remaining 30% as a testing set.

2.4 Genetic operations

Genetic operations in our program were implemented as follows. The first operation is the selection. After calculation of fitness for each individual in a generation, the best 20% of individuals go to a new generation. Then tournament based selection is applied and 30% of a current population is selected for a new generation. The rest of a new generation is created by applying the mutation (with probability of 5%) and the recombination (with probability of

95%). At the mutation we randomly select position in a tree and then we change the function in selected node, or index of the input vector if the node is a leaf (5% probability), or we delete whole sub-tree and randomly recreate it again. In case of the recombination, we pick up two trees, then we select a position in each tree and we interchange the sub-trees rooted at the selected position. The tournament selection picks up the better of these new trees into a new generation.

3. RESULTS

At the beginning of our paper we introduced problem that we try to solve. It is basically a creating one layer for more complex, multi-layer, fuzzy logic system. This single layer is responsible for processing input data, what is in our case the speech spectrum of isolated phoneme. Later in the article we presented complexity of this problem for classical analysis at the set of more than 4000 samples of five phonemes. We chose two of them which are the most similar for the experiments with genetic programming that we performed. Figures 5 and 6 show results of function evolution.

Experiment "A" with results from Figure 5 used the fitness F_A introduced in 3.3. Maximization of this fitness results in tend to search for solution at which the responses of function for different phonemes should be explicitly the most distant as possible (in one dimensional space).

Experiment "B" with results from Figure 6 used number of correctly recognized samples from training set (fitness F_B). Unlike the first experiment, this one used fitness directly focused on maximization of number of recognized samples. This resulted in increased number of recognized samples from training set, but not necessarily in such increase at testing data. Figures 7 and 8 prove that there is no direct proportionality between number of recognized samples from training set and testing set.

Table 1 summarizes results that we obtained by genetic programming. To evaluate success of recognition, we can compare results of experiments with recognition abilities of analytic approach introduced in section 2. Functions found by genetic programming recognized more than 75% of testing data, while previous analysis completely failed at recognition between similar phonemes ("aa" and "ao"). We consider these results as a proof that application of genetic programming in our work is appropriate.

Table 1: Results of the experiments stopped after crossing 3500 generations with 100 individuals each. On a system with Dual-Core CPU @2,5GHz, experiment A run about 16 hours and experiment B around 19 hours.

Experiment	Fitness	Correctly recognized samples from training set (1203 samples)	Correctly recognized samples from testing set (514 samples)	μ_1	σ_1	μ_2	σ_2
A	1.1817	978 (81.29%)	**395 (76.85%)**	0.1845	0.00048	0.2252	0.00041
B	1023	1023 (85.04%)	**391 (76.07%)**	0.6328	0.00382	0.5336	0.00247

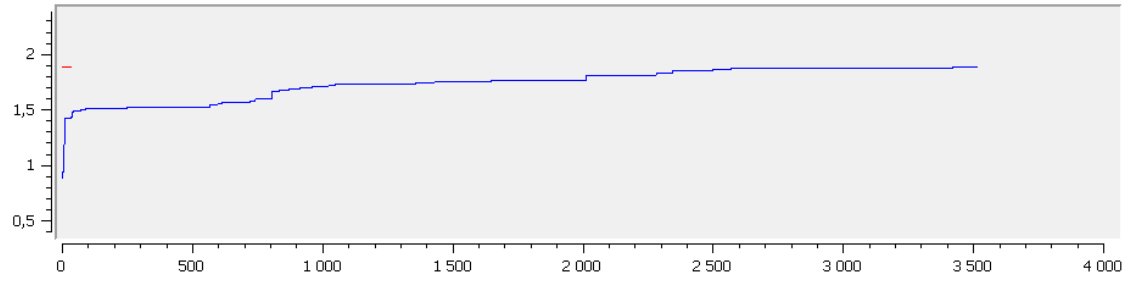

Figure 5: Dependency between generation and fitness F_A.

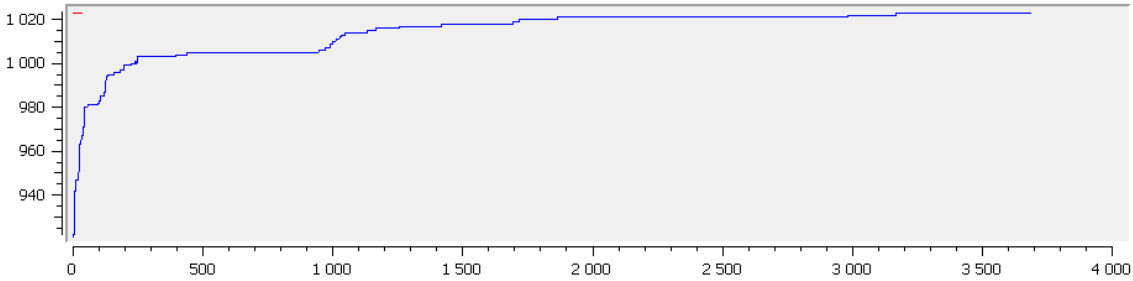

Figure 6: Dependency between generation and fitness F_B.

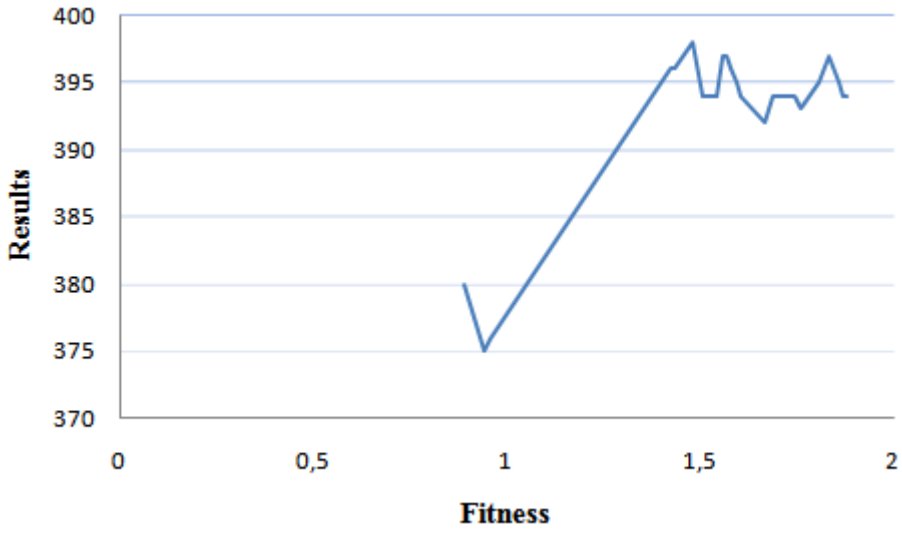

Figure 7: Dependency between fitness $F_A$and number of correctly recognized samples from testing data (maximum number of recognized samples was 398).

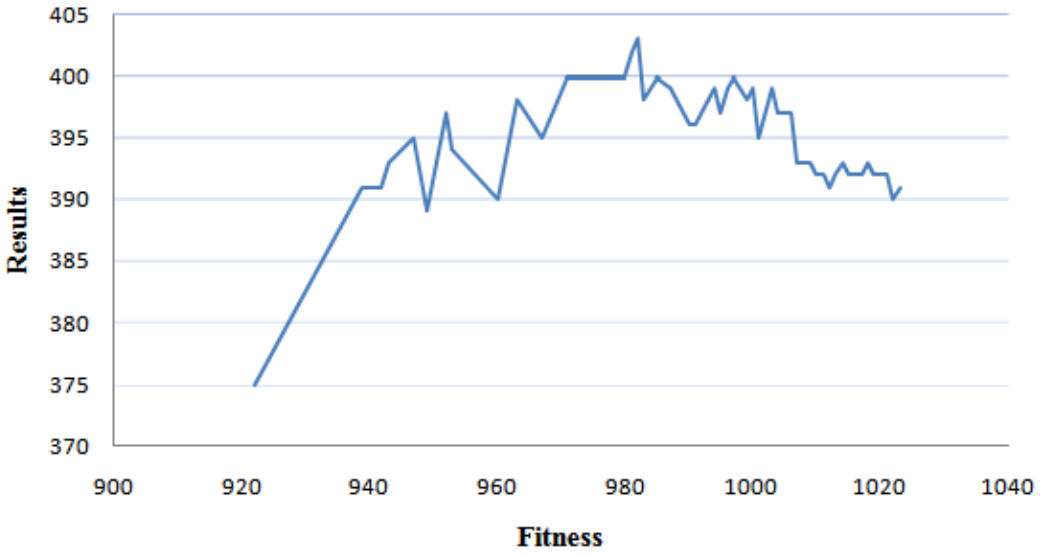

Figure 8: Dependency between fitness F_B and number of correctly recognized samples from testing data (maximum number of recognized samples was 403).

References

[1] FOLTÁN, S. *Network of fuzzy flip-flops used for speech recognition*. Banská Bystrica: Science and Research Institute, 2011. pp. 55-57. ISBN 978-80-557-0252-0.

[2] KLÍMO, M., ŠUCH, O. *Memristors can implement fuzzy logic*. http://arxiv.org/abs/1110.2074v1, 2011

[3] SEKAJ, I. Evolučné výpočty a ich využitie v praxi. IRIS 2005. ISBN 80-89018-87-4

[4] HASIE, Z., TIBSHIRANI, R., FRIEDMAN, J. *The Elementsof Statistical Learning: Data mining, Inference, and Prediction*. 2nd edition, Springer 2009, ISBN-10: 0387848576.

[5] FOLTÁN, S., SMIEŠKO, J. *Car color recognition by means of principal component analysis*. Mendel 2011: 17th international conference on soft computing, 2011. pp. 432-439, ISBN 978-80-214-4302-0.