# Supervised Competition Using Joined Growing Neural Gas

Dusan Fedorcak,
Michal Podhoranyi
{dusan.fedorcak,michal.podhoranyi}@vsb.cz
IT4Innovations Center of Excellence,
VSB - Technical University of Ostrava, 17. listopadu 15/2172, 70833 Ostrava-Poruba, Czech Republic

## ABSTRACT

Competitive learning is well-known method to process data. Various goals may be achieved using competitive learning such as classification or vector quantization. In this paper, we present a different insight into the principle of supervised competitive learning. An innovative approach to the supervised self-organization is suggested. The method is based on different handling of input data labels which encode the classification. When the label has appropriate format then it is possible to use it within the competitive process in the same way as any input data element. Such approach is as effective as standard supervised methods and has some positive attributes such as the soft classification ability.

## KEYWORDS

Competitive Network, Unsupervised Learning, Supervised Learning, Vector Quantization, Growing Neural Gas, Classification

## 1 INTRODUCTION

Artificial neural networks are phenomena of modern computer science. Many research groups are interested in these systems and artificial neural networks are strong computational devices by any mean. Many types of artificial neural networks have been invented (or reinvented) so far and are applied in various scientific or industry areas.

This article is focused on competitive networks taught by supervision. An alternative to common supervised competitive learning is suggested and it is explained how labeled data may be processed within the standard metrics-based competitive learning. There are some learning methods (such as *Learning Vector Quantization*) which can utilize the competitive learning principle but are applied to labeled data. They are called *supervised competitive learning* methods and might be mentioned also as *semi-supervised learning*. The main difference between unsupervised competitive learning methods and supervised competitive learning methods is the structure of the input dataset. As usual, the supervision above the learning process must be based on some information added to the system and such information is a class label assigned to each input signal. The learning process is supervised through these labels and the common result of supervised competitive learning is a classifier trained on given patterns i.e. pairs of input signals and their labels.

Later on this article, a new method of semi-supervised learning is presented. Our research has been inspired be some very interesting ideas about learning from context. The literature suggests that in some cases it is vital to focus on understanding the input dataset and leave the classification problem until later [7]. This leads us to an idea that the pair of input data and its label should be taken as an indivisible object. The joined information may be passed to the network rather than using the label for the output correction, which is the common approach. If this idea is applied to competitive learning, useful results can be obtained. First, a brief description of the standard supervised competitive method (LVQ) will be given, therefore the difference between the standard approach and our method can be stated more easily.

## 2 COMPETITIVE LEARNING

Competitive learning, as one of the unsupervised learning representatives, is a process of learning the organization of the input dataset through a competition between neurons. Neurons are often defined as reference vectors with the same dimensionality as the input data. The competition is usually driven by some metric, for example the Euclidean metric. The learning process consists of a fixed number of repetitive steps. Every step contains several substeps:

1. Determination of the winner according to the metric
2. Adaptation of the winner using some pseudo-hebbian adaptation rule
3. Optionally, if some sort of topology is defined, the neighborhood of the winner may be adapted as well.

Most of the competitive learning methods use a very similar learning step [5]. Sometimes, the network is

able to change its structure dynamically by adding or removing neurons (*growing networks*) [6]; or some other attributes such as a local error may be used for learning purposes. The determination of the winner is usually defined as:

$$\mathbf{w}_w = \arg\min_{u \in \mathcal{N}} \|\xi - \mathbf{w}_u\| \tag{1}$$

where $A$ is the set of neurons within the network, $\mathbf{w}_w$ is the reference vector of the winning neuron and $\xi$ is the input signal passed to the network. The Hebbian adaptation is defined as:

$$\Delta \mathbf{w}_w = \eta(\xi - \mathbf{w}_w) \tag{2}$$

where $\eta$ is the learning factor which will affect the speed of adaptation.

### 2.1 Learning Vector Quantization

*Learning Vector Quantization* [1] is a prototype-based classification algorithm. First a vector quantization of a given dataset has to be found. The idea of vector quantization is very simple. It is finding a set of representatives or *code-book vectors* that provide a good approximation of the original input dataset. Almost all competitive networks work with neurons with weight vectors distributed in the input dataset. In fact, if we think of reference vectors as code-book vectors, then all competitive networks perform vector quantization. The aim of the LVQ method is to correct the competitive learning process, so that it gives better results according to the label information stored in the training set.

There are several somewhat different LVQ algorithms appearing in the literature [9] but they are all based on similar rules. The structure of the LVQ network is created by two layers. The first layer is a competitive layer and can even have, for example, SOM topology. The second layer is the output layer with a number of neurons equal to the number of classes in the training set. The output layer performs mapping from competitive units to classes defined in the training set.

Generally, there are two stages of the LVQ learning process. The first stage is pure competitive learning where the competitive layer is involved. The result of the first stage is neuron position (i.e. values of their weights vectors). These neurons will describe clusters in the input space where input signals are situated. The second stage sets the weights of neurons in the output layer to perform desired class mapping. Also, positions of competitive neurons could be changed within this stage if new positions suit the classification better.

The algorithm described above is denoted as the LVQ1 algorithm. There are some improved versions of this algorithm denoted as LVQ2.1 and LVQ3. For example the LVQ2.1 algorithm has a different adaptation step, where two closest neurons need to be found and they are adapted together but only when a special condition of different classes is met (i.e. if the closest neuron belongs to the desired class and the second one does not). Also, the ending condition may be altered to stop the algorithm when no adaptation was done after the whole training set has been processed.

## 3 JOINED GROWING NEURAL GAS

As it was mentioned in the section above, there are competitive models which can use labeled data for training and performing the classification task. Most models use labels associated with input signals for correction of the network output. Our approach is different. The label may be understood as a part of input data. The learning algorithm can remain exactly the same and the network organizes itself as usual. If the *one-of-C* coding is used for class labels, the self-organization has some interesting properties and can be used for classification directly.

It is possible to choose any competitive network for self-organization of input data created by joined information of input signal and class attributes. We have chosen the *Growing Neural Gas* [6] method for its good abilities in self-organization. The algorithm of learning the joined dataset can be described as follows:

1. Create a set $\mathcal{J} = \{\mathbf{j}_1, \mathbf{j}_2, \ldots, \mathbf{j}_p\}$. Every vector $\mathbf{j}_i$ contains joined information of input signal $\xi$ and binary class vector $\mathbf{c} = (c_1, c_2, \ldots, c_k)$ imported from each training pattern $\mathbf{p} \in \mathcal{T}$:

$$\begin{aligned} \mathbf{p} &= ((\xi_1, \xi_2, \ldots, \xi_n),\ (c_1, c_2, \ldots, c_k)) \ \rightarrow \\ \mathbf{j} &= (\xi_1, \xi_2, \ldots, \xi_n,\ c_1, c_2, \ldots, c_k) \end{aligned} \tag{3}$$

2. Initialize the GNG method. Use reference vectors $\mathbf{w}_u \in \mathbb{R}^{n+k}$.

3. Perform the standard GNG learning algorithm with a suitable learning factor etc.

When the learning phase is finished, the network can perform the classification task almost directly. Because the network has reference vectors of different dimensionality $(n + k)$ than inputs signals $(n)$, some transformations need to be done. This transformation will be discussed later. First, the effect of the joined input space will be shown.

### 3.1 Properties of joined input space

The learning algorithm of the GNG model builds a graph of connected neurons. The structure of this graph depends on training data distribution. When the input dataset is extended by class attributes, the distribution

gets changed. If two inputs are assigned to the same class, the distance between them will not change because class vectors are equal. If two inputs have different class attributes, the distance between them will be enlarged by $\sqrt{2}$ because the one-of-C coded class vectors will differ on exactly two places.

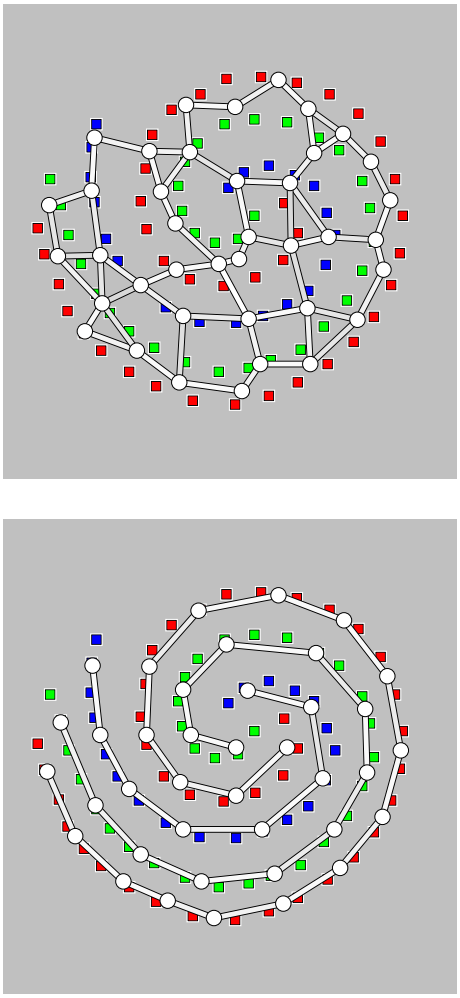

Figure 1: The difference between the standard GNG model (first) and the GNG model trained with a joined dataset (second): The training set contains two-dimensional points distributed as concentric spirals. There are three different classes in the training set denoted by colors. The standard model has two-dimensional reference vectors; the extended model has five-dimensional reference vectors (2 spatial attributes + 3 class attributes)

The normalization of input data is crucial at this point. If data is not normalized, then the importance of added distance may be significantly decreased. Unfortunately, that original dimension of the input vector will also affect the significance of added distance. If we take an arbitrary normalized input dataset of dimension $n$, then the maximum distance between any two input signals is $\sqrt{n}$. For example, if we take a 200-dimensional dataset, then maximum distance is ten times bigger than distance between two distinct classes.

However, experiments presented below show that the added distance is enough even for high-dimensional input data if the normalization is used. The figures above (fig. 1) show the difference between the standard GNG model trained with unlabeled patterns and the GNG model trained with joined dataset of inputs and labels (the color of the input signal). The difference is clearly visible. The added distance is enough to limit connections between reference vectors with different class attributes.

### 3.2 Building a classifier – Applying restricted queries

Usually, when the classification task is performed by an unsupervised competitive network, the resulting reference vectors need to be interpreted i.e. every neuron is labeled manually[1]. After the interpretation is done, the classification runs smoothly. The testing input signal is passed to the network and the winner neuron (the closest one) is found. The testing input signal is then classified according to the class label of the winner neuron.

The prediction phase of our classifier is different because the network was trained with joined training patterns. There are more network inputs than the number of elements in the testing input signal. We may think of an *Auto-associative Memory* model [8], where the incomplete information is passed to the network and the network will reconstruct missing values. We used the same idea in our classifier. First, all reference vectors have to be projected from joined space back to the input space. The projection is trivial; all class attributes of every reference vector $\mathbf{w}_u$ will be ignored. Formally, this is done by multiplying reference vectors by matrix $\mathbf{P}$:

$$\forall u \in \mathcal{N}\ ; \quad \mathbf{w}'_u = \mathbf{w}_u \cdot \mathbf{P} \tag{4}$$

$$\mathbf{P}_{n+k,n} = \begin{pmatrix} \mathbf{I}_{n,n} \\ \mathbf{0}_{k,n} \end{pmatrix} \tag{5}$$

where $\mathbf{I}_{n,n}$ is an identity matrix and $\mathbf{0}_{k,n}$ is a zero matrix.
Now, we can use a common strategy to find the winner $\mathbf{w}'_c$ between the restricted reference vectors:

$$\mathbf{w}'_w = \arg\min_{u \in \mathcal{N}} \|\xi - \mathbf{w}'_u\| \tag{6}$$

The output of the network in the predictive phase is constructed from the original reference vector of the winner i.e. the output vector $\mathbf{y} = (y_1, ..., y_k)$ is filled with class attributes of $\mathbf{w}_w$:

$$\mathbf{y} = \mathbf{w}_w \cdot \mathbf{L} \tag{7}$$

[1]Naturally, supervised competitive models such as LVQ and current model do the labeling of neurons automatically

ISBN: 978-0-9891305-5-4 ©2014 SDIWC

$$\mathbf{L}_{n+k,n} = \begin{pmatrix} \mathbf{0}_{n,n} \\ \mathbf{I}_{k,n} \end{pmatrix} \quad (8)$$

In this way, it is possible to build a classifier from almost any competitive model. In the following text, the GNG model extension will be called joined GNG or JGNG model.

### 3.3 Class Probability Prediction

As it was presented in the previous section, resulting neuron positions produced by the JGNG algorithm trained on the joined dataset (eq. 3) can be used for predictions. Due to the nature of the output vector (actually, it is a trimmed weight vector), the classifier produces output values greater than 0 and lesser than 1, i.e. the classifier makes *soft decisions*. This is highly demanded and classifiers manifesting such behavior are generally considered better. However, there is another request for soft-decisioning classifiers and it is called *probabilistic classification*.

The idea of probabilistic classification is very simple and it means that the classifier results can be understood as a posterior probability estimate for predicted classes. It is obvious that there is a condition for the output of such classifier; the sum of output values has to be 1. This condition can be met by several methods (for example MLP networks may use the *soft-max function* as the transfer function for output units [3]) but the JGNG classifier does not need any output transformation at all. The rest of this section is focused on proving the fact that the JGNG classifier makes probabilistic classifications.

The condition for probabilistic classification is $\|\mathbf{y}\|_1 = 1$ where $\mathbf{y} \in \mathbb{R}^k$ and $\mathbf{y} = (w_{n+1}, w_{n+2}, \ldots, w_{n+k})$ (eq. 7) i.e. the output must lie on the hyperplane $\rho$:

$$\rho : w_{n+1} + w_{n+2} + \ldots + w_{n+k} = 1 \quad (9)$$

The initial position of any neuron is given by its weight vector, which is initialized at random. In general, the output vector lies on $\rho$ or is inside the subspace $X$ or $X'$:

$$X \quad : \quad w_{n+1} + w_{n+2} + \ldots + w_{n+k} > 1 \quad (10)$$

$$X' \quad : \quad w_{n+1} + w_{n+2} + \ldots + w_{n+k} < 1 \quad (11)$$

The adaptation of the weight vector of the winner is defined as (see eq. 2 and 3):

$$\begin{aligned} \Delta\mathbf{w} &= \eta(\mathbf{j} - \mathbf{w}) \\ \mathbf{w} &= \mathbf{j} - \frac{\Delta\mathbf{w}}{\eta} \end{aligned} \quad (12)$$

where $\mathbf{j}$ is a random pattern chosen from the training set and the learning factor is $0 < \eta < 1$. If any neuron is inside $X$, the following inequation is fulfilled:

$$\begin{aligned} \left(j_{n+1} - \frac{\Delta w_{n+1}}{\eta}\right) + \ldots + \left(j_{n+k} - \frac{\Delta w_{n+k}}{\eta}\right) &> 1 \\ j_{n+1} + \ldots + j_{n+k} - \frac{\Delta w_{n+1} + \ldots + \Delta w_{n+k}}{\eta} &> 1 \end{aligned}$$

With the help of one-of-C coding ($j_{n+1} + j_{n+2} + \ldots + j_{n+k} = 1$ for any training pattern) we may continue to:

$$\frac{1}{\eta} \sum_{i=n+1}^{n+k} \Delta w_i < 0 \quad (13)$$

We are almost finished because vector $\Delta\mathbf{w}^{out} = (\Delta w_{n+1}, \ldots, \Delta w_{n+k})$ denotes the change of the neuron's position in the subspace from which the output vector $\mathbf{y}$ is generated (eq. 7). In other words, the inequation above (eq. 13) tells us that the scalar product of $\Delta\mathbf{w}^{out}$ and the vector $\mathbf{n} = (1, 1, \ldots, 1) \in \mathbb{R}^k$ is negative ($\eta \in \mathbb{R}^+$). Obviously, the vector $\mathbf{n}$ is the normal vector of the hyperplane $\rho$. The negative scalar product and the fact that $\mathbf{w} \in X$ means that the vector $\Delta\mathbf{w}^{out}$ is **pointing towards** the hyperplane $\rho$. If we change the position of the weight vector $\mathbf{w} \in X'$ (eq. 11) i.e. the neuron lies "under" the hyperplane $\rho$, the scalar product $\Delta\mathbf{w}^{out} \cdot \mathbf{n}$ become positive. Again, this means that the change vector is pointing towards hyperplane $\rho$.

We showed that for any valid training pattern and any initial position of neurons, all neurons will converge to positions which meets the condition for probabilistic classification. Of course, there some other conditions (e.g. a reasonable number of iterations of the GNG algorithm or a suitable value of learning factor $\eta$) but in practice, these can be easily achieved.

## 4 EXPERIMENTAL RESULTS

The following section present some experiments processed with the JGNG method as they are compared to the standard LVQ method.

### 4.1 Concentric spirals dataset

The first experiment has been run on the concentric spirals dataset already presented above. The advantage of this dataset is that it is possible to visualize the result of classification in two dimensions. Results are shown in the pictures below (fig. 2).

### 4.2 MNIST database

The dataset we used for another experiment is the popular MNIST hand-written digit database [10]. The database contains 60,000 gray scale images. Every image is a 28x28 square grid with the picture of a hand-written digit. All digits are centered and size-normalized. Every image is labeled with the number of

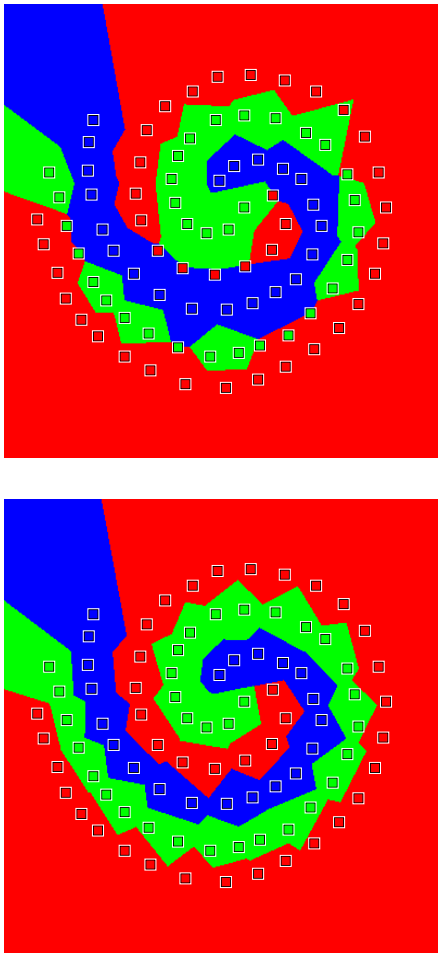

Figure 2: The difference between the standard LVQ classifier (first fig.) and the JGNG classifier (second fig.). The training set contains two-dimensional points (squares) distributed as concentric spirals. There are three different classes in the training set denoted by color. Colored areas show predicted class values of an arbitrary point within the input space.

a digit written on the image. There was no preprocessing in my experiment. By contrast, the real classifier will use some features extracted from raw pixel data, but we want to test our method and find out if it can be applied to high-dimensional non-transformed input space. Again, the LVQ classifier was run on the same dataset for comparison purposes. Results are shown in the following table (tb. 1):

Results show that our classifier is slightly better than the standard LVQ method. The improvement is not as significant as in the spiral dataset experiment but there are some other interesting attributes. The most important is the difference between the form of an answer by each classifier. While the LVQ method does the "hard" classification of the given input, the JGNG model performs soft classification. The following pictures show such behavior in some uncertain cases:

Table 1: Experimental Results: Experiments were performed on the training part of the MNIST dataset. The training sequences were run 50 times for every configuration to lower the randomness of the initial neuron distribution. The JGNG model configuration was tuned to have the same final number of neurons as the LVQ method. Error values were computed from cross testing where 90% of patterns are used for the training process and the rest are used for testing purposes. Mean error value donates how many test patterns have been incorrectly classified.

| **Net.** | **Neur.** | **Iter.** | **MSE %** | **SD** |
|---|---|---|---|---|
| LVQ | 50 | 20,000 | 18.38 | 2.23 |
| JGNG | 50 | 20,000 | 15.21 | 1.56 |
| LVQ | 100 | 40,000 | 15.10 | 2.96 |
| JGNG | 100 | 40,000 | 10.05 | 1.45 |

4 . . . 0.32
7 . . . 0.58
9 . . . 0.09

3 . . . 0.03
5 . . . 0.07
8 . . . 0.63
9 . . . 0.25

2 . . . 0.40
3 . . . 0.56

Figure 3: Uncertain Cases in JGNG driven MNIST dataset; The reference vector position in the extended input space will produce soft answers (class probability) where the highest value may denote the class prediction. Low values within class vectors were omitted.

### 4.3 Intrusion detection dataset

The next experiment was run on intrusion detection dataset. The original dataset was prepared by the 1998 DARPA Intrusion Detection Evaluation Program by MIT Lincoln Labs [11]. The dataset contains 24 attack types that could be classified into four main categories, namely *Denial of Service* (DOS), *Remote to User* (R2L), *User to Root* (U2R) and *Probing*. The original data contains 744 MB data with 4,940,000 records. The dataset contains 41 attributes for each record plus one class label. There are various features in every record like features examining past connections to the same host or failed logins etc. The class label was transformed to the one-of-C class vector (41 original input variables plus 5 class variables where the first class denotes a non-attacking record). The following table (tbl. 2) shows the results of this experiment:

Another interesting feature of the GNG method is the possibility to visualize the structure of the network and information about the dataset distribution can be obtained through this structure if appropriate visualiza-

ISBN: 978-0-9891305-5-4 ©2014 SDIWC

Table 2: Experimental Results: The experiments were performed on the training part of the DARPA intrusion detection dataset. The training sequences were run 100 times for every configuration to lower the randomness of the initial neuron distribution. The rest of configuration is the same as in the experiment before.

| Net. | Neur. | Iter. | MSE % | SD |
|---|---|---|---|---|
| LVQ | 25 | 20,000 | 13.66 | 3.06 |
| JGNG | 25 | 20,000 | 6.47 | 3.08 |
| LVQ | 75 | 60,000 | 6.56 | 1.31 |
| JGNG | 75 | 60,000 | 2.27 | 1.19 |

tion method is used. [4] Moreover, some other techniques of visualization has been used in this experiment. For example, class vectors are visualized as circles divided into colored arcs where every arc denotes one class variable and the radius of the arc is equal to the probability value.

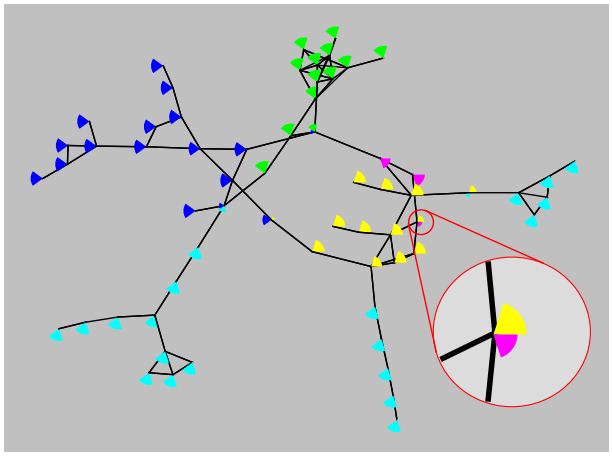

Figure 4: Structure of the JGNG network trained to detect and classify intrusions. The network is projected into two-dimensions and colored arcs denote class part of each weight vector. The detail view shows one neuron which produces soft classification. If such neuron is the winner of the competition in the predicting stage, then the prediction will contain two most probable classes.

## 5 CONCLUSION

An innovative approach to the classification task done by competitive network has been presented. The innovation lies in joining the input data with the desired class information. Usually, the supervised training uses the input data for the adaptation of network weights but desired value (or values) are used in terms of correcting such adaptation process. The joined approach uses all input and desired output data together as a single joined input vector. The desired output has to be a one-of-C coded class vector but this is very common for classification purposes. Such joined vector is passed to the network in the learning stage and no other supervision is needed.

Additionally, we have proved that the output of the network fulfills the condition of probabilistic classification without any correction. Moreover, the extended weight space gives the network more freedom in the learning phase and the final distribution of neurons can be much better than in the standard case, especially on boundaries between different classes of inputs (see fig. 1). Such outcome has been experimentally proven on several datasets. The MNIST experiment shows the classification on real data with high dimension and it is very clear that our assumptions about the soft classification were correct. All experiments were done with the joined modification of the known Growing Neural Gas model developed by B. Fritzke.

### Future Work

One of the known and useful applications of the competitive network is the weight initialization of the RBF network. The RBF network contains so-called local neurons with a kind of Radial Basis Function as their threshold function [2]. We are thinking of possible application of the presented Joined Growing Neural Gas method and we believe that the boundary preserving behavior of the joined input approach can result in better distribution of the RBF layer and can speed up the ongoing back propagation learning process of the output layer. Moreover, we suggest additional initialization of the *output layer* of the RBF network. The idea is that the output of the joined network is not only the distribution of units, but that there is also the output classification vector which can be used as a template for output units of the RBF network. Of course, experiments must be provided to support the premise that such initialization steps will speed up the fine tuning driven by back propagation.

## ACKNOWLEDGEMENT

This article has been elaborated in the framework of the project New creative teams in priorities of scientific research, reg. no. CZ.1.07/2.3.00/30.0055, supported by Operational Programme Education for Competitiveness and co-financed by the European Social Fund and the state budget of the Czech Republic.

## REFERENCES

[1] Ahalt S. C., Krishnamurthy A. K., Chen P., Melton D. *Competititve learning algorithms for vector quantization.* Neural Networks, 1990

[2] Buhmann, M. D. *Radial basis functions: theory and implementations.* Cambridge University Press, 2003.

[3] Bridle, J. S. *Probabilistic interpretation of feedfor-*

*ward classification network outputs, with relationships to statistical pattern recognition.* Neurocomputing: Algo-rithms, Architectures and Applications, pages 227236, 1990.

[4] Fedorcak, D., Vondrak, I, *Force-based Visualization for Competitive Learning Methods.* in conference proceeding ESM 2007. Ghent: EUROSIS. 2007. 331-335. EUROSIS. 978-90-77381-36-6

[5] Fritzke B., *Some Competitive Learning Methods.* `http://www.neuroinformatik.ruhr-unibochum.de/ini/VDM/research/gsn/JavaPaper/`

[6] Fritzke B., *A Growing Neural Gas Network Learns Topologies* in Advances in Neural Information Processing Systems 7, pages 625-632. MIT Press, Cambridge MA, 1995.

[7] Hinton, G. E. and Salakhutdinov, R. R. *Reducing the dimensionality of data with neural networks.* Science, 313(5786):504507, July 2006.

[8] Hopfield, J. J. *Neural networks and physical systems with emergent collective computational abilities*, in proceedings of the National Academy of Sciences of the USA, vol. 79 no. 8 pp. 2554-2558, April 1982.

[9] Kohonen, T., *New developments of learning vector quantization and self-organizing map.* In Symposium on Neural Networks, Osaka, Japan, 1992.

[10] LeCun, Y. *The mnist database of handwritten digits.* `http://yann.lecun.com/exdb/mnist`, 1998.

[11] MIT Lincoln Laboratory Cyber Systems and Technology. *Darpa intrusion detection data sets.* `http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html`, 1998.