

# MLP and SVM Networks – a Comparative Study

*Stanislaw Osowski<sup>1,2</sup>, Krzysztof Siwek<sup>1</sup> and Tomasz Markiewicz<sup>1</sup>*

<sup>1</sup>Warsaw University of Technology  
 Institute of the Theory of Electrical Engineering, Measurement and Information Systems  
 00-661 Warsaw, pl. Politechniki 1  
<sup>2</sup>Military University of Technology  
 POLAND  
 Tel. +48-22-660-7235  
 E-mail : sto@iem.pw.edu.pl

## ABSTRACT

*The paper presents the comparative analysis of two most important neural networks: the multilayer perceptron (MLP) and Support Vector Machine (SVM). The most effective learning algorithms have been implemented in a uniform way using Matlab platform and undergone the comparison with respect to the complexity of the structure as well as the accuracy and the calculation time for the solution of different learning tasks, including classification, prediction and regression. The results of numerical experiments are given and discussed in the paper.*

## 1. INTRODUCTION

An artificial neural network (ANN) is an abstract computational model of the human brain. Similar to the brain ANN is composed of artificial neurons, regarded as the processing units, and the massive interconnection among them. It has the unique ability to learn from the examples and to generalize, i.e., to produce the reasonable outputs for new inputs not encountered during a learning process. The distinct features of ANN are as following: learning from examples, generalization ability, non-linearity of processing units, adaptativity, massive parallel interconnection among processing units and fault tolerance.

The neural networks may be regarded as the universal approximators of the measured data in the multidimensional space. They realize two types of approximation: the global and local one. The most important example of global network is the multilayer perceptron (MLP), employing the sigmoidal activation function of neurons. In MLP the neurons are arranged in layers, counting from the input layer (the set of input nodes), through the hidden layers, up to the output layer. The interconnections are allowed only between two neighbouring layers. The network is feedforward, i.e., the processing signals propagate from input to the output side.

The most representative example of local neural network is the Support Vector Machine (SVM), of the Gaussian kernel function. It is a two layer neural network employing hidden layer of radial units and one output neuron. The procedure of creating this network and

learning its parameters is organized in the way in which we deal only with kernel functions instead of direct processing of hidden unit signals.

This paper will summarize and compare these two networks: MLP and SVM. The comparison will be done with respect to the complexity of the structure as well as the accuracy of results for the solution of different learning tasks, including classification, prediction and regression problem. Special emphasis will be given to the generalization ability of the learned structures acquired in different learning processes.

## 2. GRADIENT LEARNING ALGORITHMS OF MLP

The learning process of MLP network is based on the data samples, composed of the N-dimensional input vector  $\mathbf{x}$  and the M-dimensional desired output vector  $\mathbf{d}$ , called destination. By processing the input vector  $\mathbf{x}$  the MLP produces the output signal vector  $\mathbf{y}(\mathbf{x}, \mathbf{w})$ , where  $\mathbf{w}$  is the vector of adapted weights. The error signal produced actuates a control mechanism of the learning algorithm. The corrective adjustments are designed to make the output signal  $y_k$  ( $k=1, 2, \dots, M$ ) to the desired response  $d_k$  in a step by step manner.

The learning algorithm of MLP is based on the minimization of the error function defined on the learning set  $(\mathbf{x}_i, \mathbf{d}_i)$  for  $i=1, 2, \dots, p$  using an Euclidean norm

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^p \|\mathbf{y}(\mathbf{x}_i, \mathbf{w}) - \mathbf{d}_i\|^2 \quad (1)$$

The minimization of this error leads to the optimal values of weights. The most effective methods of minimization are the gradient algorithms, from which the most effective is the Levenberg Marquard algorithm for medium size networks and conjugate gradient for large size networks. Generally in all gradient algorithms the adaptation of weights is performed step by step according to the following scheme

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta \mathbf{p}(k) \quad (2)$$

In this relation  $\mathbf{p}(k)$  is the direction of minimization in  $k$ th step and  $\eta$  is the adaptation coefficient. Various learning methods differ in the way the value of  $\mathbf{p}(k)$  is generated.

In Levenberg-Marquardt approach the least square formulation of learning problem is exploited  $E(\mathbf{w}) = 0.5 \sum_{i=1}^M (y_i(\mathbf{w}) - d_i)^2$  and solved by using second order method of Newton type

$$\mathbf{p}(k) = -\mathbf{G}(k)^{-1} \mathbf{g}(k) \quad (3)$$

where  $\mathbf{g}(k) = \frac{\partial E}{\partial \mathbf{w}(k)}$  is the gradient of error function (1)

and  $\mathbf{G}(k)$  – the approximated Hessian, determined by applying the Jacobian matrix  $\mathbf{J}(k)$

$$\mathbf{G}(k) = \mathbf{J}(k)^T \mathbf{J}(k) + \nu \mathbf{I} \quad (4)$$

In this equation the Jacobian matrix  $\mathbf{J}$  is equal  $\mathbf{J} = \frac{\partial \mathbf{e}}{\partial \mathbf{w}}$ ,

and  $\mathbf{e} = [y_1(\mathbf{w}) - d_1, \dots, y_M(\mathbf{w}) - d_M]^T$ . The variable  $\nu$  is the Levenberg-Marquard parameter adjusted step by step in a way to provide the positive definiteness of Hessian  $\mathbf{G}$  (the value of  $\nu$  is eventually reduced to zero).

In conjugate gradient approach, most effective for large networks, the direction  $\mathbf{p}$  is evaluated according to the formula

$$\mathbf{p}(k) = -\mathbf{g}(k) + \beta \mathbf{p}(k-1) \quad (5)$$

where the conjugate coefficient  $\beta$  is usually determined according to the Polak-Ribiere rule

$$\beta = \frac{\mathbf{g}(k)^T (\mathbf{g}(k) - \mathbf{g}(k-1))}{\mathbf{g}(k-1)^T \mathbf{g}(k-1)} \quad (6)$$

In the weight update equation (2) the learning coefficient  $\eta$  should be adjusted by the user. It is done usually by applying so called adaptive way [7], taking into account the actual progress of minimization of the error function.

### 3. SUPPORT VECTOR MACHINE NETWORK

Support Vector Machine (SVM) is a linear machine working in the highly dimensional feature space formed by the nonlinear mapping of the N-dimensional input vector  $\mathbf{x}$  into a K-dimensional feature space ( $K > N$ ) through the use of a mapping  $\phi(\mathbf{x})$ . The way in which SVM network is created differs for the classification and regression tasks [5,6], although both transform the learning task to the quadratic problem.

#### 3.1 Classification mode

In the classification mode the equation of the hyperplane separating two different classes is given by the relation

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \sum_{j=1}^K w_j \phi_j(\mathbf{x}) + w_0 = 0, \quad \text{where the vector}$$

$\phi(\mathbf{x}) = [\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \dots, \phi_K(\mathbf{x})]^T$  is composed of activation functions of hidden units with  $\phi_0(\mathbf{x}) = 1$  and

$\mathbf{w} = [w_0, w_1, \dots, w_K]^T$  is the weight vector of the network.

The most distinctive fact about SVM is that the learning task is reduced to quadratic programming by introducing

the so-called Lagrange multipliers  $\alpha_i$ . All operations in learning and testing modes are done in SVM using kernel functions satisfying Mercer conditions [5]. The kernel is defined as  $K(\mathbf{x}, \mathbf{x}_i) = \phi^T(\mathbf{x}_i) \phi(\mathbf{x})$ . The well known kernels include radial Gaussian, polynomial, spline or linear function.

The final problem of learning SVM, formulated as the task of separating learning vectors  $\mathbf{x}_i$  into two classes of the destination values either  $d_i = 1$  or  $d_i = -1$ , with maximal separation margin, is reduced to the dual maximization problem of the quadratic function [5,6]

$$\max Q(\alpha) = \sum_{i=1}^p \alpha_i - \frac{1}{2} \sum_{i=1}^p \sum_{j=1}^p \alpha_i \alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (7)$$

with the constraints  $\sum_{i=1}^p \alpha_i d_i = 0$ ,  $0 \leq \alpha_i \leq C$ , where  $C$  is a

user-defined constant and  $p$  is the number of learning data pairs  $(\mathbf{x}_i, d_i)$ .  $C$  represents the regularizing parameter and determines the balance between the complexity of the network, characterized by the weight vector  $\mathbf{w}$  and the error of classification of data. For the normalized input signals the value of  $C$  is usually much higher than 1 and adjusted by cross validation.

The solution of (7) with respect to the Lagrange multipliers produces the optimal weight vector  $\mathbf{w}_{opt}$  as

$$\mathbf{w}_{opt} = \sum_{i=1}^{N_s} \alpha_{oi} d_{oi} \phi(\mathbf{x}_{oi}). \quad \text{In this equation } N_s \text{ means the}$$

number of support vectors, i.e. the learning vectors  $\mathbf{x}_i$ , for which the relations  $d_i \left( \sum_{j=1}^K w_j \phi_j(\mathbf{x}_i) + w_0 \right) \geq 1 - \xi_i$  ( $\xi_i \geq 0$  -

the nonnegative slack variables of the smallest possible values) are fulfilled with the equality sign [5,6]. The output signal  $y(\mathbf{x})$  of the SVM network in the retrieval mode (after learning) is determined as the function of kernels

$$y(\mathbf{x}) = \sum_{i=1}^{N_s} \alpha_{oi} d_{oi} K(\mathbf{x}_{oi}, \mathbf{x}) + w_0 \quad (8)$$

and the explicit form of the nonlinear function  $\phi(\mathbf{x})$  need not be known. The value of  $y(\mathbf{x})$  greater than 0 is associated with 1 (membership of the particular class) and the negative one with -1 (membership of the opposite class). Although SVM separates the data only into two classes, the recognition of more classes is straightforward by applying either “one against one” or “one against all” methods [9].

#### 3.2 Regression mode

The learning task in this mode is transformed to the minimization of the error function, defined through the  $\varepsilon$ -insensitive loss functions  $L_\varepsilon(d, y(\mathbf{x}))$  [5,6]

$$L_\varepsilon(d, y(\mathbf{x})) = \begin{cases} |d - y(\mathbf{x})| - \varepsilon & \text{for } |d - y(\mathbf{x})| \geq \varepsilon \\ 0 & \text{for } |d - y(\mathbf{x})| < \varepsilon \end{cases} \quad (9)$$

where  $\varepsilon$  is the assumed accuracy. The learning problem is defined as the minimization of the error function

$E = \frac{1}{p} \sum_{i=1}^p L_\varepsilon(d_i, y(\mathbf{x}_i))$  at the upper bound on the weight

vector  $\mathbf{w}$ . Introducing the slack variables  $\xi_i$  and  $\xi'_i$  the learning problem can be redefined to the form similar to the classification mode, i.e., the minimization of the cost function

$$\phi(\mathbf{w}, \xi, \xi') = C \left[ \sum_{i=1}^p (\xi_i + \xi'_i) \right] + \frac{1}{2} \mathbf{w}^T \mathbf{w} \quad (10)$$

at the functional constraints  $d_i - \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) \leq \varepsilon + \xi_i$  and  $\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) - d_i \leq \varepsilon + \xi'_i$ , for  $\xi_i \geq 0$  and  $\xi'_i \geq 0$ . The constant C in equation (10) is a user specified parameter. The variables:  $\varepsilon$  and C are free parameters that control the VC dimension of the approximating function [5,6]. Both must be selected by the user.

The solution of so defined constrained optimization problem is solved in practice by introducing the Lagrange multipliers  $\alpha_i$ ,  $\alpha'_i$  responsible for the functional constraints, and by solving the appropriate quadratic programming task [5,6]. Once again the solution of the problem depends only on the kernel functions defined identically as in the classification mode. After solving the quadratic learning problem the optimal values of Lagrange multipliers are obtained. Denoting them by  $\alpha_{oi}$  and  $\alpha'_{oi}$  SVM network output signal  $y(\mathbf{x})$  is expressed as

$$y(\mathbf{x}) = \sum_{i=1}^{N_s} (\alpha_{oi} - \alpha'_{oi}) K(\mathbf{x}, \mathbf{x}_i) + w_0 \quad (11)$$

The important advantage of SVM approach is the transformation of the learning task to the quadratic programming problem. For this type of optimization there exist many highly effective learning algorithms [6,8], leading in almost all cases to the global minimum of the cost function and to the best possible choice of the parameters of the neural network. The presented above learning algorithms related to MLP and SVM have been implemented using Matlab platform [7] and tested on three tasks of the classification, prediction and regression type.

#### 4. THE RESULTS OF NUMERICAL EXPERIMENTS

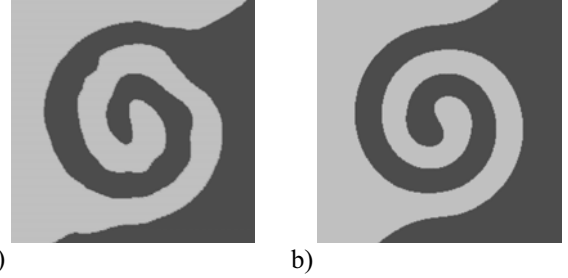
##### 4.1 Two-spiral classification problem

Two spiral problem belongs to the most demanding classification tasks [3]. The classification points distributed along two interlocking spirals go around the origin and belong to two classes (Fig. 1). The difficulty of this problem is due to the fact that small difference in x-y placement of the point may result in completely different class membership, since both classes are very close to each other. The problem is a serious challenge to all learning algorithms [2,3]. We have solved the problem using both: MLP network trained by using Levenberg-Marquardt algorithm and SVM with radial basis function trained by applying Platt method [8]. The training time of MLP was approximately 10 times longer than SVM.



**Fig. 1.** The 2-spiral problem training point distribution

Both networks have classified the data correctly. The results of testing the trained networks on the 2-spiral data scanned over x-y reception field are presented in Fig. 2.



**Fig. 2.** The output pattern of the reception field as the input is scanned over x-y space: a) MLP, b) SVM

The figures show the output of an optimal 25 hidden-unit MLP network (left) and SVM of radial kernel (right) employing 86 support vectors, as the input is scanned over the entire x-y field. Both networks properly classified all 200 training and all testing points. It can be seen that the reception field is interpolated fairly smoothly almost everywhere in the data field. More smooth reception field obtained for the SVM means, that the SVM solution is of better quality with respect to generalization. Moreover the training time of MLP was almost 10 times longer.

##### 4.2 Time series prediction problem

The next experiment is concerned on the prediction problem of the Mackey-Glass time series in chaotic mode, generated using the following differential equation [4]

$$\frac{dy(t)}{dt} = -by(t) + \frac{ry(t-\tau)}{1 + y^{10}(t-\tau)} \quad (12)$$

with the parameters  $\tau = 21$ ,  $r=0.2$ ,  $b=0.1$  and initial conditions  $y(t-\tau)=0.5$  for  $0 \leq t \leq \tau$ , and step size equal 2s. To predict the signal value  $y_i$ , the MLP and SVM input vector  $\mathbf{x}_i$  was equal  $\mathbf{x}_i = [y_{i-1}, y_{i-2}, \dots, y_{i-M}]^T$ . In our experiments we have chosen  $M=2$  and 1200 data samples generated according to the equation (12). The samples of points (1-600) have been used as the training set and the other samples (601-1200) as the testing set. The accuracy of prediction, as well as the complexity of the best trained MLP network and the best SVM network of radial kernel functions, have been compared in the testing range 601-1200 and set in Table 1. The values in

the Table are the averages of 10 different runs of the learning algorithms.

**Table 1.** The summary of results for time series prediction problem

Network	No of hidden units	Mean relative error	Max error
MLP	14	1.38%	5.94%
SVM	357	1.40%	4.83%

The prediction errors have been calculated only for testing data, not taking part in learning process. They reflect the generalization ability of both networks. As it is seen both networks behave well, although the MLP network seems to be a bit better with respect to the mean relative error (1.38% instead of 1.40%). However SVM is better with respect to the maximum relative errors (4.83% compared to 5.94%). On the other side it should be observed that the complexity of MLP is much lower than SVM (14 units instead of 357).

### 4.3 The artificial nose regression problem

The last comparison of the networks will be done on the real life problem: the calibration of the artificial nose [2]. The so called artificial nose is composed of an array of semiconductor sensors generating the signals proportional to the resistance dependent on the presence of particular gas. The normalized sensor signals are processed in the calibrating network, usually of the neural type. The role of neural calibrator is to calculate the gas concentration on the basis of the normalized sensor signals. In this application we have checked the MLP network and SVM of radial kernel function as the calibrators. In our experiments we have considered 4 toxic gases: carbon oxide, methane, propane/buthane and methanol vapour. As the sensors a small array of five semiconductor oxide sensing elements with various compositions has been used (TGS-815, TGS-822, TGS-842 Figaro sensors and NAP-11A, NAP-11AE Nemoto sensors). This array of sensors was exposed to various mixtures of air with these four pollutants of different concentrations.

The training data set consisting of 5 measured sensor signals of 340 known gas mixtures was used as the input for the network. A set of 80 further cases was used only for testing. Different concentrations (random distribution) of gases have been used in experiments, changing from 0 to 1500ppm. In the experiments all of them have been normalized to the range (0, 1) simply by dividing all numbers by the maximum value of concentration.

The number of input nodes of both networks was the same and equal to the number of sensors (5). MLP network used 4 output neurons, each responsible for the concentration of individual gas. SVM employs only one output unit, so 4 different networks should be trained, each responsible for one particular gas. The number of hidden units (K) for all trained networks are given in Table 1. It presents also the average absolute errors (mean and maximum values) of estimation of concentration of

four gases used in experiments by applying MLP network and SVM of radial kernel. They are the mean of 10 different runs of learning algorithm.

**Table 2.** The summary of average absolute errors of 4 gases estimation

Network	MLP			SVM		
	K	Mean $\times 10^{-4}$	Max $\times 10^{-4}$	K	Mean $\times 10^{-4}$	Max $\times 10^{-4}$
Gas 1	15	3.43	10.50	332	12.42	68.03
Gas 2	15	2.73	6.51	309	8.89	56.13
Gas 3	15	1.25	5.23	211	8.48	43.21
Gas 4	15	0.87	3.18	159	4.49	28.01
Average	15	2.07	10.50	253	8.57	68.03

As it is seen the MLP network was much better in this regression task. At much lower complexity (15 hidden sigmoidal neurons compared to 253 radial units) the obtained errors, both mean and maximum, are evidently smaller smaller. Moreover it should be noted that SVM solution needed training 4 different one-output networks, while MLP structure of 4 output neurons was only one, common for all four gases.

## 5. CONCLUSIONS

The numerical experiments performed for both: MLP and SVM networks have confirmed that both solutions are very well suited for classification, regression and prediction tasks. In classification mode the unbeatable is SVM, while in regression better generalization ability possesses MLP. The observed differences in performance are in most cases negligible. However the main difference is in the complexity of the networks. The MLP network implementing the global approximation strategy usually employs very small number of hidden neurons. On the other side the SVM is based on the local approximation strategy and uses large number of hidden units. The great advantage of SVM approach is the formulation of its learning problem, leading to the quadratic optimization task. It greatly reduces the number of operations in the learning mode. It is well seen for large data sets, where SVM algorithm is usually much quicker.

## REFERENCES

- [1] S. Haykin, Neural networks, a comprehensive foundation, Macmillan, 1994, N. Y.
- [2] S. Osowski, K. Brudzewski, Fuzzy self-organizing hybrid neural network for gas analysis, IEEE Trans. IM, 2000, vol.49, pp. 424-428
- [3] S.E. Fahlman, C. Lebiere, The cascade-correlation learning, in "Advances in NIPS2", D. Touretzky, Ed., 1990, pp. 524-532
- [4] E. Chang, S. Chen, B. Mulgrew, Gradient RBF networks for nonlinear and nonstationary time series prediction, IEEE Trans. On Neural Networks, 1996, vol. 7, pp. 190-194
- [5] V. Vapnik, "Statistical learning theory", Wiley, N.Y., 1998
- [6] A. Smola, B. Scholkopf, "A tutorial on support vector regression", NeuroColt Technical Report NV2-TR-1998-030, 1998, Matlab neural network toolbox, Math Works, Natick, 2001
- [7] J. Platt, "Fast training of SVM using sequential optimization", (in "Advances in kernel methods – support vector learning", B. Scholkopf, C. Burges, A. Smola eds), MIT Press, Cambridge, 1998, pp. 185-208
- [8] C. W. Hsu, C. J. Lin, A comparison of methods for multi-class support vector machines, IEEE Trans. NN, vol. 13, pp.415-425, 2002