Stanisław OSOWSKI
Institute of the Theory of Electrical Engineering,
Measurements and Information Systems
Warsaw University of Technology, Warsaw
Military University of Technology, Warsaw

MLP AND SVM CLASSIFIERS FOR FAULT DETECTION

Keywords

Diagnostics, neural classifiers, SVM, MLP, electrical circuits.

Abstract

The paper presents a comparative analysis of two of the most important neural network classifiers: the multilayer perceptron (MLP) and Support Vector Machine (SVM) in application to diagnostic problems. The structure as well as learning algorithms of both networks have been presented and compared. The results of numerical experiments comparing the performance of both classifiers on the artificial and real life problems are presented and discussed.

Introduction

An artificial neural network (ANN) is an abstract computational model of the human brain [4,11]. 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 examples and to generalize, i.e., to produce the reasonable outputs for new inputs not encountered during a learning process. The network trained on the set of learning samples acquires the ability of proper (required) behaviour on the new data, sharing the similar

properties as the learning data. To the distinct features of ANN belong the following: learning from examples, generalization ability, non-linearity of processing units, adaptability, massive parallel interconnections among processing units and fault tolerance. In contrary to the classical electronic circuits, where the fault of a single element ruins the performance of the whole circuit, the neural network is resistive to the fault. Moreover the technique of cutting individual weight connections is often used as the regularization technique, leading to the improvements of the network in the testing mode.

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 feed forward, i.e., the processing signals propagate from input to the output side.

The most representative example of the local network is the Support Vector Machine (SVM) of the Gaussian kernel function. It is a two-layer network employing one hidden layer of radial units and one output neuron. The procedure of creating this network and learning its parameters is organized in a way in which we adjust simultaneously the number of hidden units, their parameters and the weights of their interconnections with the output.

This paper will summarize and compare these two neural network classifiers: MLP and SVM. The comparison will be done with respect to the complexity of the structure as well as the learning algorithms. Special emphasis will be given to the generalization ability of the learned structures acquired in different learning processes.

1. Multilayer perceptron

The multilayer perceptron (MLP) network consists of many simple neuron-like processing units of sigmoidal activation function grouped together in layers. The general structure of the MLP classifier is presented in Fig. 1.

It contains input and output layers and one or two hidden layers (in most cases one hidden layer is sufficient). The input layer consists of the nodes, in which the excitations in the form of input signals of vector **x** are applied. The output layer is formed by the sigmoidal neurons. In the case of classifier the number of output nodes is equal to the number of classes, where the class is coded in a binary way (1 – represents the recognized class and 0 – lack of membership to the particular class). The MLP belongs to the neural networks of

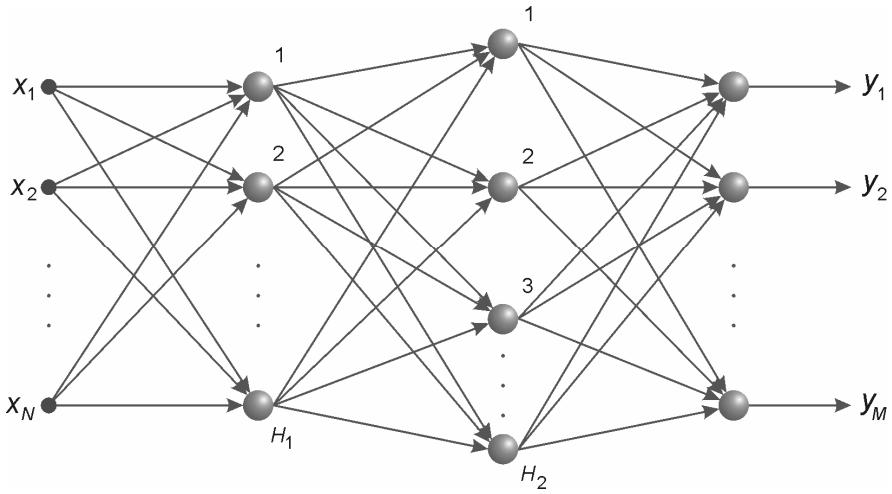


Fig. 1. The general structure of MLP classifier

supervised learning. For the purpose of learning p pairs of vectors ($\mathbf{x}_i$, $\mathbf{d}_i$), representing the problem under consideration, are generated, where $\mathbf{x}$ is N-dimensional input vector and $\mathbf{d}$ - the M-dimensional desired output vector (destination). The information contained in the vector $\mathbf{x}$ put to the input of network is processed locally in each unit by computing the dot product between the corresponding input vector and the weight vector of the neuron. Before training, the weights are initialized randomly.

The learning process of MLP network is based on the learning data samples ($\mathbf{x}_i$, $\mathbf{d}_i$). 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. Training the network to produce a desired output vector $\mathbf{d}_i$ when presented with an input vector $\mathbf{x}_i$ (i=1, 2, ..., p) involves systematically changing the weights of all neurons until the network produces the desired output within a given tolerance. The procedure is repeated over the entire training set.

From the mathematical point of view 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,…,p using an Euclidean norm

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

The most effective methods of minimization are based on gradient. For medium size networks (below 1000 weights) the Levenberg-Marquard algorithm is regarded as the best one. In the case of large networks the conjugate gradient

method is usually the best. Generally, in all gradient algorithms, the adaptation of weights is performed step by step according to the following scheme

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

In this relation **p**(k) is the direction of minimization in kth step and η is the adaptation coefficient. Various learning methods differ in the way the value of **p**(k) is generated.

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

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

where $\mathbf{g}(k) = \dfrac{\partial E}{\partial \mathbf{w}(k)}$ is the gradient of error function (1) and G(k) – the approximated Hessian, determined by applying the Jacobian matrix J(k)

$$\mathbf{G}(k) = \mathbf{J}(k)^T \mathbf{J}(k) + v1 \tag{4}$$

In this equation the Jacobian matrix J is equal $\mathbf{J} = \dfrac{\partial \mathbf{e}}{\partial \mathbf{w}}$, and $\mathbf{e} = \left[y_1(\mathbf{w}) - d_1, ..., y_M(\mathbf{w}) - d_M\right]^T$. The variable v is the Levenberg-Marquardt parameter adjusted step by step in a way to provide the positive definiteness of Hessian G (the value of v is eventually reduced to zero).

In conjugate gradient approach, the most effective method for large networks, the direction p is evaluated according to the formula

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

where the conjugate coefficient β is usually determined according to the Polak-Ribiere rule

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

In the weight update equation (2) the learning coefficient η should be adjusted by the user. It is usually done by applying so called adaptive way [4, 13], taking into account the actual progress of minimization of the error function.

The properly trained neural network acquires an unique generalization ability. To obtain this ability a lot of conditions should be fulfilled. First of all, the learning data should be typical for the modelled process and contain the characteristic examples of it. The number p of the learning pairs (x_i, d_i) should be sufficiently higher than the number of weights existing in the network. According to Vapnik [12], good generalization is always observed if the ratio of the number of learning values and the number of weights is higher than 20. However, it is not a necessary condition and good generalization is possible even with a smaller learning set. The learning process should be performed using efficient algorithms and should be fixed within limited time to avoid over fitting.

In the Institute of the Theory of Electrical Engineering, Measurements and Information Systems of Warsaw University of Technology the program *MLP* for training and testing the multilayer perceptron network was developed. It was written on the Matlab platform and has been implemented in the form of the graphical user interface presented in Fig. 2.

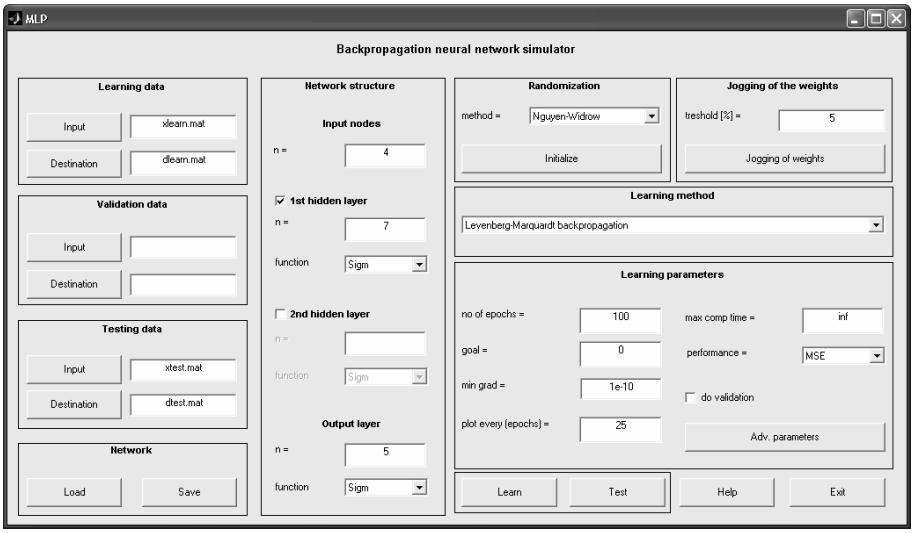


Fig. 2. The user interface for training and testing the MLP network

The definition of the network structure is very simple. The number of input and output nodes is set automatically by the program on the basis of learning data. The number of hidden layers and neurons in each of them is adjusted by the user (two hidden layers are possible). The program implements 6 different learning algorithms (the steepest descent, the steepest descent with momentum, BFGS of the variable metric, Levenberg-Marquardt, conjugate gradient and RPROP [11]). The learning step η is adjusted in an adaptive way. The learning,

validation and testing data are defined in the form of matrix **X** (the set of horizontal vectors $\mathbf{x}_i$) and matrix **D** composed of destination vectors $\mathbf{d}_i$. They are given in the form of files.

The learning phase starts after pressing the *Learn* button. Before doing it, the user has to adjust the number of learning epochs, the way of calculating the error (MSE, SSE or MAE) and also the stopping conditions in the form of the value of goal and the minimal gradient. The actual learning error is plotted parallel to the progress of the weight adaptation. The output vector y(x) is available in the working space of Matlab as the variable y.

.

2. Support Vector Machine classifier

Support Vector Machine [2, 9, 10] is a linear machine working in the highly dimensional feature space formed by the nonlinear mapping of the N-dimensional input vector **x** into a *K*-dimensional feature space (K>N) through the use of a mapping $\varphi(\mathbf{x})$. The SVM network recognizes between two classes, coded as d_i=1 and d_i=-1. In the classification mode the equation of the separating hyperplane is given by the following relation $y(\mathbf{x}) = \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b = \sum_{j=1}^{K} w_j\varphi_j(\mathbf{x}) + b = 0$, where the vector $\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_1(\mathbf{x}),...,\varphi_K(\mathbf{x})]^T$ is composed of activation functions of hidden units and $\mathbf{w} = [w_1,...,w_K]^T$ is the weight vector of the network. The parameters of the equation of the separating hyperplane y(**x**) are adjusted in a way to maximize the distance between the closest representatives of both classes. Mathematically the primary learning problem [9, 12] is defined as the minimization of the objective function $\phi(\mathbf{w},\boldsymbol{\xi})$

$$\phi(\mathbf{w},\boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{p}\xi_i \tag{7}$$

at the following linear constraints (i = 1, 2, ..., p)

$$\begin{aligned} &d_i\left(\mathbf{w}^T\varphi(\mathbf{x}) + b\right) \geq 1 - \xi_i \\ &\xi_i \geq 0 \end{aligned} \tag{8}$$

The first term in equation (4) corresponds to the maximization of the margin of separation. The constant C is the regularization parameter responsible for the minimization of the learning errors. The higher is its value the bigger impact of this term on the final parameters of the hyperplane.

The most distinctive fact about SVM is that the learning task is reduced to the quadratic programming by introducing the so-called Lagrange multipliers α_i. All operations in learning and testing modes are done in SVM by using kernel functions satisfying Mercer conditions [12]. The kernel is defined as

$$K(\mathbf{x},\mathbf{x}_i)=\boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}) \tag{9}$$

The most often used kernels include radial Gaussian, polynomial, spline or linear functions [9]. 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 [7, 9, 12]

$$\max \quad Q(\boldsymbol{\alpha})=\sum_{i=1}^{p}\alpha_i-\frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p}\alpha_i\alpha_j d_i d_j K(\mathbf{x}_i,\mathbf{x}_j) \tag{10}$$

with the constraints $\sum_{i=1}^{p}\alpha_i d_i=0,\ 0\le\alpha_i\le C$, where C is a user-defined regularization constant (hyperparameter) and p is the number of learning data pairs ($\mathbf{x}_i$, d_i). The regularizing parameter C determines the balance between the complexity of the network, characterized by the weight vector $\mathbf{w}$ and the error of the classification of data. For the normalized input signals the value of C is usually much higher than 1 and adjusted by the cross validation procedure. The solution of (10) is done with respect to the Lagrange multipliers, on the basis of which the optimal weight vector $\mathbf{w}_{opt}$ is determined, as

$$\mathbf{w}_{opt}=\sum_{i=1}^{N_S}\alpha_{oi}d_{oi}\boldsymbol{\varphi}(\mathbf{x}_{oi}) \tag{11}$$

In this equation N_s means the number of support vectors, i.e. the learning vectors $\mathbf{x}_i$, for which the relations $d_i\left(\sum_{j=1}^{K}w_j\varphi_j(\mathbf{x}_i)+w_0\right)\ge 1-\xi_i$ ($\xi_i\ge 0$ - the nonnegative slack variables of the smallest possible values) are fulfilled with the equality sign [9]. The output signal $y(\mathbf{x})$ of the SVM network in the retrieval mode (after learning) is determined as the function of kernels

$$y(\mathbf{x})=\sum_{i=1}^{N_S}\alpha_{oi}d_i K(\mathbf{x}_{oi},\mathbf{x})+w_o \tag{12}$$

with $w_0 = b$. Observe that the explicit form of the nonlinear function $\varphi(\mathbf{x})$ need not be known. Fig. 3 presents the network form that may be associated with the final expression describing the output signal of the SVM.

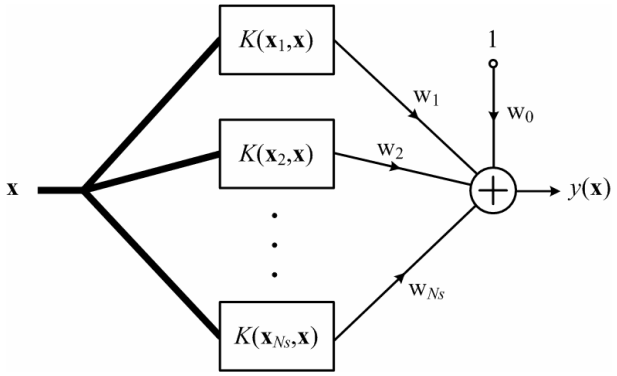


Fig. 3 The final structure of SVM network

In the classification mode 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 into two classes only, the recognition of more classes is straightforward by applying either "one against one" or "one against all" methods [5]. The more powerful is "one against one" approach in which many SVM networks are trained to recognize between all combinations of two classes of data. For M classes we have to train M(M-1)/2 individual SVM networks. In the retrieval mode the vector **x** belongs to the class of the highest number of winnings in all combinations of classes. In "one against all" we train only *M* SVM networks recognizing between the particular class and the rest. In the retrieval mode the vector **x** belongs to the class of the highest value of the discriminating function $y(\mathbf{x})$. It should be mentioned that there are some modifications of SVM problem formulation leading to the multiclass recognition in one network structure [5]. However the most popular approaches rely on the standard 2-class problem formulation and application of either "one against one" or "one against all" methods [9].

The important advantage of the SVM approach is the transformation of the learning task to the quadratic programming problem. For this type of optimisation there exist many very effective learning algorithms [7, 9] leading, in almost all cases, to the global minimum of the cost function and to the best possible choice of parameters of SVM networks. To the most known actual learning methods belong the modified sequential programming method of Platt, the LSVM procedure of Mangasarian and the active variable strategy implemented in SVM^{Light} of T. Joachims [7, 9]. Fig. 4 presents the graphical user interface of SVM developed in the Institute of the Theory of Electrical Engineering, Measurements and Information Systems. It enables the performance of the efficient learning and testing of SVM structure at the

classification (SVC) and regression (SVR) tasks and different kernel functions (linear, radial Gaussian, polynomial and sigmoidal). The classes are coded in practice using the natural number notation (the numbers from 1 to *M*).

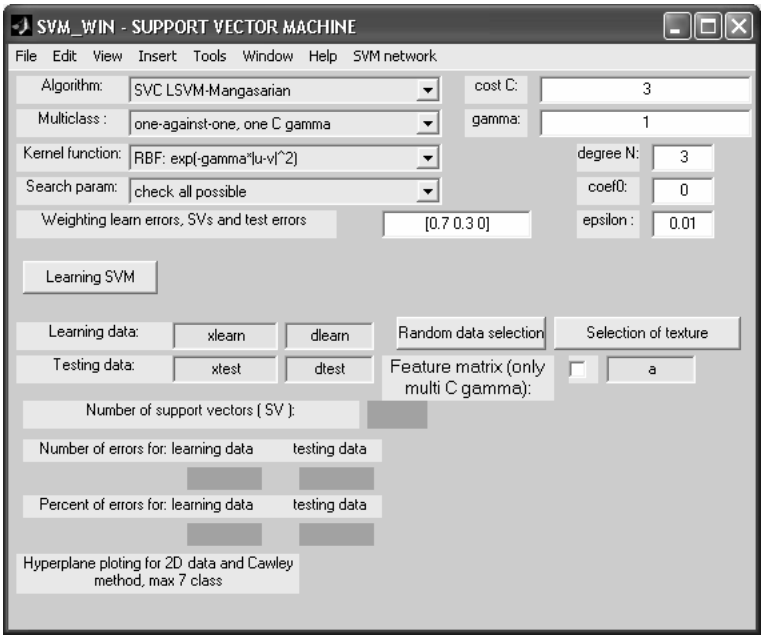


Fig. 4. The graphic user interface for SVM networks

3. The neural networks for fault detection

The neural classifier structures presented in the previous sections are the ideal executive devices in the technical diagnostic problems, especially fault detection and location. The diagnosis of any device or process is understood as the recognition of its actual state on the basis of the external measurements [6]. The measured values should be characteristic for the network or device under consideration. In the case of electrical networks they are currents and voltages measured at the accessible points (usually at the terminals).

To solve any diagnostic task efficiently we have to develop a full scheme of signal preprocessing. The most important stage is the generation of the diagnostic features, on the basis of which the process will be recognized. There are many different methods enabling the creation of features. Typically they involve scaling and standardization of the measured signals, transformation of them (FFT, wavelets, etc.), application of different decomposition and approximation approaches (Principal Component Analysis – PCA or strictly related to them Singular Value Decomposition – SVD, eigen decomposition,

orthogonal polynomial approximation, etc.), linear adaptive modeling (AR, MA, ARX, ARMAX, etc.), numerical characterization using statistical description (moments, cumulants, etc.). Usually we apply many different ways of characterizing the process in a parallel way. As a result, we get the characterization of the process by applying many different features.

It is well known that individual features have a different impact on the process of pattern recognition. A good feature should be very stable for samples belonging to the same class (the smallest possible variance), and at the same time it should differ significantly for different classes. The feature assuming similar values for different classes has no discriminative power and may be treated as the noise from the classification point of view. Thus, the main problem in machine learning is to find out the optimal set of features according to their importance for the problem solution. Note that the elimination of some features leads to the reduction of the dimensionality of the feature space and improvement of performance of the classifier in the testing mode at the data not taking part in learning.

Hence the important step in many classification problems is the selection of the features. Its main task is to arrange the features in an order depicting their class discrimination abilities. Only the most important features selected in this way are considered as the candidates for the input vector x. To get the best results of recognition we should apply the optimal set of features. There are many techniques of feature selection [3, 10]. To the most popular belong principal component analysis, projection pursuit, correlation existing among features, correlation between the features and the classes, analysis of mean and variance of the features belonging to different classes, application of linear SVM feature ranking, etc.

Especially interesting is the application of the linear SVM network for feature selection. There are two approaches possible. In the first one the predictive power of the single feature for a classification task is characterized by the value of error function minimized by a one-dimensional linear SVM trained to classify learning samples on the basis of only one feature of interest. The smaller this error, the better is the quality of the feature. We train as many linear SVM networks as are the number of features. This criterion may be used to rank features on the basis of the learning errors committed by the trained SVM networks and to select only those with an important predictive power.

It is intuitively understandable that the importance of a single feature may change when the feature is acting together with the other features. Recently authors of the writings [3] have introduced an interesting common feature selection method based on the application of linear Support Vector Machine. They proposed the ranking of the features working together as a whole set. The method is based on the idea that the absolute values of the weights of a linear classifier produce a feature ranking. The features connected with the output of

SVM through the weights of highest absolute values are regarded as the most important in the recognition process. All values of weights are arranged in decreasing order and only the most important are selected. The method proposed in [3] is recursive in nature and the elimination of features is done in many steps. The linear kernel SVM is used as the classifier, because this kernel does not deform the original impact of the feature on the result of the classification.

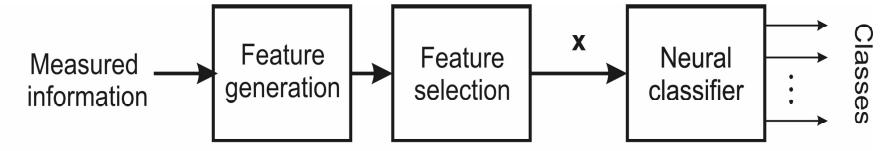


Fig. 5. The most important stages of the diagnostic system

The general scheme depicting the stages of the diagnostic process described above is presented in Fig. 5. Recall that the neural classifier is only the last stage used in an automatic diagnostic system. The efficiency of the whole system depends not only on the used classifier but also to great extend on the applied preprocessing stages, especially the way of generation and selection of the features, on the basis of which the neural classifier undertakes its decision. The better and more representative are the diagnostic features the more accurate results of the diagnosis.

4. The illustrative example

As an illustrative example we consider the recognition of single faults in the circuit of the low-pass biquadratic filter of Sallen-Key structure (Fig. 6). The filter was designed for the normalized frequency range (0, 0.5). The experiments have been performed for this normalized low-pass filter of the following nominal values of the elements: $G_1 = 1$mho, $G_3 = 0.81$mho, $G_4 = 1$mho, $C_2 = 2$F, $C_5 = 0.41$F.

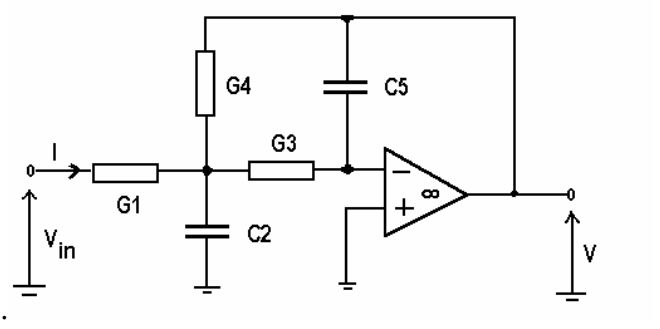


Fig. 6. The structure of the biquadratic low-pass analog filter used in the experiments

We consider here the parametric faults of the circuit elements. Faults are understood as all changes of the nominal values of conductance, inductance and capacitance ($G_n \to kG_n, L_n \to kL_n\ C_n \to kC_n$) with the value of coefficient *k* varying from zero to one minus tolerance value (the representation of the non-ideal open-circuit of the element) and from 1 plus tolerance to the infinity (the representation of the non-ideal short-circuit of the element). Each fault is associated with the tolerance of the remaining non-faulty elements changing in the experiments from 0 to 5%.

As a result of such assumptions two different kinds of faults for each element may be recognized: one corresponding to the increase of the admittance, called here the short-circuit type, and the second leading to its decrease, of the open-circuit type. At the number of circuit elements equal *n*, the number of resulting fault types is equal *2n*. The neural network should be designed in a way to recognize either appropriate fault or to indicate the normal operating condition of the circuit. It means that the number of recognized classes in this case is equal (*2n+1*).

4.1. Generation of the diagnostic features

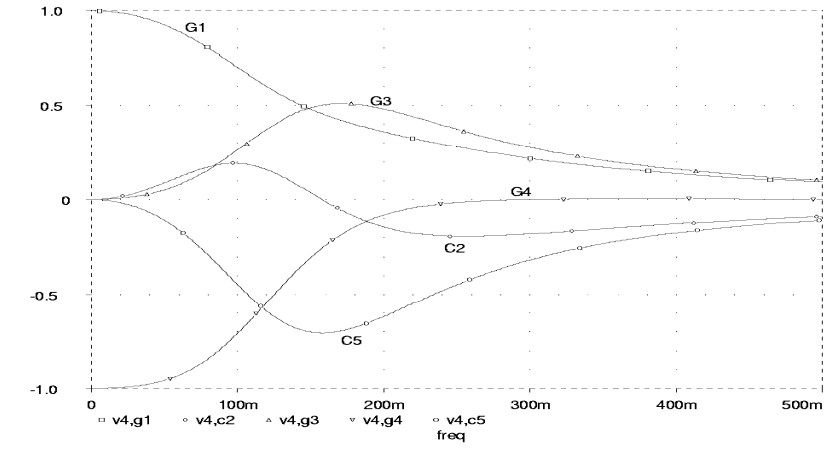
It is assumed that the only accessible points in the circuit are the input and output nodes. At voltage excitation and no load on the output side we can rely on the information contained in the input current and output voltage measured at different frequencies. The important point is the choice of the value of these frequencies providing the highest sensitivity of the recognizing system. In our solution we have adjusted them by the analysis of the sensitivity characteristics of the circuit [1].

Fig. 7 presents the exemplary relative sensitivity curves for the magnitude and phase of the output voltage $V=V_4$, obtained for this filter using PCNAP analyzer [8]. The letters G_1, G_3, G_4, C_2 and C_5 indicate what sensitivity parameter is actually considered. The points of the maximum or minimum of these curves indicate the frequencies that should be applied at the preparation of the learning and testing data for the neural network used as the recognizing system. The choice of the frequency points corresponding to the maximum or minimum of these curves provides the maximum sensitivity of the measured signal to the change of the appropriate parameter. This will help in getting the most sensitive system for fault recognition.

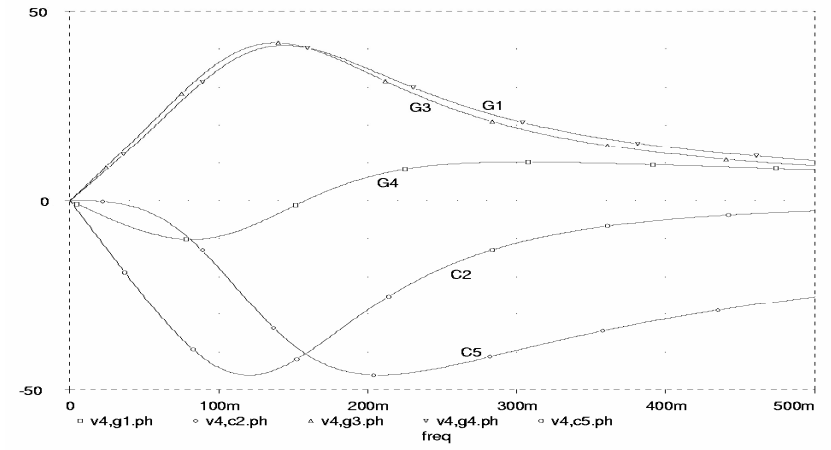
To enhance the differences among various faults we take here the learning signals as the magnitude and phase frequency characteristics of the circuit, considering the relative differences between the faulty and non-faulty modes of circuit operation. Applying the general notation x for either output voltage V or

input current I (magnitude or phase) we define their relative difference at the frequency f_i, as follows

$$x_r(f_i) = \frac{x_f(f_i) - x_n(f_i)}{x_n(f_i)} \tag{13}$$



a)



b)

Fig. 7. The sensitivity curves of the biquadratic filter: a) the magnitude, b) the phase of the output voltage. The x-axis represents the normalized frequency

The subscript *f* in this expression is related to the faulty state and *n* for non-faulty (normal) operation of the circuit.

4.2. Selection of the features

The maximum size feature vector **x** that may be generated in the circuit according to the presented procedure is given by

$$\mathbf{x} = [abs(\mathbf{V}_r), \arg(\mathbf{V}_r), abs(\mathbf{I}_r), \arg(\mathbf{I}_r)] \tag{14}$$

where the vectors: $\mathbf{V}_r = [V_r(f_1), V_r(f_2), ..., V_r(f_{m1})]$ and $\mathbf{I}_r = [I_r(f_1), I_r(f_2), ..., I_r(f_{m2})]$ represent the relative changes of the voltages and currents of the terminals at different frequencies f_i, normalized according to the relation (13), *abs* stands for the magnitude and *arg* for the phase of the corresponding complex values.

The next problem that should be solved is to determine if the measured data successfully separating different states of the circuit. The question is whether or not they are sufficient and optimal for recognition between different fault types. To solve this problem, we have applied principal component analysis of the data [4,11]. PCA is a method of transforming the original data set represented by vector samples into a new set of vector samples with reduced dimensions. The goal of PCA transformation is to concentrate the information about the differences between samples into a small number of dimensions (possible two) with low information loss. Presenting the results of PCA on the plane allows the user to assess the recognition capability of the presented feature set in a visual way.

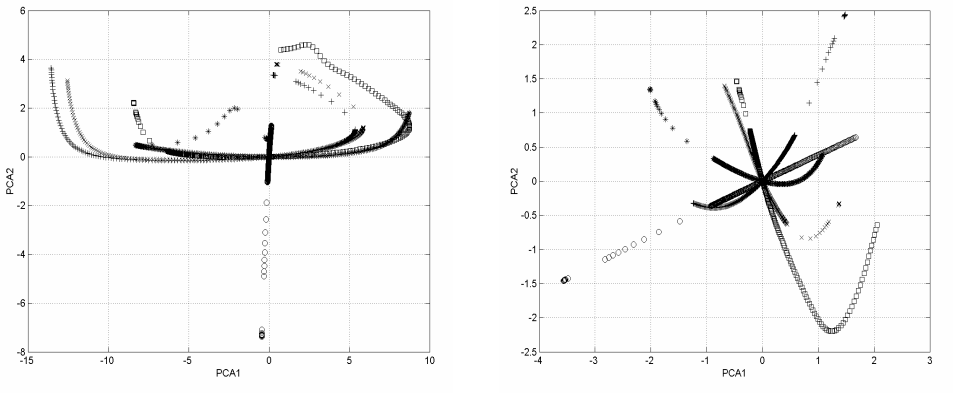
We have checked the PCA data distributions corresponding to different representations of the input vector **x,** containing: a) both magnitudes and phases of the output voltage and input current and b) only chosen representations of the output voltage and input currents. It means that many different forms of the vector **x** should be considered, including (14) and many combinations of the reduced forms, for example $\mathbf{x} = [abs(\mathbf{V}_r), \arg(\mathbf{V}_r), abs(\mathbf{I}_r)]$, $\mathbf{x} = [abs(\mathbf{V}_r), \arg(\mathbf{V}_r), \arg(\mathbf{I}_r)]$, $\mathbf{x} = [abs(\mathbf{V}_r), abs(\mathbf{I}_r), \arg(\mathbf{I}_r)]$, $\mathbf{x} = [\arg(\mathbf{V}_r), abs(\mathbf{I}_r), \arg(\mathbf{I}_r)]$ $\mathbf{x} = [abs(\mathbf{V}_r), \arg(\mathbf{V}_r)]$, $\mathbf{x} = [abs(\mathbf{V}_r)]$, $\mathbf{x} = [\arg(\mathbf{V}_r)]$, etc. The PCA transformation allows to visualize the distribution of the trajectories in a 2-dimensional graphic system corresponding to the faults of different circuit elements for the values of k changing from zero to infinity.

Fig. 8a presents the results of PCA at full feature vector described by (14) and Fig. 8b corresponding to the optimal feature vector **x** from which the phase information of the input current has been removed

$$\mathbf{x} = [abs(\mathbf{V}_r), \arg(\mathbf{V}_r), abs(\mathbf{I}_r)] \tag{15}$$

The x-axis represents the first principal component and y-axis the second largest principal component. The data forming the graphs correspond to

different values of elements, changing in experiments from zero to infinity. Point (0,0) on the figures corresponds to the nominal values of all parameters. Each fault type has been associated with different symbol (G1 $\rightarrow$o, G3 $\rightarrow$x, G4 $\rightarrow$+, C2 $\rightarrow$* and C5 $\rightarrow\square$). All graphs corresponding to the changes of the parameter values of different elements cross this particular nominal point.



a) b)

Fig. 8. The results of the PCA of the data corresponding to the voltage and current measurements of the filter: a) full feature vector containing magnitude and phase information of the voltage and current, b) the feature vector deprived of phase information of current

It is seen from the figures, that full feature vector described by (14) does not provide the best separation of different states of the circuit, since many trajectories overlap each other, especially in the neighborhood of the nominal point (0, 0). The PCA analyses of different feature representations have shown that the best separation of different states of circuit parameters are obtained when we take into account the reduced feature vector defined by (15). As shown, all five trajectories, corresponding to faults of each element of the circuit, are almost ideally separated for all considered values of coefficient *k*, changing from zero to infinity.

4.3. The results of fault recognition using SVM classifier

Different classifiers could be applied for the final fault recognition. We have tried two neural networks: MLP and SVM. However much better results have been obtained at SVM classification and the results presented here will be limited to this classifier only. In our experiments we have recognized two types of fault of each element: the non-ideal short circuit and non-ideal open circuit of it. At five passive elements of the circuit it means 11 classes (10 faults plus the

normal operation of the circuit). Since the SVM network has only one output neuron capable of recognizing between two classes, this multi-class recognition problem needs application of the one-against-one strategy [5], resulting in learning many $(11\times10/2=55)$ independent SVM classifiers. We have applied the Gaussian kernel SVM networks. The numbers of hidden neurons of these SVM networks as well as their weights have been adjusted in the learning process, performed here by applying the Platt algorithm [7]. We have used 500 learning data pairs corresponding to different cases of single faults of the elements and to the normal operation of the circuit, all associated with the tolerance of the non-faulty elements.

After learning phase, all parameters of SVM have been frozen and the network tested by using additional data samples. To check the generalization properties of SVM network in the testing mode, different values of the faulty element parameters have been used, especially those on the border of the tolerance limit. Additionally, all measurements have also been associated with a random variation of the parameters of other non-faulty elements within the tolerance limit.

The first experiments have been made at zero tolerance of elements. Zero tolerance means, that any deviation of the parameter value is regarded as a fault. It is just a pure theoretical case, very demanding for the recognition of faults, especially in the region very close to the nominal values of parameters. Only normal operation of the circuit is trivial and its recognition is possible without any errors. Most misclassifications made by the neural network classifier correspond to the faults placed in a close vicinity of the normal operating conditions of the circuit. However, the average testing error in this case did not exceed 0.48%.

Table 1. The average misclassification rate of the faulty element at 5% tolerance (the average percentage errors for each fault type)

Faulty element	Fault of the short-circuit type	Normal operation	Fault of the open-circuit type	Mean error
G_1	0	0	0	0
G_3	1.6%	0.8%	0.8	1.07%
G_4	0.8%	4.2%	0	1.67%
C_2	1.6%	3.3%	0.8%	1.90%
C_5	2.5%	8.3%	0.8%	3.87%
Mean	1.30%	3.32%	0.48%	1.70%

Inclusion of any tolerance of elements introduces some margin of normal operation. Any values of elements within tolerance limit are regarded as a normal state of the circuit. The other values of parameters mean faults. As usual,

all faults have been associated with the tolerance of the remaining, non-faulty element values. Table 1 presents the results of testing the trained SVM recognizing system at 5% tolerance of elements. The numbers given in the table denote the statistical average errors of the recognition of proper fault type of element. The faulty element values have been placed uniformly in the considered fault range. 600 cases of faults of each element have been tested, from which 240 were of the open-circuit type, 240 of the short-circuit type and 120 within tolerance limits, regarded here as the normal operation of the circuit. As it is seen the average errors of recognition are of very limited values. The maximum average error for the most difficult case (C5) did not exceed 9% and the mean of the average errors for all considered cases (including faults and normal operation) is equal 1.70%. Recall also that all tests have been performed for the data fuzzified by the tolerance of all non-faulty elements. However, even in this demanding case the mean of all average errors, calculated as the ordinary mean of all misclassification rates did not exceed the value of 1.70%. The main source of this error is the misclassification rate at normal operation of the circuit.

Conclusions

The paper has presented the new approach to the fault location in an analog circuit, using artificial neural networks of MLP and SVM type. The neural classifiers stand for the universal solution of many recognition tasks of either a technical or non-technical nature. The automatic diagnostic system needs the generation of the features well characterizing the process, selection of them to discover the most discriminating features and as the final step – the final classification on the basis of the applied feature vector. We have illustrated the whole procedure with the example of recognition of the parametric fault of the low-pass biquadratic RC filter.

The distinct advantage of the proposed solution is its high accuracy and great speed of operation. Once the network has been trained, the recognition of fault is achieved immediately, irrespective of the size of the circuit. Thus the solution is suited for real time applications for fault location. The neural network solution of the classifier is of very good generalization ability. Trained on the limited number of representative examples of each fault, the network is able to recognize the non-ideal (parametric) fault in the wide range of changed parameter values and at some assumed tolerance of the non-faulty elements.

Acknowledgements

Research work carried out within the research project "Development of technical systems for the prevention of technical risk and disasters recovery" in The Multi-Year Programme entitled "Development of innovativeness systems of manufacturing and maintenance 2004-2008".

References

1. Alippi C., Catelani M., Fort A.: Automatic selection of test frequencies for the diagnosis of soft faults in analog circuits, IEEE Instr. Measurem. Techn. Conf., Anchorage, 2002, p. 1503-1507.
2. Burges C.J.: A tutorial on Support Vector Machines for pattern recognition, (in "Knowledge discovery and data mining ", ed. Usama Fayyad, Kluwer, 2000, p. 1-43).
3. Guyon I., Elisseeff A.: An introduction to variable and feature selection, JMLR, 3, 1158–1182, 2003.
4. Haykin S., Neural networks, comprehensive foundation, Prentice Hall, 1999, New Jersey.
5. Hsu C.W., Lin C.J.: A comparison methods for multi class support vector machines, IEEE Trans. Neural Networks, 13, (2002), p. 415-425.
6. Liu R.: Testing and diagnosis of analog circuits and systems, Van Nostrand Reinhold, 1991, New York.
7. Platt L.: Fast training of SVM using sequential optimization, (in Scholkopf B., Burges B., Smola A., Eds.: Advances in kernel methods – support vector learning. Cambridge: MIT Press), (1998), p. 185-208.
8. Rubner-Petersen T.: NAP2 – a nonlinear analysis program for electronic circuits, user manual, University, 1972.
9. Schölkopf B., Smola A.: Learning with Kernels, 2002, Cambridge, MA: MIT Press.
10. Schurmann J.: Pattern classification, a unified view of statistical and neural approaches, 1996, Wiley, New York.
11. Osowski S.: Sieci neuronowe do przetwarzania informacji, Oficyna Wydawnicza PW, 2001.
12. Vapnik V.: Statistical learning theory, 1998, Wiley, New York.
13. Neural Network Toolbox of Matlab, user manual, MathWorks, Natick, 2001.

Reviewer:
Andrzej ŚWIERNIAK

Klasyfikatory neuronowe MLP i SVM dla potrzeb diagnostyki

Słowa kluczowe

Klasyfikatory neuronowe MLP i SVM, sztuczna inteligencja, diagnostyka.

Streszczenie

Praca przedstawia dwa rozwiązania klasyfikatorów neuronowych na potrzeby diagnostyki. Jednym z nich jest perceptron wielowarstwowy (ang. MultiLayer Perceptron – MLP), drugim sieć wektorów podtrzymujących (ang. Support Vector Machine (SVM). Przedstawiono struktury oraz podstawowe metody uczenia takich sieci. Działania obu klasyfikatorów sprawdzono i porównano na problemach testowych, zarówno typu syntetycznego, jak i problemie rzeczywistym rozpoznawania uszkodzeń elementów w rzeczywistym układzie filtru elektrycznego.