Classifier's Complexity Control while Training Multilayer Perceptrons

Šarūnas Raudys

Institute of Mathematics and Informatics
Akademijos 4, Vilnius 2600, Lithuania
e-mail: raudys@das.mii.lt

Abstract. We consider an integrated approach to design the classification rule. Here qualities of statistical and neural net approaches are merged together. Instead of using the multivariate models and statistical methods directly to design the classifier, we use them in order to whiten the data and then to train the perceptron. A special attention is paid to magnitudes of the weights and to optimization of the training procedure. We study an influence of all characteristics of the cost function (target values, conventional regularization parameters), parameters of the optimization method (learning step, starting weights, a noise injection to original training vectors, to targets, and to the weights) on a result. Some of the discussed methods to control complexity are almost not discussed in the literature yet.

1 Introduction

A number of years a scientific discussion about preference of parametric and nonparametric classification rules is going on. Some scientists advocate that it is not wise to make assumptions about the type and parameters of the multivariate distribution densities, to estimate these characteristics from the training set data and only then to construct the classifier (see e.g., [15]). Instead of constructing the parametric statistical classifiers they advocate that much better way is to make assumptions about a structure of the classification rule (for example a linear discriminant function in a space of original or transformed features) and then to estimate unknown coefficients (weights) of the discriminant functions directly from the training data. This is a typical formulation of the classifier design problem utilized in *artificial neural network* (ANN) theory. Another point of view is characteristic to *statistical pattern recognition*. This approach advocates that making assumptions about the type of the distribution density function is a some sort of introducing a prior information into the classifier's design process. In a case, when this additional information is correct, it can reduce the classification error [4]. Therefore, up to know the discussion between followers of both approaches is lasting on.

An important recent result obtained in a channel of the ANN theory is a demonstration that in training, the non-linear SLP evolves: one can obtain several standard statistical classification and prediction rules of different complexity [9]. It

F.J. Ferri et al. (Eds.): SSPR&SPR 2000, LNCS 1876, pp. 32-44, 2000.

was shown that conditions E exist where after the first iteration of the gradient minimization (Back Propagation - BP) training algorithm performed in a batch mode, one can obtain the well known Euclidean distance classifier (EDC). For this one needs to start training from zero initial weight vector, a mean of the training data should be moved into the center of co-ordinates. In further iterations, one can obtain a classical regularized discriminant analysis (RDA) and succeeding one is moving towards the standard linear Fisher classifier. If the number of dimensions exceeds the training set size then we are approaching the Fisher classifier with the pseudo-inversion of the sample covariance matrix. In further iterations, we have a kind of a robust classifier which is insensitive to outliers, atypical training set vectors distant from the discriminant hyper-plane. If the weights of the perceptron are large, we move towards a minimum empirical error classifier. If we have no empirical errors, we are approaching the maximal margin (the support vector) classifier. We can train the perceptron in a space of new features which can be obtained by nonlinear transformations of the original n features. Then the number of types of the classification rules can be increased.

The evolution of the non-linear SLP can be utilized *to integrate the statistical and neural net theory based approaches* to design classification and prediction rules [10]. In this approach, instead of designing parametric statistical classifiers we use the training set based information (sample means, conventional, constrained, or regularized estimates of the covariance matrix) in order to whiten the distribution of vectors to be classified. Thus, we transform the original data into the spherical one and have a good initialization: after the whitening transformation and the first BP iteration, we obtain EDC which for the spherical Gaussian data is the best sample based classifier. In the original feature space, EDC is equivalent to the statistical classifier which could be obtained by utilizing "the training set based information" directly. It can happen that the parametric assumptions utilized to transform the data are not correct absolutely. Then in further perceptron training, we have a possibility to improve the decision boundary. In case we train and stop correctly, we can obtain an "optimal" solution.

In order to obtain a full spectrum of the statistical classifiers we need to control the training process purposefully. The complexity of the classifier is very important factor that influence a performance. In the small training sample case, it is worth using simple classifiers, and in the large sample cases, it is preferable to use complex ones [6, 8]. Thus, a problem arises about how to choose a classifier of optimal complexity.

2 The Perceptron and Its Training Rule

In this paper we restrict ourselves with an analysis of the single layer and one hidden layer perceptron (SLP and MLP) classifiers used to obtain the classification rule in a two category case. We consider a simple back propagation training technique based on a gradient descent minimization procedure. We discuss an influence of all parameters of the cost function as well as the optimization procedure on the complexity of the

classification rule obtained. SLP has a number of inputs x_1 , x_2 , ... , x_n and an output o which is calculated according to equation

$$o = f(\boldsymbol{V}^T \boldsymbol{X} + w_0), \tag{1}$$

where $f(net)$ is a non-linear activation function, e.g. a tanh function

$$f(net) = \tanh(net) = (e^{net} - e^{-net})/(e^{net} + e^{-net}), \tag{2}$$

w_0, $\boldsymbol{V}^T = (v_1 , v_2 , ... , v_n)$ are weights of the discriminant function (DF), to be learned during training and $\boldsymbol{V}^T$ denotes the transpose of vector $\boldsymbol{V}$.

To find the weights we have to minimize a certain cost function. Most popular is a sum of squares cost function

$$cost_t = \frac{1}{N_1 + N_2} \sum_{i=1}^{2} \sum_{j=1}^{N_i} (t_j^{(i)} - f(\boldsymbol{V}' \boldsymbol{X}_j^{(i)} + v_o))^2 + \lambda \boldsymbol{V}^T \boldsymbol{V} \tag{3}$$

where $t_j^{(i)}$ is a desired output (a target) for $\boldsymbol{X}_j^{(i)}$, j-th training set observation from class ω_i, N_i is the number of training vectors from class ω_i.

The term $\lambda \boldsymbol{V}^T\boldsymbol{V}$ is called a "weight decay" term, where λ is a positive constant called the regularization parameter. If activation function (2) is used then typically one uses $t_j^{(1)} = -1$ $t_j^{(2)} = 1$ either $t_j^{(1)} = -0.8$, $t_j^{(2)} = 0.8$. Outputs of the activation function (2) vary in an interval (-1, 1). Therefore, the target values -1 and 1 we will refer as limiting ones, and values -0.8 and 0.8 - as "close". The one hidden layer MLP considered in the present paper consists from one output neuron with the transfer function (1) and a number of hidden neurons with the same sort of the activation functions. In the gradient descent optimization, we find the weights in an iterative way. At a step t we update the weight vector $\boldsymbol{V}_{(t)}$ according to equation

$$\boldsymbol{V}_{(t+1)} = \boldsymbol{V}_{(t)} - \eta \frac{\partial cost_t}{\partial \boldsymbol{V}}, \quad v_{0(t+1)} = v_{0(t)} - \eta \frac{\partial cost_t}{\partial v_0}, \tag{4}$$

where η is a learning-step, and $\partial cost_t / \partial \boldsymbol{V}$ is a gradient of the cost function.

There exist a number of techniques used to control the complexity of ANN. Descriptions of these techniques are dispersed in a number of scientific journals devoted to Artificial neural networks, Statistical pattern recognition, Data analysis methods, Multivariate statistical analysis. A nice explanation and comparison of several methods (noise injection, sigmoid scaling, target smoothing, penalty terms, an early stopping and pruning the networks) can be found in [3, 7, 12, 13].

However, there exist more factors which practically participate in determining the network's complexity and which likely are not well known to the connectionist community. We present a systematic analysis of all complexity control techniques which are used or can be used in SLP and MLP design. One more emphasis is that in preceptron training, *several factors act simultaneously* and in order to control the

network complexity the designer should take into account ***all*** of them. We consider a question how the parameters enumerated affect complexity, however, do not discuss a problem how to choose a set of their optimal values.

3 The Number of Iterations

In the nonlinear SLP training, with an increase in the number of iterations one can obtain *seven statistical classifiers of different complexity* (EDC, RDA, standard Fisher LDF, Pseudo-Fisher LDF, robust DA, minimum empirical error classifier, and the maximum margin (support vector) classifier [9]. Doubtless, the number of iterations affect the classifiers type while training MLP too.

4 The Weight Decay Term

Nowadays it is the most popular technique utilized to control the complexity of ANN [3]. Addition of the term $\lambda \boldsymbol{V}^T\boldsymbol{V}$ to the standard cost function reduces the magnitudes of the weights. For very small weights the activation function acts as a linear one. Suppose, $t_1 = -1$ and $t_2 = 1$, we have the same number of training vectors from each class ($N_2 = N_1$) we add the "weight decay" term $\lambda \boldsymbol{V}^T\boldsymbol{V}$ and instead of the nonlinear activation function we use the linear one, i.e. $o(net)=net$. Then equating the derivatives (4) to zero and solving resulting equations we can show that the "weight decay" term guides to RDA. (see e.g. [11]). An alternative regularization term is $+ \lambda(\boldsymbol{V}^T\boldsymbol{V} - c^2)^2$. Here a positive parameter c^2 controls the magnitude of the weights and acts as the traditional regularizer.

5 The Antiregulatization

After starting from small or even zero values the weights of the perceptron are increasing gradually [9]. The nonlinear character of the activation function assists in obtaining the robust DA, the robust regularized DA, the minimum empirical error and the maximum margin classifiers. The magnitudes of the weights, however, depend on separability of the training sets (the empirical classification error). Small weights can prevent us to obtain anyone from the complex classifiers, such as the robust DA, the robust regularized DA, the minimum empirical error and the maximum margin classifiers.

If the training sets overlap, the weights can not be increased very much. Thus, in highly intersecting pattern classes case, in order to be able to control a type of the classifier obtained at the end, we have to increase the weights artificially. For this purpose, we can *subtract the weight decay term instead of adding* it. Then we obtain large weights and begin to minimize the empirical classification error. This technique is called *antiregularization*. It was demonstrated that for non-Gaussian data

characterized by a number of outliers, the antiregularisation technique can reduce the generalization error radically. Often the traditional weight decay term destabilizes the training process. In this sense, the more complicated regularization term + $\lambda(\boldsymbol{V}^T\boldsymbol{V}- c^2)^2$ is more preferable.

6 A Noise Injection

A noise injection is one more very popular regularization factor. A noise can be added to inputs of the network, to outputs, to the targets, as well as to the weights of the network (see e.g. [2, 3]).

6.1 A noise Injection to Inputs. In this approach, we add a zero mean and small covariance noise vectors $\mathbf{n}_\alpha$ to each training vector during every particular training iteration:

$$\boldsymbol{X}_{j\beta}^{(i)} = \boldsymbol{X}_{j}^{(i)} + \mathbf{n}_{j\beta}^{(i)}, \quad (\beta = 1, 2, \ldots, t_{max}), \tag{5}$$

where t_{max} is the number of training sweeps.

This technique is called also jittering of the data. While injecting noise $m = t_{max}$ times to each training vector we "increase" the number of observations m times. For the linear classification, we can find a *link between a noise injection and RDA*. Let the mean of noise vector $\mathbf{n}_{j\beta}^{(i)}$ be zero and a covariance matrix be $\lambda\mathbf{I}$. Then the covariance matrix of random vectors $\boldsymbol{X}_{j\beta}^{(i)}$ (β=1, 2, …, m) will be $\Sigma_i + \lambda\mathbf{I}$, where Σ_i is a true covariance matrix of $\boldsymbol{X}_{j}^{(i)}$. Consider now a sample covariance matrix of a training set composed from vectors (j=1, 2, …, N_i β = 1, 2, ... , m). When $m\to\infty$, the new sample estimate of the covariance matrix tends to

$$\hat{\Sigma}_{N}^{(i)} = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (\boldsymbol{X}_{j}^{(i)} - \hat{\boldsymbol{M}}_{\boldsymbol{j}}^{(i)})(\boldsymbol{X}_{j}^{(i)} - \hat{\boldsymbol{M}}_{\boldsymbol{j}}^{(i)})^T + \lambda\mathbf{I}. \tag{6}$$

The covariance matrix (6) coincides with the "ridge" (regularized) estimate used in RDA. Minimization of the mean square error criterion (3) and the weight decay term is equivalent to RDA. Consequently, when $m \to \infty$, a noise injection technique also approximates the regularization approach using the weight decay.

Adding a noise essentially amounts to introducing of a new information which states that the space between the training vectors is not empty. It is very important in MLP design, since the network (more precisely the cost function to be minimized) "does not know" that the pattern space between the training observation vectors is empty. A noise injected to inputs introduces this information. In practice, it is important to choose a right value of λ, the noise variance. Similarly to RDA and smoothing in the Parzen window classifiers, an optimal value of λ (the regularization parameter or the noise variance) depends upon the number of training vectors and a complexity of the problem.

6.2 A noise Injection to the Weights and to the Outputs of the Network. Adding a noise to the weights is a common practice to tackle local extreme problems in optimization. In addition to fighting the local minima problems, the noise injection to the weights increases the overlap of the training sets. Consequently, the noise injection to the weights reduces their magnitudes and acts as a tool to control the complexity of the classification rule. In the MLP classifier design, the noise injection into the weights of the hidden layer acts as a noise injection into inputs of the following layer.

During the training process a noise can be added also to the outputs of the networks (or the targets). Such noise injection also reduces magnitude of the weights and influences a complexity of the classification rule. In all cases, an optimal value of a noise variance should be selected. Typically in situations with unknown data, the variance is determined by the cross validation technique.

A drawback of training with the noise injection is that it usually requires to use small learning-step and many sample presentations, *m,* over the noise. In the high-dimensional space, this problem is particularly visible.

6.3 A "Colored" Noise Injection to Inputs [5, 14]. As a rule, one injects to inputs a "white" noise. In the high dimensional classification problems, data is situated in nonlinear subspaces of much lower dimensionality than a formal number of the features. It means that for large λ *the spherical noise injection can distort a data configuration*. In a *k-NN* directed noise injection, *k-NN* clustering should be used to determine the local intrinsic data dimension; the added noise should be limited to this intrinsic dimension. Then we have a minor data distortion, and in comparison with the spherical noise injection we obtain a gain. An alternative way is to use k nearest neighbors to calculate a singular sample covariance matrix $\hat{\Sigma}_{ij}^{k}$ around $\boldsymbol{X}_{j}^{(i)}$, and then to add a Gaussian $N(\mathbf{0}, \lambda \hat{\Sigma}_{ij}^{k})$ noise. Thus, instead of a "white" noise we are adding a "colored" one. Purposeful transformations of training images and signals performed prior to the feature extraction is a colored noise injection too. The optimal number of nearest neighbors k, and a noise variance λ should be determined in an experimental way.

7 Target Values' Control

Let us analyze the classification problem into two pattern classes with SLP trained by the sum of squares cost function and tanh activation function. Suppose targets t_1 and t_2 are close to each other, e.g. t_1 = - 0.2 and t_2= 0.2. Then *after* minimization of the cost function we obtain small values of the sums $G_{ij} = \sum_{\alpha=1}^{n} v_\alpha x_{\alpha j}^{(i)} + v_0$ and, consequently, small weights. As a result, the weighted sums G_{ij} are varying in a small interval around zero. Essentially, *the activation function $f(G)$ is acting as a linear*

function. Hence, with the close targets we will not obtain the robust classification rule, the minimum empirical error and maximal margin classifiers. When $t_1 \rightarrow$ -1 and $t_2 \rightarrow$ 1 the weights will increase. Then SLP will begin to ignore training vectors distant from the decision hyper-plane. Thus, the target values can be utilized as a tool to control the "robustness" of the classification rule. To obtain the minimum empirical error and the maximal margin classifier we need to use the target values very close to the limit activation function values (-1 and 1 for the tanh activation function). Values outside interval (-1, 1) assist in fast growth of the magnitudes of the weights, speed up the training process, however, increase a probability to be trapped into the local minimum.

8 The Learning Step

The learning-step η affects the training speed. Small η forces the weights to be small for a long time. Thus, while training the non-linear SLP the small learning step assists in obtaining a full sequence of the regularized classifiers and the linear discriminant function with the conventional or with the pseudo-inverse covariance matrix. The large learning step speeds up the weights growth, however, it can stop the training process just after the first iteration. If one fulfils the conditions E and uses very large η, then after the first iteration one can obtain enormous weights. The gradient of the cost function can become very small, and the training algorithm can stop just after the first iteration. Hence, one gets EDC and does not move further. Large and intermediate learning step values can stop the training process soon after the RDA is obtained and does not allow to obtain RDA with small regularization parameter. The large learning-step leads to large weights immediately. They are increasing a possibility to be trapped into local mininima.

If a constant learning step is utilized, then with an increase in the magnitude of the weights the training process slows down and therewith stops. One can say *an effective value of* η *decreases*. In this case, the fixed value of the learning step can prevent the algorithm to get the minimum empirical error or the maximum margin classifier. One of solutions to overcome this difficulty is to use a *variable learning step* η, *where after a certain number of iterations* η *is either increased or reduced.* In the back propagation training, in order to obtain the maximal margin classifier it was suggested *to increase learning step* η *exponentially*, e.g. $\eta = \eta_0 (1+\varepsilon)^t$, where t is the iteration number and ε is a small positive constant chosen in a trial and error way. The large learning step can prevent from obtaining the standard statistical linear classifiers, however, can allow to get the robust regularized discriminant function. A degree of regularization and the robustness can be controlled by the learning step value.

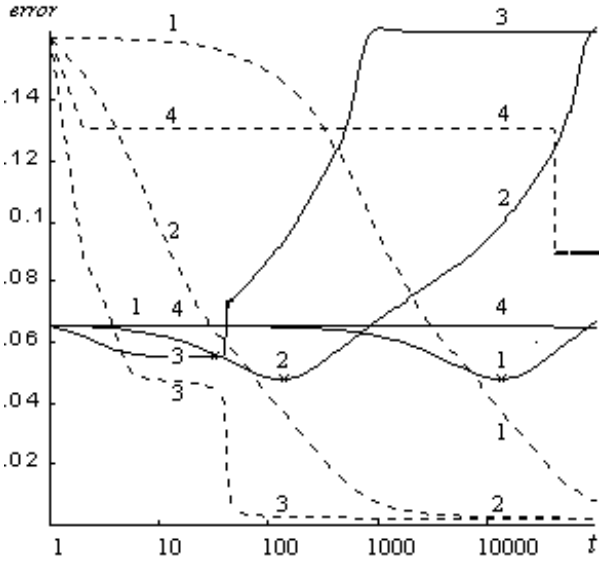

Fig. 1. Generalization (solid) and mean square (dots) errors as a function of the number of iterations t for different values of η: 1 - η = 0.001, 2 - η =0.1, 3 - η =10, 4 - η = 1000.

Example 1. Above, in Fig. 1 we had presented four pairs of learning curves "the generalization error (solid curves) and the empirical (training set) classification error (dots) as functions of the number of iterations t for different η values. In this experiment, we used two 12-variate Gaussian pattern classes with correlated features (ρ = 0.2), the Mahalanobis distance δ=3.76 (the Bayes error ε_B = 0.03), sigmoid activation function, the targets t_1=0.1, t_2=0.9, the training set composed from 10 +10 vectors. Both almost straight lines 4 show that for η = 1000 the adaptation process after the very first iteration stops practically. We have extremely slow (at the beginning) training when the learning-step is too small (curves 1 for η = 0.001). In this example, the best learning step's value is η = 0.1 and results in the smallest generalization error (curve 2).

9 Optimal Values of the Training Parameters

In SLP training, the "weight decay", the spherical noise injection, as well as early stopping can be equivalent to the classical RDA. In all four complexity control methods, we need to choose optimal values of λ or the optimal number of iterations. It was shown theoretically that in RDA, the optimal value of the regularization parameter λ depends on N, the training set size: $\lambda_{opt.}$ decreases when N increases [11]. Unfortunately, analytical equations were derived for exactly known data model

(multivariate Gaussian with known parameters), for very small λ values and cannot be applied in real situations.

Example 2. In Fig. 2 we present average (over 20 independent experiments) generalization error values as functions of λ, the conventional regularization parameter in RDA (*a*), the regularization parameter in the "weight decay" procedure (*b*), and the noise variance in the "noise injection" regularization (*c*). In this experiment, we used artificial 8-variate Gaussian GCCM data with strongly dependent features.

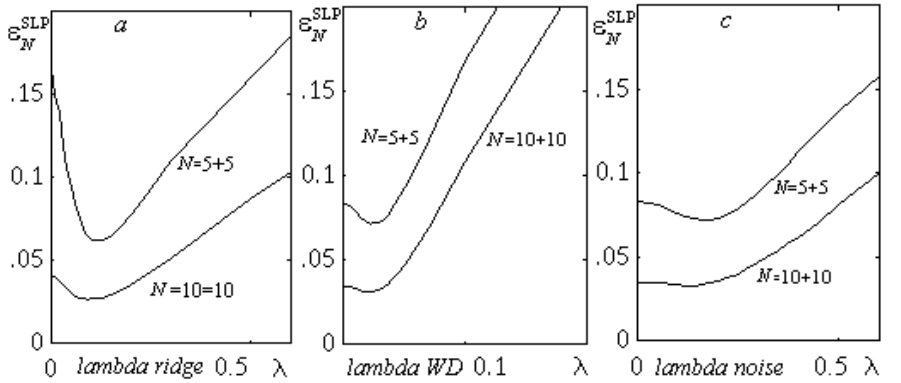

Fig. 2. Three types of regularization of the SLP classifiers for two sizes of the training sets (N_1=N_2=5 and 10): *a* - the regularized discriminant analysis with the "ridge" estimate of the covariance matrix, *b* - "weight decay", *c* - "noise injection" to inputs.

For each value of *N,* the number of training examples, all three pairs of the graphs approximately exhibit the same generalization error at minima points and explain by example *the equivalence of all* 3 *regularization techniques*. This experiments confirms theoretical conclusion: the optimal values of the regularization parameters decrease when the number of training examples increases. In SLP training, the number of iterations, the learning step, the targets, a noise, the weight decay term, the non-linearity of the activation function are acting together. An influence of one regularization factor is diminished by other ones. This circumstance causes additional difficulties in determining the optimal values of the complexity control parameters.

The targets, the learning-step and even the number of iterations do not act in a visible way. Thus, they can be considered as *automatic regularisers*. In the neural network training, however, their effects can not be ignored.

Considerations presented in this section are valid for the gradient type search techniques used to find the weights. Utilization of more sophisticated optimization techniques such as the second order Newton method or its modifications can speed up the training process, however, at the same time introduces definite shortcomings. For example, in a case of the linear pereceptron, the second order (Newton) method, in principle, allows to minimize the cost function just in one step, however, prevents

from obtaining RDA. *In this sense, the BP algorithm is more desirable*. Nevertheless, this shortcoming of the Newton method can be diminished by changing the target values, introducing the weight decay term either by a noise injection.

10 Learning Step in the Hidden Layer of MLP

In comparison with SLP, in the MLP training, several additional aspects arise. The learning-step is the first important *secret factor* that affects the complexity. In the gradient descent optimization procedure, a fixed value of the learning-step η causes a random fluctuation of the parameters to be optimized (in our case, the components of the weight vector). Asymptotically, for fixed η, and large t, the number of iterations, the weight vector $\boldsymbol{V}_{(t)}$ oscillates around $\hat{\boldsymbol{V}}_N$, an optimal weight vector for this particular training set. In his classical paper [1], Amari has shown that asymptotically, for large t, the weight $\boldsymbol{V}_{(t)}$ is a random Gausian vector $N(\hat{\boldsymbol{V}}_N, \eta\Sigma_{\boldsymbol{V}_t})$, where $\Sigma_{\boldsymbol{V}_t}$ is a covariance matrix that depends on data, a configuration of the network and peculiarities of the cost function near the optimum. In optimization, in order to find the minimum exactly the parameter η should converge to zero with a certain speed.

Random character of the hidden units' weight vectors $\boldsymbol{V}_{(t)}$ suggests that the weights and outputs of the hidden layer neurons are random variables. Thus, *we have a chaos (a process noise) inside the feed-forward neural network*. Random outputs serve as random inputs to the single layer perceptrons of the output layer. Consequently, in the output layer, *we have a data jittering*. A noise variance depends on the value of the learning-step η_h used to adapt the hidden units.

Hence, in MLP training, the learning-step η_h in the hidden layer, the traditional noise injection, the weight decay term as well as the number of iterations are producing *similar effects*. Here, we remind once more that when the hidden units' weights become large the nonlinear character of the activation function flattens the minima of the cost function, reduces the gradient and diminishes the variance of the noise injected to the output layer. Hence, we have an automatic reduction of the process noise variance.

Example 3. The effects of regularization of MLP (MLP with 8 inputs, 8 hidden neurons and one output) performed by three different, but at the same time similar techniques: weight decay, jittering of the training data and by "jittering the inputs of the output layer: controlled by η_h, the learning-step of the hidden neurons (*eta hidden*), are illustrated in Fig. 3. In this experiment, we use the artificial non-Gaussian data where vectors of the first class are inside the second class. The training sets size was N=140 (70 observation vectors from each class), and the test set Nt =1000 = 500+500. In Fig. 3 we have average values of the generalization error estimated from 7 experiments.

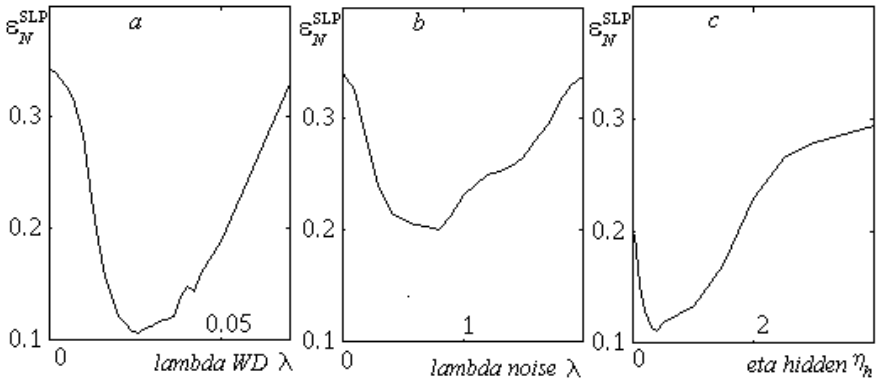

Fig. 3. Three types of regularization of the MLP classifier: *a* - weight decay, *b* -uniform noise injection into the inputs, *c* - learning step of the hidden layer neurons.

This experiment confirms the theoretical conclusion that the learning step η_h used to train the hidden layer neurons have a similar effect as the weight decay and a noise injection. The parameter η_h can be used to control the classifier's complexity. At the same time, the learning step is the obligatory parameter in BP training. Hence, the learning step η_h acts in inexplicit way and affects influences of other regularization tools like the weight decay parameter, the variance of a noise added to inputs, the target values and others.

11 Sigmoid Scaling

The SLP classifier performs classification of an unknown vector $\boldsymbol{X}$ according to a sign of the discriminat function $g(\boldsymbol{X}) = v_1 x_1, v_2 x_2, \ldots, v_n x_n + v_0$. Multiplication of all weights, $v_1, v_2, \ldots, v_n, v_0$, by a positive scalar α does not change the decision boundary. In the MLP classifiers, however, the effect of the multiplication of the hidden layer weights has more important consequences. A proportional increase in the magnitude of the hidden units weights forms the decision boundary with sharp angles. A proportional decrease in the magnitudes of all hidden layer weights smoothes the sharp angles and changes the complexity of the MLP classifier. Thus, a control of the magnitudes of the hidden layer weights is one of possible techniques which could be utilized to determine the classifier's complexity.

Example 4. In Fig. 4 we have 15 + 15 bi-variate training vectors from two artificially generated pattern classes where vectors from the first class are inside the second one and there is a ring-shaped margin between the pattern classes. An optimal (Bayes) decision boundary is depicted as a bold circle 3 in Fig. 4. *a*.

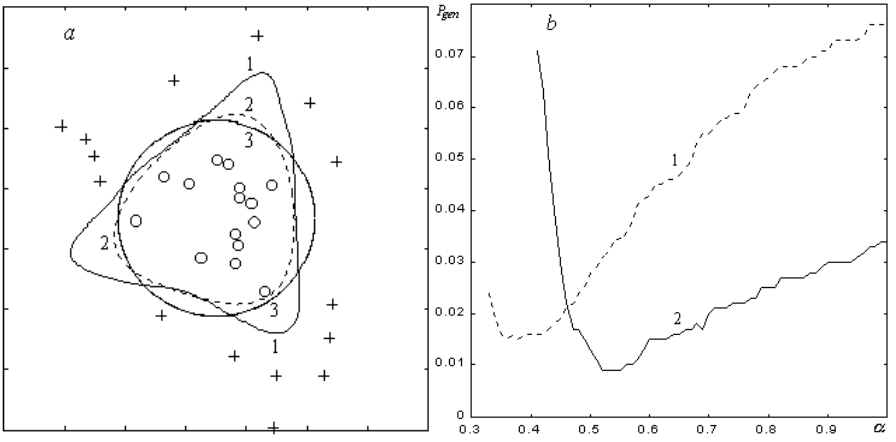

Fig. 4. *a:* 1- decision boundary of overtrained MLP, 2 - the smoothed by the factor α=0.53 decision boundary of the optimally stopped MLP, 3 - an optimal (Bayes) decision boundary. *b* - the generalization error as a function of the scaling parameter α: 1 - scaling of the hidden layer weights of the overtrained perceptron, 2 - scaling of the optimally stopped perceptron.

After training MLP with 10 hidden units for 10000 iterations in a batch mode, we obtained a nonlinear decision boundary 1 with 7.6 % generalization error (for estimation we used 500+500 test vectors). In this experiment, we observed a significant overtraining: an optimal stopping resulted in two times smaller (3.4 %) generalization error. The proportional reduction of all hidden layer weights by factor α=0.35 smoothed the decision boundary and diminished the generalization error until 1.5 % (see a behavior of curve 1 in Fig. 7). A notably better smooth decision boundary (curve 2 in Fig 4 *a*) was obtained after smoothing of the decision boundary of the optimally stopped MLP - the generalization error was reduced from 3.4 % of errors until 0.09 % (curve 2 in Fig 4 *b* with a minimum at α_{opt} = 0.53). Thus, in order to reduce the generalization error we had to optimize both the number of iterations *t* and the weights scaling parameter α.

As a possible generalization of the weights scaling technique, the magnitudes of the weights can be controlled individually for each hidden unit. An alternative to the scaling of the hidden layer weights by factor α is the supplementary regularization term $\lambda(\boldsymbol{V}_{hidden}^{T}\boldsymbol{V}_{hidden} - h^2)^2$ added to the cost function. Here the parameter *h* controls the magnitudes of the hidden layer weights $\boldsymbol{V}_{hidden}$.

At an end **we summarize** that the statistical hypothesis about data structure utilized to transform the data prior to train the perceptron highly affect the weights initialization, complexity and the performance of the SLP and MLP classifiers. All parameters of the cost function and the optimization procedure are acting together and jointly affect the complexity. While seeking for an optimal estimate of one of them the researcher should have in mind the values of other complexity control parameters.

References

1. Amari, S.: A theory of adaptive pattern classifiers. - IEEE Trans. Electron. Computers, EC-16 (1967) 623-625
2. An, G. : The effects of adding noise during backpropagation training on generalization performance, Neural Computation, 8 (1996) 643-674
3. Bishop, C..M.: Neural Networks for Pattern Recognition. Oxford Univ. Press (1995)
4. Devijver, P.A., Kittler, J. Pattern Recognition. A statistical approach, Precentice-Hall International, Inc., London (1982).
5. Duin, R.P.W.: Small sample size generalization. *Proc. 9th Scandinavian Conference on Image Analysis*, June 6-9, 1995, Uppsala, Sweden (1995)
6. Kanal L., Chandrasekaran, B.: On dimensionality and sample size in statistical pattern recognition. Pattern Recognition, 3 (1971) 238-255
7. Mao, J. Jain, A.: Regularization techniques in artificial neural networks .In: Proc. World Congress on Neural Networks, Portland, July 1993.
8. Raudys, S. On the problems of sample size in pattern recognition. In: Proc., 2nd All-Union Conf. Statist. Methods in Control Theory; Moscow, Nauka, 64-67 (1970)
9. Raudys S.: Evolution and generalization of a single neurone. I. SLP as seven statistical classifiers. Neural Networks, 11 (1998) 283-296.
10. Raudys S.: Statistical and Neural Classification Algorithms. An Integrated Approach. Springer, London (2001)
11. Raudys S, Skurikhina, M.,.Cibas, T., Gallinari, P.: Ridge estimates of the covariance matrix and regularization of artificial neural network classifier. Pattern Recognition and Image Processing, Int. J. of Russian Academy of Sciences, Moscow, 1995, N4 633-650
12. Reed R.: Pruning Algorithms - A Survey. IEEE Transactions on Neural Networks, 4, (1993) 740-747
13. Reed R., Marks II, R.J., Oh, S.: Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter, IEEE Transactions on Neural Networks, 6 (1995) 529-538.
14. Skurichina M., Raudys, S.. Duin, R.P.W.: K-nearest neighbors directed noise injection in multilayer perceptron training, IEEE Trans. on Neural Networks, 11 (2000) 504-511
15. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Berlin (1995)