PERGAMON

Neural Networks 13 (2000) 507–523

Neural Networks

www.elsevier.com/locate/neunet

Evolution and generalization of a single neurone. III. Primitive, regularized, standard, robust and minimax regressions

Š. Raudys*

Institute of Mathematics and Informatics, Akademijos 4, Vilnius 2600, Lithuania

Received 2 November 1998; accepted 16 February 2000

Abstract

We show that during training the single layer perceptron, one can obtain six conventional statistical regressions: a primitive, regularized, standard, the standard with the pseudo-inversion of the covariance matrix, robust, and minimax (support vector). The complexity of the regression equation increases with an increase in the number of iterations. The generalization accuracy depends on the type of the regression obtained during the training, on the data, learning-set size, and, in certain cases, on the distribution of components of the weight vector. For small intrinsic dimensionality of the data and certain distributions of components of the weight vector the single layer perceptron can be trained even with very short learning sequences. The type of the regression obtained in SLP training should be controlled by the sort of cost function as well as by training parameters (the number of iterations, learning step, etc.). Whitening data transformation prior to training the perceptron is a tool to incorporate a prior information into the prediction rule design, and helps both to diminish the generalization error and the training time.

Keywords: Single-layer perceptron; Prediction; Regression; Generalization error; Overtraining; Dimensionality; Complexity; Sample size

1. Introduction

Two most important problems in statistical data analysis are prediction and classification (0, 1 loss). In the prediction task, we are trying to predict the value of a continuous variable, say y, according to a set of predictors $(x_1, x_2, \ldots, x_p) = \mathbf{x}'$. An example is a linear regression $y = \mathbf{w}'\boldsymbol{x} + w_0$, where w_0 and $\mathbf{w} = (w_1, w_2, \ldots, w_p)'$ are the weights of the regression equation. In order to find the weights, one needs a training-set data $(y_1, \boldsymbol{x}_1)$, $(y_2, \boldsymbol{x}_2)$, …, $(y_N, \boldsymbol{x}_N)$. In the artificial neural network (ANN) approach to find the weights, one minimizes a certain cost (loss) function, e.g.

$$\text{Cost} = \frac{1}{N}\sum_{i=1}^{N} \Psi(y_i - f(\mathbf{w}'\boldsymbol{x}_i + w_0)), \tag{1}$$

where $\Psi(c)$ is a chosen pattern error (loss) function, and $f(s)$ a non-linear activation function. In the classification task we have 0–1 loss, i.e. the loss is equal 0 if the vector to be classified is recognized correctly, and the loss is equal to 1 if we have a misclassification. In the prediction problem, we have a continuous loss. The most popular loss function is the sum of squares function $\Psi(c) = c^2$. In prediction, one can use a scaled sigmoid activation function $f(s) = 1/(1 + \exp(-\alpha s)) - 0.5$, where α is a scaling parameter. If $f(s) = s$, and $\Psi(c) = c^2$ we have a linear single layer perceptron (SLP). If $f(s)$ is a non-linear function, we have a non-linear SLP. Note, if the perceptron's weights are small, all values $s_i = \mathbf{w}'\boldsymbol{x}_i + w_0$ are small too. Then $f(s) = s$, and, practically, we have the linear SLP.

In the ANN approach, the weight vector is found in an iterative procedure where the cost function is minimized. One of the simplest procedures is the delta rule (back propagation, gradient descent) where the weight vector is adapted according to iterative rule

$$\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} - \eta\frac{\partial \text{cost}_l}{\partial \mathbf{w}}, \tag{2}$$

where η is a learning-step.

In Part I (Raudys, 1998a) it was shown that in training, the *non-linear SLP classifier evolves*. In the evolution process, the weights of the perceptron increase, and the cost function of the sum of squares changes gradually. If certain conditions are satisfied, the decision boundary of SLP can become identical or close to that of seven regular statistical classifiers: (1) the Euclidean distance classifier; (2) the regularized linear discriminant analysis; (3) the standard Fisher linear discriminant function; (4) the Fisher

* Fax: + 370-2-729-209.
E-mail address: raudys@das.mii.lt (Š. Raudys).

0893-6080/00/$ - see front matter
PII: S0893-6080(00)00025-3

linear discriminant function with a pseudo-inversion of the covariance matrix; (5) the generalized Fisher discriminant function; (6) the minimum empirical error classifier, and (7) the maximum margin (support vector) classifier. The complexity of the classifier changes during the training process. At first, we estimate only mean vectors of the pattern classes, then gradually we begin to estimate the covariance matrix, and, at the very end of the training process, we use only highest order statistical moments.

One may hope that in the prediction task (regression), we also have a similar evolution process. Up to now, it was known that at the beginning of the training of the linear single layer perceptron, we have the regularization and at the end, we have a standard sum of squares regression (Sjoberg & Ljung, 1992; Wang & Venkatesh, 1994). In the finite learning-set case, the generalization prediction error of the standard regression is determined by a simple expression (Davisson, 1965)

$$E\sigma^2_{\text{generalization}} = \sigma^2_{\text{ideal}} \frac{N}{N-p}. \tag{3}$$

The above equation shows that the generalization error decreases when the learning set size N increases. When N is smaller than p, the dimensionality, an asymptotic analysis performed by a statistical mechanics approach shows that the learning curve $E\sigma^2_{\text{prediction}} = f(N)$ has a peaking character: when N increases from 1 up to p, the generalization error decreases at first, and then begins to increase (for the prediction task see e.g. Bös, 1996; Krogh & Hertz, 1992; and for the classification task, Raudys, 1998b; Raudys & Duin, 1998). For the regularized regression obtained after few first iterations, we have much more complex equations (Wang & Venkatesh, 1994). One may hope that while training the non-linear SLP, one can obtain a wide assortment of regression rules too. The objective of the present paper is to consider this problem more exhaustively. A thorough unified analysis is most easy to accomplish in the simple SLP case. This analysis gives new information and ideas, which help to understand the complex learning behavior of multilayer perceptrons. In Section 2, we present several types of the linear regressions commonly used in statistical data analysis. In Section 3, we analyze regressions that can be obtained while training the non-linear SLP by the standard back propagation rule (BP) and by using various types of pattern error function $\Psi(c)$. In Section 4, we use standard multivariate analysis techniques to derive the generalization error formulae for the regressions discussed in the previous two sections. We show that there, similarly to the classification task (see e.g. Raudys, 1967, 1998b), the intrinsic dimensionality of the data, as well as components of the optimal weight vector $\mathbf{w}^*$ play an important role. In Section 5, we discuss tools and criteria which can be used to control the type and complexity of the regression equation obtained while training the SLP. In Section 6, we present a discussion and concluding remarks.

2. Primitive, regularized, standard, robust and minimax (support vector) regressions in statistical data analysis

2.1. Standard regression

In order to simplify analytical derivations in a major part of this paper, without loss of generality we assume that the learning set data is "normalized", i.e. the sample mean of vector

$$\begin{bmatrix} y \\ \mathbf{X} \end{bmatrix}$$

is equal to zero. For linear regressions, this assumption leads to $\hat{w}_0 = 0$. In the standard least square approach, we use a sum of squares pattern error function $\Psi(c) = c^2$. After minimization of the sum of the squares cost function we obtain

$$\hat{w}^{\text{ST}} = \mathbf{S}_{XX}^{-1}\mathbf{S}_{Xy}, \tag{4}$$

where $\mathbf{S}_{XX}$ and $\mathbf{S}_{Xy}$ are block components of a conventional sample maximum likelihood estimate of the covariance matrix

$$\Sigma = \begin{bmatrix} \Sigma_{yy} & \Sigma_{yX} \\ \Sigma_{Xy} & \Sigma_{XX} \end{bmatrix}$$

of a vector

$$\begin{bmatrix} y \\ \mathbf{X} \end{bmatrix}.$$

In multivariate analysis, the weight vector (4) is obtained as a conditional mean $\mathrm{E}[y|\mathbf{X}]$ of the Gaussian vector

$$\begin{bmatrix} y \\ \mathbf{X} \end{bmatrix}.$$

For this, instead of true components of the covariance matrix Σ(we use sample estimates. We refer to this type of regression as *the standard* one.

2.2. "Primitive" regression

Some investigators in the field of applied research are unfamiliar with statistical methods. They normalize the data by making all sample variances of y and $\mathbf{X}$ to be equal to 1, and intuitively are using correlation coefficients between y and components of the vector $\mathbf{X}$ as components of the weight vector $\mathbf{w}$. In this case, we have a weight vector

$$\hat{\mathbf{w}}^{\text{PRIM}} = \mathbf{S}_{Xy}, \tag{5}$$

We will refer to this type of regression as the *primitive regression*. Note, if $\Sigma = \mathbf{I}$, then in the multivariate Gaussian case, with an increase in N, the learning-set size, this approach leads to the optimal prediction rule. Otherwise we will obtain a bias.

2.3. Regularized regression

When N, the number of the learning-set vectors, is small, one cannot invert the sample covariance matrix S_{XX}. One of the possibilities to overcome this kind of difficulty is to add small positive constants, λ, to all diagonal elements of the matrix S_{XX} (Harley, 1963, 1965; Hoerl & Kennard, 1970). This is *regularized regression* (RR):

$$\hat{\mathbf{w}}^{\mathrm{RR}} = (\mathbf{S}_{XX} + \mathbf{I}\lambda)^{-1}\mathbf{S}_{Xy}. \tag{6}$$

2.4. Standard regression with a pseudo-inverse

When the sample size N is smaller than p, the number of features, the matrix S_{XX} becomes singular. An alternative to regularized regression is to *ignore directions* corresponding to zero eigenvalues. For this we have to perform a singular value decomposition of $\mathbf{S}_{XX}$: $\mathbf{S}_{XX} = \mathbf{TDT}'$. Let $\mathbf{d}$ be the $r \times r$ diagonal matrix corresponding to r non-zero eigenvalues in $\mathbf{D}$, and r be the rank of S_{XX}. Then the pseudo-inverse of the matrix $\mathbf{S}_{XX}$ is defined as

$$\mathbf{S}_{XX}^{+} = \mathbf{T}\begin{bmatrix} \mathbf{d}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\mathbf{T}'.$$

Regularized regression can also be expressed in terms of eigenvalues and eigenvectors: $\mathbf{S}_{XX} + \mathbf{I}\lambda = \mathbf{T}(\mathbf{D} + \mathbf{I}\lambda)\mathbf{T}'$. Thus, in regularized regression, we are adding constant λ to all eigenvalues of matrix $\mathbf{S}_{XX}$.

2.5. Robust regression

In various real world problems, sometimes we have atypical training samples where one or several components are recorded with errors or affected by some abnormal noise. In statistics, these atypical observation vectors are called *outliers*. The outliers affect the estimates of the mean vector and the covariance matrix of

$$\begin{bmatrix} y \\ \mathbf{X} \end{bmatrix}$$

and reduce the prediction accuracy. There are a number of techniques called *robust statistics* to deal with such problems (see e.g. Huber, 1981). In a regular robust approach, instead of the standard maximum likelihood estimate of the matrix

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{yy} & \mathbf{S}_{yX} \\ \mathbf{S}_{Xy} & \mathbf{S}_{XX} \end{bmatrix},$$

one uses a robust (weighted) estimate. As the robust estimates of the mean and the covariance matrix one can use

$$\hat{\mu}_{\mathrm{ROBUST}} = \frac{\sum_j \gamma(\mathbf{X}_j)\mathbf{X}_j}{\sum_j \gamma(Xj)}, \qquad \text{and}$$

$$\hat{\Sigma}_{\mathrm{ROBUST}} = \frac{\sum_j \gamma(\mathbf{X}_j)(\mathbf{X}_j - \hat{\mu}_{\mathrm{ROBUST}})(\mathbf{X}_j - \hat{\mu}_{\mathrm{ROBUST}})'}{\sum_j \gamma(\mathbf{X}_j)}, \tag{7}$$

where $\gamma(\mathbf{X}_j)$ is a weighting factor, which decreases monotonically with increase in distance

$$D(\mathbf{X}_j, \hat{\mu}_{\mathrm{ROBUST}}) = (\mathbf{X}_j - \hat{\mu}_{\mathrm{ROBUST}})'(\mathbf{X}_j - \hat{\mu}_{\mathrm{ROBUST}}).$$

For example

$$\gamma(\mathbf{X}_j) = 1 \text{ if } D(\mathbf{X}_j, \hat{\mu}_{\mathrm{ROBUST}}) < 2p, \text{ and } D(\mathbf{X}_j - \hat{\mu}_{\mathrm{ROBUST}})$$
$$= 1/D(\mathbf{X}_j - \hat{\mu}_{ROBUST}) \text{ otherwise.}$$

2.6. Minimax (support vector) regression

Real world problems exist where one can *assume* that in the linear model $y = \mathbf{w}\mathbf{x}' + w_0 + \xi$, a noise ξ is a random variable distributed uniformly in an unknown interval $(-a,b)$. In such cases, instead of minimizing the sum of squares cost (1), some researchers minimize *a maximal deviation* of learning-set observations y_j from predicted values $y_j^{\mathrm{predicted}} = \hat{\mathbf{w}}'\mathbf{x}_j + w_0$, $j = 1, 2, \ldots, N$. This type of regression is called *a minimax regression*. Only a small number (R) of most distant vectors $\mathbf{X}_{\mathrm{SV1}}$, $\mathbf{X}_{\mathrm{SV2}}$, …, $\mathbf{X}_{\mathrm{SVR}}$, determine a position of the prediction equation $y = \hat{\mathbf{w}}'\mathbf{x} + w_0$. These R vectors are referred to as *supporting vectors*. Then the prediction (regression) rule is called *support vector regression*. It is analogous to the support vector classification rule (see e.g. Cortes & Vapnik, 1995) and SV regression (Drucker, Burges, Kaufman, Smola & Vapnik, 1996).

3. Evolution of the SLP in the training process

3.1. Primitive regression

Consider the iterative gradient descent training procedure (2) of the linear single layer perceptron where a sum of squares cost function

$$\mathrm{Cost} = \frac{1}{2}\frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}'\mathbf{x}_i)^2, \tag{8}$$

is minimized. In the above equation, we added 1/2 for cosmetic reasons, and according to the assumptions

$$\sum_{i=1}^{N} y_i = 0, \qquad \sum_{i=1}^{N} \mathbf{x}_i = \mathbf{0},$$

we have skipped the threshold w_0. Then

$$\frac{\partial \text{cost}}{\partial \mathbf{w}} = -\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i y_i - \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i'\mathbf{w}_{(t)}\right) = -\mathbf{S}_{Xy} + \mathbf{S}_{XX}\mathbf{w}_{(t)}, \tag{9}$$

and

$$\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} + \eta(\mathbf{S}_{Xy} - \mathbf{S}_{XX}\mathbf{w}_{(t)}) = (\mathbf{I} - \eta\mathbf{S}_{XX})\mathbf{w}_{(t)} + \eta\mathbf{S}_{Xy}. \tag{10}$$

Let us start training from zero initial weights, i.e. $\mathbf{w}_{(0)} = \mathbf{0}$. Then after the first iteration we have

$$\mathbf{w}_{(1)} + \eta\mathbf{S}_{Xy}. \tag{11}$$

The above equation indicates that for $\eta = 1$ after the first iteration we obtain primitive regression. Let now, prior to training, we perform *a linear transformation of the data*: $\mathbf{Z} = \mathbf{FX}$, where $\mathbf{F} = \mathbf{D}^{-1/2}\mathbf{T}'$, and $\mathbf{D}$ is the $p \times p$ diagonal matrix composed from eigenvalues of matrix $\mathbf{S}_{XX}$, and $\mathbf{T}$ is the $p \times p$ orthonormal matrix of the eigenvectors of $\mathbf{S}_{XX}$. Then after the first iteration

$$\mathbf{w}_{\mathbf{Z}(1)} = \frac{\eta}{N}\sum_j \mathbf{Z}_j y_j = \frac{\eta}{N}\sum_j \mathbf{D}^{-1/2}\mathbf{T}'\mathbf{X}_j y_j = \eta\mathbf{D}^{-1/2}\mathbf{T}'\mathbf{S}_{Xy},$$

and the prediction equation

$$y(\mathbf{Z}) = \mathbf{Z}'\mathbf{w}_{Z(1)} = \eta(\mathbf{X}'\mathbf{T}\mathbf{D}^{-1/2})\mathbf{D}^{-1/2}\mathbf{T}'\mathbf{S}_{X_y} = \mathbf{X}'\eta\mathbf{S}_{XX}^{-1}\mathbf{S}_{Xy} = \eta\mathbf{X}'\hat{\mathbf{w}}^{\mathrm{ST}}, \tag{12}$$

where $\hat{\mathbf{w}}^{\mathrm{ST}}$ has been defined in Eq. (4).

Thus, for $\eta = 1$, after the first iteration in the transformed $\mathbf{Z}$ space, we obtain the standard regression in the original $\mathbf{X}$ space. This is an important conclusion, which can be used to incorporate additional information (statistical hypothesis concerning the distribution model of vector $\mathbf{X}$) into the perceptron design. Actually it is a way how to integrate statistical and neural network approaches to design the linear prediction equation.

3.2. Regularized regression

Consider the learning process further in the second and following iterations. After the second iteration we obtain

$$\mathbf{w}_{(2)} = (2\eta\mathbf{I} - \eta^2\mathbf{S}_{XX})\mathbf{S}_{Xy}, \qquad \text{and further}$$

$$\mathbf{w}_{(3)} = (3\eta\mathbf{I} - 3\eta^2\mathbf{S}_{XX} + \eta^3\mathbf{S}_{XX}^2)\mathbf{S}_{Xy}, \qquad \vdots$$

$$\mathbf{w}_{(t)} = \sum_{s=1}^{t} C_t^s(-1)^{s-1}\eta^s\mathbf{S}_{XX}^{s-1})\mathbf{S}_{Xy} = [\mathbf{I} - (\mathbf{I} - \eta\mathbf{S}_{XX})^t]\mathbf{S}_{XX}^{-1})\mathbf{S}_{Xy} \tag{13}$$

For small η, we can use only the first terms of Eq. (13)

$$\mathbf{w}_{(t)} = \left(t\eta\mathbf{I} - \frac{t(t-1)}{2}\eta^2\mathbf{S}_{XX}\right)\mathbf{S}_{Xy} = t\eta[\mathbf{I} - \lambda\mathbf{S}_{XX})]\mathbf{S}_{Xy}, \tag{14}$$

where $\lambda = [(t-1)\eta/2]$.

Use of an expansion $(\mathbf{I} - \lambda\mathbf{S}_{XX})^{-1} = \mathbf{I} + \lambda\mathbf{S}_{XX} - \ldots.$ for very small λ results in

$$\mathbf{w}_{(t)} = \frac{2t}{t-1}(\mathbf{S}_{XX} + \lambda\mathbf{I})^{-1}S_{Xy}. \tag{15}$$

For small η (and t too) the resulting weight vector is proportional to that of the regularized regression $\hat{\mathbf{w}}^{\mathrm{RR}}$ (Eq. (6)). This means, after the first few iterations we have a notable regularization (large $\lambda = [2/(t-1)\eta]$). The degree of regularization diminishes monotonically with increase in t, the number of iterations.

3.3. Standard regression

Eq. (15) is valid only when $\lambda = [(t-1)\eta/2]$ is small. In addition, it does not show the behavior of the weight vector when the number of iterations increases without bound. Let the learning step η diminish with increase in the number of iterations with a certain speed. Then we can arrive to a minimum of the cost function (see e.g. Amari, 1967). Equating (9) to zero results in

$$\mathbf{w}_{(t\to\infty)} = \mathbf{S}_{XX}^{-1}\mathbf{S}_{Xy} = \hat{\mathbf{w}}^{\mathrm{ST}}, \tag{16}$$

i.e. the standard regression for the centered data.

3.4. Standard regression with the pseudoinversion of the covariance matrix

In Sections 3.2 and 3.3, we have assumed the sample covariance matrix $\mathbf{S}_{XX}$ to be non-singular. When p, the dimensionality of vector $\mathbf{X}$, exceeds N, the number of learning examples, this matrix becomes singular. Let the rank of $\mathbf{S}_{XX}$ be r, and the singular value representation

$$\mathbf{S}_{XX} = \mathbf{T}\begin{bmatrix}\mathbf{d} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{T}',$$

where $\mathbf{T}$ is the orthogonal eigenvector matrix, and $\mathbf{d}$ the $r \times r$ diagonal eigenvector matrix corresponding to r non-zero eigenvalues of $\mathbf{S}_{XX}$. Let us denote

$$\mathbf{U} = \mathbf{T}'\mathbf{X} = \begin{bmatrix}\mathbf{V}\\ \mathbf{U}_2\end{bmatrix}, \qquad \mathbf{S}_{UU} = \mathbf{T}'\mathbf{S}_{XX}\mathbf{T} = \begin{bmatrix}\mathbf{d} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix},$$

$$\mathbf{S}_{Uy} = \mathbf{T}'\mathbf{S}_{Xy} = \begin{bmatrix}\mathbf{S}_{Vy}\\ \mathbf{S}_{U_2y}\end{bmatrix},$$

where $\mathbf{V}$ is r-variate vector column. Utilization of the representation (13) gives that after the tth iteration, the regression

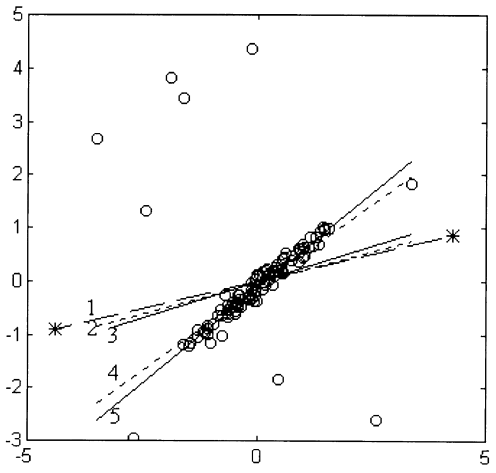

Fig. 1. Robust SLP regression: 1—the standard regression; 2,4—Robust SLP (18) with $\alpha = 0.3$, and $\alpha = 6$ (Graph 4); 3,5—Robust SLP (22) with $\alpha = 0.3$, and $\alpha = 5$ (Graph 5).

equation

$$y_{(t)}(\mathbf{X}) = \mathbf{X}'\mathbf{w}_{(t)} = \mathbf{X}'\mathbf{T}\mathbf{T}'\mathbf{w}_{(t)}$$

$$= \mathbf{X}'\mathbf{T}\sum_{s=1}^{t} C_t^s(-1)^{s-1}\eta^s(\mathbf{T}'\mathbf{S}_{XX}\mathbf{T})^{s-1})\mathbf{T}'\mathbf{S}_{Xy}$$

$$= \mathbf{V}'\sum_{s=1}^{t} C_t^s(-1)^{s-1}\eta^s\mathbf{d}^{s-1}\mathbf{S}_{Vy}. \quad (17)$$

Representation (17) means that in this (singular) case, the iterative search for the weights of the perceptron is performed in a subspace of r eigenvectors corresponding to r non-zero eigenvalues of $\mathbf{S}_{XX}$. This is a more condensed and rigid proof than that presented in the first part of the paper (Raudys, 1998a). This proof is also valid for the coefficients of the linear classifier obtained while training the SLP classifier in the very small learning-set case.

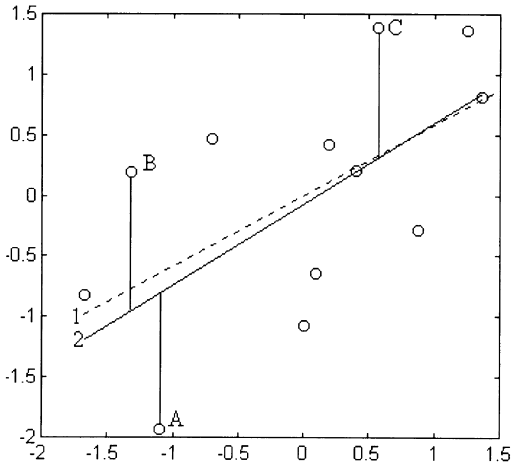

Fig. 2. Minimax (support vector) regression: 1—the standard regression, 2—the minimax SLP predictor for $\alpha = 10$; A, B, C—supporting vectors.

3.5. Robust regression

Up to this moment we have considered the linear single-layer perceptron. Consider now the *non-linear* SLP trained with the following cost function

$$\text{Cost} = \frac{1}{2}\frac{1}{N}\sum_{i=1}^{N}(f(y_i) - f(\mathbf{w}'\mathbf{x}_i + w_0))^2, \quad (18)$$

where $f(s)$ is a non-linear activation function which saturates when $|s|$ increases (again we assume sample means, $\bar{y}$ and $\bar{\mathbf{X}}$, to be zero). For example

$$f(s) = \frac{1}{1 + \mathrm{e}^{-\alpha s}} - 0.5, \quad (19)$$

where a positive scalar α(controls the degree of non-linearity—the proportion of observations with reduced influence while determining the weights of the regression.

The non-linear activation function diminishes the influence of overly small and overly large values of y_i (as well as $\mathbf{w}'\mathbf{x}_i + w_0$) and reduces the influence of outliers distant from $\bar{y} = (1/N)\sum_i y_i$ and $\mathbf{w}'\bar{\mathbf{X}}$, ($\bar{\mathbf{X}} = (1/N)\sum_i \mathbf{X}_i$). In comparison with standard robust regression (see Section 2.5), an effect of only a portion of the outliers is reduced.

Another possible cost function, which can be used to obtain the robust regression is

$$\text{Cost} = \frac{1}{2}\frac{1}{N}\sum_{i=1}^{N}\Psi(\alpha(y_i - (\mathbf{w}'\mathbf{x}_i + w_0))), \quad (20)$$

where and $\Psi(s)$ is a non-linear pattern error function, which saturates where a difference $|s| = |y_i - (\mathbf{w}'\mathbf{x}_i + w_0)|$ is large. The positive scalar α controls the degree of non-linearity—the proportion of ignored observations (robustness). An example of the robust pattern error function can be

$$\Psi(s) = \begin{cases} 1 - \cos(s) & \text{if } -\pi \le s \le \pi \\ 2 & \text{otherwise} \end{cases}. \quad (21)$$

This cost functions ignores observations with large prediction errors. When α is small almost all observations contribute to the regression coefficients. When α is large, only a proportion of observations is taken into account. In Fig. 1, we present an example of the use of adaptive robust regression. The Gaussian strongly correlated data is contaminated by Gaussian large variance noise. We see the standard regression (line 1) fails to find the right prediction function. Robust adaptive regression with cost (18) and $\alpha = 6$ performs much better (line 4). Robust adaptive regression with cost functions (20) and (21) and $\alpha = 5$ (line 5) practically ignores all atypical observation vectors, but fails to do this when α(is small (line 3).

3.6. Minimax (support vector) regression

In order to minimize the maximal deviations $|y_i - (\mathbf{w}'\mathbf{x}_i + w_0)|$ among all N training vectors, one can use the cost function (20) with the activation function whose value

$\vartheta(s)$ increases enormously with $|s|$. An example

$$\varphi(s) = \exp(\alpha s^2) - 1. \tag{22}$$

A relative contribution of each learning vector into the sum (20) is determined by a scaling coefficient α and the distance $|y_i - (\mathbf{w}'\mathbf{x}_i + w_0)|$. Learning vectors with small deviations $|y_i - (\mathbf{w}'\mathbf{x}_i + w_0)|$ have zero or very small contribution into the cost function. In a limit, where α is large, and the training process finishes, only a small number of the most distant vectors contribute into the cost function. Therefore, only a small number of training vectors determine the values of the regression coefficients. We refer these vectors as *supporting ones*. In Fig. 2, we demonstrate a utilization the SLP to obtain the minimax (support vector) regression. In this example, the variable x is Gaussian, positive deviations ξ of variable $y = x + \xi$ are distributed uniformly in the interval (0,0.5), while negative deviations appear in the interval (−4,0). Only three training vectors (A, B, and C) serve as the supporting vectors. We see a notable difference between the standard and minimax regressions.

In order to find the maximal deviations $|y_i - (\mathbf{w}'\mathbf{x}_i + w_0)|$ exactly we need to use large α values. Two numerical problems arise. First of all, large α values cause many local minima of the cost function to appear in the multivariate weight space. In addition, for large α, the convergence of the algorithm becomes unstable. One needs significant effort in order to choose the proper learning-step value. Our experiments have shown that a good strategy is to increase α with any increase in the number of iterations.

4. The generalization error

In order to obtain an explicit formula for an expected square prediction error $\mathrm{E}\varepsilon^2 = \mathrm{E}(y - y_{\mathrm{pred}})^2$ of the first four linear predictors obtained in the linear SLP back propagation training we assume

$$y = \mathbf{X}'\mathbf{w}^* + w_0^* + \xi, \tag{23}$$

where $\mathbf{w}^* = (w_1^*, w_2^*, \ldots, w_p^*)'$, w_0^* are true weight values, $\mathbf{X}$ is a random Gaussian N($\mathbf{0}$, Σ) vector, and ξ is independent zero Gaussian N$(0, \sigma^2)$ noise. Moreover, in order to simplify the analysis assume $\bar{y} = 0$ and $\bar{\mathbf{X}} = 0$. Then for the linear models we have $\hat{w}_0 = 0$.

4.1. Standard regression

For model (23), the weight vector

$$\hat{\mathbf{w}}^{\mathrm{STAND}} = \left(\frac{1}{N}\sum_j \mathbf{X}_j'\right)^{-1} \frac{1}{N}\sum_j \mathbf{X}_j y_j,$$

can be rewritten in a following way:

$$\hat{\mathbf{w}} = \left(\frac{1}{N}\sum_j \mathbf{X}_j\mathbf{X}'_j\right)^{-1}\left(\frac{1}{N}\sum_j \mathbf{X}_j(\mathbf{X}'_j\mathbf{w}^* + \xi_j)\right)$$

$$= \mathbf{w}^* + \left(\frac{1}{N}\sum_j \mathbf{X}_j\mathbf{X}'_j\right)^{-1}\left(\frac{1}{N}\sum_j \mathbf{X}_j\xi_j\right). \tag{24}$$

Then for vector $(y, \mathbf{X}')'$ the prediction error

$$y_{\mathrm{pred}} - y = \mathbf{X}'\hat{\mathbf{w}} - \mathbf{X}'\mathbf{w}^* - \xi = \mathbf{X}'(\hat{\mathbf{w}} - \mathbf{w}^*) - \xi$$

$$= \mathbf{X}'\left(\frac{1}{N}\sum_j \mathbf{X}_j\mathbf{X}_j'\right)^{-1}\left(\frac{1}{N}\sum_j \mathbf{X}_j\xi_j\right) - \xi.$$

Taking into account that for independent learning-set vectors

$$\mathrm{E}\mathbf{X}\xi = \mathbf{0}, \qquad \mathrm{E}\frac{1}{N}\sum_j \mathbf{X}_j y_j = \mathbf{0},$$

$$\mathrm{E}\xi_i\xi_j = \begin{cases} 0 & \text{if } j \neq i \\ \sigma^2 & \text{if } j = i \end{cases},$$

and

$$\mathrm{E}\left(\frac{1}{N}\sum_j \mathbf{X}_j\mathbf{X}'_j\right)^{-1} = \Sigma\frac{N}{N-p-1},$$

the expected square prediction error

$$\mathrm{E}\varepsilon^2_{\mathrm{STANDARD}}$$

$$= \mathrm{E}\left\{\mathbf{X}'\left(\frac{1}{N}\sum_j \mathbf{X}_j\mathbf{X}'_j\right)^{-1}\left(\frac{1}{N^2}\sum_{ij} \mathbf{X}_j\xi_j\xi_i\mathbf{X}_i\right)\right.$$

$$\left.\times\left(\frac{1}{N}\sum_j \mathbf{X}_j\mathbf{X}'_j\right)^{-1}\mathbf{X}\right\} + \mathrm{E}\xi^2 = \sigma^2\left(1 + \frac{p}{N-p-1}\right). \tag{25}$$

The result does not depend on Σ. Expression (25) was firstly obtained by Davisson (1965).

4.2. Primitive regression

The weight vector

$$\hat{\mathbf{w}}^{\mathrm{PRIM}} = \frac{1}{N}\sum_j \mathbf{X}_j y_j = \left(\frac{1}{N}\sum_j \mathbf{X}_j(\mathbf{X}'_j\mathbf{w}^* + \xi_j)\right)$$

$$= \mathbf{S}_{XX}\mathbf{w}^* + \mathbf{S}_{X\xi},$$

where

$$\mathbf{S}_{XX} = \frac{1}{N}\sum_j \mathbf{X}_j\mathbf{X}'_j,\ \mathbf{S}_{X\xi} = \frac{1}{N}\sum_j \mathbf{X}_j\xi_j.$$

The prediction error

$$y_{\text{pred}} - y = \mathbf{X}'\hat{\mathbf{w}}^{\text{PRIM}} - \mathbf{X}'\mathbf{w}^* - \xi$$

$$= \mathbf{X}'(\mathbf{S}_{XX}\mathbf{w}^* + \mathbf{S}_{X\xi}) - \mathbf{X}'\mathbf{w}^* - \xi$$

$$= \mathbf{X}'(\mathbf{S}_{XX} - \mathbf{I})\mathbf{w}^* - \mathbf{X}'\mathbf{w}^* - \xi.$$

Then the expected square prediction error

$$\mathrm{E}\varepsilon^2 = \mathrm{E}\,\mathrm{tr}\{\mathbf{w}^*(\mathbf{S}_{XX} - \mathbf{I})\mathbf{X}\mathbf{X}'(\mathbf{S}_{XX} - \mathbf{I})\mathbf{w}'^* + \mathbf{S}_{X\xi}\mathbf{S}'_{x\xi}\mathbf{X}\mathbf{X}'\} + \mathrm{E}\xi^2.$$

By means of standard methods of multivariate statistical analysis, and after some simple but tedious algebra, we derive

$$\mathrm{E}\varepsilon^2_{\text{PRIMITIVE}} = \sigma^2 + T_{\text{A}} + T_{\text{B}}, \tag{26}$$

where

$$T_{\text{A}} = \mathbf{w}'_{\text{F}}*(\Delta - \mathbf{I})^2\mathbf{w}^*_{\text{F}},$$

$$T_{\text{B}} = \frac{1}{N}\sigma^2\{\mathrm{tr}(\Delta^2) + \mathbf{w}'_{\text{F}}*(\Delta^2 + \mathrm{tr}(\Delta^2)\mathbf{I})\mathbf{w}^*_{\text{F}}\},$$

Δ is the $p \times p$ diagonal matrix composed from eigenvalues of Σ, such that $\Gamma\Sigma\Gamma' = \Delta$, Γ is the orthonormal eigenvector matrix of Σ, and

$$\mathbf{w}^*_{\text{F}} = \Delta^{1/2}\Gamma\mathbf{w}^* = (w_1 = w_2 = \cdots = w_p)'.$$

To obtain Eq. (26) we had to prove the following three equalities:

$$\mathrm{E}\left\{\left(\sum_i \mathbf{U}_i\mathbf{U}'_i\right)\Delta^2\left(\sum_i \mathbf{U}_i\mathbf{U}'_i\right)\right\} = N\,\mathrm{tr}(\Delta^2)\mathbf{I} + N(N+1)\Delta^2,$$

$$\mathrm{tr}(\Sigma\mathbf{w}'^*\mathbf{w}^*) = \mathrm{tr}(\mathbf{w}^*_{\text{F}}\mathbf{w}'_{\text{F}}*),$$

and

$$\mathrm{tr}(\Sigma^2\mathbf{w}^*\mathbf{w}^{*'}) = \mathrm{tr}(\Delta\mathbf{w}^*_{\text{F}}\mathbf{w}'_{\text{F}}*) = \mathbf{w}'_{\text{F}}*\Delta\mathbf{w}^*_{\text{F}},$$

where $\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N$ are independent random Gaussian $N(\mathbf{0},\mathbf{I})$ vectors.

The expression (26) shows the prediction generalization error is composed from three terms:

- σ^2—a term which characterizes the variance of noise in the model (23)—this term is fixed and does not depend on the learning-set size;
- $T_{\text{A}} = \mathbf{w}'_{\text{F}}*(\Delta - \mathbf{I})^2\mathbf{w}^*_{\text{F}}$ indicates the price we pay for ignoring estimate $\mathbf{S}_{XX} = (1/N)\sum_j \mathbf{x}_j\mathbf{x}'_j$ of the covariance matrix Σ_{xx} (in the standard regression we do not ignore $\mathbf{S}_{XX}$, and do not have this term). The term T_{A} is non-negative. If $\Sigma = \mathbf{I}$, we have $\Delta = \mathbf{I}$. then $T_{\text{A}} = 0$;
- $T_{\text{B}} = (1/N)\sigma^2\{\mathrm{tr}(\Delta^2) + \mathbf{w}'_{\text{F}}*(\Delta^2 + \mathrm{tr}(\Delta^2)\mathbf{I})\mathbf{w}^*_{\text{F}}\}$, indicates the influence of the learning-set size. Contrary to the generalization error expression for the standard regression (25), this term depends both on the covariance matrix Σ, and on the optimal weight vector $\mathbf{w}^*$. If the covariance matrix $\Sigma = \mathbf{I}$, this term becomes $(1/N)\sigma^2(p + 2\mathbf{w}'^*\mathbf{w}^*)$.

Let ρ be a multiple correlation coefficient. It is easy to find

$$\mathbf{w}'^*\mathbf{w}^* = \frac{\sigma^2\rho^2}{1-\rho^2}.$$

Then

$$T_{\text{B}} = \frac{1}{N}\sigma^2\left(p + 2\frac{\sigma^2\rho^2}{1-\rho^2}\right).$$

We see that the sensitivity of primitive regression to the learning-set size depends of the true multiple correlation coefficient. For standard regression, we had an analogous term

$$T_{\text{B}}^{\text{Stand}} = \sigma^2 - \frac{p}{N-p-1},$$

which does not depend on the multiple correlation coefficient.

In the small learning-set size case, when dimensionality p of the vector $\mathbf{X}$ is close to N, the term $T_{\text{B}}^{\text{Stand}}$ is higher than the term T_{B}. For large N, however, the term $T_{\text{B}}^{\text{Stand}}$ can become lower than T_{B}. In the small learning-set size case, this means that primitive regression can outperform standard regression. Here we can conclude that in such case, it is not worth training the perceptron for many iterations. On the contrary, in situations where primitive regression loses against standard regression, i.e. in the large learning-set size case, it is worth training the perceprton for many iterations.

In situations with $\Sigma \neq \mathbf{I}$, the difference between standard and primitive regression can become particularly large. In this case,

$$\mathbf{w}'_{\text{F}}*\mathbf{w}^*_{\text{F}} = \mathbf{w}'^*\Sigma\mathbf{w}^* = \frac{\sigma^2\rho^2}{1-\rho^2}.$$

In order to observe an influence of Σ and $\mathbf{w}^*_{\text{F}}$ on the generalization error *let us analyze a model with the following constraints* on eigenvalues of the covariance matrix Σ and on components of the vector $\mathbf{w}^*_{\text{F}}$:

$$\delta_1 = 1, \quad \delta_2 \ldots = \delta_p = k_\delta,$$

$$w_1 = \frac{\sigma\rho}{\sqrt{(1-\rho^2)(1+(p-1)k_{\text{w}}^2}},$$

$$w_2 = w_3 = \cdots = w_p = w_1 k_w.$$

This model is determined by two parameters k_δ, and k_{w}. Results of numerical calculations performed for $p = 50$, $\rho = 0.9$, and $N = 60$ are presented in Tables 1 and 2.

Consider two particular cases:

A. A model where the largest eigenvalue (δ_1) of the

Table 1
The bias term in Eq. (26) $\sqrt{T_A}$ for different k_w and k_δ ($p = 50, \rho = 0.9$)

$k_w \backslash k_\delta$	0.001	1	1000
0.001	0.0001	0 (no bias)	0.75
0.1	0.55 (case A1)	0 (no bias)	116.2
1.0	1.27	0 (no bias)	201.4 (case B1)
1000	1.29	0 (no bias)	203.4

covariance matrix Σ is much larger than the other ones, and when the first component (w_1) of the weight vector $\mathbf{w}_F^*$ (in the direction of the largest eigenvalue of Σ) is much larger than the other ones (e.g. a case A1: $k_\delta = 0.001$, and $k_w = 0.1$). Then we have at most a small bias (0.55). In the small learning-set case, primitive regression ($\sqrt{E\varepsilon^2_{PRIMITIVE}} = 1.59$) outperforms the standard one dramatically (for $p = 50$, $N = 60$ we have $\sqrt{E\varepsilon^2_{STANDARD}} = 2.56$).

B. Models of the data where primitive regression performs badly (large eigenvalues $\delta_2, \ldots, \delta_p$ and large weights $w_2, \ldots, w_p$). For example, for $k_\delta = 1000$ and $k_w = 1$, (a case B1), we have ($\sqrt{E\varepsilon^2_{PRIMITIVE}} = 290.9$, and $\sqrt{E\varepsilon^2_{STANDARD}} = 2.56$. This theoretical analysis can unveil situations where primitive regression performs well. In such cases (e.g. cases where k_δ, k_w and N are small), is worth training perceptron only for a small number of iterations. Eqs. (25) and (26), however indicate that with an increase in the learning-set size, the situation can differ. E.g. for $N = 300$ for the case A1 we have $\sqrt{E\varepsilon^2_{STANDARD}} = 1.10$, though, $\sqrt{E\varepsilon^2_{PRIMITIVE}} = 1.56$.

In Fig. 3a and b, we present *simulation results with the linear single-layer perceptron*. Fig. 3a is obtained for data model A1. For $N = 60$ (Graph 1) while training the SLP from zero initial weights with $\eta = 1$, after the first iteration we obtained the generalization error 1.55. After 150,000 iterations, we obtained the generalization error 2.53. Both values are very close to the theoretical expected values $\varepsilon_{PRIMITIVE} = 1.59$, and $\varepsilon_{STANDARD} = 2.56$. For $N = 300$ (Graph 2), after the first iteration we have 1.55, and after 2000 iterations 1.10, very close to 1.56 and 1.10, the theoretical values for primitive and standard regressions. We see, for such type of data primitive regression is useful. Table 2 shows that particularly high effectiveness of the primitive regression (a brief training) comes to light if data with $k_w = 0.001$ and $k_\delta = 0.001$ is used. Then after

Table 2
The generalization error $\sqrt{E^2_{\varepsilon_{PRIMITIVE}}}$ ($p = 50, \rho = 0.9, N = 60$)

$k_w \backslash k_\delta$	0.001	1	1000
0.001	1.08	2.34	207.3
0.1	1.59 (case A1)	2.34	238.7
1.0	2.29	2.34	290.9 (case B1)
1000	2.31	2.34	292.4

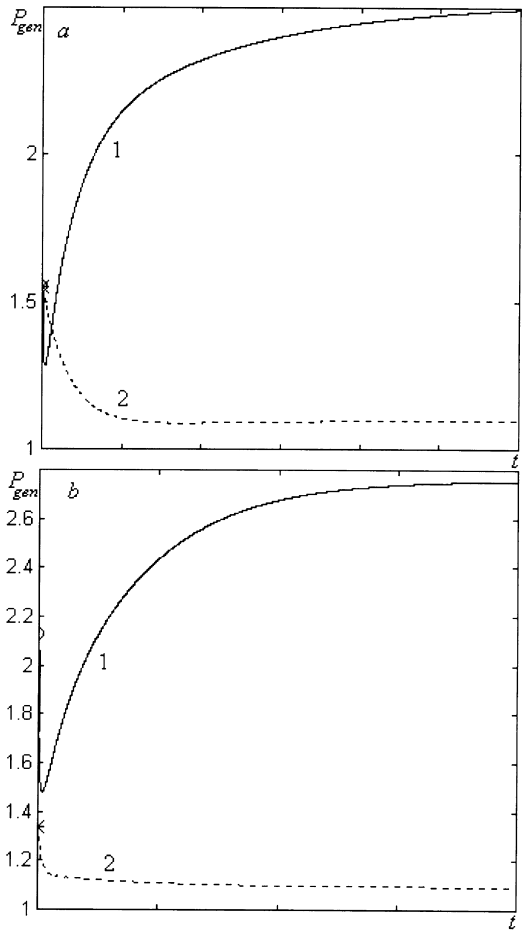

Fig. 3. Generalization error of linear SLP as a function of the number of iterations: (a) data A: 1—$N = 60$, 2—$N = 300$; (b) data B: 1 $N = 60$, 2—$N = 300$.

the first iteration, we should obtain the generalization error 1.08, and an essentially higher error (2.56) at the end of the training process. This means that after the first iteration we have $(2.56 - 1)/(1.08 - 1) \approx 20$ times smaller increase in the generalization error than at the end—a tremendous overtraining! Small k_δ is characteristic of data with small intrinsic dimensionality (the dimensionality of the subspace where data points $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3, \ldots$ are situated). Table 2 shows that for small k_δ (low intrinsic dimensionality) we have the lowest increase in generalization error. It agrees with a similar conclusion obtained for an analogous classification algorithm—the Euclidan distance classifier (see e.g. Raudys, 1967, 1998b)—in both algorithms, the Euclidean distance classifier and primitive regression, only sample means are used to find the weights of the algorithms.

Table 2 shows that there also exist situations where primitive regression and brief training of the SLP is not a good choice. For example, for the data model B with $k_w = 1$ and $k_\delta = 1000$ (a high intrinsic dimensionality case) and $N = 60$ from Table 2, we find $\varepsilon_{PRIMITIVE} = 290.9$—a very high generalization error, considerably higher than the error of a untrained SLP with weight vector $\mathbf{w} = \mathbf{0}$. Training with

$\eta = 1$ leads to $\varepsilon_{\text{gen}} = 273$ after the first iteration, and to a divergence of the back propagation algorithm later. Use of much smaller learning step $\eta = 0.001$ results in curve 1 in Fig. 3b with $\varepsilon_{\text{gen}} = 2.75$ after 4000 iterations (recall, for standard regression the expected prediction error $\varepsilon_{\text{STANDARD}} = 2.56$). For learning-set size $N = 300$ theory states $\varepsilon_{\text{SPRIMITIVE}} = 223$, and $\varepsilon_{\text{STANDARD}} = 1.10$. BP training of the SLP with $\eta = 1$ after the first iteration resulted in $\varepsilon_{\text{gen}} = 229$, and further divergence of BP training. For small learning-step ($\eta = 0.01$) we have no divergence: after the first iteration we obtained a much smaller generalization error, $\varepsilon_{\text{gen}} = 1.34$, and after 400 iterations, $\varepsilon_{\text{gen}} = 1.09$. For this type of the data, experiment confirms the theoretical conclusion: the SLP should be trained until the end: no early stopping is necessary.

Two conclusions follow:

—for the high-dimensional Gaussian data the starting (after the first iteration) and the final generalization errors of the linear SLP can be predicted with fairly high accuracy,

—the learning-steps value is an important tool to control convergence and the generalization error of the perceptron.

4.3. Regularized regression

The weight vector

$$\hat{\mathbf{w}}^{\text{RR}} = \left(\frac{1}{N} \sum_j \mathbf{X}_j \mathbf{X}'_j + \mathbf{I}\lambda \right)^{-1} \frac{1}{N} \sum_j \mathbf{X}_j y_j$$

$$= (\mathbf{S}_{XX} + \mathbf{I}\lambda)^{-1} (\mathbf{S}_{XX}\mathbf{w}^* + \mathbf{S}_{X\xi}).$$

Use of an expansion $(\mathbf{S}_{XX} + \mathbf{I}\lambda)^{-1} = \mathbf{S}_{XX}^{-1} - \lambda \mathbf{S}_{XX}^{-2} + \cdots$ and rejecting terms of order λ^2 and higher, for very small λ we can write:

- the prediction error

$$y_{\text{pred}} - y = \mathbf{X}'\hat{\mathbf{w}}^{\text{RR}} - \mathbf{X}'\mathbf{w}^* - \xi$$

$$= \mathbf{X}'(\mathbf{S}_{XX} + \mathbf{I}\lambda)^{-1}(\mathbf{S}_{XX}\mathbf{w}^* + \mathbf{S}_{X\xi}) - \mathbf{X}'\mathbf{w}^* - \xi$$

$$= -\lambda \mathbf{X}'\mathbf{S}_{XX}^{-1}\mathbf{w}^* + \mathbf{X}'\mathbf{S}_{XX}^{-1}\mathbf{S}_{X\xi} - \lambda \mathbf{S}_{XX}^{-2}\mathbf{S}_{X\xi} - \xi,$$

- and the expected square prediction error

$$\text{E}\varepsilon^2 = \text{E tr}\{[\mathbf{S}_{XX}^{-1}\mathbf{S}_{X\xi}\mathbf{S}'_{X\xi}\mathbf{S}_{XX}^{-1} - 2\lambda \mathbf{S}_{XX}^{-1}\mathbf{S}_{X\xi}\mathbf{S}'_{X\xi}\mathbf{S}_{XX}^{-2}]\mathbf{X}\mathbf{X}'\}$$

$$+\text{E}\xi^2 = \sigma^2 + \frac{1}{N}\sigma^2 \text{ tr E}\{\text{tr}(\mathbf{S}_{XX}^{-1}\Sigma) - 2\lambda \mathbf{S}_{XX}^{-2}\Sigma\}$$

$$= \sigma^2\{1 + \frac{1}{N}(\text{tr E}\mathbf{S}_{\text{I}}^{-1} - 2\lambda \text{E}\mathbf{S}_{\text{I}}^{-2}\Delta^{-1})\},$$

where Δ is a diagonal matrix composed from eigenvalues of the true covariance matrix Σ, $\mathbf{S}_{\mathbf{I}} = \Delta^{-1/2}\Gamma'\mathbf{S}_{XX}\Gamma\Delta^{-1/2}$ (it is a random Wishart W(N,**I**) matrix).

Γ is the eigenvector matrix of Σ, in singular value representation $\Sigma = \Gamma\Delta\Gamma'$, and

It is known (see e.g. Raudys, 1972) that

$$\text{E}\mathbf{S}_{XX}^{-1} = \mathbf{I}\frac{N}{N-p-1}, \qquad \text{and}$$

$$\text{E}\mathbf{S}_{\text{XX}}^{-2} = \mathbf{I}\frac{N^2(N-1)}{(N-p)(N-p-1)(N-p-3)}.$$

Therefore

$$\text{E}\varepsilon_{\text{RR}}^2$$

$$= \sigma^2\left(1 + \frac{p}{N-p-1} - 2\lambda \frac{\text{tr}(\Delta^{-1})N(N-1)}{(N-p)(N-p-1)(N-p-3)}\right)$$

$$= \text{E}\varepsilon_{\text{STANDARD}}^2 + T_\lambda, \tag{27}$$

where

$$T_\lambda = -2\lambda\sigma^2 \frac{\text{tr}(\Delta^{-1})N(N-1)}{(N-p)(N-p-1)(N-p-3)}.$$

Eq. (27) indicates that generalization error depends on λ, and parameter $\text{tr}(\Delta^{-1})$ of the distribution density function of **X**. The term T_λ tries to reduce the cost we need to pay to estimate the covariance matrix $\mathbf{S}_{XX}$. The simulation experiments reported in Section 4.2 confirm that while training the SLP and moving from primitive regression towards the standard one we have smaller generalization error. This is a consequence of the influence of the third term in Eq. (27).

4.4. Standard regression with the pseudoinversion of the covariance matrix

The weight vector

$$\hat{\mathbf{w}}^{\text{PINV}} = \mathbf{T}\begin{bmatrix} \mathbf{d}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \frac{1}{N} \sum_j \mathbf{T}'\mathbf{X}_j(\mathbf{X}'_j\mathbf{T}\mathbf{T}'\mathbf{w}^* + \xi_j)$$

$$= \mathbf{T}\left[\begin{pmatrix} \mathbf{w}_{\text{T1}} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \frac{1}{N}\sum_i \mathbf{d}^{-1}\mathbf{U}_{1i}\xi_i \\ \mathbf{0} \end{pmatrix} \right],$$

where $\mathbf{w}_{1i}$, $\mathbf{U}_{1i}$ are composed from the first r components of vectors

$$\mathbf{w}_{\text{T}} = \mathbf{T}'\mathbf{w}^* = \begin{pmatrix} \mathbf{w}_{\text{T1}} \\ \mathbf{w}_{\text{T2}} \end{pmatrix},$$

$$\mathbf{U}_i = \mathbf{T}'\mathbf{X}_i = \begin{pmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{pmatrix},$$

Table 3
Expected values $E\sum_j d_j^{-1}$ of the sum of inverse eigenvalues of the Wishart $W(N, \mathbf{I}_p)$ random matrix for $p = 20$

N	2	4	6	8	10	12	14	16	18
$E\sum d_j^{-1}$	0.24	1.08	2.73	5.78	11.2	20.7	38.2	87.6	274

respectively, and $\mathbf{T} = ((t_{is}))$ and

$$\mathbf{d} = \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & d_r \end{pmatrix}$$

has been defined in Section 3.4. To prove the above representation, we have used the representation

$$\mathbf{T}'\frac{1}{N}\sum_j \mathbf{X}_j\mathbf{X}_j' \quad \mathbf{T} = \frac{1}{N}\sum_j \mathbf{U}_j\mathbf{U}'_j = \begin{bmatrix} \mathbf{d} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Then the prediction error for vector $(y,\ \mathbf{X}')'$

$$y_{\text{pred}} - y = \mathbf{X}'\hat{\mathbf{w}}^{\text{PINV}} - \mathbf{X}'\mathbf{w}^* - \xi$$

$$= \mathbf{U}_1'\left(\frac{1}{N}\sum_j \mathbf{d}^{-1}\mathbf{U}_{1j}\xi_j\right) - \mathbf{U}_2'\mathbf{w}_{T2} - \xi,$$

and the expected square prediction error

$$E\varepsilon^2 = E\ \text{tr}\left(\frac{1}{N^2}\sum_{ij}\mathbf{d}^{-1}\mathbf{U}_{1i}\xi_i\xi_j\mathbf{U}'_{1j}\mathbf{d}^{-1}\right)\mathbf{U}_1\mathbf{U}'_1\}$$

$$+ E\ \text{tr}(\mathbf{w}_{T2}\mathbf{w}'_{T2}\mathbf{U}_2\mathbf{U}'_2) + E\xi^2.$$

In order to obtain a simple and easy to analyze formula for the prediction error, assume $\mathbf{X} \sim N(\mathbf{0}, \mathbf{I}_p)$. Consequently $\mathbf{U} = \mathbf{T}'\mathbf{X} \sim N(\mathbf{0}, \mathbf{I})$, and

$$E\varepsilon^2_{\text{PINV}} = \sigma^2 + T_A + T_B, \tag{28}$$

where

$$T_A = \sigma^2\frac{1}{N}E\sum_j d_j^{-1},$$

$$T_B = \sigma^2\frac{1}{N}\sum_{ij} w_i w_j E\sum_s t_{si}t_{sj}.$$

Eq. (28) indicates that the generalization error depends on components $w_1, w_2, \dots, w_p$ of the models weight vector $\mathbf{w}^*$.

Let us consider *two extreme cases*:

(a) all components of vector $\mathbf{X}$ are equally correlated with y, and

(b) only one (say, the first) component of $\mathbf{X}$ is correlated with y.

Let the coefficient of multiple correlation for model (23) be equal to ρ. Then for case (a)

$$w_1 = w_2 = \dots = w_p = \sigma\rho/\sqrt{p(1-\rho^2)},$$

and for case (b)

$$w_1 = \sigma\rho/\sqrt{p(1-\rho^2)}, w_2 = \dots = w_p = 0.$$

When $N < p$ the eigenvector matrix $\mathbf{T}$ becomes a random matrix. From the orthonormality condition ($\mathbf{T}\mathbf{T}' = \mathbf{I}$) for large p, we have

$$E\sum_s t_{si}t_{sj} = \begin{cases} 1/p & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases}.$$

Consequently, for both cases (a) and (b)

$$T_B = \sigma^2\frac{1}{N}\sum_{ij} w_i w_j \quad E\sum_s t_{si}t_{sj} = \sigma^2\frac{1}{N}\frac{\rho^2\sigma^2}{(1-\rho^2)}.$$

The term $T_A = \sigma^2(1/N)E\sum_j d_j^{-1}$ characterizes eigenvalues of the singular random $\mathbf{S}_{XX}$ matrix, having Wishart $W(N, \mathbf{I}_p)$ distribution, and can be evaluated by numerical methods. In Table 3 we present several values of Eig (N, p) found for $p = 20$.

We see that the expected values $E\sum_j d_j^{-1}$ increase with N exponentially. We can use these values to calculate theoretical estimates of the generalization error of the standard regression when $N < p$

$$E\varepsilon^2_{\text{PINV}} = \sigma^2 + \sigma^2\frac{1}{N}E\Sigma_{d_j}^{-1} + \sigma^2\frac{1}{N}\frac{\rho^2}{(1-\rho^2)}\left(1 - \frac{N}{p}\right). \tag{29}$$

The expression (29) for the generalization error is composed from three terms:

- σ^2—a term which characterizes the variance of noise in the model (23), an asymptotic (minimal, ideal) prediction error;
- $\sigma^2(1/N)E\Sigma_{d_j}^{-1}$ increases with N (when $N < p$);
- $\sigma^2\frac{1}{N}\frac{\rho^2}{(1-\rho^2)}\left(1 - \frac{N}{p}\right)$ decreases with N.

Numerical analysis of Eq. (29) shows that with increase in the learning-set size N, from 1 up to p, the generalization error decreases at first, comes to a minimum, and begins to increase when N approaches p (theoretically until infinity). This agrees with conclusions obtained earlier from the statistical mechanics approach (see e.g. Bös, 1996; Krogh & Hertz, 1992) and that obtained for the classification task (Raudys, 1998b; Raudys & Duin, 1998). When $N > p$, we have standard regression where the generalization error declines monotonically with increase in the learning-set size N (Eq. (25)). In Section 3.4, we demonstrated that in the large learning-set case (when $N > p$) back propagation training of a linear SLP leads to standard regression and

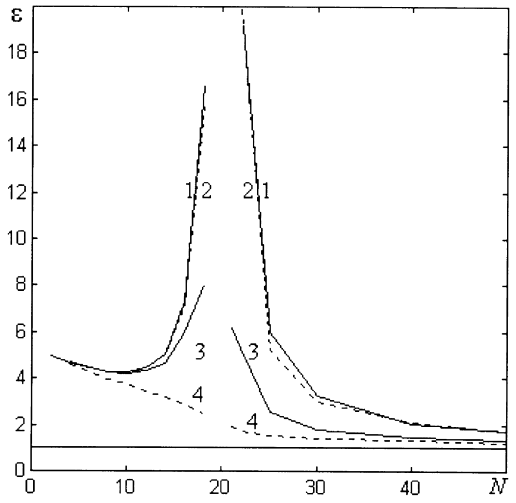

Fig. 4. Generalization of the standard regression (with the pseudoinversion of the covariance matrix, if $N < p$) : 1—theory, 2—experiment, 3—SLP after 2000 (if $N < p$), or 10,000 (if $N > p$) iterations, 4—optimally stopped SLP.

when $N < p$ to standard regression with the pseudo-inverse. Therefore, one can expect a similar peaking behavior of the fully trained linear SLP used as the predictor. The peaking behavior, however, can be abolished if we stop the training process optimally. In Section 2 we have shown that early stopping prevents us from obtaining standard regression and forces the perceptron to act as regularized regression acts (recall that in regularized regression *we add positive values* λ to all eigenvalues of the sample covariance matrix $\mathbf{S}_{XX}$). Simulation experiments show that in very small sample cases, regularized regression is a better strategy than standard regression with pseudo-inversion where *we ignore directions* corresponding to zero eigenvalues.

In Fig. 4, we present theoretical (1) and experimental (2, 3 and 4) graphs of dependence of the generalization error on the number of learning examples N: Graph 2 corresponds to the standard regression with pseudo-inversion, Graph 3 to SLP after $t_{\max} = 2000$ iterations ($t_{\max} = 10,000$ when $N > p$), and Graph 4 to optimally stopped SLP ($\mathbf{w}_{(0)} = \mathbf{0}$; batch training; $\eta = 0.1$ when $N < p$, and $\eta = 0.3$ when $N > p$). Gaussian N($\mathbf{0}$,$\mathbf{I}_p$) 20-variate data; all prediction variables $x_1, x_2 \ldots, x_p$ are equally correlated with y; the coefficient of the multiple correlation $\rho = 0.9$. For $N < p$, the theoretical graph was calculated from Eq. (29), and for $N > p$ from (25). The empirical graphs are average values obtained from 500 (when $N < p$) or 50 (when $N > p$) independent experiments. We see, theoretical (1) and experimental (2) graphs (for standard regression with pseudo-inversion of the covariance matrix) are very close. The Graph 3 for the exhaustively trained SLP resembles the graphs 1 and 2. The optimally stopped SLP (Graph 4) performs at best. All first three graphs exhibit the peaking behavior when N is close to p, however, the optimally stopped perceptron (the Graph 4) does not. It advocates once more that use of pseudo-inversion is not the best choice in the regression design process.

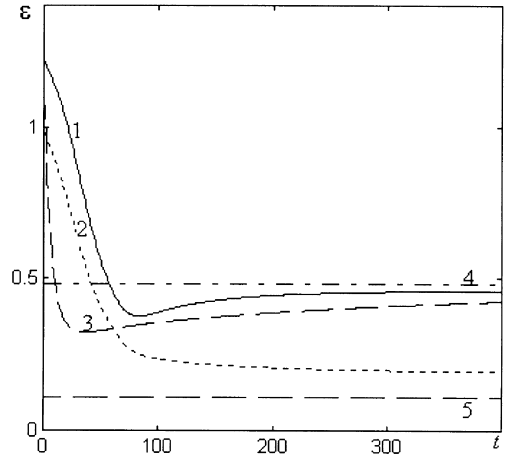

Fig. 5. Generalization of robust regression. 1, 2—robust SLP (test-set and training-set); 3—standard linear SLP (test-set); 4,5—standard regression (test-set and training-set).

4.5. Robust regression

No theoretical results are available in the literature. To explain principal tendencies in the small learning-set behavior, we performed a number of simulation experiments. In Section 3.5, we demonstrated a simple example where robust regressions outperforms the standard one if the data is contaminated by a noise (Fig. 1). Standard regression is appropriate for multivariate Gaussian data. In comparison with standard regression, in robust regression most distant observation vectors have less weighty contributions while determining the weights of the linear regression equation. Therefore, for Gaussian data use of robust regression can lead to an increase in the prediction generalization error. As a typical example in Fig. 5, we present learning curves $\varepsilon = f(t)$ of standard and robust SLP regression trained with the cost function (20) and $\alpha = 3$. We used 50-variate Gaussian data; all p variables $x_1, x_2 \ldots, x_p$ were equally correlated among themselves ($\rho = 0.3$); before training the perceptrons the data was normalized by subtracting a sample mean vector and dividing all variables by their standard deviations; the learning-set size $N = 60$, the learning rate $\eta = 0.001$. Curves 1 and 2 correspond to generalization and training-set errors of robust SLP, and curve 3 to the generalization error of the standard linear SLP. Straight lines 4 and 5 correspond to the generalization and training-set errors ($\sigma_{\text{gen}} = 0.481$, $\sigma_{\text{training}} = 0.111$) of the standard regression. The minimal generalization error of the standard SLP ($\sigma_{\text{gen}} = 0.324$) is lower than the minimal value of the generalization error of the robust SLP ($\sigma_{\text{gen}} = 0.377$). For scaling $\alpha = 10$ the robust SLP results in much higher error ($\sigma_{\text{gen}} = 1.2$). For $\alpha = 0.1$, however, the cost function of robust regression is actually the quadratic function. Thus, for $\alpha = 0.1$ we have $\sigma_{\text{gen}} = 0.324$, i.e. the same value as with standard regression.

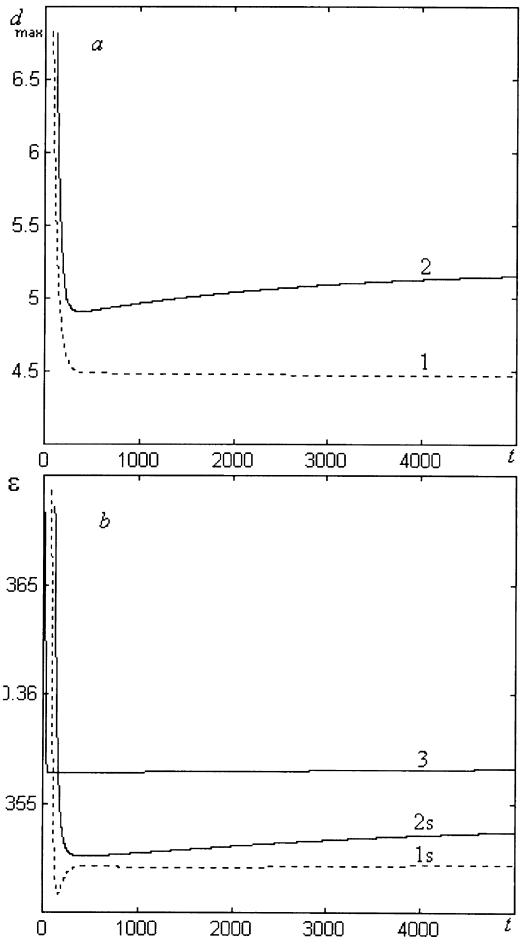

Fig. 6. The minimax regression. (a) The maximal distance d_{max} as a function of the number of iterations: 1—training-set, 2—test-set. (b) The standard deviation as a function of the number of iterations t: 1s—training-set, 2s—test; 3—standard linear SLP test-set.

The experimental studies demonstrated a preference of the cost functions (20) and (21) over the cost (18) and (19). In Gaussian cases, utilization of the robust cost function in the SLP training results in no gain. In both, Gaussian and non-Gaussian cases, the generalization error of the linear SLP perceptron with optimally chosen α and t is lower than that for standard regression. Here the regularized and robust regressions act together. Thus, we obtain *a robust regularized regression*. To obtain the best results we need to choose both the optimal number of iterations t and the non-linearity parameter α. The value of the learning-step η also is an important parameter. In order to obtain fast convergence, the parameter η should depend on α.

The priority of robust regression unveils only in non-Gaussian cases with atypical observations. As a characteristic example, we can mention a stock market closing index prediction task. The learning set was composed from 500 25-variate observation vectors and had a number of atypical observations. Standard regression resulted in the test-set prediction error $\varepsilon_{STAND} = 0.257$. Use of the optimally stopped linear SLP (optimal regularized regression) applied to normalized data (see Section 5.3) resulted in a little bit better prediction error: $\varepsilon_{SLP} = 0.223$. Robust regression with cost functions (20) and (21)) resulted in $\varepsilon_{ROBUST} = 0.242$ (for $\alpha = 0.6$), $\varepsilon_{ROBUST} = 0.125$ (for $\alpha = 17$), and $\varepsilon_{ROBUST} = 0.126$ (for $\alpha = 20$). Hence, selection of optimal values of α and t allowed the gereralization error to be reduced 2.05 times (in comparison with the standard regression), and in all experiments we present generalization error values of optimally stopped SLP.

In Section 4.1, we showed that in the low intrinsic dimensionality case, the generalization error of primitive regression can become very low. Regularized regression is an intermediate case between standard and primitive regressions. The generalization error of the regularized regression also can be adequately low. In the optimized robust perceptron, we have regularized robust regression. This means that in the low intrinsic dimensionality cases, the SLP robust regularized regression can have good small learning set size properties too.

4.6. Minimax (support vector) regression

No theoretical results are available in the literature here either. In standard regression, we minimize the mean square error. In minimax adaptive regression, we minimize the distances from the prediction hyperplane to the farthest learning-set vectors. This distance diminishes during the training. The maximal distance between the test-set vectors and the hyperplane (generalization), however, exhibits overtraining behavior: it diminishes at first, reaches a minimum and then begins to increase. The mean square errors, both in training and in testing, typically exhibit overtraining. The overtraining, however, there occurs earlier.

For a rough evaluation of relations between the generalization error, dimensionality p, and the learning-set size N, we performed a number of simulation experiments with Gaussian and non-Gaussian data. We found that an increase in the generalization error mainly depends on the ratio N/p, as predicted by theory for the linear regression models. For example, for data model (23) with a *uniform random noise* ξ in the interval $(-0.25, +0.25)$, for $\sigma = \sqrt{E\xi^2} = 0.1445$, and according to Eq. (25) the generalization error of the standard sum of squares regression $\sigma_{gen} \approx 0.37$ when $N = 1.2p$. In 50 repetitions of the experiments, we obtained approximately the same values of σ_{gen} for all four dimensionalities of the feature space ($p = 10, 20, 30,$ and 50) tested: $\sigma_{PRED} \approx 0.38$ for standard regression, $\sigma_{gen} \approx 0.19$ for optimally stopped minimax SLP, and $\sigma_{gen} \approx 0.23$ for minimax SLP after 1000 iterations with $\alpha = 5$, and $\eta = 0.5$. In a further increase in the number of iterations, the generalization error gradually approached 0.38, the generalization error of the standard regression. The non-linear character of the cost function (20), however, often oppress this process, and we need tremendous numbers of iterations in order to become close to standard regression. Fortunately, the best generalization is obtained much earlier.

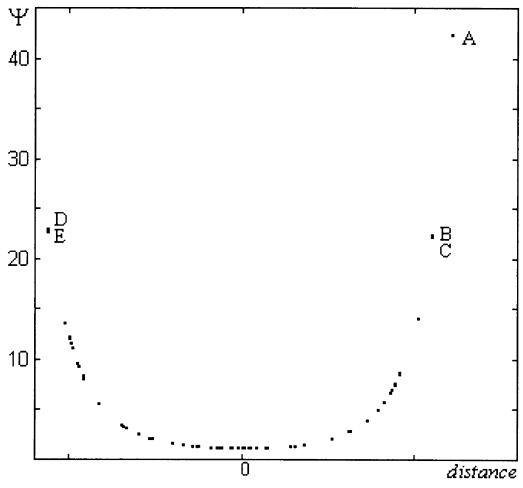

Fig. 7. Contributions Ψ of 60 learning-vectors to the cost function after 5000 iterations.

For multivariate *Gaussian* and even *non-Gaussian data*, typically the generalization mean square error of the minimax adaptive regression is higher than that of the standard optimally stopped SLP. Data models where the intrinsic dimensionality of the data is very low and noise ξ in the data model (23) is *uniformly* distributed constitute exceptions. In such cases, the generalization error of the optimally stopped minimax SLP regression is frequently somewhat lower than that of the standard optimally stopped SLP.

In Figs. 6a, b and 7 we present such a *non-typical example*. In this example, the intrinsic dimensionality of 49-variate vector $\mathbf{X}$ is practically equal to 1: all components of vector $\mathbf{X}$ are equally correlated ($\rho = 0.3$), maximal deviations of $\mathbf{X}$ in directions of 48 eigenvalues of $\mathbf{X}$ are 0.0005; and the last eigenvalue is equal to 1; noise ξ in the model (23) is uniform in the interval $(-0.25, +0.25)$. In Fig. 6a, we depict the dependence of the maximal distance between the training set vectors and the prediction hyperplane (curve 1) and between the test-set vectors and the hyperplane (curve 2) on the number of iterations. We see that the training distance diminishes constantly, while the testing distance exhibits peaking behavior. Fig. 6b shows the mean square deviations as functions of the number of iterations: 1s, the training-set error; 2s, the test-set error and 3, the test-set error of the standard linear SLP perceptron. All three curves peak. In this low intrinsic dimensionality case, nevertheless, adaptive minimax regression outperforms standard adaptive regression according to the minimal distance and the mean square deviation.

This experiment (non-typical), however, indicates that there is no need to train the perceptron until the support vector regression is obtained. In other experiments performed with non-uniform noise and especially in experiments with higher intrinsic dimensional data, the necessity to stop training the minimax SLP earlier is much more obvious. Fig. 7 shows contributions of sixty 50-variate training-set vectors (training data) to the cost function after 5000 iterations. In this experiment 5000 iterations correspond to overtrained perceptron. We see five learning-set vectors (A, B, C D and E) contribute at most to the cost function value, however the remaining vectors (15 at least) also have notable influence. A more general conclusion obtained from this and other experiments, is that in spite of the fact that we are rather far away from the support vector regression, the maximal distance and the minimal mean square error among *the test-set vectors* are obtained substantially earlier.

To obtain the support vector classifier while training the non-linear SLP, we need to have no classification errors among the learning-set vectors, to use limit target values ($t_1 = 0$ and $t_2 = 1$ for the sigmoid activation function $f(c) = 1/(1 + \exp(-c))$), and gradually increase the learning-step η with an increase in the number of BP iterations (Raudys, 1998a). In regression, however, we need to use a special pattern error function (Eq. (22) for example) and gradually increase the scaling parameter α with an increase in the number of BP iterations. Simulation experiments show that in both tasks, classification and regression, convergence to the support vector machine is slow. In both tasks, nevertheless, the best generalization is obtained far earlier before the support vector machine is obtained.

5. Complexity control

5.1. A necessity

We have shown that while training a SLP, we can obtain six different types of regression, which should be used in different circumstances. When we have the multivariate Gaussian data with low intrinsic dimensionality, the primitive regression or briefly trained linear SLP can become very useful. For certain distributions of the components of the weight vector and eigenvalues of the covariance matrix of the vector $\mathbf{X}$, however, the primitive regression can perform very poorly. For such types of data in the large learning-set case ($N > 2p$), we need to use standard regression, i.e. to use the linear SLP and train it almost until the end. If the learning-set size is small, we need to use regularized regression, i.e. to stop training earlier. When the data is contaminated by noise and we have a great percentage of atypical observations, outliers, the robust regression is preferable over all other types of the regression rules. In such cases, we need to use a non-linear SLP with cost functions (18) or (20) and (21). In some cases, the maximal accuracy prediction is of the minimax regression (SLP with pattern error function (22) as discussed in Section 3.6).

5.2. Tools

Which tools can be used to control the result obtained? First of all is the *number of iterations* in the gradient (BP) training algorithm. Use of second-order optimization methods such as the Newton algorithm can lead to the standard regression in one single learning iteration. In such

cases, we fail to obtain regularized regression. This explains why simple gradient training methods often outperform more sophisticated second-order methods.

The value of the *learning-step* η also provides a very important tool, which affects the result. Typically the learning-step controls the speed of the learning process. Too small values of η make the training very slow, while too large ones can lead to local minima and/or to the divergence of the algorithm. It is known (see e.g. Amari, 1967) that the value of the learning-step controls the variance of the final weight vector. According to this theory, in order to find the minimum of the cost function, the learning-step should diminish with the increase in the number of iterations. It is absolutely true only in the linear SLP training. Here the cost function does not change with an increase in the number of iterations. In the non-linear SLP, the cost function changes and one has to control the magnitude of the learning step with any increase in the number of iterations. One more remark concerning the learning-step η: in the non-linear SLP, overly large values of η can saturate the cost function and stop further training at all. In order to obtain primitive regression in one single iteration, we need to use $\eta = 1$. We can decrease this value in further training. For some data models, however, we need to choose a very small learning-step value just for the first iteration (see the simulation experiments mentioned at the very end of the Section 4.2).

The types of *activation and cost functions* are of great importance too, and allow one to choose between linear and non-linear perceptrons, and obtain robust and/or minimax regressions with different properties. We have discussed these properties above, in Section 3. There, the values of the scaling parameter, α, play a very important role. Small values of these parameters move the cost functions (18) and/or (20) and (21) to the standard sum of squares cost and to the linear SLP, while in the large-learning-set case ($N > 2p$), large values of α control the degree of ignored distant observations. In minimax regression, the parameter α controls the proportion of the most distant observations, which contribute to the final position of the regression equation. Consequently, the optimal α values should be chosen in dependence with the structure of the data.

Besides the standard cost function of the type (1), an additional regularization term can be added (Hinton, 1987). Addition of the simple weight decay term $+\lambda_R \mathbf{w}'\mathbf{w}$ has a regularization effect: it can be shown that λ_R plays the role of term λ in the regularized regression discussed in Sections 2.3 and 3.3 (see also Sjoberg & Ljung, 1992). Large weights can help to obtain the minimax (support vector) regression easier. Then we can use the regularization term $+\lambda_R(\mathbf{w}'\mathbf{w} - h^2)^2$, where parameter h controls the magnitudes of the weights. Selection of an optimal set of training parameters and a way to control the training process constitute the topic of the next few sections.

5.3. Data complexity control

A very important tool, which can help to control the type of regression obtained in adaptive training is *weight initialization*. A successful initialization yields smaller generalization errors and permits this to be obtained faster (Raudys & Amari, 1998). We have seen that the transit of the coordinate center into the sample mean of the learning data set helps one to obtain the primitive regression in one iteration. In some situations, this type of regression is the best choice (Section 4.2). Therefore, in such situations, the weight vector after the very first iteration can become a very good starting position for further training.

Moreover, for many models of the data and configurations of the components of the ideal weight vector $\mathbf{w}^*$ primitive regression requires fewer training vectors than the more complex standard regression (Sections 4.1 and 4.2). For some configurations, the difference can be very large. Iterative BP training of the single layer perceptron becomes very slow when variances of vector $\mathbf{X}$ are different in various directions, i.e. when the eigenvalues of the covariance matrix Σ are essentially different (Le Cun, Kanter & Solla, 1991).

The above arguments advocate that it would be desirable, prior to training the perceptron, to transform the data in such a way that to have the spherical Gaussian distribution of vector $\mathbf{X}$. We can try to do this by transforming the data by means of rotation and scaling:

$$\mathbf{X}_{\text{new}} = \mathbf{D}^{-1/2}\mathbf{T}'\mathbf{X},$$

where $\mathbf{D}$ and $\mathbf{T}$ are $p \times p$ diagonal eigenvalue and $p \times p$ eigenvectors matrix of the covariance matrix Σ. Instead of Σ, we can use the sample covariance matrix $\mathbf{S}_{XX}$. Then the learning-set covariance matrix of $\mathbf{X}_{\text{new}}$ will be an identity matrix $\mathbf{I}$. After the first learning iteration in the transformed space we obtain $(\mathbf{X}_{\text{new}})'\hat{\mathbf{w}}^{\text{PRIM}} = \mathbf{X}'\mathbf{S}_{XX}^{-1}\mathbf{S}_{Xy}$, i.e. we have standard regression (4) in the original (X) space (Section 3.1, Eq. (12)).

In the singular value decomposition $\mathbf{T}\mathbf{S}_{XX}\mathbf{T}' = \mathbf{D}$, instead of the standard maximum likelihood estimate $\mathbf{S}_{\text{XX}} = (1/N)\sum_j \mathbf{X}_j\mathbf{X}_j$ one can use some additional information concerning the structure of the covariance matrix Σ. For example, use of an assumption that components of vector $\mathbf{X}$ are realizations of a stationary Gaussian autoregressive process of order h, leads to a *constrained estimate of the covariance matrix* and reduces the number of parameters to be estimated from the learning data from $p(p+1)/2$ to h. In the case, when assumptions concerning the structure of the covariance matrix Σ are approximately correct, in subsequent training of the SLP, one can obtain a significant gain (see e.g. experimental results obtained for the classification task in Raudys & Saudargiene, 1998). Much smaller generalization errors also can be obtained if instead of the standard maximum likelihood estimate $\mathbf{S}_{XX}$, we use the regularized sample estimate of the covariance matrix (Raudys, 2000). Utilization of constrained estimates of the covariance

matrix in order to perform the data transformation $\mathbf{X}_{new} = \mathbf{D}^{-1/2}\mathbf{T}'\mathbf{X}$, is in fact an incorporation of additional statistical information into the perceptron design. If, in addition to the conventional learning-set, some more information is available, possibly this information can be used to transform the data in such a way that make the terms T_A and T_B in Eq. (26) as small as possible. This is the subject of further investigations.

5.4. Optimal complexity

In principle, one can use analytical formulae for the expected prediction error $E\varepsilon_{prediction}$, to calculate this error for the number of regression models, and then to choose the best. An example has been presented at the end of Section 4.2. Unfortunately, a number of obstacles exist. First of all, $E\varepsilon_{prediction}$ is a mean value, and practically for each particular learning-set, we have random deviations from the expected (mean) values. Second, in order to use analytical formulae we need to know several true parameters of the data model. For example, for the primitive regression these are the components of the optimal weight vector and the eigenvalues. Sample estimates of these parameters are not exact and, no doubt, result in random errors while calculating $E\varepsilon_{prediction}$. Third, the true data model, as a rule, is unknown. Use of the simplified multivariate models, such as the multivariate Gaussian distribution, causes additional bias errors. And, last, for a majority of real world data models analytical equations are not derived (e.g. the robust and minimax regression, or the standard, primitive regressions for non-Gaussian data models) either are only approximate (e.g. Eq. (27) for the regularized regression is valid only for very small λ). Therefore, a purely analytical route to the best model choice is impractical.

A standard practical way of *model choice* in statistical inference is *cross validation*. There, one splits the design-set vectors into two parts. One part is used for *training* and the second one for *validation* (*testing*). In order to use the design-set vectors more economically sometimes one uses a *rotation* method: there one splits the design-set into k parts and uses $k-1$ parts of the data for training, and the remaining one part for testing. This procedure is repeated k times, and an average generalization error value of all k experiments is used to select the best model. One more method is the *bootstrap* method (Efron, 1979). There, in order to estimate the bias of the apparent (learning-set, or resubstitution) prediction error the following computer intensive procedure is used. From N design-set vectors, one forms a random bootstrap learning-set composed from N randomly chosen vectors. The model is tested twice: (a) on the bootstrap learning-set, and (b) on all N original design-set vectors. The procedure is repeated k times. The mean value Δ_{boot} of the difference between the two estimates (a and b) is estimated over k runs of the experiment. The difference Δ_{boot} is used to estimate a bias of the resubstitution (learning-sets) error estimate. In principle, this computer-intensive method can be applied to choose the best type of regression in the SLP training.

One more possibility to form a pseudo-validation-set is a *noise injection*. The injection of the Gaussian spherical noise $N(0,\mathbf{I}\lambda)$ asymptotically (when a number of the noise injections tends to infinity) is equivalent to usage of regularized regression, where we add component $\mathbf{I}\lambda$ to the sample estimate of the covariance matrix. Therefore Gaussian spherical noise $N(0,\mathbf{I}\lambda)$ injection *can not be used* for a correct non-biased model choice. To improve statistical classifiers and neural networks Duin (1993) suggested injecting k-NN directed noise. In his proposal, for each learning-vector $\mathbf{x}_j$, one injects noise in directions of few nearest neighbors to $\mathbf{x}_j$. Skurichina and Duin (1997) and Skurichina, Raudys and Duin (2000) suggest to use $k = 2$. Contrary to traditional regularized classification or regression, this approach introduces new information concerning the local structure of the data, what is different from assumptions used to construct a regularized regression. Therefore, in some circumstances, the k-NN directed noise injection can be used to form an additional validation-set for the model choice. This problem deserves further analysis.

6. Concluding remarks

If appropriately used, the theoretical findings concerning the evolution in the single-layer perceptron training process can become very useful. The main theoretical results presented in this paper are the following:

1. While training the SLP we can obtain six different types of the regression, starting from the simplest one in the statistical sense, and going on to more complex ones.
2. The main tools, which can help to control the complexity of the prediction rule are: the number of learning iterations, the learning-step's value control during the iterative training process, the type of activation and pattern error functions, as well as the scaling parameter α value, which determines the shape of the pattern error functions. The data transformation prior to training the perceptron is a very important tool and can help to utilize additional information about the structure of the data presented in a form of a statistical hypothesis about the structure of the covariance matrix.
3. The analytical formulae of the expected generalization error derived for the linear SLP shows that the relationship between generalization error and learning-set size depends on the regression type, and on the data. The relationship can depend on components of the vector $\mathbf{w}^*$ too. The intrinsic dimensionality of the data is of great importance also. For certain configurations of the data (low intrinsic dimensionality, special structure of the components of vector $\mathbf{w}^*$), the perceptron can be perfectly trained with very short learning sequences.

In order to train the SLP in the best way we recommend:

1. Before training, the single-layer perceptron to move the coordinate center to the zero point.
2. To transform the data in order to have it approximately spherical distributed data with unit variance of all components of vector $\mathbf{X}_{\text{new}}$. We suggest to start training from zero initial weight vector and for a short time to test several learning-steps values. It is a way to integrate the statistical and neural net approaches to design linear prediction rules.
3. To analyze bi-variate scatter diagrams of some pairs of components of $\mathbf{X}_{\text{new}}$, and if we see that the data is obviously non-Gaussian, and/or contaminated by outliers to choose the robust cost function with the cost functions (20) and (21), and to train the perceptron with different values of the scaling parameter α. If there are no visible outliers to use the standard sum of squares cost function or use small α. In the Appendix, we present MATLAB code for robust regression (20) with (21).
4. To form the validation-set in one way or another. To use this set in order to choose the optimal value of α and the optimal numbers of iterations t for each particular α value.

The analysis performed explains particular theoretical questions, however some important problems remain unsolved. First of all, there are no theoretical formulae for the generalization error of the robust and the minimax regressions yet, as well as more exact non-asymptotic expressions for regularized and standard regression with pseudo-inversion in a general case (when $\Sigma \neq \mathbf{I}$). It would be interesting to find out how to use additional information in order to transform the data in such a way that the terms T_{A} and T_{B} in Eq. (26) become small.

The single-layer perceptron and the back propagation training are simplified mathematical models of complex information processing processes, which take place in Nature. In the BP training, we begin from the simplest in the statistical sense models (the Euclidean distance classifier or primitive regression) and gradually step-by-step move towards more complex models (regularized and robust procedures, the support vector machines). One may guess that this is a natural way of development. During the last few decades statisticians and engineers understood that while designing the decision making algorithms from experimental data one needs to move from simple algorithms to complex ones. The artificial neuron does this in a natural way. Statisticians required several decades to develop a number of statistical classification and regression rules: Fisher (1936) proposed his linear discriminant function more than six decades ago, and Vapnik his support vector machine (see Cortes & Vapnik, 1995) only recently. The neuron implements these algorithms in a natural way. One may guess that this, the Nature inspired, way of maturation is one of the most successful ones.

Acknowledgements

The author thanks Prof. Roy Davies for friendly aid in preparing a final version of the paper.

Appendix

```
% Find robust regression by
%   the nonlinear single layer perceptron
% author Sarunas Raudys
% <raudys@das.mii.lt>
% A input N*p array - training-set
% Y target N*1 array- training-set
% At input Nt*p array - test-set
% Yt target Nt*1 array- test-set
% iter - number of iterations
% step - learning-step
% Wstart - 1*(p + 1) starting weight
% vector
% alfa - scaling parameter
% W - 1*(p + 1) final weight vector
% et - generalization error history in
% iter training iterations
% prior to training we recommend:
% - to substract from A,Y; At,Yt the sample
% means of A,Y;
% - to use Wstart = zeros(1,p + 1);
% - whitening of a distribution of input
% vector can be useful
function [W,et] = robustpc
(A,Y,At,Yt,iter,step,Wstart,alfa)
[N, p] = size(A);
[Nt, pt] = size(At);
W = Wstart;
stepalfa = step/alfa;
AA = [A, ones(N,1)];
AAt = [At, ones(Nt,1)];
for i = 1:iter
  dist = alfa*(Y - AA * W);
  ind = find(abs(dist) < pi);
  W = W + stepalfa
  * sin(dist(ind))*AA(ind,:);
  dt = AAt*W - Yt;
  et(i) = sqrt(dt*dt./Nt);
end
return
```

References

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computations, EC-16*, 299–307.

Bös, S. (1996). Learning curves of on-line and off-line training. *Proceedings of ICANN'96, Bohum*.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*, 273–297.

Davisson, L. D. (1965). The prediction error of stationary Gaussian time series of unknown covariance. *IEEE Transactions on Information Theory, IT-11*, 527–532.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1996). Support vector regression machines. In M. C. Mozer, M. I. Jordan & T. Petsche, *Advances in neural information processing systems. Proceedings of the 1996 conference* (pp. 155–161). vol. 9. Cambridge, MA: MIT Press (A Bredfor Book).

Duin, R. P. W. (1993). Superlearning capabilities of neural networks. *Proceedings of the 8th Scandinavian conference on image analysis* (pp. 547–554). NOVIM, Norwegian Society for Image Processing and Pattern Recognition, Tromso, Norway.

Efron, B. (1979). Bootstrap methods: another look at the jacknife. *Annals of Statistics, 7*, 1–26.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics, London, 7 (2), 179–188.*

Harley, T. J. (1963). Pseudoestimates versus pseudo-inverses for singular sample covariance matrices. Section 2 in Report No 5, Contract DA-36-039-SC-90742, AD427172 (September 1963).

Harley, T. J. (1965). MS thesis, Moore School of Electrical Engineering, University of Pennsylvania.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for orthogonal problems. *Technometrics, 12*, 55–67.

Hinton, G. (1987). Learning translation invariant recognition in massively parallel networks. In J. W. de Bakker, A. J. Nijman & P. C Treleaven, *Proceedings of the PARLE conference on parallel architectures and languages* (pp. 1–13). Berlin: Springer.

Huber, P. J. (1981). *Robust statistics*. New York: Wiley.

Krogh, A., & Hertz, J. A. (1992). A simple weight decay can improve generalization. In J. Moody, S. J. Hanson & R. Lippmann, *Advances in neural information processing* (pp. 950–957). vol. 4.

Le Cun, Y., Kanter, I., & Solla, S. (1991). Eigenvalues of covariance matrices: application to neural-network learning. *Physical Review Letters, 66* (18), 2396–2399.

Raudys, Š. (1967). On determining the training sample size of a linear classifier. In N. Zagoruiko, *Computing systems. Institute of Mathematics, Academy of Sciences USSR* (pp. 79–87). vol. 28. Novosibirsk: Nauka (in Russian).

Raudys, Š. (1972). On the amount of a priori information in designing the classification algorithm. *Proceedings of the Academy of Sciences of USSR, Technical, Cybernetics, 4*, 168–174 (in Russian).

Raudys, Š. (1998). Evolution and generalization of a single neurone. I. Single-layer perceptron as seven statistical classifier. *Neural Networks, 11*, 283–296.

Raudys, Š. (1998). Evolution and generalization of a single neurone. II. Complexity of statistical classifiers and sample size considerations. *Neural Networks, 11*, 297–313.

Raudys, Š. (2000). Scaled rotation regularization. *Pattern Recognition, 33*, 1–10.

Raudys, Š., & Amari, S. (1998). Effect of initial values in simple perception. *Proceedings of the 1998 IEEE World Congress on Computational Intelligence, IJCNN98*, 1998.

Raudys, Š., & Duin, R. P. W. (1998). Expected classification error of the Fisher classifier with pseudo-inverse covariance matrix. *Pattern Recognition Letters, 19*, 385–392.

Raudys, Š., & Saudargiene, A. (1998). Structures of covariance matrices in the classifier design. *Advances in pattern recognition, Springer Lecture notes in computer science. (Proceedings of the Joint IAPR International Workshops/SSPR98 and SPR98, Sydney, Australia, August 11–13, 1998)*, vol. 1451. Berlin: Springer (pp. 583–592).

Skurichina, M., & Duin, R. P. W. (1997). Scale dependence of noise injection in perceptron training. In P. Pudil, J. Novovicova, J. Grim, *Proceedings of the IAPR Workshop on Statistical Techniques in Pattern Recognition, June 9–11*, 153–158.

Skurichina, M., Raudys, S., & Duin, R. P. W. (2000). K-Nearest neighbors directed noise injection in multilayer perceptron training. *IEEE Transactions on Neural Networks, 11* (2), 504–511.

Sjoberg, J., & Ljung, L. (1992). *Overtraining, regularization, and searching for minimum in neural networks. Technical report.* Department of Electrical Engineering, Linkoping University, S-581 83 Linkoping, Sweden, February.

Wang, C., & Venkatesh, S. S. (1994). Temporal dynamics of generalization in neural networks. *Proceedings of NIPS-7*, pp. 262–270.