Annals of the University of Bucharest (mathematical series)
4 (LXII) (2013), 491–511

Sub-optimal Logistic Models for Classification and Prediction Tasks

Iuliana Paraschiv-Munteanu and Luminiţa State

Communicated by Ioan Tomescu

Abstract - Learning from data means to have a learning method that is an algorithm implemented in software that estimates an unknown dependency between a system's inputs and outputs from the available data, namely from known samples. Once such a dependency has been accurately estimated, it can be used for prediction of future system outputs for known input values.

In order to learn a classification system based on a known finite size input-output samples, a probabilistic parametric logistic model is proposed in the paper, and several estimation methods to approximate the parameters are considered. Their performances are analyzed on experimental basis from two points of view, namely the quality of the learned model expressed in terms of the values of the likelihood function and the behavior in classifying new data coming from the same repartitions expressed in terms of the empirical error. The estimation process is implemented on linearly separable data, while in testing the resulted performance new linearly separable and non-linearly separable data are used. Four suboptimal types of estimations are introduced in the second section of the paper. The comparative analysis developed in the second and the third sections of the paper pointed out their better behavior as compared to the estimates computed using the learning algorithms MSE, perceptron, Mays and SVM.

Key words and phrases : logistic model, learning from data, supervised learning, maximum likelihood principle, minimum squared errors, prediction, classification.

Mathematics Subject Classification (2010) **:** 62J05, 62J12.

1. Introduction

We consider the learning environment described in [9], [10] and [13]. Let **S** be a system that for any n-dimensional input x computes an m-dimensional output y according to an unknown law. In the simplest approach we can assume that the output y is uniquely determined by the input x. However, the output can be influenced by a series of unobservable factors, and the dependency between the inputs and outputs of **S** could be of non-deterministic type. Consequently, in a more sophisticated approach we are forced to take

into account a non-deterministic dependency, modelled for instance in probabilistic terms, as a reasonable hypothesis concerning the unknown law. In our model we consider the Generator, denoted by **G**, the source that generates the inputs. Mainly, there are two ways to model **G**, namely when the mechanism of generating inputs is known by the observer and when the law according to which the inputs are generated is also unknown, respectively. The third component of our learning environment denoted by **L**, is responsible with possible models of the unknown dependency corresponding to **S**. The learning component **L** implements a class of hypothesis (models) Ω, such that to each particular hypothesis $\omega \in \Omega$ corresponds a function $\varphi_\omega : \mathcal{X} \to \mathcal{Y}$ defined on the space of inputs $\mathcal{X}$ and taking values in the space of outputs $\mathcal{Y}$ ([2], [3], [7], [9]). For each particular input x_0, $\widehat{y}_0 = \varphi_\omega(x_0)$ is the estimate of the **S**'s output corresponding to x_0 in case of the model ω. Being given a criterion function $\mathcal{C}$ that expresses numerically the fitness of each model with respect to the available evidence E, about **S**, the best model $\omega_0(E)$ is a solution of the optimization problem

$$\arg\left(\text{optimize}_{\omega\in\Omega}\, \mathcal{C}(\omega, E)\right) . \tag{1.1}$$

In the case of supervised learning the available evidence E is represented by a finite set of pairs $\{(x_i, y_i)\ , 1 \le i \le N\} \subset \mathcal{X} \times \mathcal{Y}$, where each y_i is the actual output of **S** for the input x_i.

Conventionally, when the output space of **S** is a finite set, the problem is referred as a classification problem. Typically, in case of a classification problem the output of **S** is either 0 or 1, these values corresponding to the recognition of the class from which each input comes. The use of a set of linear hypotheses ω for discriminating between two input classes yields poor results, mainly because in order to bring the computed values to the domain $\{0, 1\}$ some threshold should be imposed. An alternative way is to combine the linear regression with the logistic function in order to model the classes in the output space in terms of a Bernoulli probability distribution ([8]). Following the brief presentation of the logistic model, the ML estimate of the probabilistic model on the space of outputs of **S** is derived in the third section of the paper. In the final part of the paper a series of conclusions are presented concerning the performance of the estimated model derived on experimental basis.

2. Supervised Learning Schemes of the Logistic Model for Classification Purposes

We denote by n the dimension of the space from which **G** selects the inputs loaded into **S**, where for each input x the system **S** computes either 0 or 1. At the level of the component **L** each hypothesis ω corresponds to a $(n+1)$-dimensional parameter β. For any $\beta \in \mathbf{R}^{n+1}$ let $g_\beta : \mathbf{R}^{n+1} \to (0, 1)$ defined

by $g_\beta(z) = h\left(\beta^T z\right)$, where h is the logistic function $h(u) = \dfrac{1}{1+e^{-u}}$.

Our approach is based on a probabilistic model of Bernoulli type concerning the behavior of **S**, that is if β is the current hypothesis, then for each input x, $g_\beta(z)$, $1-g_\beta(z)$ are taken as the probabilities that **S** emits the outputs $y=1$, $y=0$, respectively,

$$\begin{aligned} &P\left(y=1 \mid x,\beta\right) = g_\beta(z)\,, \\ &P\left(y=0 \mid x,\beta\right) = 1-g_\beta(z)\,. \end{aligned} \tag{2.1}$$

In other words, each hypothesis ω implemented at the level of the learning component **L** is represented by a $(n+1)$-dimensional parameter β, the predicted values of **L** concerning the unknown input-output dependency of **S**, being given by the probability distribution

$$p\left(y \mid x,\beta\right) = \left(g_\beta(z)\right)^y \left(1-g_\beta(z)\right)^{1-y}\ ,\ y\in\{0,1\}\,,\ x\in\mathbf{R}^n,\ z=\begin{pmatrix}1\\ x\end{pmatrix}. \tag{2.2}$$

Obviously,

$$P\left(y=1 \mid x,\beta\right) \geq P\left(y=0 \mid x,\beta\right) \Longleftrightarrow g_\beta(z) \geq \frac{1}{2} \Longleftrightarrow \left\lfloor g_\beta(z)+\frac{1}{2}\right\rfloor = 1\,,$$

and

$$P\left(y=1 \mid x,\beta\right) < P\left(y=0 \mid x,\beta\right) \Longleftrightarrow g_\beta(z) < \frac{1}{2} \Longleftrightarrow \left\lfloor g_\beta(z)+\frac{1}{2}\right\rfloor = 0\,,$$

that is the prediction $y_\beta(x) = \left\lfloor g_\beta(z)+\dfrac{1}{2}\right\rfloor$ corresponds to the most probable output according to the model β.

Also, since

$$g_\beta(z) \geq \frac{1}{2} \Longleftrightarrow \beta^T z \geq 0\,,$$

we get that each model β defines a partition of the input space into two regions that approximate the classes of inputs for which **S** emits the outputs 1/0, respectively, these regions being separated by the hyperplane H_β : $\beta^T z = 0$. The class of inputs corresponding to the predicted output $y_\beta = 1$ is the hyperspace $S^+\left(H_\beta\right) \cup H_\beta$, where $S^+\left(H_\beta\right) = \left\{x\in\mathbf{R}^n \mid \beta^T\begin{pmatrix}1\\ x\end{pmatrix} > 0\right\}$.

We consider a supervised framework as the base for identifying the optimal hypothesis from the point of view of least means errors criterion, that is the search for optimal hypothesis is developed using the information contained by a finite set of labeled examples

$$\mathcal{S}_N = \left\{(x_i,y_i) \mid x_i\in\mathbf{R}^n,\ y_i\in\{0,1\},\ 1\leq i\leq N\right\}.$$

Being given the model β, the predicted output for each input x_i is $y_\beta(x_i) = \left\lfloor g_\beta(z_i) + \frac{1}{2} \right\rfloor$, where $z_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$.

The problem of learning from data the most suitable model β can be approached many ways, by adopting different point of views expressed in terms of performance functions used to evaluate to quality of each model with respect to the evidence represented by $\mathcal{S}_N$.

2.1. Minimum Squared Errors (MSE) Estimates of the Logistic Model

The performance of each model β is evaluated by the MSE criterion function

$$F_N(\beta) = \frac{1}{N} \sum_{i=1}^{N} \|y_i - g_\beta(z_i)\|^2,$$

and the optimal model determined on the basis of $\mathcal{S}_N$ is a solution of the optimization problem

$$\widehat{\beta}_{MSE} = \arg \left(\min_{\beta \in \mathbf{R}^{n+1}} F_N(\beta) \right) . \tag{2.3}$$

The computational difficulties of deriving exact solution of the problem (2.3) justify the use of iterative learning schemes and heuristic approximations, yielding to sub-optimal solutions, the quality of each sub-optimal solution being evaluated experimentally.

Denoting by $\nabla_\beta F_N(\beta)$ the gradient of the objective function F_N, the space of critical points is the set of the solutions of the system

$$\nabla_\beta F_N(\beta) = 0 . \tag{2.4}$$

Using straightforward computations we get

$$\nabla_\beta F_N(\beta) = \frac{2}{N} \sum_{i=1}^{N} (g_\beta(z_i) - y_i)\, g_\beta(z_i)\, (1 - g_\beta(z_i))\, z_i ,$$

where $z_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$.

Moreover, the gradient of the objective function $F_N(\beta)$ can be written in compact form as

$$\nabla_\beta F_N(\beta) = \frac{2}{N} Z D_1(\beta)\, (D_1(\beta) - \mathbf{I}_N)\, (Y - D_1(\beta) u) ,$$

where Z, Y are the matrices of examples and the corresponding labels,

$$Z = (z_1, \ldots, z_N) , \quad Y = (y_1, \ldots, y_N)^T , \tag{2.5}$$

$D_1(\beta)$ denotes the diagonal matrix having as diagonal entries $g_\beta(z_i)$, $1 \leq i \leq N$,

$$D_1(\beta) = \operatorname{diag}\left(g_\beta(z_1), \ldots, g_\beta(z_N)\right), \tag{2.6}$$

and

$$u = (1, \ldots, 1)^T \in \mathbf{R}^N. \tag{2.7}$$

Unfortunately, the system (2.4) can not be solved and only numerical methods and iterative learning schemes of gradient decent type can be used to approximate the optimal MSE model. The computation of the hessian matrix of the objective function $F_N(\beta)$ yields to

$$H_\beta\left(F_N(\beta)\right) = \frac{2}{N} Z D_1(\beta)\left(-3D_1(\beta)^2 + 2D_1(\beta) + 2\widetilde{Y}D_1(\beta) - \widetilde{Y}\right)\left(\mathbf{I}_N - D_1(\beta)\right) Z^T, \tag{2.8}$$

where the matrix $\widetilde{Y}$ is the diagonal matrix whose entries are the labels $y_1, \ldots, y_N$.

If β is an approximation of a solution of system (2.4) obtained by a numerical method, it is computationally difficult to check the positive definiteness of $H_\beta\left(F_N\left(\widetilde{\beta}\right)\right)$, especially because of its dependency on the set of examples Z. Several tests pointed out that there are chances that $H_\beta\left(F_N(\beta)\right)$ is in general positive semi-definite in the sense that we did not find any set of examples Z yielding to a negative-definite matrix $H_\beta\left(F_N\left(\widetilde{\beta}\right)\right)$.

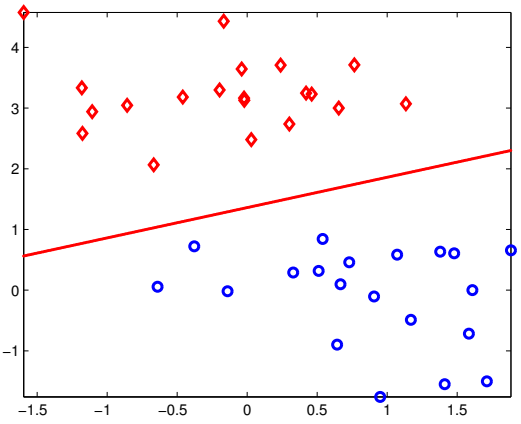

Figure 1. *The sub-optimal classifier* $H_{\beta^\star}$.

The alternative approach is to develop iterative learning schemes allowing the computation of approximates of local minima of the objective function $F_N(\beta)$. For instance, the standard gradient descent algorithm and its stochastic variant use the updating rules

$$\beta \leftarrow \beta - \rho Z D_1(\beta)\left(D_1(\beta) - \mathbf{I}_N\right)\left(Y - D_1(\beta)u\right),$$

and

$$\beta \leftarrow \beta - \rho\left(g_\beta(z_i) - y_i\right) g_\beta(z_i)\left(1 - g_\beta(z_i)\right) z_i,$$

respectively, where $\rho > 0$ is the learning rate.

From experimental point of view there are not major differences between the sub-optimal solutions computed by batch and stochastic gradient learning algorithms. Most of the tests were performed on multi-dimensional data simulated from Gaussian repartitions.

For instance, in Figure 1 the results obtained in case of two dimensional input data generated from the Gaussian repartitions of means $(0,3)^T$ and $(1,0)^T$, respectively, and equal covariance matrices the generated data were linearly separable. The variation of the performance function $F_N(\beta)$ is presented in Figure 2. After 5000 iterations, the fluctuations of $F_N(\beta)$ became insignificant around the parameter value $\beta^\star = (2.05,\, 0.96,\, -1.62)^T$. The data and the resulted separating hyperplane $H_{\beta^\star}$ are represented in Figure 1. The quality of the learned model $\beta^\star$ to explain the behavior of $\mathbf{S}$ concerning the sequence of input $x_1, \ldots, x_N$ can be expressed in terms of the likelihood function

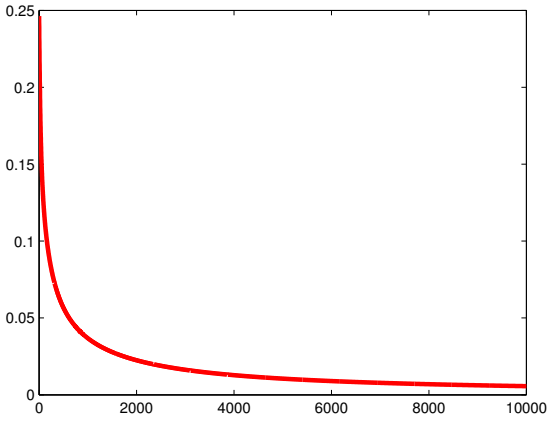

Figure 2: *The variation of the criterion function.*

$$L\left(\beta^\star, \mathcal{S}_N\right) = \prod_{i=1}^{N} P\left(y = y_i \mid x = x_i, \beta^\star\right) = \prod_{i=1}^{N} \left(g_{\beta^\star}\left(z_i\right)\right)^{y_i} \left(1 - g_{\beta^\star}\left(z_i\right)\right)^{1-y_i}.$$

In case of our example we obtained 0.44.

2.2. Maximum Likelihood (ML) Estimates of the Logistic Model

In the framework of the probabilistic model (2.1) the capacity of each model β to explain the sequence of labeled examples $\mathcal{S}_N$ is given by the value of the likelihood function

$$L\left(\beta, \mathcal{S}_N\right) = \prod_{i=1}^{N} p\left(y_i \mid x_i, \beta\right) = \prod_{i=1}^{N} \left(\frac{g_\beta\left(z_i\right)}{1 - g_\beta\left(z_i\right)}\right)^{y_i} \left(1 - g_\beta\left(z_i\right)\right).$$

Therefore, $L\left(\beta, \mathcal{S}_N\right)$ can be taken as a performance function, the best model being a solution of the optimization problem

$$\widehat{\beta}_{ML} = \arg\left(\max_{\beta \in \mathbf{R}^{n+1}} L\left(\beta, \mathcal{S}_N\right)\right), \tag{2.9}$$

or, equivalently a solution of the optimization problem involving the log-likelihood function $l(\beta, \mathcal{S}_N) = \ln(L(\beta, \mathcal{S}_N))$, that is

$$\widehat{\beta}_{ML} = \arg\left(\max_{\beta \in \mathbf{R}^{n+1}} \left(\sum_{i=1}^{N} y_i \ln\left(\frac{g_\beta(z_i)}{1 - g_\beta(z_i)}\right) + \sum_{i=1}^{N} \ln(1 - g_\beta(z_i))\right)\right) \quad (2.10)$$

The space of critical points of $l(\beta, \mathcal{S}_N)$ is the set of solutions of the system

$$\nabla_\beta l(\beta, \mathcal{S}_N) = 0\,, \quad (2.11)$$

where $\nabla_\beta l(\beta, \mathcal{S}_N)$ stands for the gradient of the log-likelihood function.

Using straightforward computations, for $1 \le k \le n+1$,

$$(\nabla_\beta l(\beta, \mathcal{S}_N))_k = \frac{\partial l(\beta, \mathcal{S}_N)}{\partial \beta_k} = \sum_{i=1}^{N} \left(\frac{(y_i - g_\beta(z_i))}{g_\beta(z_i)(1 - g_\beta(z_i))} \frac{\partial g_\beta(z_i)}{\partial \beta_k}\right),$$

and since

$$\frac{\partial g_\beta(z_i)}{\partial \beta_k} = g_\beta(z_i)(1 - g_\beta(z_i)) z_i^{(k)},$$

we finally get

$$\nabla_\beta l(\beta, \mathcal{S}_N) = \sum_{i=1}^{N} (y_i - g_\beta(z_i)) z_i,$$

that is the compact form is

$$\nabla_\beta l(\beta, \mathcal{S}_N) = ZY - Z \begin{pmatrix} g_\beta(z_1) \\ \vdots \\ g_\beta(z_N) \end{pmatrix},$$

where $z_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$.

Obviously for any $\beta \in \mathbf{R}^{n+1}$ and $1 \le p, k \le n+1$,

$$\frac{\partial^2 l(\beta, \mathcal{S}_N)}{\partial \beta_p \partial \beta_k} = \frac{\partial}{\partial \beta_p}\left(\sum_{i=1}^{N} (y_i - g_\beta(z_i)) z_i^{(k)}\right) = -\sum_{i=1}^{N} z_i^{(k)} \left(\frac{\partial}{\partial \beta_p} g_\beta(z_i)\right) =$$
$$-\sum_{i=1}^{N} \left(g_\beta(z_i) z_i^{(p)}\right)\left((1 - g_\beta(z_i)) z_i^{(k)}\right).$$

therefore the Hessian matrix of the log-likelihood function is

$$H(l(\beta, \mathcal{S}_N)) = \left\| \frac{\partial^2 l(\beta, \mathcal{S}_N)}{\partial \beta_p \partial \beta_k} \right\| = -Z D_1(\beta) (\mathbf{I}_N - D_1(\beta)) Z^T. \quad (2.12)$$

For any $v \in \mathbf{R}^{n+1} \setminus \{\mathbf{0}\}$,

$$v^T H(l(\beta, \mathcal{S}_N))v = -\left(Z^T v\right) D_1(\beta) \left(\mathbf{I}_N - D_1(\beta)\right) Z^T v = \\ -\sum_{i=1}^{N} \left(Z^T v\right)_i^2 g_\beta(z_i)\left(1 - g_\beta(z_i)\right) \leq 0 \tag{2.13}$$

therefore for all values of the parameter $\beta \in \mathbf{R}^{n+1}$, $H(l(\beta, \mathcal{S}_N))$ is a symmetric negative semi-definite matrix.

The critical points of $l(\beta, \mathcal{S}_N)$ are the values of the parameter β, solutions of the system

$$Z \begin{pmatrix} g_\beta(z_1) \\ \vdots \\ g_\beta(z_N) \end{pmatrix} = ZY. \tag{2.14}$$

In order to find a solution β of (2.14), one should determine first the vector $\begin{pmatrix} g_\beta(z_1) \\ \vdots \\ g_\beta(z_N) \end{pmatrix}$. A computable solution $\begin{pmatrix} g_\beta(z_1) \\ \vdots \\ g_\beta(z_N) \end{pmatrix}$ of (2.14) expressed in terms of the generalized inverse (Penrose pseudo-inverse, [6]) is

$$\begin{pmatrix} g_\beta(z_1) \\ \vdots \\ g_\beta(z_N) \end{pmatrix} = Z^+ ZY. \tag{2.15}$$

Indeed, using the well known properties of the Penrose pseudo-inverse, $ZZ^+ZY = ZY$, that is (2.15) satisfies (2.14).

In case of most iterative algorithms that compute the generalized inverse of a matrix (for instance, the Greville's method, [11], [12]) the number of iterations equals the number of columns of the processed matrix. Consequently, being given the fact that Z has N columns, the evaluation of Z^+ could become computationally too expensive in case of large size data sets. In the particular case when ZZ^T is invertible, the complexity of the computation scheme can be slightly decreased because $Z^+ = \left(ZZ^T\right)^{-1} Z$. Indeed, using the well known properties of the generalized inverse, if ZZ^T is invertible, then

$$\left(ZZ^T\right)^{-1} = \left(ZZ^T\right)^+ = \left(Z^T\right)^+ Z^+ = \left(Z^+\right)^T Z^+,$$

therefore

$$\left(ZZ^T\right)^{-1} Z = \left(Z^+\right)^T Z^+ Z = \left(Z^+\right)^T \left(Z^+ Z\right)^T = \left(Z^+ ZZ^+\right)^T = Z^+,$$

and consequently, (2.15) becomes

$$\begin{pmatrix} g_\beta(z_1) \\ \vdots \\ g_\beta(z_N) \end{pmatrix} = Z^T \left(ZZ^T\right)^{-1} ZY. \tag{2.16}$$

The aim is to find a parameter value $\widehat{\beta}_{ML}$ that maximizes the log-likelihood function. Two main problems arise in solving (2.15). On one hand, since the entries of Z and Y are determined by the set of examples, some entries of Z^+ZY could happen to lay outside the interval $(0,1)$. Unfortunately, a long series of tests based on simulated data confirmed that this case occurs very often. On the other hand, even when all entries of Z^+ZY belong to $(0,1)$, the resulted solution $\widetilde{\beta}$ is not guaranteed to maximize the log-likelihood function because $H\left(l\left(\widetilde{\beta},\mathcal{S}_N\right)\right)$ can happen to be singular.

If all entries of Z^+ZY (or $Z^T\left(ZZ^T\right)^{-1}ZY$, in case it exists) belong to the interval $(0,1)$, in order to find a suitable parameter $\widetilde{\beta}$ we should proceed as follows. If we denote $v_i = (Z^+ZY)_i$, $1 \leq i \leq N$, then (2.15) can be written as

$$-\beta^T z_i = \ln\left(\frac{1}{v_i} - 1\right), \quad 1 \leq i \leq N, \tag{2.17}$$

its compact form being

$$\beta^T Z = V,$$

or equivalently,

$$Z^T\beta = V^T,$$

where the entries of V are $\ln\left(\frac{v_i}{1-v_i}\right)$, $1 \leq i \leq N$.

Using standard MSE arguments, by minimizing $\left\|Z^T\beta - V^T\right\|$, we get

$$\widehat{\beta} = \left(Z^T\right)^+ V^T = \left(Z^+\right)^T V^T = \left(VZ^+\right)^T. \tag{2.18}$$

We refer $\widehat{\beta}$ as a quasi-optimal model.

In case some of the entries of Z^+ZY lay outside the interval (0,1), obviously it does not exists $\beta \in \mathbf{R}^{n+1}$ when such that (2.15) holds. Based on the continuous dependency of the log-likelihood function on β, several attempts to overpass this difficulty could be intuitively justified, yielding to sub-optimal models. One possibility is to try to find out a point in the $(n+1)$-dimensional hypercube $(0,1)^{n+1}$ at the smallest distance to Z^+ZY, that is to minimize the objective function $\varphi(\beta) = \left\|\left(g_\beta(z_1),\ldots,g_\beta(z_N)\right)^T - Z^+ZY\right\|^2$, yielding to a sub-optimal model

$$\widehat{\beta}^{(1)} = \arg\left(\min_{\beta\in\mathbf{R}^{n+1}}\varphi(\beta)\right). \tag{2.19}$$

Another idea is to use a "brute force" approach inspired by the expression of the quasi-optimal model (2.18) that would result if all entries of Z^+ZY were inside the interval $(0,1)$. Let $\varepsilon \in (0,1)$ be a relatively small threshold value and define

$$h_i = \begin{cases} \left(Z^+ZY\right)_i & , \text{ if } (Z^+ZY)_i \in (0,1) \\ \varepsilon & , \text{ if } (Z^+ZY)_i \leq 0 \\ 1-\varepsilon & , \text{ if } (Z^+ZY)_i \geq 1 \end{cases}, \quad 1 \leq i \leq N.$$

If we denote by $\widehat{V}$ the vector having as entries $\ln\left(\frac{h_i}{1-h_i}\right)$ then $\widehat{\beta}^{(2)}$ is an ε-sub-optimal model, where

$$\widehat{\beta}^{(2)} = \left(\widehat{V}Z^{+}\right)^{T} \tag{2.20}$$

Finally, standard learning schemes of gradient ascent and stochastic gradient ascent types to maximize the log-likelihood function could be a base for providing sub-optimal models. The adaptive learning of a sub-optimal ML-model using a gradient based approach can be described as follows. Obviously,

$$\nabla_{\beta} l\left(\beta, \mathcal{S}_N\right) = Z\left(Y - D_1(\beta)u\right), \tag{2.21}$$

where the matrix $D_1(\beta)$ and the vector u are given by (2.6) and (2.7).

Let $\beta_0 \in \mathbf{R}^{n+1}$ be an arbitrary parameter value, $\rho > 0$ a small learning rate and $\mathcal{C}$ a conventionally stopping condition expressed in terms of an upper threshold of a number of iterations and/or a threshold to control the accuracy. The updating phases of the gradient ascent and stochastic gradient ascent learning algorithms to maximize the log-likelihood function are

$\beta^{(old)} \leftarrow \beta_0$
repeat
  $D_1 \leftarrow \mathtt{diag}\left(g_{\beta^{(old)}}(x_1), \ldots, g_{\beta^{(old)}}(x_N)\right)$
  $\beta^{(new)} \leftarrow \beta^{(old)} + \rho Z\left(Y - D_1 u\right)$
  $\beta^{(old)} \leftarrow \beta^{(new)}$
until $\mathcal{C}$

and

$\beta^{(old)} \leftarrow \beta_0$
repeat
  $\beta \leftarrow \beta^{(old)}$
  for $i = 1, N$
    for $k = 1, n+1$
      $\beta_k^{(new)} \leftarrow \beta_k + \rho z_i^{(k)}\left(y_i - g_{\beta}(z_i)\right)$
    endfor
    $\beta \leftarrow \beta^{(new)}$
  endfor
  $\beta^{(old)} \leftarrow \beta^{(new)}$
until $\mathcal{C}$

respectively.

The sub-optimal models $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ computed by these two algorithms respectively are represented by the parameter value $\beta^{(old)}$ resulted when the stopping condition $\mathcal{C}$ holds.

2.3. The performance analysis of ML-estimates of the logistic model

The performance analysis of the ML-estimates of the logistic model obtained by the approaches presented in the previous section is performed experimentally on design and test data generated from the same repartitions. The quality of each computed model β has to be evaluated from two points of view. On one hand, since each parameter value β corresponds to a particular hypothesis implemented at the level of the component **L** one way to evaluate its performance in estimating the unknown input-output dependency of **S** is to express quantitatively its capacity to predict the output of **S** in terms of the values of the likelihood function taken on the examples used in the design phase as well as on test data generated from the same repartitions. In other words, the generating mechanism **G** according to which the inputs from the input space are selected is stationary during the learning and the test phases. On the other hand, for each model β, since for any input x, $g_\beta(z) \geq \frac{1}{2} \Leftrightarrow \beta^T z \geq 0$, where $z = \begin{pmatrix} 1 \\ x \end{pmatrix}$, another way to express the quality of β is by its performance when it is used as a linear classifier. The aim of this section is to provide a series of experimentally derived conclusions concerning the qualities of the proposed sub-optimal ML estimates from these two points of view. In our tests the data were represented by samples of different sizes generated from multidimensional Gaussian repartitions. One of the conclusions is that the sub-optimal models $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ are equivalent from both points of view.

From the point of view of the criterion referring to the predictive capacity our tests pointed out that models $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ supply the best estimates of the unknown input-output dependency of **S**, the values of the likelihood function being around 0.95. Concerning the estimate $\widehat{\beta}^{(2)}$ the values of the likelihood function were around 0.84, the values corresponding to the likelihood function in case of $\widehat{\beta}^{(1)}$ being very small. The generalization capacities of $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$, $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ expressed in terms of the values of likelihood function are quite small but again the models $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ supply the largest values. Used as parameters of linear classifiers

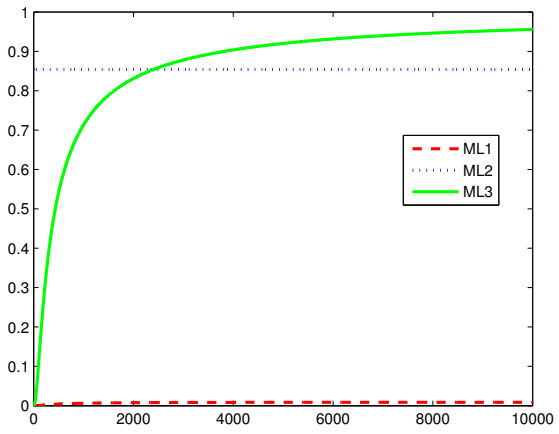

Figure 3: *The variation of the likelihood function.*

all models managed to correctly classify the design data sets.

In most cases $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ classify without errors most of the test data, while the error rates corresponding to $\widehat{\beta}^{(2)}$ and $\widehat{\beta}^{(1)}$ were about 4% and 1.5% respectively. We can conclude that the best logistic models are given by the solutions computed by the gradient and stochastic gradient respectively viewed as ML-suboptimal hypotheses as well as the parameters of linear classifiers in the input space. However, the solution $\widehat{\beta}^{(2)}$ is computed directly from data, without invoking an iterative method proves almost as good as the solution $\widehat{\beta}^{(3)}$, $\widehat{\beta}^{(4)}$ also in learning the parameters of the logistic model and as classifier, while the solution $\widehat{\beta}^{(1)}$ supplies poor behavior from both points of view.

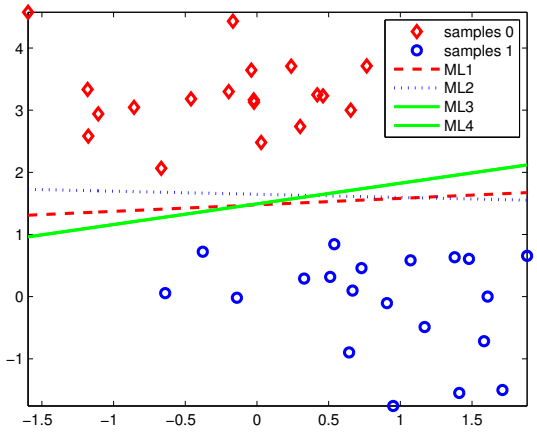

Figure 4. *The ML-estimated linear classifiers .*

In the following we present the results of one of the series of tests performed in order to establish experimentally based conclusions concerning the qualities of the ML-estimates $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$, $\widehat{\beta}^{(3)}$, and $\widehat{\beta}^{(4)}$.

The test data containing 20 examples from each class were generated from the two dimensional Gaussian repartitions $\mathcal{N}(\mu_i, \Sigma_i)$, $i \in \{1, 2\}$, where

$$\mu_1 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \ \mu_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

$$\Sigma_1 = \Sigma_2 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}.$$

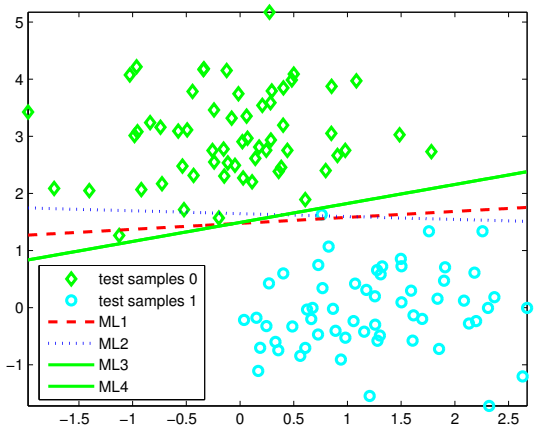

Figure 5: *The performance of the ML-estimated linear classifiers on new test data.*

The setting of the correction parameter $\varepsilon \in (0, 1)$ in case of the solution $\widehat{\beta}^{(2)}$ was 10^{-10} and the stopping condition in computing the solutions $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ was set in terms of a threshold on the maximum number of iterations combined with a condition imposed on the value of the likelihood function.

The values of the likelihood function corresponding to the resulted ML-estimates were

$$L\left(\widehat{\beta}^{(1)},\mathcal{S}_N\right)=0.0085,\quad L\left(\widehat{\beta}^{(2)},\mathcal{S}_N\right)=0.8542,$$
$$L\left(\widehat{\beta}^{(3)},\mathcal{S}_N\right)=0.9557,\quad L\left(\widehat{\beta}^{(4)},\mathcal{S}_N\right)=0.9557.$$

The variation of the values of the likelihood functions during the learning processes yielding to the ML estimates $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$, $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ is represented in Figure 3. Suitable estimates $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$ $\widehat{\beta}^{(3)}$, and $\widehat{\beta}^{(4)}$ were used notations ML1, ML2, ML3 and ML4 respectively.

The linear classifiers of parameters $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$, $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ are represented in Figure 4. Note that the classifiers of parameters $\widehat{\beta}^{(3)}$, and $\widehat{\beta}^{(4)}$ are equivalent. In order to test the generalization capacity of each of these models as well as their corresponding performance as linear classifiers we used new linearly separable samples of sizes 60 generated from the same repartitions. The values of the likelihood function were

$$L\left(\widehat{\beta}^{(1)},\mathcal{S}_N\right)=0.0315\cdot 10^{-6},\quad L\left(\widehat{\beta}^{(2)},\mathcal{S}_N\right)=0.0593,$$
$$L\left(\widehat{\beta}^{(3)},\mathcal{S}_N\right)=0.2276,\quad L\left(\widehat{\beta}^{(4)},\mathcal{S}_N\right)=0.2277.$$

In Figure 5 are shown the linear classifiers of parameters $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$, $\widehat{\beta}^{(3)}$, $\widehat{\beta}^{(4)}$, respectively, and the new linearly separable test data.

Similar results were obtained in case of different tests performed on other class repartitions in case of linearly separable design and test data sets.

The error rates corresponding to the classifiers of parameters $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ were zero while two examples were misclassified in case of $\widehat{\beta}^{(1)}$ and 5 errors in case of $\widehat{\beta}^{(2)}$. In order to test the performance of the learned ML estimates on possibly non-linearly separable data sets generated from the same classes, a set of 100 new examples coming from each class was generated providing a non-linearly separable data sets of size 200. In Figure 6 are shown the data set and the linear classifiers of parameters $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$, $\widehat{\beta}^{(3)}$, and $\widehat{\beta}^{(4)}$. In case of the linear classifiers of parameters $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ four examples were incorrectly

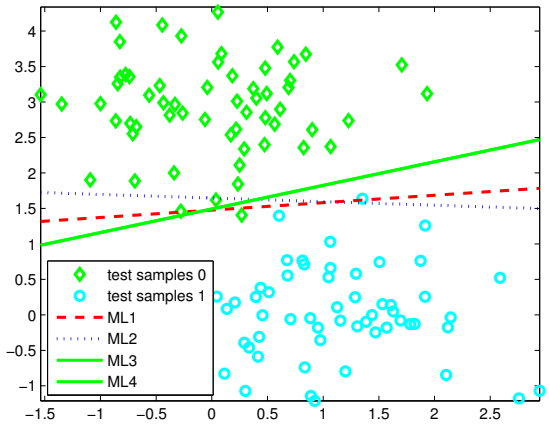

Figure 6: *The performance of ML-estimated linear classifiers on new non-linearly separable test data.*

classified while the classifiers of parameters $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$ misclassified 5 and 6 examples respectively.

The conclusions concerning the performance of ML-estimated linear classifiers concerning both the quality of the learned model expressed in terms of the values of the likelihood function and the empirical error of the resulted linear classifier are summarized in Table 1. Similar results were obtained incase of several simulated design/test data coming from different Gaussian repartitions.

Table 1. *The performance of the ML-estimated classifiers.*

	$\widehat{\beta}^{(1)}$	$\widehat{\beta}^{(2)}$	$\widehat{\beta}^{(3)}$	$\widehat{\beta}^{(4)}$
The value of likelihood function for linearly separable design samples	0.008	0.854	0.956	0.956
The value of likelihood function for new linearly separable test samples	$3 \cdot 10^{-6}$	0.003	0.164	0.164
The value of likelihood function for new non-linearly separable test samples	$1 \cdot 10^{-14}$	$2 \cdot 10^{-10}$	$4 \cdot 10^{-6}$	$4 \cdot 10^{-6}$
The value of empirical classification error for linearly separable design samples	0	0	0	0
The value of empirical classification error for new linearly separable test samples	1/120	2/120	1/120	1/120
The value of empirical classification error for new non-linearly separable test samples	4/400	8/400	6/400	6/400

According to the results of a long series of tests, the logistic model is best learned in case of the methods of computing the estimates $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ followed by the method yielding to $\widehat{\beta}^{(2)}$ while the estimate $\widehat{\beta}^{(1)}$ supplies poor results. The quality of the corresponding learned models is confirmed by the values of the likelihood functions on new linearly separable test data. The values of the likelihood function taken on non-linearly separable test data generated from the same repartitions are, as it is expected pretty small, but again, the estimates $\widehat{\beta}^{(3)}$ and $\widehat{\beta}^{(4)}$ prove significantly better than $\widehat{\beta}^{(1)}$ and $\widehat{\beta}^{(2)}$. Consequently, the learning algorithms of gradient ascent type compute the best estimates of the parameter β in case of the logistic model. From the point of view of classification purpose the linear classifiers of parameters $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$, $\widehat{\beta}^{(3)}$, and $\widehat{\beta}^{(4)}$ prove the same quality in case of linearly separable design and new test data. However, in case of non-linearly separable test data, it seems that the classifier of parameter $\widehat{\beta}^{(1)}$ is slightly better than the linear classifiers of parameters $\widehat{\beta}^{(2)}$, $\widehat{\beta}^{(3)}$, $\widehat{\beta}^{(4)}$.

3. Comparative analysis of different learning algorithms

3.1. Brief presentation of some learning algorithms in designing linear classifiers

So far, there have been proposed a large class of supervised learning algorithms for the parameters of linear classifiers, some of them reducing to the optimization of a conventionally selected criterion function. In this section, for numerical reasons, a learning sequence is represented by a finite labeled set of examples, where each example is labeled by 1 or -1,

$$\mathcal{S}_N = \left\{ (x_i, y_i) \mid x_i = \left(x_i^{(1)}, \ldots, x_i^{(n)}\right)^T \in \mathbf{R}^n \,,\ y_i \in \{-1, 1\} \,,\ i = \overline{1, N} \right\} . \quad (3.1)$$

By encoding the label at the level of representation, the learning sequence is also represented

$$\mathcal{Z}_N = \left\{ z_i \mid z_i = y_i \begin{pmatrix} 1 \\ x_i \end{pmatrix} , (x_i, y_i) \in \mathcal{S}_N \,,\ i = \overline{1, N} \right\} . \quad (3.2)$$

The learning sequence is linearly separable if there exists $w \in \mathbf{R}^{n+1}$, $w = \begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix} \in \mathbf{R}^{n+1}$, such that for each i, $1 \leq i \leq N$, $h_w(x_i) > 0$, if $y_i = 1$, and $h_w(x_i) < 0$, if $y_i = -1$, where $h_w(x) = w^T \begin{pmatrix} 1 \\ x_i \end{pmatrix}$.

Obviously, being given $w \in \mathbf{R}^{n+1}$, the example (x_i, y_i) is correctly classified by h_w if $w^T z_i > 0$. For each example (x_i, y_i), we denote by

$$H_i = \left\{ w \in \mathbf{R}^{n+1} \mid w^T z_i = 0 \right\} ,$$

the hyperplane whose parameters are the entries of z_i and

$$S^+ H_i = \left\{ w \in \mathbf{R}^{n+1} \mid w^T z_i > 0 \right\} ,$$

the positive hyperspace with respect to H_i. Therefore the learning sequence $\mathcal{Z}_N$ is linearly separable if $\bigcap_{i=1}^{N} S^+ H_i \neq \emptyset$. We denote by $\bigcap_{i=1}^{N} S^+ H_i$ the set of solutions and by $\bigcap_{i=1}^{N} \left(S^+ H_i \bigcup H_i \right)$ the set of admissible solutions of the classification problem.

For each $w \in \mathbf{R}^{n+1}$, we denote by

$$\mathcal{Z}(w) = \left\{ z_i \in \mathcal{Z}_N \mid w^T z_i \leq 0 \right\} ,$$

the set of misclassified examples.

The function $f : \mathbf{R}^{n+1} \to [0, +\infty)$ is a criterion function if f is minimized on the set of admissible solutions. An example of criterion function is ([5])

$$f(w) = \begin{cases} -\sum\limits_{z_i \in \mathcal{Z}(w)} w^T z_i & , \quad \text{if } \mathcal{Z}(w) \neq \emptyset \\ 0 & , \quad \text{if } \mathcal{Z}(w) = \emptyset \end{cases}$$

Denoting by $\rho > 0$ the value of a conventionally selected learning rate, the updating rule obtained by applying gradient descent method to minimize f is

$$w \leftarrow w + \rho \sum_{z_i \in \mathcal{Z}(w)} z_i \,, \text{ if } \mathcal{Z}(w) \neq \emptyset,$$

the stopping condition being expressed in terms of the cardinal of $\mathcal{Z}_w$.

It is interesting to note that in case $\rho = 1$, the updating rule used by the stochastic gradient method becomes " if $\left(w^T z_i \leq 0\right)$ then $w \longleftarrow w + z_i$ ", which is the perceptron learning algorithm.

In case $\mathcal{Z}_N$ is linearly separable, in order to assure good generalization capacities, we would like to compute a solution as equidistant as possible to the each of the hyper-planes H_i, $1 \leq i \leq N$. Unfortunately, most of the learning algorithms fail to compute such a solution. In order to prevent de computation of a solution placed "too close" to one or more hyper-planes a conventionally selected parameter $b > 0$ could be used to define the set of misclassified examples. In such a case for each $w \in \mathbf{R}^{n+1}$ we define

$$\mathcal{Z}(w, b) = \left\{ z_i \in \mathcal{Z}_N \mid w^T z_i \leq b \right\},$$

and, consequently, the updating rules of the learning algorithms developed by applying the gradient descent and stochastic gradient methods become

" if $\mathcal{Z}(w, b) \neq \emptyset$ then $w \longleftarrow w + \rho \sum\limits_{z_i \in \mathcal{Z}(w,b)} z_i$ "

and

" if $\left(w^T z_i \leq b\right)$ then $w \longleftarrow w + \rho z_i$ ", respectively.

The class of "relaxation methods" ([5], [14]) uses the criterion function

$$g(w) = \begin{cases} \frac{1}{2} \sum\limits_{z_i \in \mathcal{Z}(w,b)} \frac{\left(w^T z_i - b\right)^2}{\|z_i\|^2} & , \quad \text{if } \mathcal{Z}(w, b) \neq \emptyset \\ 0 & , \quad \text{if } \mathcal{Z}(w, b) = \emptyset \end{cases}$$

If we denote by $H_{i,b}$ the hyper-plane of equation $w^T z_i - b = 0$, the distance from any $\widetilde{w} \in \mathbf{R}^{n+1}$ to $H_{i,b}$ is $d\left(\widetilde{w}, H_{i,b}\right) = \dfrac{\left|\widetilde{w}^T z_i - b\right|}{\|z_i\|}$. In a way, w induces a "tension"expressed by $d^2\left(w, H_{i,b}\right)$ on each $H_{i,b}$ such that $z_i \in \mathcal{Z}(w, b)$, therefore the value of $g(w)$ gives the overall "tension" induced by

w. According to this point of view, for fixed $b > 0$ the value $g(w)$ can be taken as measure of the degree of fitness of the hyper-plane of equation $w^T z - b = 0$ to separate $\mathcal{S}_N$ in the space of examples.

Obviously, the updating rules of the gradient descent algorithm and its sequential variant (stochastic gradient) for minimizing the criterion function g are

" if $\mathcal{Z}(w,b) \neq \emptyset$ then $w \longleftarrow w - \rho \sum_{z_i \in \mathcal{Z}(w,b)} \frac{w^T z_i - b}{\|z_i\|^2} z_i$"

and

" if $w^T z_i \leq b$ then $w \longleftarrow w - \rho \frac{w^T z_i - b}{\|z_i\|^2} z_i$", respectively.

The stochastic gradient descent algorithm where $\rho \in (0,2)$ is known as the Mays learning scheme ([12], [5]).

Nowadays, a class of the most popular methods for designing linear and non-linear classifiers, referred as Support Vector Machines (SVM), are based on the concept of support vector ([15]). The classifiers obtained by SVM approaches have remarkable properties in assuring generalization capacities ([2], [3], [4]). In our approaches we considered only the simplest type of SVM developed for designing a linear classifier when the learning sequence is linearly separable. Let $\mathcal{S}_N$ be a labeled sequence of n-dimensional examples of size N and $\mathcal{Z}_N$ the sequence of $(n+1)$-dimensional representations as in (3.1) and (3.2). The aim is to determine the parameters w and b of a hyper-plane $H_{w,b}$ in the n-dimensional space of examples such that the point $\widetilde{w} = \begin{pmatrix} b \\ w \end{pmatrix}$ is as equidistant as possible of the hyper-planes $H_i : \ \widetilde{w}^T z_i = 0$ in the $(n+1)$-dimensional space of parameters. From mathematical point of view the linear SVM problem is expressed as the constrained quadratic optimization problem (QP)([1])

$$\begin{cases} \min\limits_{w \in \mathbf{R}^n, b \in \mathbf{R}} \|w\|^2 \\ \\ y_i \left(w^T x_i + b \right) \geq 1 \,, \quad i = \overline{1, N} \,, \end{cases} \tag{3.3}$$

its dual problem being the constrained QP-problem

$$\begin{cases} \max\limits_{\alpha = (\alpha_1, \ldots, \alpha_N)} \left(-\frac{1}{2} \sum\limits_{i=1}^{N} \sum\limits_{k=1}^{N} \left(\alpha_i \alpha_k y_i y_k \left(x_i^T x_k \right) \right) + \sum\limits_{i=1}^{N} \alpha_i \right) \\ \\ \alpha_i \geq 0 \,, \quad \forall \, 1 \leq i \leq N \,, \\ \\ \sum\limits_{i=1}^{N} \alpha_i y_i = 0 \,. \end{cases} \tag{3.4}$$

If $\alpha^* = (\alpha_1, \ldots, \alpha_N)$ is a solution of (3.4) then $w^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i$. The support vectors are the examples x_i for which $\alpha_i^* \neq 0$. The bias parameter b can not be determined by solving the SVM-QP problem is computed in terms support vectors. There are several ways to determined the bias b, for instance

$$b^* = \frac{1}{|\mathcal{S}^*|} \sum_{x_i \in \mathcal{S}^*} \left(y_i - (w^*)^T x_i \right).$$

where

$$\mathcal{S}^* = \left\{ x_i \,|\, (x_i, y_i) \in \mathcal{S}_N,\, y_i \left((w^*)^T x_i + b \right) = 1 \right\} = \{ x_i \,|\, (x_i, y_i) \in \mathcal{S}_N,\, \alpha_i^* > 0 \},$$

is the set of support vectors ([1]).

3.2. Comparative analysis of the estimates computed by ML, MSE, perceptron, Mays and SVM learning algorithms

The aim of this section is to present conclusions concerning the performance of the estimates of the parameter β computed by the MSE and ML-based methods presented in section 2 and the learning schemes presented in section 3.1. The comparative analysis aims to derive conclusions concerning the qualities of the estimates computed by the previous mentioned methods from the following point of views:

1. The quality of the learned logistic model expressed in terms of the values of the likelihood function evaluated on the design and new test data respectively.

2. The quality of the resulted linear classifier expressed in terms of the empirical error evaluated on the basis of linear/nonlinear test data generated from the same multivariate repartitions

3. The generalization capacity corresponding to the resulted models evaluated in terms of the values of the likelihood function and the empirical error on new test data generated from the same multivariate repartitions.

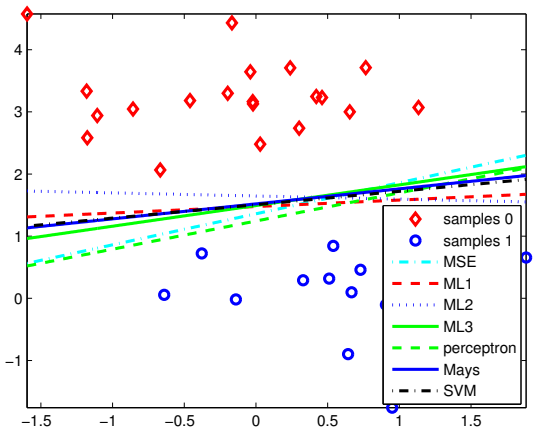

Figure 7: *Linear classifiers on linearly separable design data.*

The analysis is performed exclusively from experimental point of view, the tests being developed on a long series of simulated data generated from multivariate Gaussian repartitions.

The results resumed in the tables and figures included in this section correspond the following test. The design data set is linearly separable and contains 40 examples generated from the two-dimensional Gaussian repartitions $\mathcal{N}(\mu_i, \Sigma_i)$, $i = 1, 2$, $\mu_1 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$, $\mu_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\Sigma_1 = \Sigma_2 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$, each class containing 20 examples.

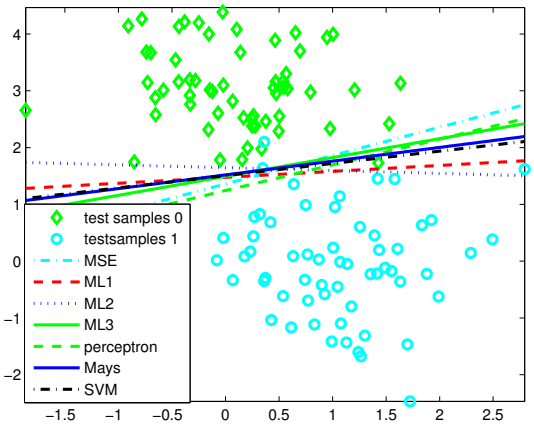

Figure 8: *The behavior of the designed linear classifiers on new test data.*

The tests were performed on two test data sets generated from the same repartitions, the former being linearly separable and the latter being a non-linearly separable one of sizes 60 and 200 respectively. The values of the likelihood function and the empirical classification error are summarized in Table 1 and Table 2. The linear classifiers computed on the test data are showed in Figure 7 and their performance on test data is presented in Figure 8. The variation of the values of the likelihood function during the learning process is depicted in Figure 9.

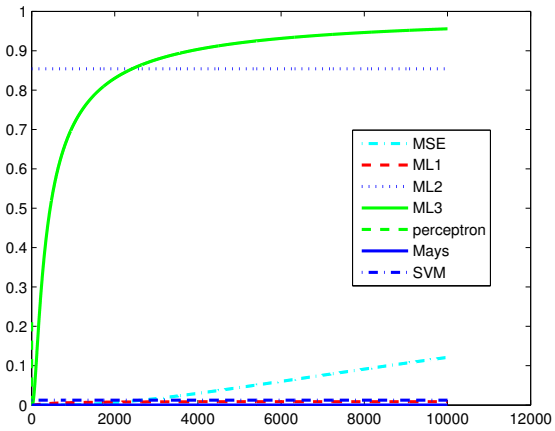

Figure 9: *The variations of the likelihood functions L.*

On the basis of information supplied by Table 1 and Table 2 (resulted by a test performed on the same design and test data) also these eight estimates have a similar behaviors for classification purposes, the estimates $\beta^\star$, β_{perc}, β_{Mays} and β_{SVM} fail to learn the logistic model, the values of the likelihood

Table 2: *The quality of the learned model and the resulted performance for classification purposes.*

	$\beta^\star$	β_{perc}	β_{Mays}	β_{SVM}
The value of likelihood function for linearly separable design samples	0.121	0.231	$1 \cdot 10^{-5}$	0.124
The value of likelihood function for new linearly separable test samples	$3 \cdot 10^{-4}$	0.005	$5 \cdot 19^{-15}$	$1 \cdot 10^{-7}$
The value of likelihood function for new non-linearly separable test samples	$1 \cdot 10^{-14}$	$1 \cdot 10^{-10}$	$1 \cdot 10^{-18}$	$1 \cdot 10^{-18}$
The value of empirical classification error for linearly separable design samples	0	0	0	0
The value of empirical classification error for new linearly separable test samples	0	0	0	0
The value of empirical classification error for new non-linearly separable test samples	4/400	4/400	3/400	3/400

function being very small for both, design and test data, even when the data are linearly separable.

4. Conclusions

The research reported in the paper is focused on a series of parametric approximation methods in a logistic approach of classification and prediction tasks. The parameters of the probabilistic logistic model presented in Section 2 are estimated using the MSE and ML methods. In order to overpass the difficulties pointed out in Section 2.2, four suboptimal estimations $\widehat{\beta}^{(1)}$, $\widehat{\beta}^{(2)}$, $\widehat{\beta}^{(3)}$, and $\widehat{\beta}^{(4)}$ are introduced and a comparative analysis of their qualities is performed. The experimental analysis pointed out good behavior in case of linearly separable design data sets from both points of view, the capacity of learning the logistic model and classification of new linearly separable and non-linearly separable data sets, respectively. In the next section, estimations of the parameters of the logistic model are computed on the basis of some of the most frequently used learning algorithms as MSE, perceptron, Mays and SVM.

In the final part of the third section a series of conclusions established on experimental basis concerning their performance in learning the logistic model and classifying new data sets are reported. The results confirmed significantly better performance of the suboptimal estimates introduced in

the second section as compared to the estimates computed by the algorithms MSE, perceptron, Mays and SVM.

References

[1] S. Abe, *Support Vector Machines for Pattern Classification*, Springer-Verlag, 2005, 2010.

[2] E. Alpaydin, *Introduction to Machine Learning*, The MIT Press, Massachusetts, 2010.

[3] V. Cherkassky and F. Mulier, *Learning from Data Concepts, Theory, and Methods (Second Edition)*, John Wiley & Sons, Inc., 2007.

[4] C. Cortes and V. Vapnik, Support-vector networks, *Mach. Learn.*, **20**, *3* (1995), 273-297.

[5] R.O. Duda, P.E. Hart and D.G. Stork, *Pattern Classification*, 2nd Edition,New York: John Wiley & Sons, 2001.

[6] A.J. Izenman, *Modern Multivariate Statistical Techniques: Regression, Classification and Manifold Learning*, Springer, 2008.

[7] S. Marsland, *Machine Learning: An Algorithmic Perspective*, CRC Press, Taylor & Francis Group, Boca Raton - London - New York, 2009.

[8] K.E. Muller and P.W. Stewart, *Linear Model Theory - Univariate, Multivariate and Mixed Models*, Wiley-Interscience, 2006.

[9] I. Paraschiv-Munteanu, Model-free approaches in learning the multivariate linear regressive models, *An. Univ. Craiova Ser. Mat. Inform.*, **38**, *2* (2008), 87-94.

[10] I. Paraschiv-Munteanu, Multivariate linear systems for learning from data, *Ann. Univ. Buchar. Math. Ser.*, **2 (LX)**, *2* (2011), 179-206.

[11] I. Paraschiv-Munteanu and L. State, Learning from Data using Multivariate Linear Models, *Coping with Complexity, COPCOM 2011*, Editura Casa Cărţii de Ştiinţă, D. Dumitrescu and others (eds.), pp. 20-31, 2011.

[12] L. State and I. Paraschiv-Munteanu, *Introducere in teoria statistica a recunoasterii formelor*, Editura Universitatii din Pitesti, Romania, 2009.

[13] L. State and I. Paraschiv-Munteanu, A Probabilistic Model-free Approach in Learning Multivariate Noisy Linear Systems, *Proceedings 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing SYNASC-2011*, D. Wang and others (eds.), IEEE Computer Society Conference Publishing Services, pp. 239-246.

[14] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 4th Edition, Elsevier Inc., 2008.

[15] V.N. Vapnik, *Statistical Learning Theory*, New York, Wiley, 1998.

Iuliana Paraschiv-Munteanu
University of Bucharest, Faculty of Mathematics and Computer Science
14 Academiei St., Bucharest 010014, Romania
E-mail: pmiulia@fmi.unibuc.ro

Luminiţa State
University of Piteşti, Faculty of Mathematics and Computer Science
1 Târgu din Vale St., Piteşti 110040, Romania
E-mail: lstate@clicknet.ro