Numerical analysis of least squares and perceptron learning for classification problems

N.B. When citing this work, cite the original published paper.

(article starts on next page)

Numerical analysis of least squares and perceptron learning for classification problems

Larisa Beilina

Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, SE-42196 Gothenburg, Sweden.; larisa@chalmers.se

Abstract: This work presents study on regularized and non-regularized versions of perceptron learning and least squares algorithms for classification problems. The Fréchet derivatives for least squares and perceptron algorithms are derived. Different Tikhonov's regularization techniques for choosing the regularization parameter are discussed. Numerical experiments demonstrate performance of perceptron and least squares algorithms to classify simulated and experimental data sets.

Keywords: Classification problem, linear classifiers, least squares algorithm, perceptron learning algorithm, Tikhonov's regularization.

MSC: 65F22, 68T05, 90C47.

1. Introduction

Machine learning is a field of artificial intelligence which gives computer systems the ability to "learn" using available data. Recently machine learning algorithms become very popular for analyzing of data and make prediction [1–4]. Linear models for classification is a part of supervised learning [1]. Supervised learning is machine learning task of learning a function which transforms an input to an output data using available input-output data. In supervised learning, every example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used then for analyzing of new examples. Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, decision trees and support vector machines. In this work we will concentrate attention on study the linear and logistic regression algorithms. Supervised learning problems are further divided into Regression and Classification problems. Both problems have as goal the construction of a model which can predict the value of the dependent attribute from the attribute variables. The difference between these two problems is the fact that the attribute is numerical for regression and logical (belonging to class or not) for classification.

In this work are studied linear and polynomial classifiers, more precisely, the regularized versions of least squares and perceptron learning algorithms. The WINNOW algorithm for classification is also presented since it is used in numerical examples of Section 6 for comparison of different classification strategies. The classification problem is formulated as a regularized minimization problem for finding optimal weights in the model function. To formulate iterative gradient-based classification algorithms the Fréchet derivatives for the non-regularized and regularized least squares algorithms are presented. The Fréchet derivative for the perceptron algorithm is also rigorously derived.

Adding the regularization term in the functional leads to the optimal choice of weights such that they make a trade-off between observed data and obtaining a minimum of this functional. Different rules are used for choosing the regularization parameter in machine learning, and most popular are early stopping algorithm, bagging and dropout techniques [5], genetic algorithms [6], particle swarm optimization [7,8], racing algorithms [9] and Bayesian optimization techniques [10,11]. In this work are presented the most popular a priori and a posteriori Tikhonov's regularization rules for choosing the regularization parameter in the cost functional. Finally, performance of non-regularized versions of all classification algorithms with respect to applicability, reliability and efficiency is analyzed on simulated and experimental data sets [12,13].

The outline of the paper is as follows. In Section 2 are briefly formulated non-regularized and regularized classification problems. Least squares for classification are discussed in Section 3. Machine learning linear and polynomial classifiers are presented in Section 4. Tikhonov's methods of regularization for classification problems are discussed in Section 5. Finally, numerical tests are presented in Section 6.

2. Classification problem

The goal of regression is to predict the value of one or more continuous target variables $t = \{t_i\}, i = 1, ..., m$ by knowing the values of input vector $x = \{x_i\}, i = 1, ..., m$. Usually, classification algorithms are working well for linearly separable data sets.

Definition 1. Let A and B are two data sets of points in an n-dimensional Euclidean space. Then A and B are linearly separable if there exist $n + 1$ real numbers $\omega_1, ..., \omega_n, l$ such that every point $x \in A$ satisfies $\sum_{i=1}^{n} \omega_i x_i > l$ and every point $x \in B$ satisfies $\sum_{i=1}^{n} \omega_i x_i < -l$.

The classification problem is formulated as follows:

- Suppose that we have data points $\{x_i\}, i = 1, ..., m$ which are separated into two classes A and B. Assume that these classes are linearly separable.
- Our goal is to find the decision line which will separate these two classes. This line will also predict in which class will the new point fall.

In the non-regularized classification problem the goal is to find optimal weights $\omega = (\omega_1, ..., \omega_M)$, M is the number of weights, in the functional

$$F(\omega) = \frac{1}{2}\|t - y(\omega)\|^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i(\omega))^2 \tag{1}$$

with m data points. Here, $t = \{t_i\}, i = 1, ..., m$, is the target function with known values, $y(\omega) = \{y_i(\omega)\} := \{y(x_i, \omega)\}, i = 1, ..., m$, is the classifiers model function.

Algorithm 1 Gradient Algorithm for classification.

- Step 1. Initialization:

  - Assume that every training example $\mathbf{x} = (x_1, ..., x_m)$ is described by m attributes with values $x_i = 0$ or $x_i = 1$.
  - Label examples of the first class with $t(\mathbf{x}) = 1$ and examples of the second class $t(\mathbf{x}) = 0$. Assume that all examples where $t(\mathbf{x}) = 1$ are linearly separable from examples where $t(\mathbf{x}) = 0$.
  - Denote by $y(\mathbf{x}, \omega)$ the classifier's model function.
  - Initialize weights $\omega^0 = \{\omega_i^0\}, i = 1, ..., M$ to small random numbers. Compute the sequence of $\omega_i{}^N$ for all $N > 0$ in the following steps.
- Step 2. Compute gradient

$$G_i^k = -(t - y(\omega_i^k)) \cdot y'_{\omega_i}(\omega_i^k) + \gamma\omega_i^k, i = 1, ..., M. \tag{2}$$

- Step 3. Update the unknown parameter $\omega := \omega^{k+1}$ using (2) as

$$\omega_i^{k+1} = \omega_i^k + \eta G_i^k, \tag{3}$$

  where η is the learning rate or step size in the gradient update which is usually taken as $\eta = 0.5$ [4].
- Step 4. For the tolerance $0 < \theta < 1$ chosen by the user, stop computing the functions ω_i^k and set $\omega_i^N = \omega_i^k$ if either $\|G_i(\omega_i^k)\|_{L_2} \le \theta$, or norms $\|G_i(\omega_i^k)\|_{L_2}$ abruptly grow, or computed $\|\omega_i^k\|_{L_2}$ are stabilized. Otherwise, set $k := k + 1$ and go to Step 2.

To find optimal vector of weights $\omega = \{\omega_i\}, i = 1, ..., M$ in the regularized classification problem, the regularization term is added to the functional (1):

$$F(\omega) = \frac{1}{2}\|t - y(\omega)\|^2 + \frac{1}{2}\gamma\|\omega\|^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i(\omega))^2 + \frac{1}{2}\gamma\sum_{j=1}^{M}|\omega_j|^2. \tag{4}$$

Here, γ is the regularization parameter, $\|\omega\|^2 = \omega^T\omega = \omega_1^2 + ... + \omega_M^2$, M is the number of weights. In order to find the optimal weights in (1) or in (4), the following minimization problem should be solved

$$\min_{\omega} F(\omega). \tag{5}$$

Thus, we seek for a stationary point of (1) or (4) with respect to ω such that

$$F'(\omega)(\bar{\omega}) = 0, \tag{6}$$

where $F'(\omega)$ is the Fréchet derivative acting on $\bar{\omega}$.

More precisely, for the functional (4) we get

$$F'(\omega)(\bar{\omega}) = \sum_{i=1}^{M} F'_{\omega_i}(\omega)(\bar{\omega}_i),$$

$$\frac{\partial F}{\partial \omega_i}(\omega)(\bar{\omega}_i) := F'_{\omega_i}(\omega)(\bar{\omega}_i) = -(t - y) \cdot y'_{\omega_i}(\bar{\omega}_i) + \gamma\omega_i(\bar{\omega}_i), \; i = 1, ..., M. \tag{7}$$

The Fréchet derivative of the functional (1) is obtained by taking $\gamma = 0$ in (7). To find optimal vector of weights $\omega = \{\omega_i\}, i = 1, ..., M$ can be used the **Algorithm 1** as well as least squares or machine learning algorithms.

For computation of the learning rate η in the **Algorithm 1** usually is used optimal rule which can be derived similarly as in [14]. However, as a rule take $\eta = 0.5$ in machine learning classification algorithms [4]. Among all other regularization methods applied in machine learning [5–11], the regularization parameter γ can be also computed using the Tikhonov's theory for inverse and ill-posed problems by different algorithms presented in [15–19]. Some of these algorithms are discussed in Section 5.

3. Least squares for classification

The linear regression is similar to the solution of linear least squares problem and can be used for classification problems appearing in machine learning algorithms. We will revise solution of linear least squares problem in terms of linear regression.

The simplest linear model for regression is

$$f(x, \omega) = \omega_0 \cdot 1 + \omega_1 x_1 + ... + \omega_M x_M. \tag{8}$$

Here, $\omega = \{\omega_i\}, i = 0, ..., M$ are weights with bias parameter ω_0, $\{x_i\}, i = 1, ..., M$ are training examples. Target values (known data) are $\{t_i\}, i = 1, ..., N$ which correspond to $\{x_i\}, i = 1, ..., M$. Here, M is the number of weights and N is the number of data points. The goal is to predict the value of t in (1) for a new value of x in the model function (8).

The linear model (8) can be written in the form

$$f(x, \omega) = \omega_0 \cdot 1 + \sum_{i=1}^{M} \omega_i \varphi_i(x) = \omega^T \varphi(x), \tag{9}$$

where $\varphi_i(x), i = 0, ..., M$ are known basis functions with $\varphi_0(x) = 1$.

3.1. Non-regularized least squares problem

In non-regularized linear regression or least squares problem the goal is to minimize the sum of squares

$$E(\omega) = \frac{1}{2} \sum_{n=1}^{N} (t_n - f(x, \omega))^2 = \frac{1}{2} \sum_{n=1}^{N} (t_n - \omega^T \varphi(x_n))^2 := \frac{1}{2} \|t - \omega^T \varphi(x)\|_2^2 \tag{10}$$

to find optimal weights $\omega_i, i = 0, ..., M$, in the minimization problem

$$\min_{\omega} E(\omega) = \min_{\omega} \frac{1}{2} \|t - \omega^T \varphi(x)\|_2^2, \tag{11}$$

for points $x_n, n = 1, ..., N$. The problem (11) is a typical least squares problem of the minimizing the squared residuals

$$\min_{\omega} \frac{1}{2} \|r(\omega)\|_2^2 = \min_{\omega} \frac{1}{2} \|t - \omega^T \varphi(x)\|_2^2 \tag{12}$$

with the residual $r(\omega) = t - \omega^T \varphi(x)$. The test functions $\varphi(x)$ form the design matrix A

$$A = \begin{bmatrix} 1 & \varphi_1(x_1) & \varphi_2(x_1) & \dots & \varphi_M(x_1) \\ 1 & \varphi_1(x_2) & \varphi_2(x_2) & \dots & \varphi_M(x_2) \\ 1 & \varphi_1(x_3) & \varphi_2(x_3) & \dots & \varphi_M(x_3) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & \varphi_1(x_N) & \varphi_2(x_N) & \dots & \varphi_M(x_N) \end{bmatrix}, \tag{13}$$

and the regression problem (or the least squares problem) is written as

$$\min_{\omega} \frac{1}{2} \|r(\omega)\|_2^2 = \min_{\omega} \frac{1}{2} \|A\omega - t\|_2^2, \tag{14}$$

where A is of the size $N \times M$ with $N > M$, t is the target vector of the size N, and ω is vector of weights of the size M.

To find minimum of the error function (10) and derive the *normal equations*, we look for the ω where the gradient of the norm $\|r(\omega)\|_2^2 = \|A\omega - t\|_2^2 = (A\omega - t)^T(A\omega - t)$ vanishes, or where $(\|r(\omega)\|_2^2)'_\omega = 0$. To derive the Fréchet derivative, we consider the difference $\|r(\omega + e)\|_2^2 - \|r(\omega)\|_2^2$ and single out the linear part with respect to ω. More precisely, we get

$$\begin{aligned} 0 =& \lim_{\|e\|0} \frac{(A(\omega + e) - t)^T(A(\omega + e) - t) - (A\omega - t)^T(A\omega - t)}{\|e\|_2} \\ =& \lim_{\|e\|0} \frac{((A\omega - t) + Ae)^T((A\omega - t) + Ae) - (A\omega - t)^T(A\omega - t)}{\|e\|_2} \\ =& \lim_{\|e\|0} \frac{\|(A\omega - t) + Ae\|_2^2 - \|A\omega - t\|_2^2}{\|e\|_2} \\ =& \lim_{\|e\|0} \frac{\|A\omega - t\|_2^2 + 2(A\omega - t) \cdot Ae + \|Ae\|_2^2 - \|A\omega - t\|_2^2}{\|e\|_2} \\ =& \lim_{\|e\|0} \frac{2e^T(A^T A\omega - A^T t) + e^T A^T A e}{\|e\|_2} \end{aligned}$$

Thus,

$$0 = \lim_{\|e\|0} \frac{2e^T(A^T A\omega - A^T t) + e^T A^T A e}{\|e\|_2}. \tag{15}$$

The second term in (15) can be estimated as

$$\lim_{\|e\|0} \frac{|e^T A^T A e|}{||e||_2} \leq \lim_{\|e\|0} \frac{||A||_2^2 ||e||_2^2}{||e||_2} = \lim_{\|e\|0} ||A||_2^2 ||e||_2 \to 0 \tag{16}$$

Thus, the first term in (15) must also be zero, and thus,

$$A^T A \omega = A^T t \tag{17}$$

Equations (17) is a symmetric linear system of the $M \times M$ linear equations for M unknowns called normal equations.

Using definition of the residual in the functional

$$\frac{1}{2} \|r(\omega)\|_2^2 = \frac{1}{2} \|A\omega - t\|_2^2 \tag{18}$$

can be computed the Hessian matrix $H = A^T A$. If the Hessian matrix $H = A^T A$ is positive definite, then ω is indeed a minimum.

Lemma 1. *The matrix $A^T A$ is positive definite if and only if the columns of A are linearly independent, or when $rank(A) = M$ (full rank).*

Proof. . We have that $dim(A) = N \times M$, and thus, $dim(A^T A) = M \times M$. Thus, $\forall v \in R^M$ such that $v \neq 0$

$$v^T A^T A v = (Av)^T (Av) = \|Av\|_2^2 \geq 0. \tag{19}$$

For positive definite matrix $A^T A$ we need to show that $v^T A^T A v > 0$. Assume that $v^T A^T A v = 0$. We observe that $Av = 0$ only if the linear combination $\sum_{i=1}^{M} a_{ji} v_i = 0$. Here, a_{ji} are elements of row j in A. This will be true only if columns of A are linearly dependent or when $v = 0$, but this is contradiction with assumption $v^T A^T A v = 0$ since $v \neq 0$ and thus, the columns of A are linearly independent and $v^T A^T A v > 0$. $\square$

The final conclusion is that if the matrix A has a full rank ($rank(A) = M$) then the system (17) is of the size M-by-M and is symmetric positive definite system of normal equations. It has the same solution ω as the least squares problem $\min_\omega \|A\omega - t\|_2^2$ and can be solved efficiently via Cholesky decomposition [20].

3.2. Regularized linear regression

Let now the matrix A will have entries $a_{ij} = \phi_j(x_i), i = 1, ..., N; j = 1, ..., M$. Recall, that functions $\phi_j(x), j = 0, ..., M$ are called basis functions which should be chosen and are known. Then the regularized least squares problem takes the form

$$\min_\omega \frac{1}{2} \|r(\omega)\|_2^2 + \frac{\gamma}{2} \|\omega\|_2^2 = \min_\omega \frac{1}{2} \|A\omega - t\|_2^2 + \frac{\gamma}{2} \|\omega\|_2^2. \tag{20}$$

To minimize the regularized squared residuals (20) we will again derive the *normal equations*. Similarly as was derived the Fréchet derivative for the non-regularized regression problem (14), we look for the ω where the gradient of $\frac{1}{2} \|A\omega - t\|_2^2 + \frac{\gamma}{2} \|\omega\|_2^2 = \frac{1}{2} (A\omega - t)^T (A\omega - t) + \frac{\gamma}{2} \omega^T \omega$ vanishes. In other words, we consider the difference $(\|r(\omega + e)\|_2^2 + \frac{\gamma}{2} \|\omega + e\|_2^2) - (\|r(\omega)\|_2^2 + \frac{\gamma}{2} \|\omega\|_2^2)$, then single out the linear part with respect to ω to obtain:

$$\begin{aligned}
0 &= \frac{1}{2} \lim_{\|e\|0} \frac{(A(\omega + e) - t)^T (A(\omega + e) - t) - (A\omega - t)^T (A\omega - t)}{||e||_2} + \lim_{\|e\|0} \frac{\frac{\gamma}{2} (\omega + e)^T (\omega + e) - \frac{\gamma}{2} \omega^T \omega}{||e||_2} \\
&= \frac{1}{2} \lim_{\|e\|0} \frac{\|(A\omega - t) + Ae\|_2^2 - \|A\omega - t\|_2^2}{||e||_2} + \lim_{\|e\|0} \frac{\frac{\gamma}{2} (\|\omega + e\|_2^2 - \|\omega\|_2^2)}{||e||_2} \\
&= \frac{1}{2} \lim_{\|e\|0} \frac{\|A\omega - t\|_2^2 + 2(A\omega - t) \cdot Ae + \|Ae\|_2^2 - \|A\omega - t\|_2^2}{||e||_2} + \frac{\gamma}{2} \lim_{\|e\|0} \frac{\|\omega\|_2^2 + 2e^T \omega + \|e\|_2^2 - \|\omega\|_2^2}{||e||_2}
\end{aligned}$$

$$= \frac{1}{2}\lim_{\|e\|0}\frac{2e^T(A^TA\omega - A^Tt) + e^TA^TAe}{||e||_2} + \frac{\gamma}{2}\lim_{\|e\|0}\frac{2e^T\omega + e^Te}{||e||_2}.$$

The term

$$\lim_{\|e\|0}\frac{|e^TA^TAe|}{||e||_2} \leq \lim_{\|e\|0}\frac{||A||_2^2||e||_2^2}{||e||_2} = \lim_{\|e\|0}||A||_2^2||e||_2 \to 0. \tag{21}$$

Similarly, the term

$$\lim_{\|e\|0}\frac{|e^Te|}{||e||_2} = \lim_{\|e\|0}\frac{||e||_2^2}{||e||_2} \to 0. \tag{22}$$

We finally get

$$0 = \lim_{\|e\|0}\frac{e^T(A^TA\omega - A^Tt)}{||e||_2} + \frac{\gamma e^T\omega}{||e||_2}.$$

The expression above means that the factor $A^TA\omega - A^Tt + \gamma\omega$ must also be zero, or $(A^TA + \gamma I)\omega = A^Tt$, where I is the identity matrix. This is a system of M linear equations for M unknowns, the normal equations for regularized least squares.
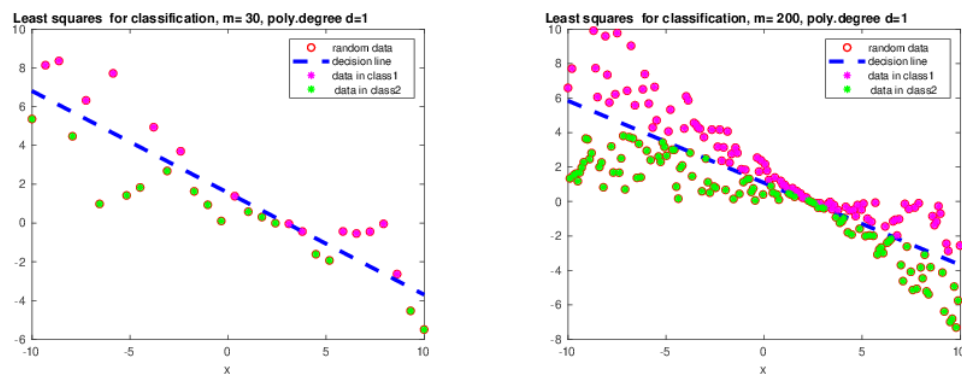

Figure 1. Least squares for classification.

Figure 1 shows that the linear regression or least squares minimization $\min_\omega \|A\omega - t\|_2^2$ for classification is working fine when it is known that two classes are linearly separable. Here the linear model equation in the problem (12) is

$$f(x,y,\omega) = \omega_0 + \omega_1 x + \omega_2 y \tag{23}$$

and the target values of the vector $t = \{t_i\}, i = 1, ..., N$ in (12) are

$$t_i = \begin{cases} 1 & \text{red points,} \\ 0 & \text{green points.} \end{cases} \tag{24}$$

The elements of the design matrix (13) are given by

$$A = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \\ \vdots & \vdots & \ddots \\ 1 & x_N & y_N \end{bmatrix}. \tag{25}$$

3.3. Polynomial fitting to data in two-class model

Let us consider the least squares classification in the two-class model in the general case. Let the first class consisting of l points with coordinates $(x_i, y_i), i = 1, ..., l$ is described by it's linear model

$$f_1(x, c) = c_{1,1}\phi_1(x) + c_{2,1}\phi_2(x) + ... + c_{n,1}\phi_n(x). \tag{26}$$

Let the second class consisting of k points with coordinates $(x_i, y_i), i = 1, ..., k$ is also described by the same linear model

$$f_2(x, c) = c_{1,2}\phi_1(x) + c_{2,2}\phi_2(x) + ... + c_{n,2}\phi_n(x). \tag{27}$$

Here, basis functions are $\phi_j(x), j = 1, ..., n$. Our goal is to find the vector of parameters $c = c_{i,1} = c_{i,2}, i = 1, ..., n$ of the size n which will fit best to the data $y_i, i = 1, ..., m, m = k + l$ of both model functions, $f_1(x_i, c), i = 1, ..., l$ and $f_2(x_i, c), i = 1, ..., k$ with $f(x, c) = [f_1(x_i, c), f_2(x_i, c)]$ such that the minimization problem

$$\min_c \|y - f(x, c)\|_2^2 = \min_c \sum_{i=1}^m (y_i - f(x_i, c))^2 \tag{28}$$

is solved with $m = k + l$. If the function $f(x, c)$ in (28) is linear then we can reformulate the minimization problem (28) as the following least squares problem

$$\min_c \|Ac - y\|_2^2, \tag{29}$$

where the matrix A in the linear system

$$Ac = y$$

will have entries $a_{ij} = \phi_j(x_i), i = 1, ..., m; j = 1, ..., n$, i.e. elements of the matrix A are created by basis functions $\phi_j(x), j = 1, ..., n$. Solution of (29) can be found by the method of normal equations derived in Section 3.1:

$$c = (A^T A)^{-1} A^T b = A^+ b \tag{30}$$

with pseudo-inverse matrix $A^+ := (A^T A)^{-1} A^T$.

For creating of elements of A different basis functions can be chosen. The polynomial test functions

$$\phi_j(x) = x^{j-1}, \ \ j = 1, ..., n \tag{31}$$

have been considered in the problem of fitting to a polynomial in examples presented in Figures 2. The matrix A constructed by these basis functions is a Vandermonde matrix, and problems related to this matrix are discussed in [20]. Linear splines (or hat functions) and bellsplines also can be used as basis functions [20].

Figures 2 present examples of polynomial fitting to data for two-class model with $m = 10$ using basis functions $\phi_j(x) = x^{j-1}, j = 1, ..., d$, where d is degree of the polynomial. Using these figures we observe that least squares fit data well and even can separate points in two different classes, although this is not always the case. Higher degree of polynomial separates two classes better. However, since Vandermonde's matrix can be ill-conditioned for high degrees of polynomial, we should carefully choose appropriate polynomial to fit data.

4. Machine learning linear and polynomial classifiers

In this section we will present the basic machine learning algorithms for classification problems: perceptron learning and WINNOW algorithms. Let us start with considering of an example: determine the decision line for points presented in Figure 3. One example on this figure is labeled as positive class, another one as negative. In this case, two classes are separated by the linear equation with three weights $\omega_i, i = 1, 2, 3$, given by

$$\omega_1 + \omega_2 x + \omega_3 y = 0. \tag{33}$$

In common case, two classes can be separated by the general equation

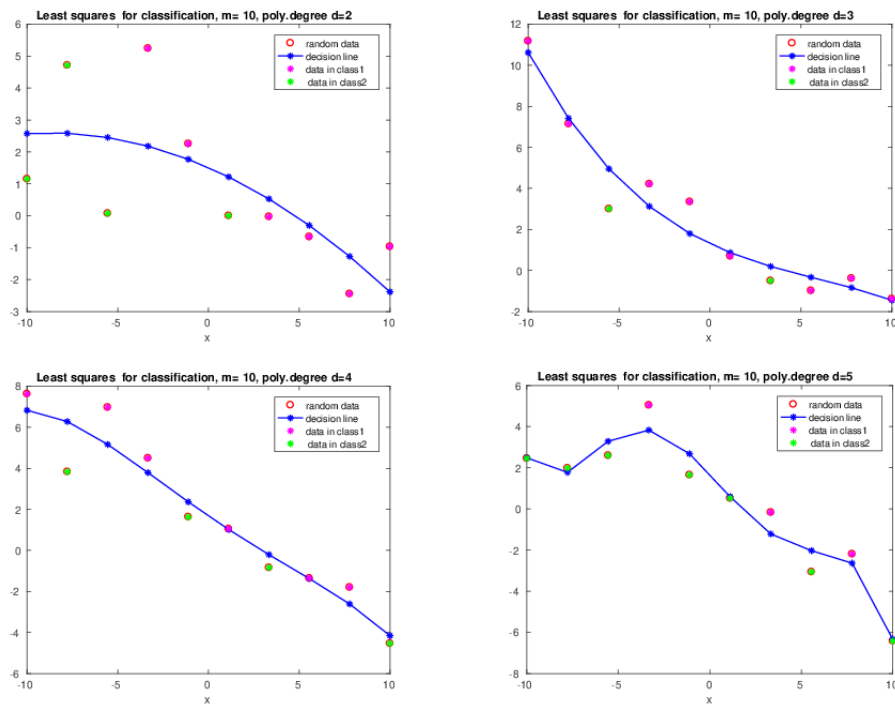$$\omega_0 + \omega_1 x_1 + \omega_2 x_2 + ... + \omega_n x_n = 0 \tag{34}$$

Figure 2. Least squares in polynomial fitting to data for different polynomials defined by (31).

Algorithm 2 Regularized perceptron learning for classification.

- Step 1. Initialization:
    - Assume that every training example $\mathbf{x} = (x_1, ..., x_n)$ is described by n attributes.
    - Label examples of the first class with $c(\mathbf{x}) = 1$ and examples of the second class as $c(\mathbf{x}) = 0$. Assume that all examples of the first class where $c(\mathbf{x}) = 1$ are linearly separable from examples of the second class where $c(\mathbf{x}) = 0$.
    - Let us denote by $h(\mathbf{x})$ the classifier's hypothesis which will have binary values $h(\mathbf{x}) = 1$ or $h(\mathbf{x}) = 0$. Initialize $h(\mathbf{x}) = 0$ for examples $c(\mathbf{x}) = 1$ and $h(\mathbf{x}) = 1$ for examples $c(\mathbf{x}) = 0$.
    - Initialize weights $\omega^0 = \{\omega_i^0\}, i = 1, ..., M$ to small random numbers. Compute the sequence of $\omega_i{}^m$ for all $m > 0$ in the following steps.
- Step 2. If $\sum_{i=0}^n \omega_i^m x_i > 0$ we will say that the example belongs to the first class and $h(\mathbf{x}) = 1$.
- Step 3. If $\sum_{i=0}^n \omega_i^m x_i < 0$ we will say that the example belongs to the second class and $h(\mathbf{x}) = 0$.
- Step 4. Update weights $\omega := \omega^{m+1} = \{\omega_i^{m+1}\}, i = 1, ..., M$ using

$$\omega_i^{m+1} = \omega_i^m + \eta \cdot ([c(\mathbf{x}) - h(\mathbf{x})] \cdot x_i + \gamma \cdot \omega_i^m), \qquad (32)$$

  where η is the learning rate usually taken as $\eta = 0.5$ [4].
- Step 5. If $c(\mathbf{x}) = h(\mathbf{x})$ for all learning examples - stop. Otherwise set $m := m + 1$ and return to step 2.

---

which also can be written as

$$\omega^T x = \sum_{i=0}^n \omega_i x_i = 0 \qquad (35)$$

with $x_0 = 1$. If $n = 2$ then the above equation defines a line, if $n = 3$ - plane, if $n > 3$ - hyperplane. The problem is to determine weights ω_i and the task of machine learning is to determine their appropriate values. Weights $\omega_i, i = 1, ..., n$ determine the angle of the hyperplane, ω_0 is called bias and determines the offset, or the hyperplanes distance from the origin of the system of coordinates.
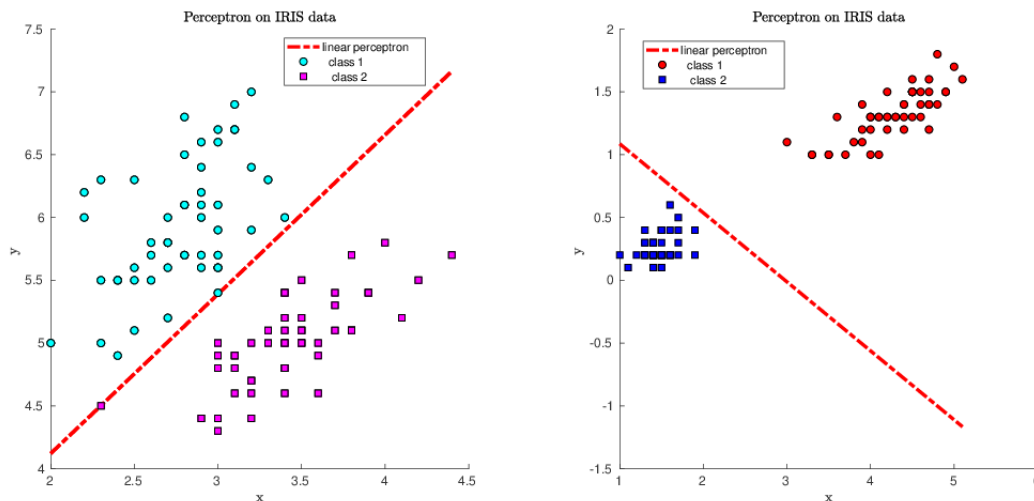
Figure 3. Decision lines computed by the perceptron learning algorithm for separation of two classes using Iris dataset [12].

4.1. Perceptron learning for classification

The main idea of perceptron is binary classification. The perceptron computes a sum of weighted inputs

$$h(x, \omega) = \omega^T x = \sum_{i=0}^{n} \omega_i x_i \qquad (36)$$

and uses the binary classification

$$y(x, \omega) = sign(h(x, \omega)). \qquad (37)$$

When weights are computed, the linear classification boundary is defined by

$$h(x, \omega) = \omega^T x = 0.$$

Thus, the perceptron algorithm determines weights in (36) via binary classification (37).

Binary classifier $y(x, \omega)$ in (37) decides whether or not an input x belongs to some specific class:

$$y(x, \omega) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} \omega_i x_i + \omega_0 > 0, \\ 0, & \text{otherwise,} \end{cases} \qquad (38)$$

where ω_0 is the bias. The bias does not depend on the input value x and shifts the decision boundary. If the learning sets are not linearly separated the perceptron learning algorithm does not terminate and will never converge and classify data properly, see Figure 7-a).

The algorithm which determines weights in (36) via binary classification (37) can be reasoned by minimization of the regularized residual

$$F(\omega) = \|r(x, \omega)\|_2^2 = \|(t - y(x, \omega))\xi_\delta(x)\|_2^2 + \frac{1}{2}\gamma\|w\|_2^2, \qquad (39)$$

where $\xi_\delta(x)$ for a small δ is a data compatibility function to avoid discontinuities which can be defined similarly with [21] and γ is the regularization parameter. Taking $\gamma = 0$ algorithm will minimize the non-regularized residual (39). Alternative, it can be minimized the residual

$$r(x, \omega) = -t^T y(x, \omega) = -\sum_{i \in M} t_i y_i = -\sum_{i \in M} t_i \, sign(h(x_i, \omega)) \qquad (40)$$

over the set $M \subset \{1, ..., m\}$ of the currently miss-classified patterns which is simplified for the case of the perceptron algorithm to the minimization of the following residual

$$r(x, \omega) = -t^T h(x, \omega) = -\sum_{i \in M} t_i h_i = -\sum_{i \in M} t_i \, sign(h(x_i, \omega)) = \sum_{i=1}^{m} |-t_i h_i|_+ \tag{41}$$

with

$$|-t_i h_i|_+ = \max(0, -t_i h_i), i = 1, ..., m.$$

The **Algorithm 2** presents the regularized perceptron learning algorithm where in update of weights (32) was used the following regularized functional

$$F(\omega) = \frac{1}{2}\|(t - y(x, \omega))\xi_\delta(x)\|_2^2 + \frac{1}{2}\gamma\|w\|_2^2 = \frac{1}{2}\sum_{i=1}^{m}((t_i - y_i(x, \omega))\xi_\delta(x))^2 + \frac{1}{2}\gamma\sum_{i=1}^{n}w_i^2. \tag{42}$$

Here, γ is the regularization parameter, t is the target function, or class c in the algorithm 2, which takes values 0 or 1.

To find optimal weights in (42) we need to solve the minimization problem in the form (5)

$$F'(\omega)(\bar{\omega}) = 0, \tag{43}$$

where $F'(\omega)$ is a Frechet derivative acting on $\bar{\omega}$. To derive $F'(\omega)$ for (39) we seek the ω where the gradient of $\frac{1}{2}\|r(x, \omega)\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2$ vanishes. In other words, we consider for (39) the difference $(\|r(x, \omega + e)\|_2^2 + \frac{\gamma}{2}\|\omega + e\|_2^2) - (\|r(x, \omega)\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2)$, then single out the linear part with respect to ω to obtain:

$$
\begin{aligned}
0 =& \frac{1}{2}\lim_{\|e\|0} \frac{\|(t - y(x, \omega + e))\xi_\delta(x)\|_2^2 + \frac{\gamma}{2}\|\omega + e\|_2^2 - \|(t - y(x, \omega))\xi_\delta(x)\|_2^2 - \frac{\gamma}{2}\|\omega\|_2^2}{\|e\|_2} \\
=& \frac{1}{2}\lim_{\|e\|0} \frac{\|(t - \sum_{i=0}^{n}\omega_i x_i - \sum_{i=0}^{n}e_i x_i)\xi_\delta(x)\|_2^2 - \|(t - y(x, \omega))\xi_\delta(x)\|_2^2}{\|e\|_2} + \lim_{\|e\|0} \frac{\frac{\gamma}{2}(\omega + e)^T(\omega + e) - \frac{\gamma}{2}\omega^T\omega}{\|e\|_2} \\
=& \frac{1}{2}\lim_{\|e\|0} \frac{\|(t - y(x, \omega) - e^T x)\xi_\delta(x)\|_2^2 - \|(t - y(x, \omega))\xi_\delta(x)\|_2^2}{\|e\|_2} + \lim_{\|e\|0} \frac{\frac{\gamma}{2}(\omega + e)^T(\omega + e) - \frac{\gamma}{2}\omega^T\omega}{\|e\|_2} \\
=& \frac{1}{2}\lim_{\|e\|0} \frac{\|(t - y(x, \omega))\,\xi_\delta(x)\|_2^2 - 2(t - y(x, \omega)) \cdot e^T x\,\xi_\delta(x) + \|e^T x\,\xi_\delta(x)\|_2^2}{\|e\|_2} \\
& -\frac{1}{2}\lim_{\|e\|0} \frac{\|(t - y(x, \omega))\,\xi_\delta(x)\|_2^2}{\|e\|_2} + \lim_{\|e\|0} \frac{\frac{\gamma}{2}(\omega + e)^T(\omega + e) - \frac{\gamma}{2}\omega^T\omega}{\|e\|_2} \\
=& \frac{1}{2}\lim_{\|e\|0} \frac{-2(t - y(x, \omega)) \cdot e^T x\,\xi_\delta(x) + \|e^T x\,\xi_\delta(x)\|_2^2}{\|e\|_2} + \frac{\gamma}{2}\lim_{\|e\|0} \frac{2e^T\omega + e^T e}{\|e\|_2}.
\end{aligned}
$$

The second part in the last term of the above expression is estimated as in (22). The second part in the first term is estimated as

$$
\begin{aligned}
\lim_{\|e\|0} \frac{|(e^T x\,\xi_\delta(x))^T\, e^T x\,\xi_\delta(x)|}{\|e\|_2} &= \lim_{\|e\|0} \frac{|(x^T e\,\xi_\delta(x))^T\, x^T e\,\xi_\delta(x)|}{\|e\|_2} \\
&\leq \lim_{\|e\|0} \frac{\|x\,\xi_\delta(x)\|_2^2\|e\|_2^2}{\|e\|_2} = \lim_{\|e\|0}\|x\,\xi_\delta(x)\|_2^2\|e\|_2 \to 0.
\end{aligned}
\tag{44}
$$

We finally get

$$0 = \lim_{\|e\|0} -\frac{x^T e(t - y(x, \omega))\,\xi_\delta(x)}{\|e\|_2} + \frac{\gamma e^T\omega}{\|e\|_2}.$$

Open J. Discret. Appl. Math. **2020**, *3(2)*, 30-49

40

The expression above means that the factor $-x^T(t - y(x, \omega))\, \xi_\delta(x) + \gamma\omega$ must also be zero, or

$$
F'(\omega)(\bar{\omega}) = \sum_{i=1}^{n} F'_{\omega_i}(\omega)(\bar{\omega}_i),
$$

$$
F'_{\omega_i}(\omega)(\bar{\omega}_i) = -(t - y) \cdot \xi_\delta(x) \cdot y'_{\omega_i}(\bar{\omega}_i) + \gamma\omega_i = -(t - y) \cdot x_i \cdot \xi_\delta(x_i) + \gamma\omega_i,\ i = 1, ..., n.
$$

$$(45)$$

The non-regularized version of the perceptron **Algorithm 2** is obtained taking $\gamma = 0$ in (45).

4.2. Polynomial of the second order

Coefficients of polynomials of the second order can be obtained by the same technique as coefficients for linear classifiers. The second order polynomial function is:

$$
\omega_0 + \omega_1 \underbrace{x_1}_{z_1} + \omega_2 \underbrace{x_2}_{z_2} + \omega_3 \underbrace{x_1^2}_{z_3} + \omega_4 \underbrace{x_1 x_2}_{z_4} + \omega_5 \underbrace{x_2^2}_{z_5} = 0. \tag{46}
$$

This polynomial can be converted to the linear classifier if we introduce notations:

$$
z_1 = x_1, z_2 = x_2, z_3 = x_1^2, z_4 = x_1 x_2, z_5 = x_2^2.
$$

Then equation (46) can be written in new variables as

$$
\omega_0 + \omega_1 z_1 + \omega_2 z_2 + \omega_3 z_3 + \omega_4 z_4 + \omega_5 z_5 = 0 \tag{47}
$$

which is already the linear function. Thus, the Perceptron learning **Algorithm 2** can be used to determine weights $\omega_0, ..., \omega_5$ in (47).

Suppose that weights $\omega_0, ..., \omega_5$ in (47) are computed. To present the decision line one need to solve the following quadratic equation for x_2:

$$
\omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_1^2 + \omega_4 x_1 x_2 + \omega_5 x_2^2 = 0 \tag{48}
$$

with known weights $\omega_0, ..., \omega_5$ and known x_1 which can be rewritten as

$$
\underbrace{\omega_5}_{a} x_2^2 + x_2 \underbrace{(\omega_2 + \omega_4 x_1)}_{b} + \underbrace{\omega_0 + \omega_1 x_1 + \omega_3 x_1^2}_{c} = 0, \tag{49}
$$

or in the form

$$
a x_2^2 + b x_2 + c = 0 \tag{50}
$$

with known coefficients $a = \omega_5, b = \omega_2 + \omega_4 x_1, c = \omega_0 + \omega_1 x_1 + \omega_3 x_1^2$. Solutions of (50) will be

$$
x_2 = \frac{-b \pm \sqrt{D}}{2a},
$$

$$
D = b^2 - 4ac. \tag{51}
$$

Thus, to present the decision line for polynomial of the second order, first should be computed weights $\omega_0, ..., \omega_5$, and then the quadratic equation (49) should be solved the solutions of which are given by (51). Depending on the classification problem and set of admissible parameters for classes, one can then decide which one classification line should be presented, see examples in section 6.

4.3. WINNOW learning algorithm

To be able compare perceptron with other machine learning algorithms, we present here one more learning algorithm which is very close to the perceptron and called WINNOW. Here is described the simplest version of this algorithm without regularization. The regularized version of WINNOW is analyzed in [22]. Perceptron learning algorithm uses additive rule in the updating weights, while WINNOW algorithm uses multiplicative rule: weights are multiplied in this rule. The WINNOW algorithm **Algorithm 3** is written for
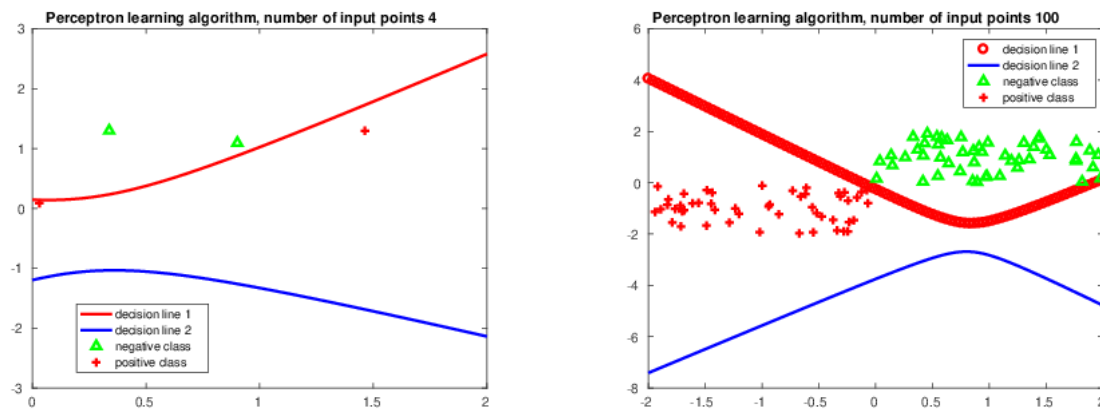
Figure 4. Perceptron learning algorithm for separation of two classes by polynomials of the second order.

Algorithm 3 WINNOW for classification

- Step 1. Initialization:

    - Assume that every training example $\mathbf{x} = (x_1, ..., x_n)$ is described by n attributes.
    - Label examples of the first class with $c(\mathbf{x}) = 1$ and examples of the second class as $c(\mathbf{x}) = 0$. Assume that all examples of the first class where $c(\mathbf{x}) = 1$ are linearly separable from examples of the second class where $c(\mathbf{x}) = 0$.
    - Initialize the classifier's hypothesis $h(\mathbf{x}) = 0$ for examples $c(\mathbf{x}) = 1$ and $h(\mathbf{x}) = 1$ for examples $c(\mathbf{x}) = 0$.
    - Choose parameter $\alpha > 1$, usually $\alpha = 2$.
    - Initialize weights $\omega^0 = \{\omega_i^0\}, i = 1, ..., M$ to small random numbers. Compute the sequence of ω_i^m for all $m > 0$ in the following steps.

- Step 2. If $\sum_{i=0}^{n} \omega_i^m x_i > 0$ we will say that the example is positive and $h(\mathbf{x}) = 1$.
- Step 3. If $\sum_{i=0}^{n} \omega_i^m x_i < 0$ we will say the the example is negative and $h(\mathbf{x}) = 0$.
- Step 4. Update weights $\omega := \omega^{m+1} = \{\omega_i^{m+1}\}$, $i = 1, ..., M$ using the formula

$$\omega_i^{m+1} = \omega_i^m \cdot \alpha^{(c(\mathbf{x}) - h(\mathbf{x})) \cdot x_i}.$$

- Step 5. If $c(\mathbf{x}) = h(\mathbf{x})$ for all learning examples - stop. Otherwise set $m := m + 1$ return to step 2.

$c = t$ and $y = h$ in (45). We will again assume that all examples where $c(\mathbf{x}) = 1$ are linearly separable from examples where $c(\mathbf{x}) = 0$.

5. Methods of Tikhonov's regularization for classification problems

To solve the regularized classification problem the regularization parameter γ can be chosen by the same methods which are used for the solution of ill-posed problems. For different Tikhonov's regularization strategies we refer to [15–19,23]. In this section we will present main methods of Tikhonov's regularization which follows ideas of [15–17,19,23].

Definition 2. Let B_1 and B_2 be two Banach spaces and $G \subset B_1$ be a set. Let $y : G \; B_2$ be one-to-one. Consider the equation

$$y(\omega) = t, \tag{52}$$

where t is the target function and $y(\omega)$ is the model function in the classification problem. Let t^* be the noiseless target function in equation (52) and ω^* be the ideal noiseless weights corresponding to t^*, $y(\omega^*) = t^*$. For every $\delta \in (0, \delta_0)$, $\delta_0 \in (0, 1)$ denote

$$K_\delta(t^*) = \left\{ z \in B_2 : \|z - t^*\|_{B_2} \leq \delta \right\}.$$

Let $\gamma > 0$ be a parameter and $R_\gamma : K_{\delta_0}(t^*) \to G$ be a continuous operator depending on the parameter γ. The operator R_γ is called the *regularization operator* for (52) if there exists a function $\gamma_0(\delta)$ defined for $\delta \in (0, \delta_0)$ such that

$$\lim_{\delta 0} \left\| R_{\gamma_0(\delta)}(t_\delta) - \omega^* \right\|_{B_1} = 0.$$

The parameter γ is called the regularization parameter and the procedure of constructing the approximate solution $\omega_{\gamma(\delta)} = R_{\gamma(\delta)}(t_\delta)$ is called the *regularization procedure*, or simply *regularization*.

One can use different regularization procedures for the same classification problem. The regularization parameter γ can be even the vector of regularization parameters depending on number of iterations in the used classification method, the tolerance chosen by the user, number of classes, etc..

For two Banach spaces B_1 and B_2 let Q be another space, $Q \subset B_1$ as a set and $\overline{Q} = B_1$. In addition, we assume that Q is compactly embedded in B_1. Let $G \subset B_1$ be the closure of an open set. Consider a continuous one-to-one function $y : G \to B_2$. Our goal is to solve

$$y(\omega) = t, \;\; \omega \in G. \tag{53}$$

Let

$$y(\omega^*) = t^*, \quad \|t - t^*\|_{B_2} < \delta. \tag{54}$$

To find an approximate solution of equation (53), we construct the Tikhonov regularization functional $J_\gamma(\omega)$,

$$J_\gamma(\omega) = \frac{1}{2} \|y(\omega) - t\|_{B_2}^2 + \frac{\gamma}{2} \|\omega\|_{B_1}^2 := \varphi(\omega) + \frac{\gamma}{2} \psi(\omega), \tag{55}$$

$$J_\gamma : G \to \mathbb{R},$$

where $\gamma = \gamma(\delta) > 0$ is a small regularization parameter.

The regularization term $\frac{\gamma}{2} \psi(\omega)$ encodes a priori available information about the unknown solution such that sparsity, smoothness, monotonicity, etc... Regularization term $\frac{\gamma}{2} \psi(\omega)$ can be chosen in different norms, for example:

- $\frac{\gamma}{2} \psi(\omega) = \frac{\gamma}{2} \|\omega\|_{L^p}^p$, $1 \le p \le 2$.
- $\frac{\gamma}{2} \psi(\omega) = \frac{\gamma}{2} \|\omega\|_{TV}$, TV means "total variation".
- $\frac{\gamma}{2} \psi(\omega) = \frac{\gamma}{2} \|\omega\|_{BV}$, BV means "bounded variation", a real-valued function whose TV is bounded (finite).
- $\frac{\gamma}{2} \psi(\omega) = \frac{\gamma}{2} \|\omega\|_{H^1}^2$.
- $\frac{\gamma}{2} \psi(\omega) = \frac{\gamma}{2} (\|\omega\|_{L^1} + \|\omega\|_{L^2}^2)$.

We consider the following Tikhonov functional for regularized classification problem

$$J_\gamma(\omega) = \frac{1}{2} \|y(\omega) - t\|_{L_2}^2 + \frac{\gamma}{2} \|\omega - \omega_0\|_{L_2}^2 := \varphi(\omega) + \frac{\gamma}{2} \psi(\omega), \tag{56}$$

where terms $\varphi(\omega), \psi(\omega)$ are considered in L_2 norm which is the classical Banach space. In (56) ω_0 is a good first approximation for the exact weight function ω^*, which is called also the first guess or the first approximation. For discussion about how the first guess in the functional (56) should be chosen we refer to [15,16,24].

In this section we will discuss following rules for choosing regularization parameter in (56):

- A-priori rule (Tikhonov's regularization)

  - For $\|t - t^*\| \le \delta$ a priori rule requires (see details in [15]):

$$\lim_{\delta 0} \gamma(\delta) \to 0, \;\; \lim_{\delta 0} \frac{\delta^2}{\gamma(\delta)} \to 0.$$

- A-posteriori rules:

  - Morozov's discrepancy principle [17,19,25].
  - Balancing principle [17].

A-priori rule and Morozov's discrepancy are most popular methods for the case when there exists estimate of the noise level δ in data t. Otherwise it is recommended to use balancing principle or other a-posteriori rules presented in [15–19,23].

5.1. The Tikhonov's regularization

The goal of regularization is to construct sequences $\left\{ \gamma\left(\delta_k \right) \right\}$, $\left\{ \omega_{\gamma(\delta_k)} \right\}$ in a stable way so that

$$\lim_{k\infty} \left\| \omega_{\gamma(\delta_k)} - \omega^* \right\|_{B_1} = 0,$$

where a sequence $\{\delta_k\}_{k=1}^{\infty}$ is such that

$$\delta_k > 0, \quad \lim_{k\infty} \delta_k = 0. \tag{57}$$

Using (54) and (56), we obtain

$$J_\gamma\left(\omega^* \right) = \frac{1}{2} \left\| y(\omega^*) - t \right\|_{B_2}^2 + \frac{\gamma(\delta)}{2} \left\| \omega^* - \omega_0 \right\|_Q^2 \tag{58}$$

$$\leq \frac{\delta^2}{2} + \frac{\gamma(\delta)}{2} \left\| \omega^* - \omega_0 \right\|_Q^2. \tag{59}$$

Let

$$m_{\gamma(\delta_k)} = \inf_G J_{\gamma(\delta_k)}\left(\omega \right).$$

By (59)

$$m_{\gamma(\delta_k)} \leq \frac{\delta_k^2}{2} + \frac{\gamma\left(\delta_k \right)}{2} \left\| \omega^* - \omega_0 \right\|_Q^2.$$

Hence, there exists a point $\omega_{\gamma(\delta_k)} \in G$ such that

$$m_{\gamma(\delta_k)} \leq J_{\gamma(\delta_k)}\left(\omega_{\gamma(\delta_k)} \right) \leq \frac{\delta_k^2}{2} + \frac{\gamma\left(\delta_k \right)}{2} \left\| \omega^* - \omega_0 \right\|_Q^2. \tag{60}$$

Thus, by (56) and (60)

$$\frac{1}{2} \left\| y(\omega_{\gamma(\delta_k)}) - t \right\|_{B_2}^2 + \frac{\gamma\left(\delta_k \right)}{2} \left\| \omega_{\gamma(\delta_k)} - \omega_0 \right\|_Q^2 = J_\gamma\left(\omega_{\gamma(\delta_k)} \right). \tag{61}$$

From (61) follows that

$$\frac{1}{2} \left\| y(\omega_{\gamma(\delta_k)}) - t \right\|_{B_2}^2 \leq J_\gamma\left(\omega_{\gamma(\delta_k)} \right), \tag{62}$$

$$\frac{\gamma\left(\delta_k \right)}{2} \left\| \omega_{\gamma(\delta_k)} - \omega_0 \right\|_Q^2 \leq J_\gamma\left(\omega_{\gamma(\delta_k)} \right). \tag{63}$$

Using (63) and then (60) one can obtain

$$\left\| \omega_{\gamma(\delta_k)} - \omega_0 \right\|_Q^2 \leq \frac{2}{\gamma\left(\delta_k \right)} J_\gamma\left(\omega_{\gamma(\delta_k)} \right) \leq \frac{2}{\gamma\left(\delta_k \right)} \cdot \left[\frac{\delta_k^2}{2} + \frac{\gamma\left(\delta_k \right)}{2} \left\| \omega^* - \omega_0 \right\|_Q^2 \right] \tag{64}$$

from what follows that

$$\left\| \omega_{\gamma(\delta_k)} - \omega_0 \right\|_Q^2 \leq \frac{\delta_k^2}{\gamma\left(\delta_k \right)} + \left\| \omega^* - \omega_0 \right\|_Q^2. \tag{65}$$

Suppose that

$$\lim_{k\infty} \gamma\left(\delta_k \right) = 0 \text{ and } \lim_{k\infty} \frac{\delta_k^2}{\gamma\left(\delta_k \right)} = 0. \tag{66}$$

Then (65) implies that the sequence $\left\{\omega_{\gamma(\delta_k)}\right\} \subset G \subseteq Q$ is bounded in the norm of the space Q. Since Q is compactly embedded in B_1, then there exists a sub-sequence of the sequence $\left\{\omega_{\gamma(\delta_k)}\right\}$ which converges in the norm of the space B_1.

To ensure (66) one can choose, for example

$$\gamma\left(\delta_k\right) = C\delta_k^\mu, \mu \in (0,2), \quad C = const. > 0, \delta \in (0,1). \tag{67}$$

Other choices of γ which satisfy conditions (66) are also possible.

In [15,20] was proposed following iterative update of the regularization parameters γ_k which satisfy conditions (66):

$$\gamma_k = \frac{\gamma_0}{(k+1)^p}, \quad p \in (0,1], \tag{68}$$

where γ_0 can be computed as in (67).

5.2. Morozov's discrepancy principle

The principle determines the regularization parameter $\gamma = \gamma(\delta)$ in (56) such that

$$\|y(\omega_{\gamma(\delta)}) - t\| = c_m \delta, \tag{69}$$

where $c_m \geq 1$ is a constant. Relaxed version of a discrepancy principle is:

$$c_{m,1}\delta \leq \|y(\omega_{\gamma(\delta)}) - t\| \leq c_{m,2}\delta, \tag{70}$$

for some constants $1 \leq c_{m,1} \leq c_{m,2}$. The main feature of the principle is that the computed weight function $\omega_{\gamma(\delta)}$ can't be more accurate than the residual $\|y(\omega_{\gamma(\delta)}) - t\|$.

For the Tikhonov functional

$$J_\gamma\left(\omega\right) = \frac{1}{2}\|y(\omega) - t\|_2^2 + \gamma\|\omega\|_2^2 = \varphi(\omega) + \gamma\psi(\omega), \tag{71}$$

the value function $F(\gamma) : \mathbb{R}^+ \to \mathbb{R}$ is defined accordingly to [23] as

$$F(\gamma) = \inf_\omega J_\gamma(\omega). \tag{72}$$

If there exists $F'_\gamma(\gamma)$ at $\gamma > 0$ then from (71) and (72) follows that

$$F(\gamma) = \inf_\omega J_\gamma\left(\omega\right) = \underbrace{\varphi'(\omega)}_{\bar\varphi(\gamma)} + \gamma\underbrace{\psi'(\omega)}_{\bar\psi(\gamma)}. \tag{73}$$

Since $F'_\gamma(\gamma) = \psi'(\omega) = \bar\psi(\gamma)$ then from (73) follows

$$\bar\psi(\gamma) = F'_\gamma(\gamma), \quad \bar\varphi(\gamma) = F(\gamma) - \gamma F'_\gamma(\gamma). \tag{74}$$

The main idea of the principle is to compute discrepancy $\bar\varphi(\gamma)$ using the value function $F(\gamma)$ and then approximate $F(\gamma)$ via model functions. If $\bar\psi(\gamma) \in C(\gamma)$ then the discrepancy equation (69) can be rewritten as

$$\bar\varphi(\gamma) = F(\gamma) - \gamma F'_\gamma(\gamma) = \frac{\delta^2}{2}. \tag{75}$$

The goal is to solve (75) for γ. Main methods for solution of (75) are the model function approach and a quasi-Newton method presented in details in [17].

5.3. Balancing principle

The balancing principle (or Lepskii, see [26,27]) finds $\gamma > 0$ such that following expression is fulfilled

$$\bar{\varphi}(\gamma) = C\gamma\bar{\psi}(\gamma), \tag{76}$$

where $C = a_0/a_1$ is determined by the statistical a priori knowledge from shape parameters in Gamma distributions [17]. When $\gamma = 1$ the method is called zero crossing method, see details in [28]. In [17] for computing γ is proposed the fixed point algorithm 4. Convergence of this algorithm is also analyzed in [17].

Algorithm 4 Fixed point algorithm

- Step 1. Start with the initial approximations γ_0 and compute the sequence of γ_k in the following steps.
- Step 2. Compute the value function $F(\gamma_k) = \inf_\omega J_{\gamma_k}(\omega)$ for (71) and get ω_{γ_k}.
- Step 3. Update the reg. parameter $\gamma := \gamma_{k+1}$ as

$$\gamma_{k+1} = \frac{\varphi(\omega_{\gamma_k})}{\psi(\omega_{\gamma_k})}$$

- Step 4. For the tolerance $0 < \theta < 1$ chosen by the user, stop computing reg.parameters γ_k if computed γ_k are stabilized, or $|\gamma_k - \gamma_{k-1}| \le \theta$. Otherwise, set $k := k+1$ and go to Step 2.
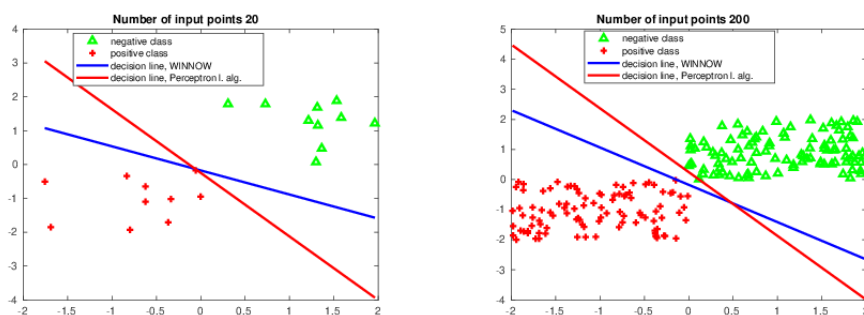

Figure 5. Perceptron learning (red line) and WINNOW (blue line) algorithms for separation of two classes.

6. Numerical results

In this section are presented several examples which show performance and effectiveness of least squares, perceptron and WINNOW algorithms for classification. We note that all classification algorithms considered here doesn't include regularization.

6.1. Test 1

In this test the goal is to compute decision boundaries for two linearly separated classes using least squares classification. Points in these classes are generated randomly by the linear function $y = 1.2 - 0.5x$ on different input intervals for x. Then the random noise δ is added to the data $y(x)$ as

$$y_\delta(x) = y(x)(1 + \delta\alpha), \tag{77}$$

where $\alpha \in (1,1)$ is randomly distributed number and $\delta \in [0,1]$ is the noise level. Then obtained points are classified such that the target function for classification is defined as

$$t_i = \begin{cases} 1, & y_\delta(x) - 1.2 + 0.5x > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{78}$$

Figures 1 present classification performed via least squares minimization $\min_\omega \|A\omega - y\|_2^2$ for the linear model function (23) with target values *t* given by (78) and elements of the design matrix *A* given by (25). Using these figures we observe that the least squares can be used successfully for classification when it is known that classes are linearly separated.

6.2. Test 2

Here we present examples of performance of the perceptron learning algorithm for classification of two linearly separated classes. Data for analysis in these examples are generated similarly as in the Test 1. Figures 3 present classification of two classes in the perceptron algorithm with three weights. Figures 4 show classification of two classes using the second order polynomial function (46) in the perceptron algorithm with six weights. We note that the red and blue lines presented in Figures 4 are classification boundaries computed via (51). Figures 5 present comparison of linear perceptron and WINNOW algorithms for separation of two classes. Again, all these figures show that perceptron and WINNOW algorithms can be successfully used for separation of linearly separated classes.
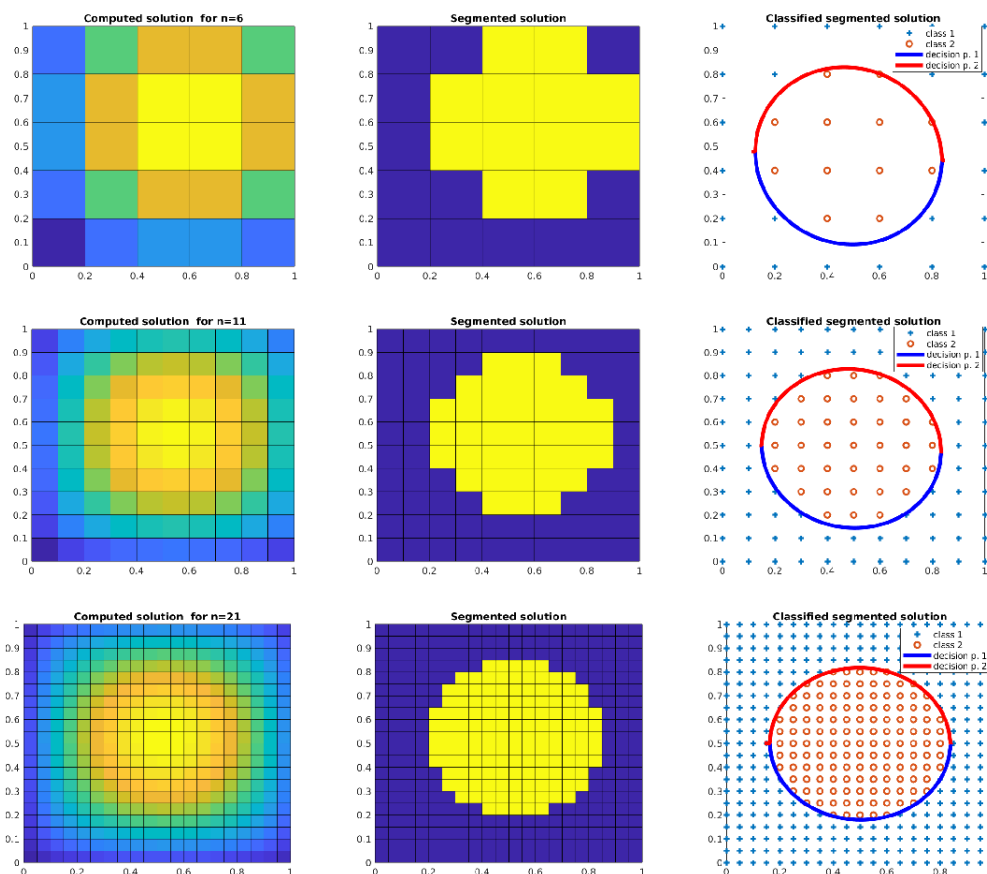

Figure 6. Classification of the computed solution for Poisson's equation (see example 8.1.3 of [20]) on different meshes using perceptron learning algorithm.

6.3. Test 3

This test shows performance of using the second order polynomial function (46) in the perceptron algorithm for classification of the segmented solution. The classification problem in this example is formulated as follows:

- Given the computed solution of the Poisson's equation $\triangle u = f$ with homogeneous boundary conditions $u = 0$ on the unit square, classify the discrete solution u_h into two classes such that the target function for classification is defined as (see middle figures of Figure 6):

$$t_i = \begin{cases} 1, & u_{h_i} > 4 & \text{(yellow points)}, \\ 0, & \text{otherwise} & \text{(blue points)}. \end{cases} \tag{79}$$

- Use the second order polynomial function (46) in the perceptron algorithm to compute decision boundaries.

Details about setup and numerical method to compute solution of Poisson's equation are presented in the example 8.1.3 of [20]. Figure 6 presents classification of the computed solution for Poisson's equation on different meshes using second order polynomial in classification algorithm. The computed solution u_h of the Poisson's equation on different meshes is presented on the left figures of Figure 6. The middle figures show segmentation of the obtained solution satisfying (79). The right figures of Figure 6 show results of applying the second order polynomial function (46) in the perceptron algorithm for classification of the computed solution u_h with target function (79). We observe that in this particular example computed decision lines correctly separate two classes even if these classes are not linearly separable.

6.4. Test 4

In this test we show performance of linear least squares together with linear and quadratic perceptron algorithms for classification of experimental data sets: database of grey seals [13] and Iris data set [12]. Matlab code to run these simulations is available for download from the link provided in [13].

Figure 7-a) shows classification of seal length and seal weight depending on the year. Figure 7-b) shows classification of seal length and seal thickness depending on the seal weight. We observe that classes on Figure 7-a) are not linearly separable and the best algorithm which separates both classes well, is the least squares. In this example, the linear and quadratic perceptron have not classified data correctly and actually, these algorithms have not converged and stopped when the maximal number of iterations (10^8) was reached. As soon as classes become linearly separated, all algorithms show good performance and computes almost the same separation lines, see Figure 7-b).

The same conclusion is obtained from separation of Iris data set [12]. Figures 8 show decision lines computed by least squares, linear and quadratic perceptron algorithms. Since all classes of Iris data set are linearly separable, all classification algorithms separate data correctly.
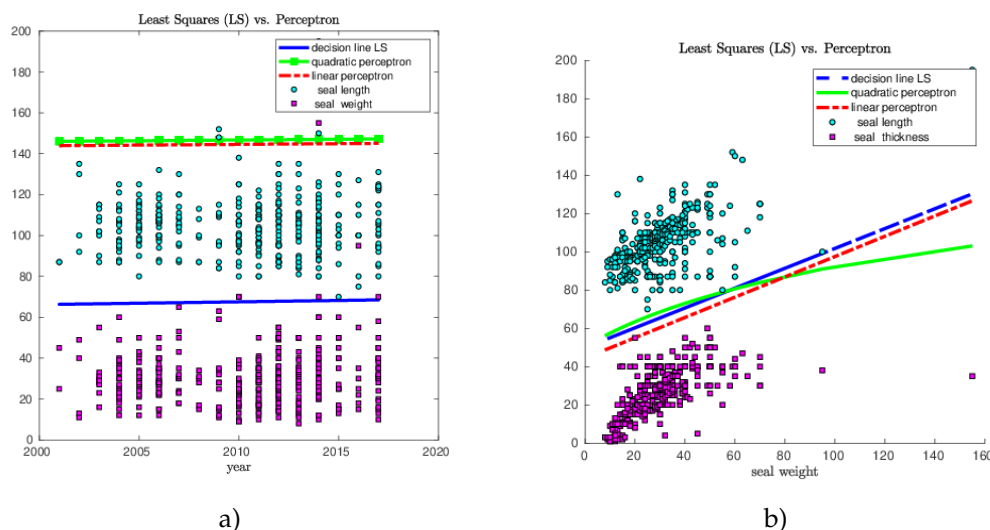

a)   b)

Figure 7. Least squares (LS) and Perceptron learning algorithm for separation of two classes using Grey Seal database [13].
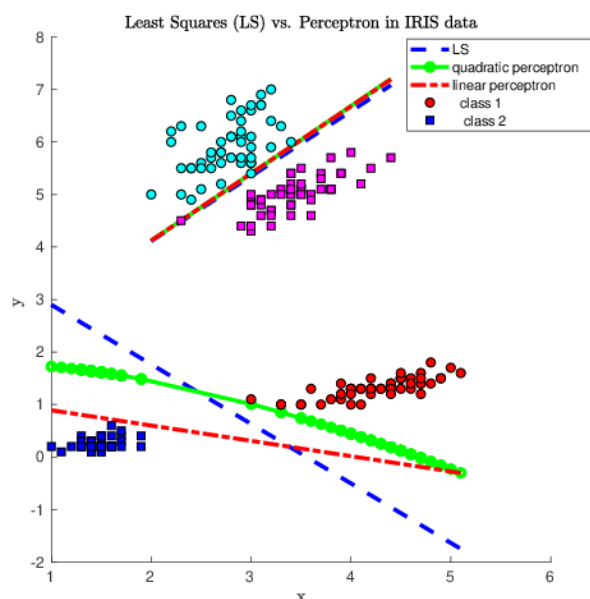
Figure 8. Least squares (LS) and Perceptron learning algorithm in classification of Iris dataset [12].

7. Conclusion

We have presented regularized and non-regularized perceptron learning and least squares algorithms for classification problems as well as discussed main a-priori and a-posteriori Tikhonov's regularization rules for choosing the regularization parameter. The Fréchet derivatives for least squares and perceptron algorithms are also rigorously derived.

The future work can be related to computation of miss-classification rates which can be done similarly with works [2,29], as well as to study of classification problem using regularized linear regression, perceptron learning and WINNOW algorithms. Other classification algorithms such that regularized SVM and kernel methods can be also analyzed. Testing of all algorithms on different benchmarks as well as extension to multiclass case should be also investigated.

Conflicts of Interest: "The author declares no conflict of interest."

Bibliography

[1] Bishop, C. M. (2006). *Pattern recognition and machine learning.* Springer.

[2] Hand, D. J., Mannila, H., & Smyth, P. (2001). *Principles of data mining (adaptive computation and machine learning)*. MIT Press.

[3] Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering, 160*(1), 3-24.

[4] Kubat, M. (2017). *An introduction to machine learning*. Springer International Publishing AG.

[5] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

[6] Tsai, J. T., Chou, J. H., & Liu, T. K. (2006). Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm. *IEEE Transactions on Neural Networks, 17*(1), 69-80.

[7] Lin, S. W., Ying, K. C., Chen, S. C., & Lee, Z. J. (2008). Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert Systems with Applications, 35*(4), 1817-1824.

[8] Meissner, M., Schmuker, M., & Schneider, G. (2006). Optimized Particle Swarm Optimization (OPSO) and its application to artificial neural network training. *BMC Bioinformatics, 7*(1), 125.

[9] Bartz-Beielstein, T., Chiarandini, M., Paquete, L., & Preuss, M. (Eds.). (2010). *Experimental methods for the analysis of optimization algorithms* (pp. 311-336). Berlin: Springer.

[10] Bergstra, J., Yamins, D., & Cox, D. D. (2013, June). Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference* (Vol. 13, p. 20). Citeseer.

[11] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951-2959).

[12] Iris flower data set. Retrieved from https://en.wikipedia.org/wiki/Iris_flower_data_set

[13] Classification of grey seals: Matlab code and database. Retrieved from http://www.waves24.com/download

[14] Persson, C. (2016). *Iteratively regularized finite element method for conductivity reconstruction in a waveguide* (Master's thesis). Chalmers University of Technology.

[15] Bakushinsky, A. B., & Kokurin, M. Y. (2005). *Iterative methods for approximate solution of inverse problems* (Vol. 577). Springer Science & Business Media.

[16] Beilina, L., & Klibanov, M. V. (2012). *Approximate global convergence and adaptivity for coefficient inverse problems.* Springer Science & Business Media.

[17] Ito, K., & Jin, B. (2015). *Inverse problems: Tikhonov theory and algorithms.* Series on Applied Mathematics, World Scientific.

[18] Kaltenbacher, B., Neubauer, A., & Scherzer, O. (2008). *Iterative regularization methods for nonlinear ill-posed problems* (Vol. 6). Walter de Gruyter.

[19] Tikhonov, A. N., Goncharsky, A. V., Stepanov, V. V., & Yagola, A. G. (2013). *Numerical methods for the solution of ill-posed problems* (Vol. 328). Springer Science & Business Media.

[20] Beilina, L., Karchevskii, E., & Karchevskii, M. (2017). *Numerical linear algebra: Theory and applications.* Springer.

[21] Beilina, L., Thành, N. T., Klibanov, M. V., & Malmberg, J. B. (2014). Reconstruction of shapes and refractive indices from backscattering experimental data using the adaptivity. *Inverse Problems, 30*(10), 105007.

[22] Zhang, T. (2001). Regularized winnow methods. In *Advances in Neural Information Processing Systems* (pp. 703-709).

[23] Tikhonov, A. N., & Arsenin, V. Y. (1977). *Solutions of ill-posed problems*. New York, 1-30.

[24] Klibanov, M. V., Bakushinsky, A. B., & Beilina, L. (2011). Why a minimizer of the Tikhonov functional is closer to the exact solution than the first guess. *Journal of Inverse and Ill-Posed Problems, 19*(1), 83-105.

[25] Morozov, V. A. (1966). On the solution of functional equations by the method of regularization. In *Doklady Akademii Nauk* (Vol. 167, No. 3, pp. 510-512). Russian Academy of Sciences.

[26] Lazarov, R. D., Lu, S., & Pereverzev, S. V. (2007). On the balancing principle for some problems of numerical analysis. *Numerische Mathematik, 106*(4), 659-689.

[27] Mathé, P. (2006). The Lepskii principle revisited. Inverse problems, 22(3), L11-L15.

[28] Johnston, P. R., & Gulrajani, R. M. (1997). A new method for regularization parameter determination in the inverse problem of electrocardiography. *IEEE Transactions on Biomedical Engineering, 44*(1), 19-39.

[29] Thomas, P. (2015, November). Perceptron learning for classification problems: Impact of cost-sensitivity and outliers robustness. In *2015 7th International Joint Conference on Computational Intelligence* (IJCCI) (Vol. 3, pp. 106-113). IEEE.