JSS30, Summer School, COM5: Machine learning in inverse and ill-posed problems

Larisa Beilina*

Department of Mathematical Sciences, Chalmers University of Technology and Gothenburg University, SE-42196 Gothenburg, Sweden

https://www.jyu.fi/en/research/

Classification algorithms
Computer Session 3

Outline

- least squares for classification problems
- Machine learning algorithms (perceptron learning, WINNOW) for classification problems

Supervised learning

- Linear regression as part of supervised learning.
- Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each 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 for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances.

Linear regression

- 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 $x = \{x_i\}, i = 1, ..., m$.
- The linear regression is similar to the solution of linear least squares problem.
- We will revise solution of linear least squares problem in terms of linear regression.

Used literature:

Miroslav Kurbat, *An Introduction to Machine Learning*, Springer, 2017.

Christopher M. Bishop, *Pattern recognition and machine learning*, Springer, 2009.

L. Beilina, E. Karchevskii, M. Karchevskii, *Numerical Linear Algebra: Theory and Applications*, Springer, 2017 – see link to GitHub with Matlab code:

`https://github.com/springer-math/Numerical_Linear_Algebra_Theory_and_Applications`

L. Beilina, Numerical analysis of least squares and perceptron learning for classification problems, `https://arxiv.org/abs/2004.01138`

Classification problem

Usually, classification algorithms are working well for linearly separable data sets.

Definition
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 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.
- The 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.

Classification problem: an example

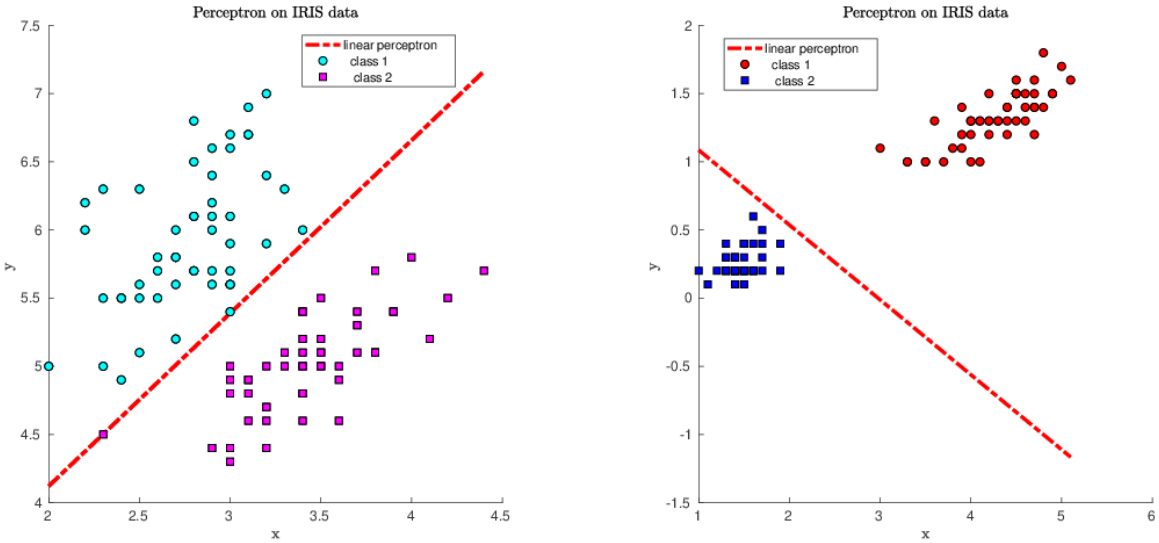


Figure: Decision lines computed by the perceptron learning algorithm for separation of two classes using Iris dataset. Test Matlab program to generate this figures on the course page.

Non-regularized classification problem

The non-regularized classification problem is formulated as a standard least squares problem.
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

$$F(\omega) = \frac{1}{2}\|t - y(\omega)\|_2^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i(\omega))^2 \tag{1}$$

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.

Regularized classification problem

In the regularized classification problem to find optimal vector of weights $\omega = \{\omega_i\}, i = 1, ..., M$, to the functional (1) is added the regularization term such that the functional is written as

$$F(\omega) = \frac{1}{2}\|t - y(\omega)\|_2^2 + \frac{1}{2}\gamma\|\omega\|_2^2 = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i(\omega))^2 + \frac{1}{2}\gamma\sum_{j=1}^{M}|\omega_j|^2. \quad (2)$$

Here, γ is the regularization parameter, $\|\omega\|_2^2 = \omega^T\omega = \omega_1^2 + ... + \omega_M^2$, M is the number of weights. In order to find the optimal weights in (1) or in (2), the following minimization problem should be solved

$$\min_{\omega} F(\omega). \quad (3)$$

Thus, we seek for a stationary point of (1) or (2) with respect to ω such that

$$F'(\omega)(\bar{\omega}) = 0, \quad (4)$$

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

More precisely, for the functional (2) we get

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

$$\frac{\partial F}{\partial \omega_i}(\omega)(\bar{\omega}_i) := F'_{\omega_i}(\omega)(\bar{\omega}_i) = -(t-y)\cdot y'_{\omega_i}(\bar{\omega}_i) + \gamma\omega_i(\bar{\omega}_i),\ i = 1, ..., M. \tag{5}$$

The Fréchet derivative of the functional (1) is obtained by taking $\gamma = 0$ in (5). To find optimal vector of weights $\omega = \{\omega_i\}, i = 1, ..., M$ can be used least squares or machine learning algorithms.

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

$$f(x, \omega) = \omega_0 \cdot 1 + \omega_1 x_1 + ... + \omega_M x_M. \tag{6}$$

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 (6).
The linear model (6) can be written in the form

$$f(x, \omega) = \omega_0 \cdot 1 + \sum_{i=1}^{M} \omega_i \varphi_i(x) = \omega_0 + \omega^T \varphi(x), \tag{7}$$

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

Non-regularized least squares problem

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

$$E(\omega) = \frac{1}{2}\sum_{n=1}^{N}(t_n - f(x,\omega))^2 = \frac{1}{2}\sum_{n=1}^{N}(t_n - \omega^T\varphi(x_n))^2 := \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2 \quad (8)$$

to find

$$\min_{\omega} E(\omega) = \min_{\omega} \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2. \quad (9)$$

The problem (9) is a typical least squares problem of the minimizing the squared residuals

$$\min_{\omega} \frac{1}{2}\|r(\omega)\|_2^2 = \min_{\omega} \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2 \quad (10)$$

with the residual $r(\omega) = t - \omega^T\varphi(x)$.

The test functions $\varphi(x)$ form the design matrix A

$$A = \begin{bmatrix} 1 & \varphi_1(x_1) & \varphi_2(x_1) & \dots & \varphi_M(x_1) \\ 1 & \varphi_1(x_2) & \varphi_2(x_2) & \dots & \varphi_M(x_2) \\ 1 & \varphi_1(x_3) & \varphi_2(x_3) & \dots & \varphi_M(x_3) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & \varphi_1(x_N) & \varphi_2(x_N) & \dots & \varphi_M(x_N) \end{bmatrix}, \tag{11}$$

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

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

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 (12):

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

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. See details in Lecture 6 for derivation.

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

$$\min_{\omega} \frac{1}{2}\|r(\omega)\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2 = \min_{\omega} \frac{1}{2}\|A\omega - t\|_2^2 + \frac{\gamma}{2}\|\omega\|_2^2. \tag{13}$$

To minimize the regularized squared residuals (13) we will again derive the *normal equations*. Similarly as was derived the Fréchet derivative for the non-regularized regression problem (12), 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:

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

The term

$$\lim_{\|e\|\to 0} \frac{|e^T A^T A e|}{\|e\|_2} \leq \lim_{\|e\|\to 0} \frac{\|A\|_2^2 \|e\|_2^2}{\|e\|_2} = \lim_{\|e\|\to 0} \|A\|_2^2 \|e\|_2 \to 0. \tag{14}$$

Similarly, the term

$$\lim_{\|e\|\to 0} \frac{|e^T e|}{\|e\|_2} = \lim_{\|e\|\to 0} \frac{\|e\|_2^2}{\|e\|_2} \to 0. \tag{15}$$

We finally get

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

The expression above means that the factor $A^T A\omega - A^T t + \gamma\omega$ must also be zero, or

$$(A^T A + \gamma I)\omega = A^T t,$$

where I is the identity matrix. This is a system of M linear equations for M unknowns, the normal equations for regularized least squares.

The normal equations for regularized least squares

$$(A^T A + \gamma I)\omega = A^T t,$$

for the square matrix $dim(A^T A + \gamma I) = M \times M$ are solved as usual system of linear equations to find vector of weights ω of the size M:

$$\omega = (A^T A + \gamma I)^{-1} A^T t,$$

The regularization parameter γ can be chosen by the same methods which are used for the solution of ill-posed problems [1,3,4] or by machine learning methods [2]. For different Tikhonov's regularization strategies for the solution of ill-posed problems look [1,3,4]. We will discuss main methods of Tikhonov's regularization which follows ideas of [1,3,4]


[1] A.B. Bakushinsky, M.Yu. Kokurin and A. Smirnova, *Iterative methods for ill-posed problems*, de Gruyter, 2011.


[2] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, `http://www.deeplearningbook.org`, 2016


[3] K. Ito, B. Jin, *Inverse Problems: Tikhonov theory and algorithms*, Series on Applied Mathematics, V.22, World Scientific, 2015.


[4] Tikhonov, A.N., Goncharsky, A., Stepanov, V.V., Yagola, A.G., *Numerical Methods for the Solution of Ill-Posed Problems*, ISBN 978-94-015-8480-7, 1995.

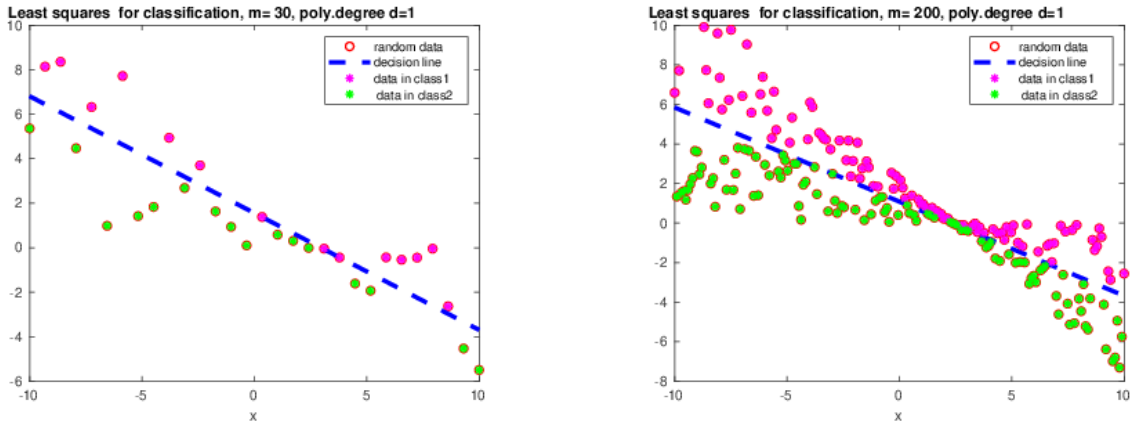


Figure: Examples of linear regression for classification for different number of input points.

Figure 2 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 (10) is

$$f(x, y, \omega) = \omega_0 + \omega_1 x + \omega_2 y \tag{16}$$

and the target values of the vector $t = \{t_i\}, i = 1, ..., N$ in (10) are

$$t_i = \begin{cases} 1 & \text{red points}, \\ 0 & \text{green points}. \end{cases} \tag{17}$$

The elements of the design matrix (11) are given by

$$A = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \\ \vdots & \vdots & \ddots \\ 1 & x_N & y_N \end{bmatrix}. \tag{18}$$

Here, (x_i, y_i) are coordinates (x, y) for all (red and green) points.

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

$$f_1(x, c) = c_{1,1}\phi_1(x) + c_{2,1}\phi_2(x) + ... + c_{n,1}\phi_n(x). \quad (19)$$

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

$$f_2(x, c) = c_{1,2}\phi_1(x) + c_{2,2}\phi_2(x) + ... + c_{n,2}\phi_n(x). \quad (20)$$

Here, basis functions are $\phi_j(x), j = 1, ..., n$.

Polynomial fitting to data in two-class model

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

$$\min_c \|y - f(x, c)\|_2^2 = \min_c \sum_{i=1}^{m} (y_i - f(x_i, c))^2 \tag{21}$$

is solved with $m = k + l$.

If the function $f(x, c)$ in (21) is linear then we can reformulate the minimization problem (21) as the following least squares problem

$$\min_c ||Ac - y||_2^2, \tag{22}$$

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 (22) can be found by the method of normal equations derived in Section 12:

$$c = (A^T A)^{-1} A^T b = A^+ b \tag{23}$$

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

$$\phi_j(x) = x^{j-1}, \quad j = 1, ..., n \tag{24}$$

have been considered in the problem of fitting to a polynomial in Lecture 6.
The matrix A constructed by these basis functions is a Vandermonde matrix, and problems related to this matrix we discussed in Lecture 6.
Linear splines (or hat functions) and bellsplines also can be used as basis functions, see Lecture 6.

Figures 3 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.

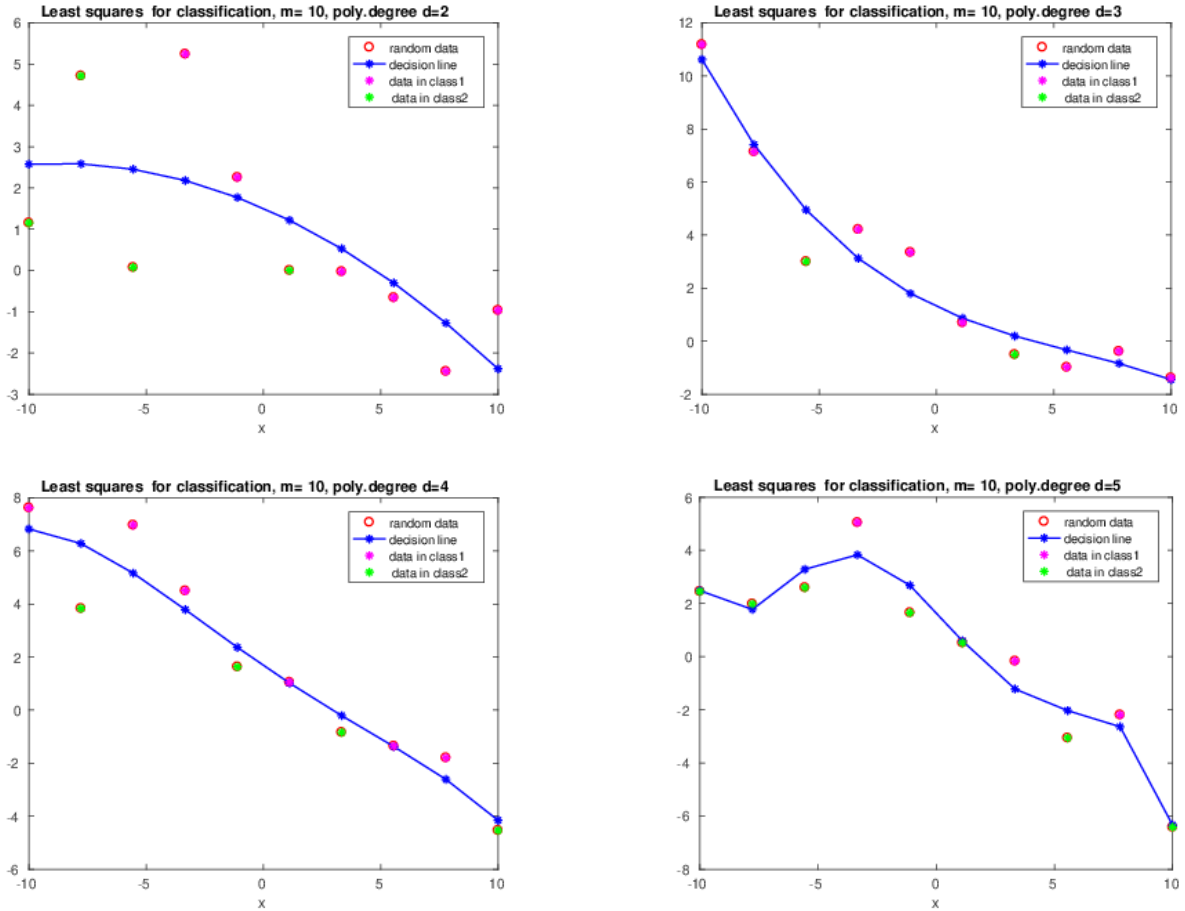


Figure: Least squares in polynomial fitting to data for different degree of polynomial in the test functions (24).

Machine learning linear and polynomial classifiers

Let us start with considering of an example: determine the decision line for points presented in the Figure. Two classes are separated by the linear equation with three weights $\omega_i, i = 1, 2, 3$, given by

$$\omega_1 + \omega_2 x + \omega_3 y = 0. \tag{25}$$

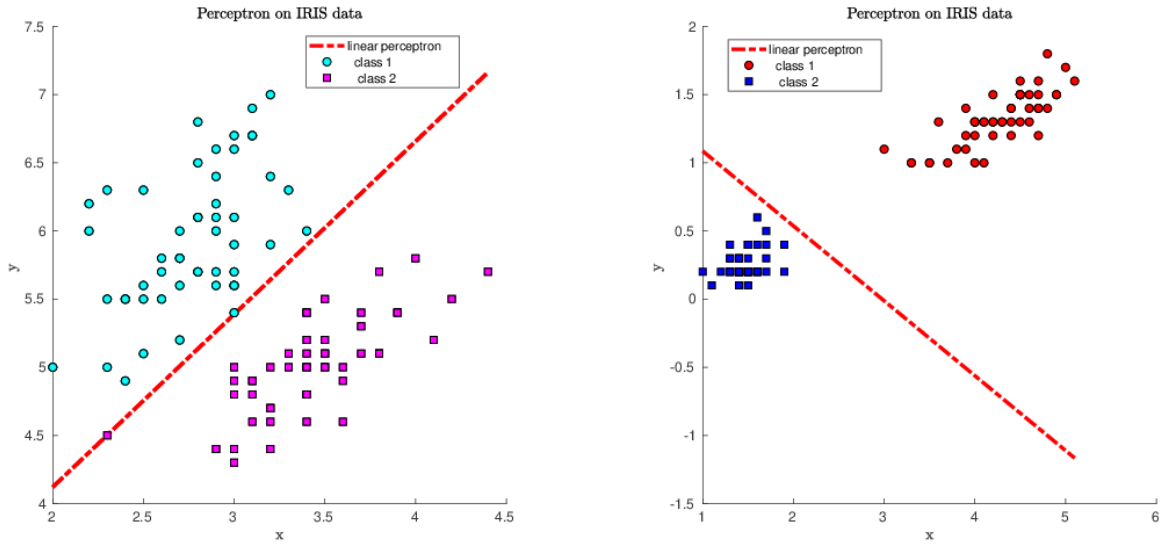


Figure: Decision lines computed by the perceptron learning algorithm for separation of two classes using Iris dataset. Test Matlab program to generate this figures on the course page.

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

$$\omega_0 + \omega_1 x_1 + \omega_2 x_2 + ... + \omega_n x_n = 0 \tag{26}$$

which also can be written as

$$\omega^T x = \sum_{i=0}^{n} \omega_i x_i = 0 \tag{27}$$

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.

We can rewrite (26) as

$$\omega_0 + \omega_1\varphi(x_1) + \omega_2\varphi(x_2) + ... + \omega_n\varphi(x_n) = 0 \tag{28}$$

or as

$$\omega^T\varphi(x) = \sum_{i=0}^{n} \omega_i\varphi(x_i) = 0 \tag{29}$$

where $\varphi(x_i), i = 1, ..., n$ are basis functions.
In the non-regularized classification problem our goal is to minimize the following functional

$$E(\omega) = \frac{1}{2}\|t - \omega^T\varphi(x)\|_2^2 = \frac{1}{2}\sum_{n=1}^{N}(t_n - \omega^T\varphi(x_n))^2. \tag{30}$$

Study of bias parameter ω_0

Let us rewrite (30) making the bias parameter ω_0 explicit

$$E(\omega) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \omega^T\varphi(x_n))^2 = \frac{1}{2}(\sum_{n=1}^{N}(t_n - \omega_0 - \sum_{j=1}^{M}\omega_j\varphi_j(x_n))^2. \quad (31)$$

Now we take derivative $E'(\omega_0) = 0$ to get

$$0 = E'(\omega_0) = (\sum_{n=1}^{N}(t_n - \omega_0 - \sum_{j=1}^{M}\omega_j\varphi_j(x_n))(-1), \quad (32)$$

or

$$0 = E'(\omega_0) = \sum_{n=1}^{N} t_n - N\omega_0 - (\omega_1\sum_{n=1}^{N}\varphi_1(x_n) + ... + \omega_M\sum_{n=1}^{N}\varphi_M(x_n)). \quad (33)$$

Study of bias parameter ω_0

Now solving (33) for ω_0 we get

$$\omega_0 = \bar{t} - \sum_{j=1}^{M} \omega_j \bar{\varphi}_j, \tag{34}$$

where

$$\begin{aligned} \bar{t} &= \frac{1}{N} \sum_{n=1}^{N} t_n, \\ \bar{\varphi}_j &= \frac{1}{N} \sum_{n=1}^{N} \varphi_j(x_n). \end{aligned} \tag{35}$$

Using (34) we conclude that bias ω_0 balance the difference between the averages of the target values and the weighted sum of the averages of the basis functions.

Perceptron learning for classification

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

$$y(x, \omega) = sign(\omega^T x) = sign(\sum_{i=0}^{n} \omega_i x_i) \tag{36}$$

and uses the binary classification to compute weights ω_i:

$$sign(\omega^T x) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} \omega_i x_i + \omega_0 > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{37}$$

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

$$\omega^T x = 0.$$

Perceptron learning for classification

Binary classifier decides whether or not an input x belongs to some specific class:

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

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.
The algorithm which determines weights via binary classification can be reasoned by minimization of the regularized residual

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

where $\xi_\delta(x)$ for a small δ is a data compatibility function to avoid discontinuities and γ is the regularization parameter. Taking $\gamma = 0$ algorithm will minimize the non-regularized residual (39).

Perceptron learning for classification

Alternative, it can be minimized the residual

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

over the set $M \subset \{1, ..., m\}$ of the currently miss-classified patterns.

Perceptron learning for classification

In the Perceptron learning algorithm which we will study in update of weights is used the following regularized functional

$$\begin{aligned} F(\omega) &= \frac{1}{2}\|(t - y(x,\omega))\xi_\delta(x)\|_2^2 + \frac{1}{2}\gamma\|w\|_2^2 \\ &= \frac{1}{2}\sum_{i=1}^{m}((t_i - y_i(x,\omega))\xi_\delta(x))^2 + \frac{1}{2}\gamma\sum_{i=1}^{n} w_i^2. \end{aligned} \tag{41}$$

Here, γ is the regularization parameter, t is the target function, or class c in the algorithm, which takes values 0 or 1.
To find optimal weights in (41) we need to solve the minimization problem in the form (3)

$$F'(\omega)(\bar{\omega}) = 0, \tag{42}$$

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.

Perceptron learning for classification

Let us consider the difference $(\|r(x,\omega+e)\|_2^2+\frac{\gamma}{2}\|\omega+e\|_2^2)-(\|r(x,\omega)\|_2^2+\frac{\gamma}{2}\|\omega\|_2^2)$ for $y(x,\omega)=\sum_{i\in M}\omega_i x_i$. We single out the linear part with respect to ω to obtain:

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

Perceptron learning for classification

The second part in the last term of the above expression is estimated as in (15). The second part in the first term is estimated as

$$\lim_{\|e\|\to 0} \frac{|(e^T x\, \xi_\delta(x))^T\, e^T x\, \xi_\delta(x)|}{\|e\|_2} = \lim_{\|e\|\to 0} \frac{|(x^T e\, \xi_\delta(x))^T\, x^T e\, \xi_\delta(x)|}{\|e\|_2}$$
$$\leq \lim_{\|e\|\to 0} \frac{\|x\, \xi_\delta(x)\|_2^2 \|e\|_2^2}{\|e\|_2} = \lim_{\|e\|\to 0} \|x\, \xi_\delta(x)\|_2^2 \|e\|_2 \to 0. \tag{43}$$

We finally get

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

Perceptron learning for classification

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

$$\begin{aligned} F'(\omega)(\bar{\omega}) &= \sum_{i=1}^{n} F'_{\omega_i}(\omega)(\bar{\omega}_i), \\ F'_{\omega_i}(\omega)(\bar{\omega}_i) &= -(t - y) \cdot \xi_\delta(x) \cdot y'_{\omega_i}(\bar{\omega}_i) + \gamma\omega_i \\ &= -(t - y) \cdot x_i \cdot \xi_\delta(x_i) + \gamma\omega_i, \ i = 1, ..., n. \end{aligned} \tag{44}$$

The non-regularized version of the perceptron algorithm is obtained taking $\gamma = 0$ in (44).

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 $c(\mathbf{x}) = 0$.
- 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 $c(\mathbf{x}) = 0$.
- 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 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 $h(x) = 1$.
- 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 $h(x) = 0$.
- Step 4: Update weight $\omega := \omega^{m+1} = \{\omega_i^{m+1}\},\ i = 1, ..., M$ using

$$\omega_i^{m+1} = \omega_i^m + \eta \cdot ([c(\mathbf{x}) - h(\mathbf{x})] \cdot x_i + \gamma \cdot \omega_i^m), \tag{45}$$

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

Perceptron learning for classification: 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:

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

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

$$\omega_0 + \omega_1 z_1 + \omega_2 z_2 + \omega_3 z_3 + \omega_4 z_4 + \omega_5 z_5 = 0 \qquad (47)$$

which is already the linear function.

Thus, the Perceptron learning algorithm 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

$$x_2 = \frac{-b \pm \sqrt{D}}{2a}, \tag{51}$$
$$D = b^2 - 4ac.$$

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 Figure.

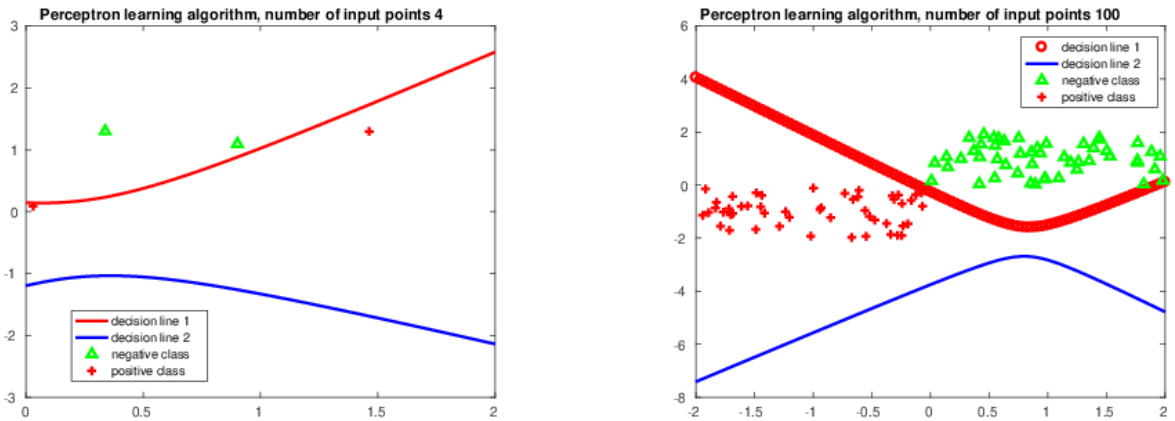


Figure: Perceptron learning algorithm for separation of two classes by polynomials of the second order.

WINNOW learning algorithm

Here we present 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. 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 which we study is written for $c = t$ and $y = h$ in (44). We will again assume that all examples where $c(\mathbf{x}) = 1$ are linearly separable from examples where $c(\mathbf{x}) = 0$.

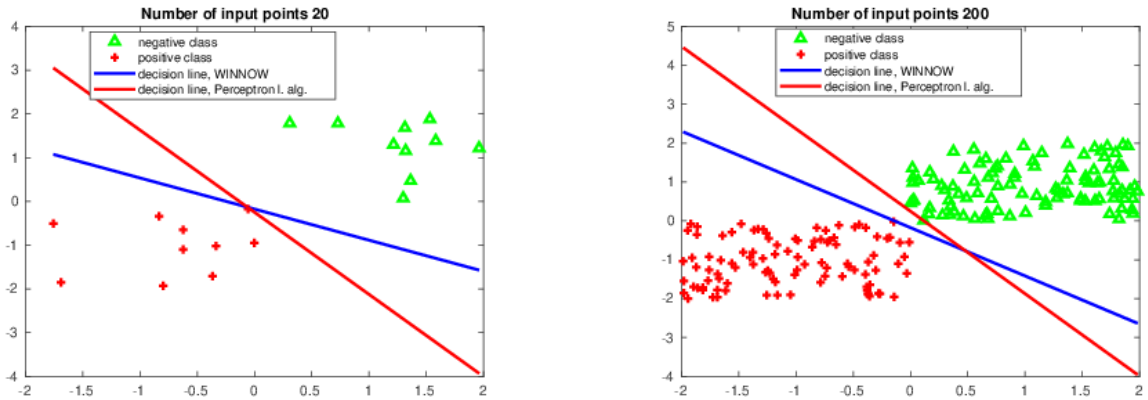


Figure: Comparison of two classification algorithms for separation of two classes: Perceptron learning algorithm (red line) and WINNOW (blue line).

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 ${\omega_i}^m$ 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 is positive and $h(x) = 1$.
- Step 3: If $\sum_{i=0}^{n} \omega_i^m x_i < 0$ we will say the the example is negative and $h(x) = 0$.
- Step 4: Update every weight 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 1.

Methods of Tikhonov's regularization

We consider the following Tikhonov functional for regularized classification problem

$$J_\gamma(\omega) = \frac{1}{2}\left\|y(\omega) - t\right\|_{L_2}^2 + \frac{\gamma}{2}\|\omega\|_{L_2}^2 := \varphi(\omega) + \frac{\gamma}{2}\psi(\omega), \qquad (52)$$

where terms $\varphi(\omega), \psi(\omega)$ are considered in L_2 norm which is the classical Banach space and γ is the regularization parameter.

Methods of Tikhonov's regularization

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.
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:

- $\frac{\gamma}{2}\psi(\omega) = \frac{\gamma}{2}\|\omega\|^p_{L^p}, \quad 1 \leq p \leq 2.$
- $\frac{\gamma}{2}\psi(\omega) = \frac{\gamma}{2}\|\omega\|_{TV}$, 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).
- $\frac{\gamma}{2}\psi(\omega) = \frac{\gamma}{2}\|\omega\|^2_{H^1}.$
- $\frac{\gamma}{2}\psi(\omega) = \frac{\gamma}{2}(\|\omega\|_{L^1} + \|\omega\|^2_{L^2}).$

We will discuss following rules for choosing γ in (52):

- A-priori rule (Tikhonov's regularization)
  - For $\|t - t^*\| \leq \delta$ a priori rule requires (see details in [1]):

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

- A-posteriori rules:
  - Morozov's discrepancy principle [2,3].
  - Balancing principle [2]

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 [2].


A.B. Bakushinsky, M.Yu. Kokurin and A. Smirnova, *Iterative methods for ill-posed problems*, de Gruyter, 2011.


K. Ito, B. Jin, *Inverse Problems: Tikhonov theory and algorithms*, Series on Applied Mathematics, V.22, World Scientific, 2015.


Tikhonov, A.N., Goncharsky, A., Stepanov, V.V., Yagola, A.G., *Numerical Methods for the Solution of Ill-Posed Problems*, ISBN 978-94-015-8480-7, 1995.

A-priori rule (Tikhonov's regularization)

For $\|t - t^*\| \leq \delta$ a priori rule requires (see details in [1]):

$$\lim_{\delta \to 0} \gamma(\delta) \to 0, \ \lim_{\delta \to 0} \frac{\delta^2}{\gamma(\delta)} \to 0. \tag{53}$$

To ensure (53) one can choose, for example

$$\gamma(\delta_k) = C\delta_k^{\mu}, \mu \in (0,2), \quad C = const. > 0, \delta \in (0,1). \tag{54}$$

Other choices of γ which satisfy conditions (53) are also possible. In [1] was proposed following iterative update of the regularization parameters γ_k which satisfy conditions (53):

$$\gamma_k = \frac{\gamma_0}{(k+1)^p}, \quad p \in (0,1], \tag{55}$$

where γ_0 can be computed as in (54).

A.B. Bakushinsky, M.Yu. Kokurin and A. Smirnova, *Iterative methods for ill-posed problems*, de Gruyter, 2011.

Morozov's discrepancy principle

The principle determines the regularization parameter $\gamma = \gamma(\delta)$ in (52) such that

$$\|y(\omega_{\gamma(\delta)}) - t\| = c_m \delta, \tag{56}$$

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

$$c_{m,1}\delta \leq \|y(\omega_{\gamma(\delta)}) - t\| \leq c_{m,2}\delta, \tag{57}$$

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 $\|y(\omega_{\gamma(\delta)}) - t\|$.

Morozov's discrepancy principle

For the Tikhonov functional

$$J_\gamma(\omega) = \frac{1}{2}\left\|y(\omega) - t\right\|_2^2 + \gamma\|\omega\|_2^2 = \varphi(\omega) + \gamma\psi(\omega), \tag{58}$$

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

$$F(\gamma) = \inf_\omega J_\gamma(\omega). \tag{59}$$

If there exists $F'_\gamma(\gamma)$ at $\gamma > 0$ then from (58) and (59) follows that

$$F(\gamma) = \inf_\omega J_\gamma(\omega) = \underbrace{\varphi'(\omega)}_{\bar{\varphi}(\gamma)} + \gamma \underbrace{\psi'(\omega)}_{\bar{\psi}(\gamma)}. \tag{60}$$

Since $F'_\gamma(\gamma) = \psi'(\omega) = \bar{\psi}(\gamma)$ then from (60) follows

$$\bar{\psi}(\gamma) = F'_\gamma(\gamma), \quad \bar{\varphi}(\gamma) = F(\gamma) - \gamma F'_\gamma(\gamma). \tag{61}$$

Morozov's discrepancy principle

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 (56) can be rewritten as

$$\bar{\varphi}(\gamma) = F(\gamma) - \gamma F'_{\gamma}(\gamma) = \frac{\delta^2}{2}. \tag{62}$$

The goal is to solve (62) for γ. Main methods for solution of (62) are the model function approach and a quasi-Newton method presented in details in [1].

K. Ito, B. Jin, *Inverse Problems: Tikhonov theory and algorithms*, Series on Applied Mathematics, V.22, World Scientific, 2015.

Balancing principle

The balancing principle (or Lepskii) finds $\gamma > 0$ such that following expression is fulfilled

$$\bar{\varphi}(\gamma) = C\gamma\bar{\psi}(\gamma), \tag{63}$$

where $C = a_0/a_1$ is determined by the statistical a priori knowledge from shape parameters in Gamma distributions [1]. When $C = 1$ the method is called zero crossing method.

K. Ito, B. Jin, *Inverse Problems: Tikhonov theory and algorithms*, Series on Applied Mathematics, V.22, World Scientific, 2015.

Balancing principle

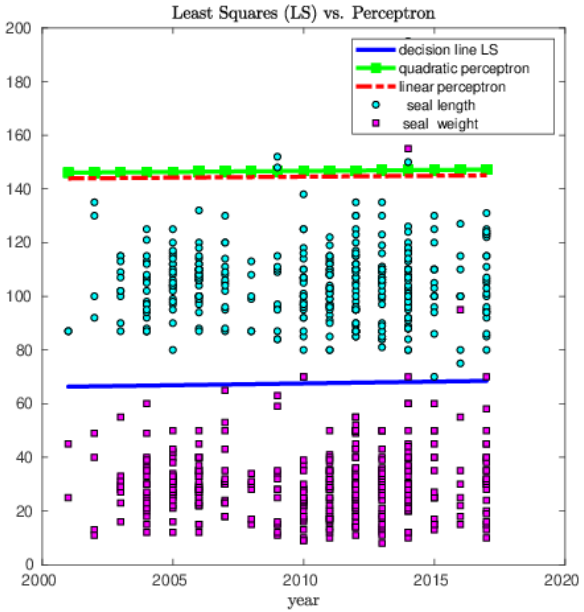
Below is presented the fixed point algorithm for solution of (63).

- Step 1. Start with the initial approximations γ_0 (for example, compute γ_0 via (54)) 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 (58) and get ω_{γ_k}.
- Step 3. Update the reg. parameter $\gamma := \gamma_{k+1}$ as

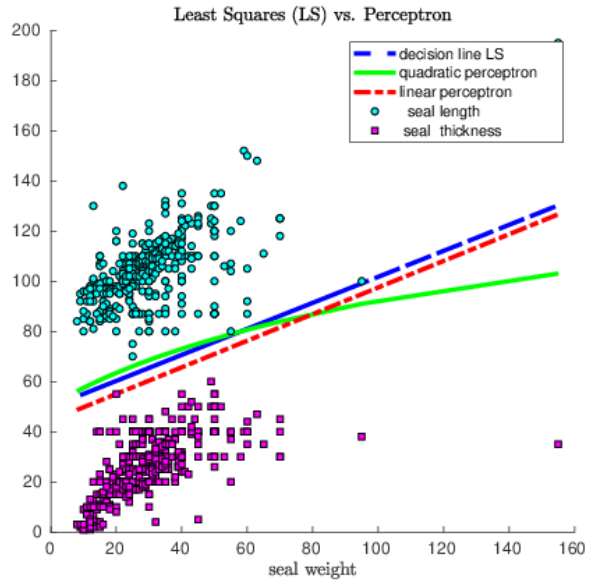
$$\gamma_{k+1} = \frac{\|\bar{\varphi}(\omega_{\gamma_k})\|_2}{\|\bar{\psi}(\omega_{\gamma_k})\|_2}$$

- 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 $|\gamma_k - \gamma_{k-1}| \leq \theta$. Otherwise, set $k := k + 1$ and go to Step 2.

Example: least squares (LS) and Perceptron learning algorithm



a)



b)

Figure: Comparison of least squares (LS) and Perceptron learning algorithm for separation of two classes using Grey Seal database `https://waves24.com/download/`.

Example: least squares (LS) and Perceptron learning algorithm

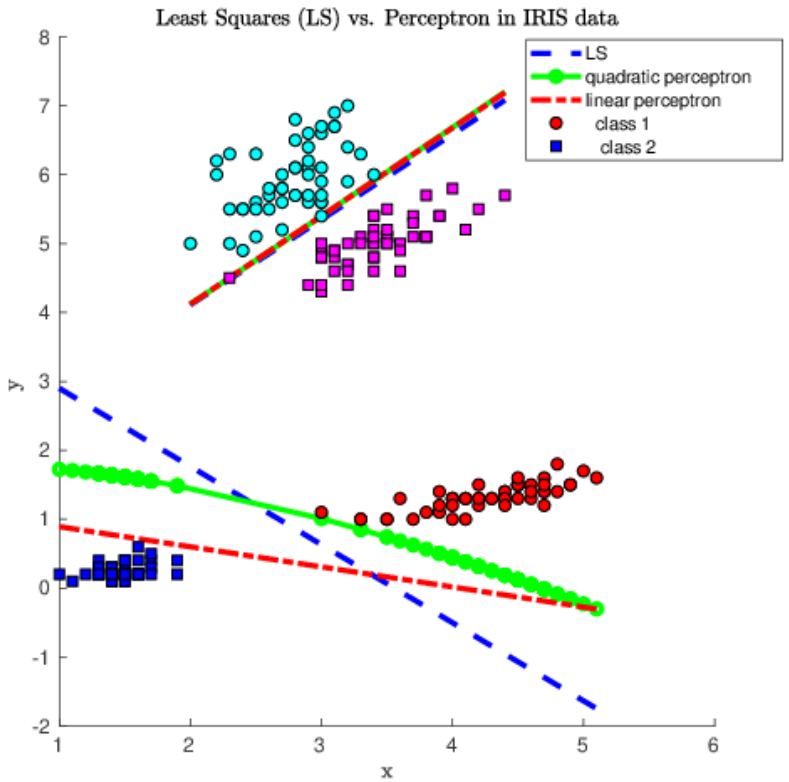


Figure: Comparison of least squares (LS) and Perceptron learning algorithm on Iris dataset `https://en.wikipedia.org/wiki/Iris_flower_data_set`.