Numerical Linear Algebra
TMA265/MMA600
Computer exercise 2:
Least squares and machine learning algorithms for classification problem

Larisa Beilina, larisa@chalmers.se

Computer exercise 2 (1 b.p.)

Least squares and machine learning algorithms for classification problem

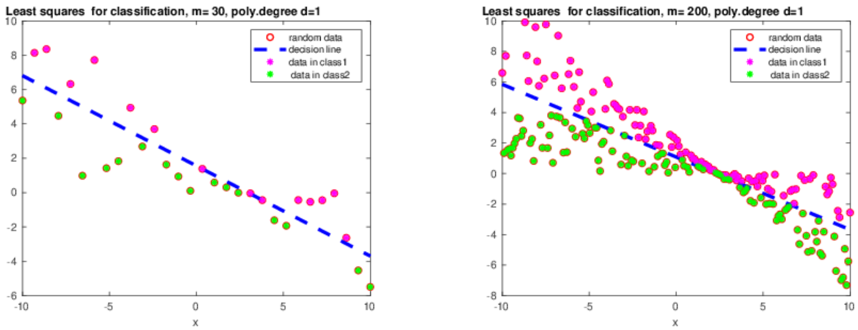

Figure 0.1: Examples of linear regression for classification for different number of input points.

In this exercise we will study different linear and quadratic classifiers: least squares classifier and perceptron learning algorithm using training sets described below.

<u>Computer exercise 2</u>

Implement in MATLAB least squares classifier and perceptron learning algorithm and present decision lines for following training sets:

- For randomly distributed data $y_i, i = 1, ..., m$ generated by the line

$$-1.2 + 0.5x + y = 0 \tag{0.1}$$

  on the interval $x = [-10, 10]$. Generate random data $(x, y_\sigma(x))$ using the formula (similar with comp.ex. 1)

$$y_\sigma(x) = y(x)(1 + \delta\alpha),$$

  where $\alpha \in (-1, 1)$ is randomly distributed number and $\delta \in [0, 1]$ is the noise level. For example, if the noise level in data is 5%, then $\delta = 0.05$. Then separate your points into 2 classes as follows: all points $(x_i, y_{\sigma i})$ which will be on the left side of the line (0.1) mark with code 1, and all points $(x_i, y_{\sigma i})$ which will be on the right side of the line (0.1) mark with code 0. In this way you will construct also your target vector t. Your points separated into 2 classes should look similarly as in Figure 0.1.

- Perform different experiments with different number of generated data points $m > 0$ which you choose as you want (for example, $m = 10, 100, 1000$).

- Add again noise as in (0.1) to already separated points into 2 classes and obtain new noisy data $y_\sigma(x)$. When you will add large noise σ you will observe that two classes are not more linearly separated. Check if classification algorithms are still working. Explain why some of algorithms are not working properly.

Optional (not necessary): compute missclassification rate E using the formula (see [4], p. 211-214):

$$\mathrm{E} = \frac{\sum_{i=1}^{K} N_{F,i}}{\sum_{i=1}^{K} (N_{T,i} + N_{F,i})}, \tag{0.2}$$

where K is the number of classes, $N_{T,i}$ is the number of points of the class i which are classified correctly, $N_{F,i}$ is the number of points of the class i which are classified wrong. Precision for class i can be computed as

$$\mathrm{P}(i) = \frac{N_{T,i}}{N_{T,i} + N_{F,j}}. \tag{0.3}$$

Try answer to the following questions:

- Analyze what happens with performance of perceptron learning algorithm if we take different learning rates $\eta \in (0, 1]$? For what values of η perceptron learning algorithm is more sensitive and when the iterative process is too slow?
- Analyze which one of the studied classification algorithms perform best and why?
- Try to explain in which case the perceptron algorithm will fail to separate data.

Hints:

1. In this exercise we will assume that we will work in domains with two classes: positive class and negative class. We will assume that each training example $\mathbf{x}$ can have values 0 or 1 and we will label positive examples with $c(\mathbf{x}) = 1$ and negative with $c(\mathbf{x}) = 0$.

2. We will also assume that these classes are linearly separable. These two classes can be separated by a linear function of the form

$$\omega_0 + \omega_1 x + \omega_2 y = 0, \tag{0.4}$$

where x, y are Cartesian coordinates. Note that the equation (0.4) can be rewritten for the case of a linear least squares problem as

$$\omega_0 + \omega_1 x = -\omega_2 y \tag{0.5}$$

or as

$$-\frac{\omega_0}{\omega_2} - \frac{\omega_1}{\omega_2} x = y. \tag{0.6}$$

3. We can determine coefficients $\omega_0, \omega_1, \omega_2$ in (0.4) by solving 2 different least squares problem:

   – By solving the following least squares problem for fitting the data y:

   $$\min_{\omega} \|A\omega - y\|_2^2 \tag{0.7}$$

   with $\omega = [\omega_1, \omega_2]^T = [-\frac{\omega_0}{\omega_2}, -\frac{\omega_1}{\omega_2}]^T$, where rows in the matrix A are given by

   $$[1, x_k], \quad k = 1, ..., m,$$

   and the known data vector is $y = [y_1, ..., y_m]^T$.

- Or by solving the following least squares problem:

$$\min_{\omega} \|f(x, y, \omega) - t\|_2^2 \tag{0.8}$$

with the linear model equation

$$f(x, y, \omega) = \omega_0 + \omega_1 x + \omega_2 y \tag{0.9}$$

and the target values of the vector $t = \{t_i\}, i = 1, ..., m$ which are defined as

$$t_i = \begin{cases} 1 & \text{point of 1 class,} \\ 0 & \text{point of 2 class.} \end{cases} \tag{0.10}$$

Thus, in the least squares problem

$$\min_{\omega} \|A\omega - t\|_2^2 \tag{0.11}$$

rows in matrix A will be:

$$[1, x_k, y_k], \quad k = 1, ..., m.$$

4. The Perceptron learning algorithm is taken from [4] and is presented below. Decision line then can be presented in Matlab for already computed weights by Perceptron learning algorithm using the formula (0.6).
5. Useful links for the literature in AI: [2, 3, 4]. Details about linear and quadratic perceptron learning algorithm and their implementation can be found in the paper *Numerical analysis of least squares and perceptron learning for classification problems* which can be downloaded from the link

   `https://arxiv.org/pdf/2004.01138.pdf`
6. Example of the Matlab code for classification of 2 classes using least squares, linear and quadratic perceptron on IRIS flower data set

   `http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/2021/Matlab/iris.csv`

   is available on the course homepage:

   `http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/2021/Matlab/testiris2.m`

   The complete dataset IRIS can be downloaded from the link:

   `https://en.wikipedia.org/wiki/Iris_flower_data_set`

Perceptron learning algorithm [4]

Assume that two classes c(**x**)=1 and c(**x**)=0 are linearly separable.

Step 0. Initialize all weights ω_i in

$$\sum_{i=0}^{n} \omega_i x_i = 0$$

to small random numbers (note $x_0 = 1$). Choose an appropriate learning rate $\eta \in (0, 1]$.

Step 1. For each training example $\mathbf{x} = (x_1, ..., x_n)$ whose class is $c(\mathbf{x})$ do:

- (i) Assign $h(\mathbf{x}) = 1$ if
$$\sum_{i=0}^{n} \omega_i x_i > 0$$
and assign $h(\mathbf{x}) = 0$ otherwise.
- (ii) Update each weight using the formula
$$\omega_i = \omega_i + \eta \cdot [c(\mathbf{x}) - h(\mathbf{x})] \cdot x_i.$$

Step 2. If $c(\mathbf{x}) = h(\mathbf{x})$ for **all training examples** stop, otherwise go to Step 1.

References

[1] L. Beilina, E. Karchevskii, M. Karchevskii, Numerical Linear Algebra: Theory and Applications, Springer, 2017.

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

[3] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`

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