Statistical Machine Learning
Linear Discriminant and Performance Metrics

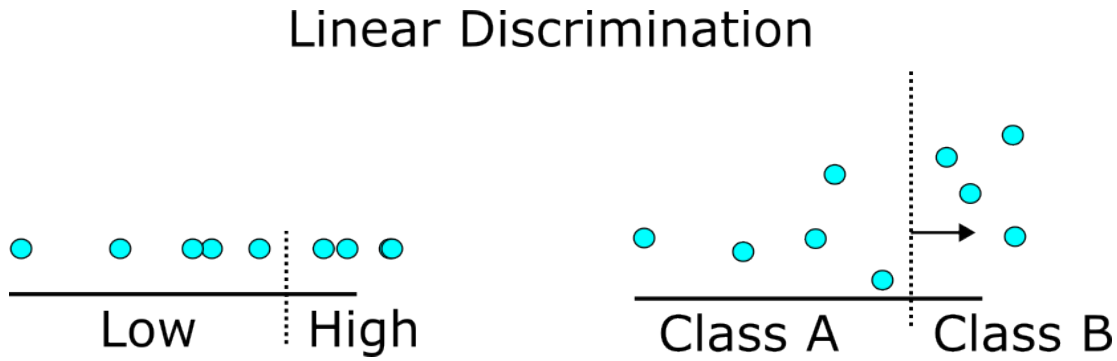
Horacio Gómez-Acevedo
Department of Biomedical Informatics

January 27, 2025

UAMS

Linear Discrimination Motivation

We regularly use thresholds to label data points.



Note that in the 2-dimensional case, we will be looking for a plane (or hyperplane in higher dimensions). One way to simplify finding for a plane is to identify a vector which is normal to the dividing plane.

Linear Discriminant Functions

A real-valued function such as

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 \tag{1}$$

is called a **discriminant function**, where **w** is refer to as **weight vector**, and w_0 is normally called **bias** (not the same as the statistical term). For the discriminant function (1) , we can define a two-category classifier (with classes w_1 and w_2) with the following rule:

- $\mathbf{x} \in w_1$ if $g(\mathbf{x}) > 0$,
- $\mathbf{x} \in w_2$ if $g(\mathbf{x}) < 0$
- if $g(\mathbf{x}) = 0$, then $\mathbf{x}$ can be assigned to either class.

Geometry related to the discriminant function

The equation $g(\mathbf{x}) = 0$ describes the **decision surface** that separates both categories. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are in the decision surface

$$\mathbf{w}^t\mathbf{x}_1 + w_0 = \mathbf{w}^t\mathbf{x}_2 + w_0 \Rightarrow \mathbf{w}^t(\mathbf{x}_1 - \mathbf{x}_2) = 0$$

This implies that the vector $\mathbf{w}$ is normal to the hyperplane H defined (generated) by the decision surface. We can represent

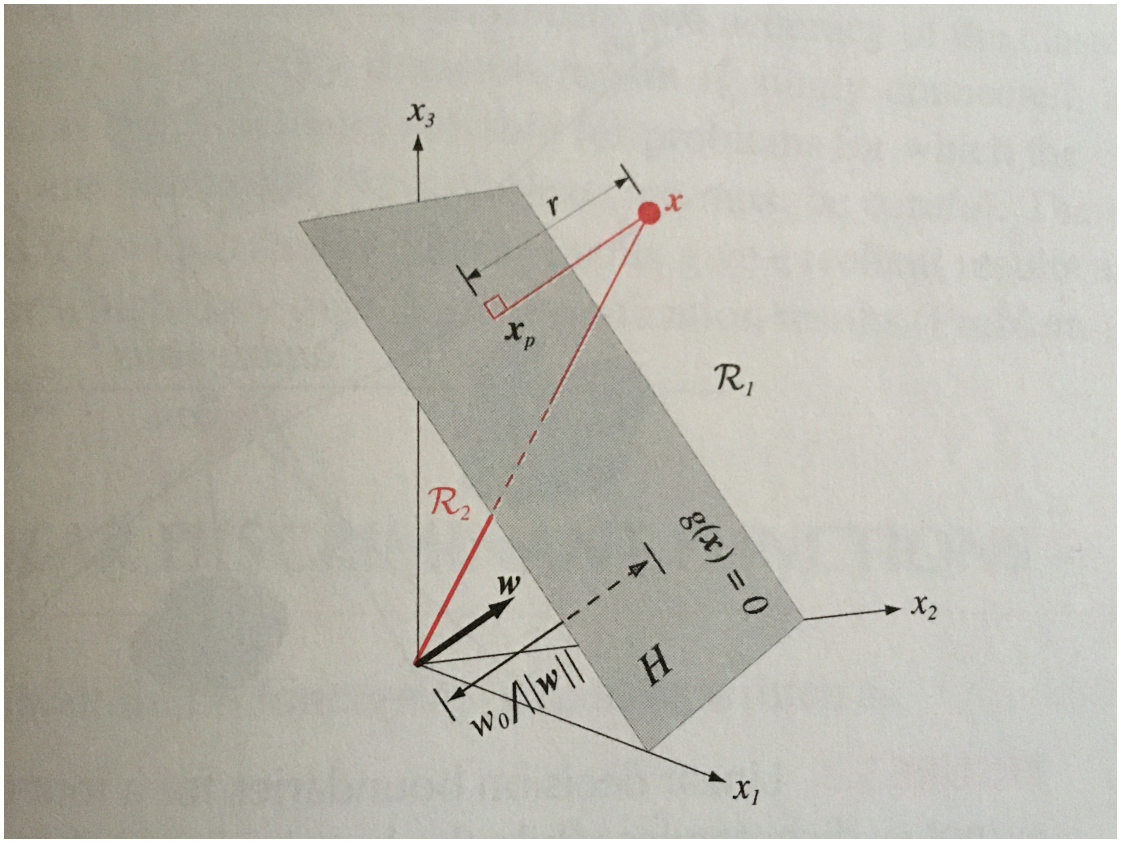
$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{||\mathbf{w}||}$$

where $||\mathbf{w}||$ is the norm of $\mathbf{w}$ (i.e., $||\mathbf{w}||^2 = \mathbf{w} \cdot \mathbf{w}$), $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto H and r is a signed distance (the sign depends on the side of the point $\mathbf{x}$ with respect to the separating plane H). More specifically,

$$r = \frac{g(\mathbf{x})}{||\mathbf{w}||}$$

Discriminant Function

We can visualize these concepts with the following graph



Versatility of the Discriminant Functions

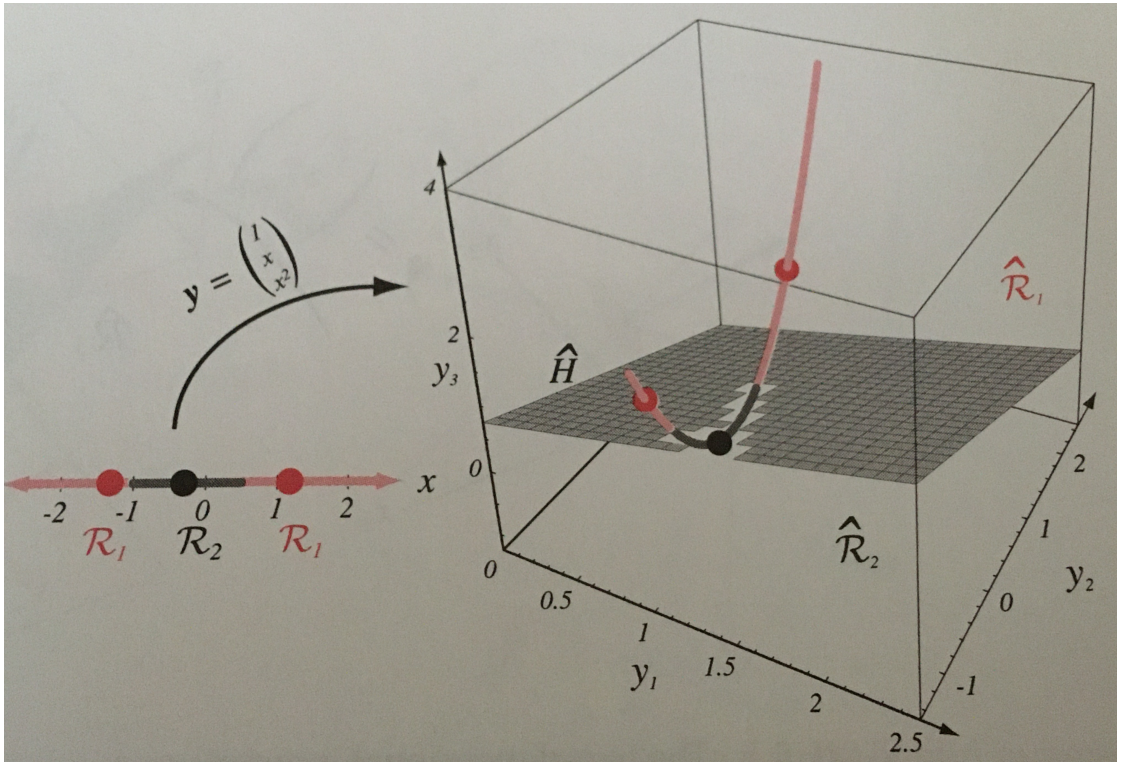
Whereas finding a point (plane or hyperplane) that splits the data into two categories, we can map our dataset to a higher dimension and find a solution.
In the next figure, if we want to separate the interval in black from the intervals marked in red, it is clear that we cannot find a point from which the left will be on one class and right into another. But, if we use a discriminant function

$$g(x) = a_0 + a_1 x + a_2 x^2$$

we can select a plane that splits those datasets.

Discriminant functions (cont)

We can visualize this process with the following figure

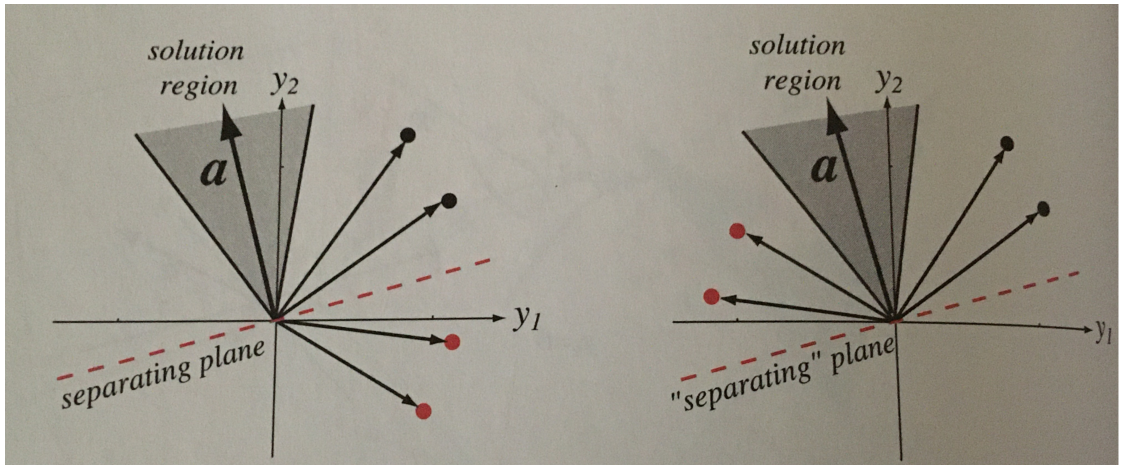


Linearly separable case

Suppose we have a set of n samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$ and some of them are labeled w_1 and some labeled w_2. We want to determine the weights $\mathbf{a}$ in a linear discriminant function $g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$
First, we proceed with some sort of "normalization" which means that we replace all samples labeled w_2 by their negatives. Then the problem reduces to find a vector $\mathbf{a}$ such that

$$\mathbf{a}^t\mathbf{y} > 0$$

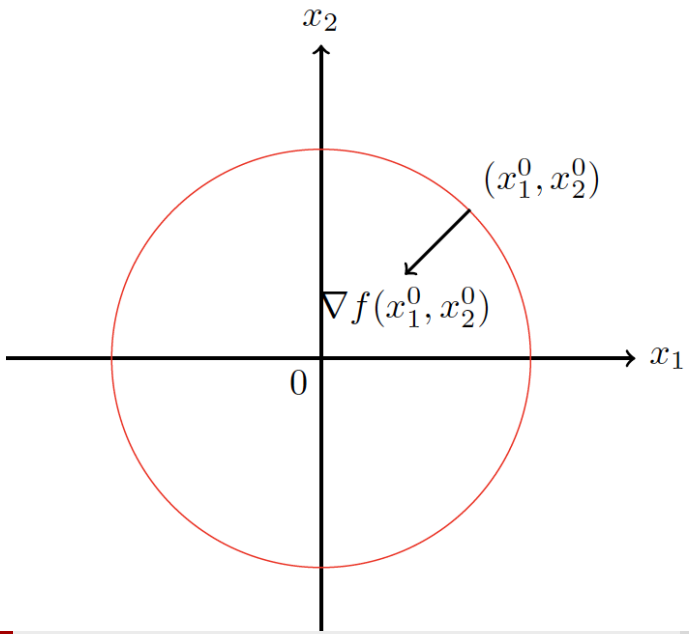
for all of the samples.



Gradient Descent Procedures

We need to find a solution to the set of linear inequalities $\mathbf{a}^t\mathbf{y}_i > 0$. There may be several solutions, to determine the "best" solution is determined by the minimization of a criterion function $J(\mathbf{a})$. The minimum (if exists) is called the "solution vector".

Recall that the gradient of a smooth function points towards the maximum for every point on the level surface. Thus, we need to use the negative of the gradient if we are to find the minimum.



Gradient Descent Procedures (cont)

The basic gradient descend goes as follows:

1. Chose an arbitrary weight vector $\mathbf{a}(1)$
2. Compute $\nabla J(\mathbf{a})(1)$
3. $\mathbf{a}(2)$ is obtained by moving the same distance from $\mathbf{a}(1)$ along the negative of the gradient.

We repeat this process following the formula

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\nabla J(\mathbf{a}(k))$$

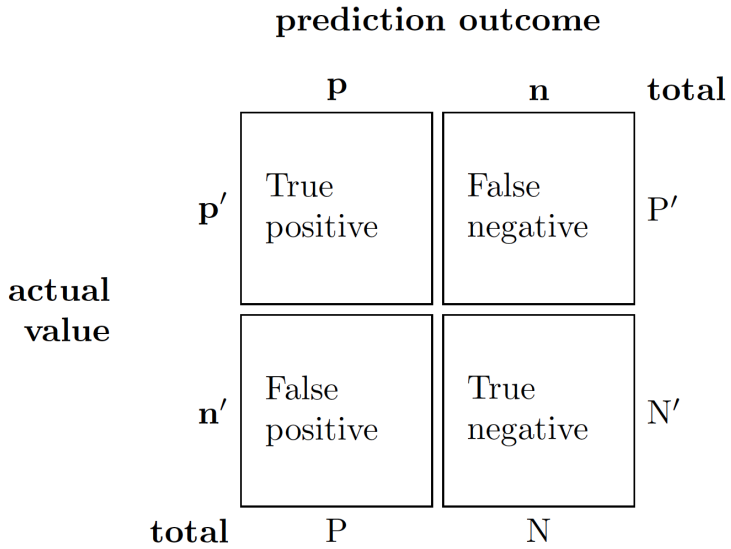
where $\eta > 0$ is called the **learning rate** that sets the step size. We hope that the weight vectors will converge to a solution minimizing $J(\mathbf{a})$

Confusion Matrix

A binary classifier placing instances in two classes w_1 and w_2 can make two types of error:

- it can assign an instance of w_1 into w_2
- it can assign an instance of w_2 into w_1

The **confusion matrix** allows us to summarize this information by putting it into a matrix in which each row represents the *actual* class, and each column represents the *predicted* class.



Useful metrics

Once you have developed your binary classifier there are some common metrics to determine how well your classifier works.
Precision is defined as

$$\text{precision} = \frac{TP}{TP + FP}$$

and the **recall** defined by the following expression

$$\text{recall} = \frac{TP}{TP + FN}$$

We also use another metric called the F_1 score that represents the harmonic mean of the precision and recall.

$$F_1 = \frac{1}{\frac{1}{2}(\frac{1}{\text{precision}} + \frac{1}{\text{recall}})} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Observations about classifier metrics

- False positives are also known as *false alarm* or type I error.
- False negative are also known as *miss* or type II error.
- Recall is also referred to as *sensitivity* or *hit rate*.
- Precision is also called *positive predictive value*.
- In theory, you would like that precision and recall to be equal to 1. But this is not possible due to the **precision-recall tradeoff**.

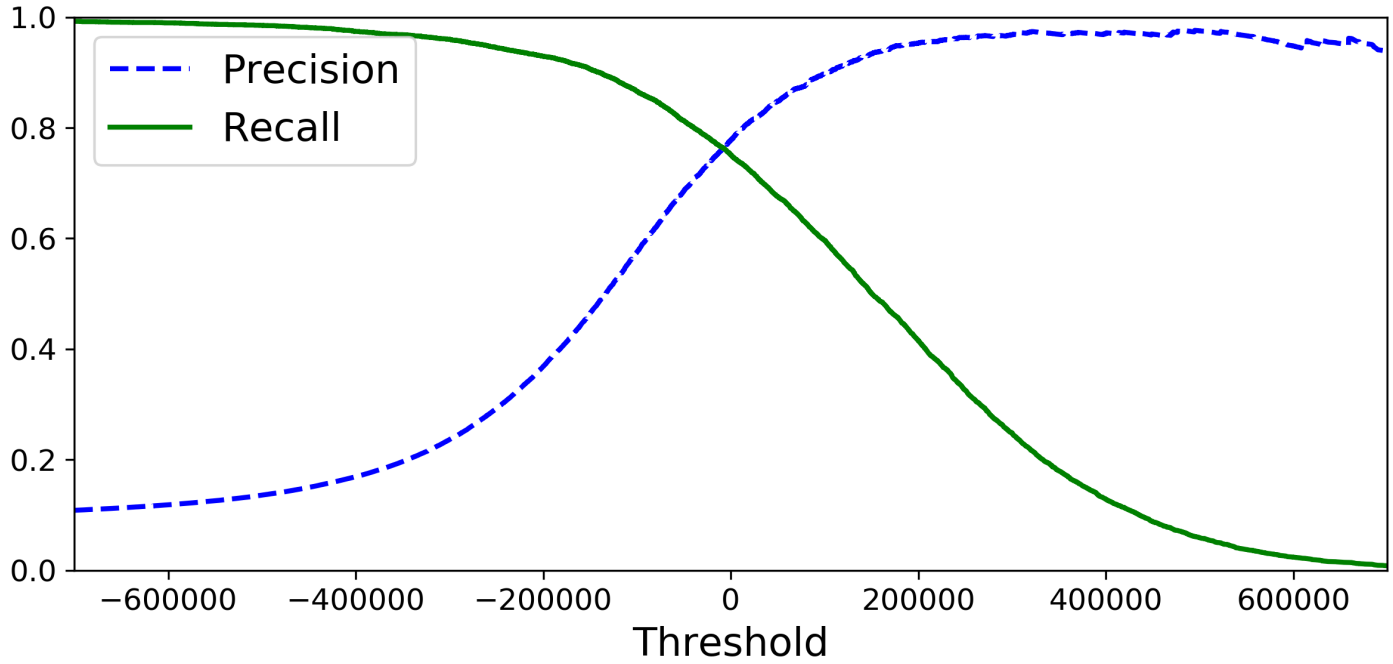
Suppose you are developing a new molecular technique (algorithm) to detect certain type of cancer

Silly Scenario 1 You have tested your tool in vitro. You would like to detect as few false positives samples as possible, thus your precision must be very high while sacrificing recall.

Silly Scenario 2. If you proceed with human samples, a false positive can be financially costly due to the extra cost of further testing, but a false negative may cause harm to patients. Thus, you would prefer higher recall sacrificing precision.

Precision-Recall function

The so-called precision-recall function is obtained when we change the bias in our discriminant function and calculate the precision and recall.



ROC curve

The **receiver operating characteristic curve (ROC)** plots the recall (true positive rate) against the *false positive rate*.

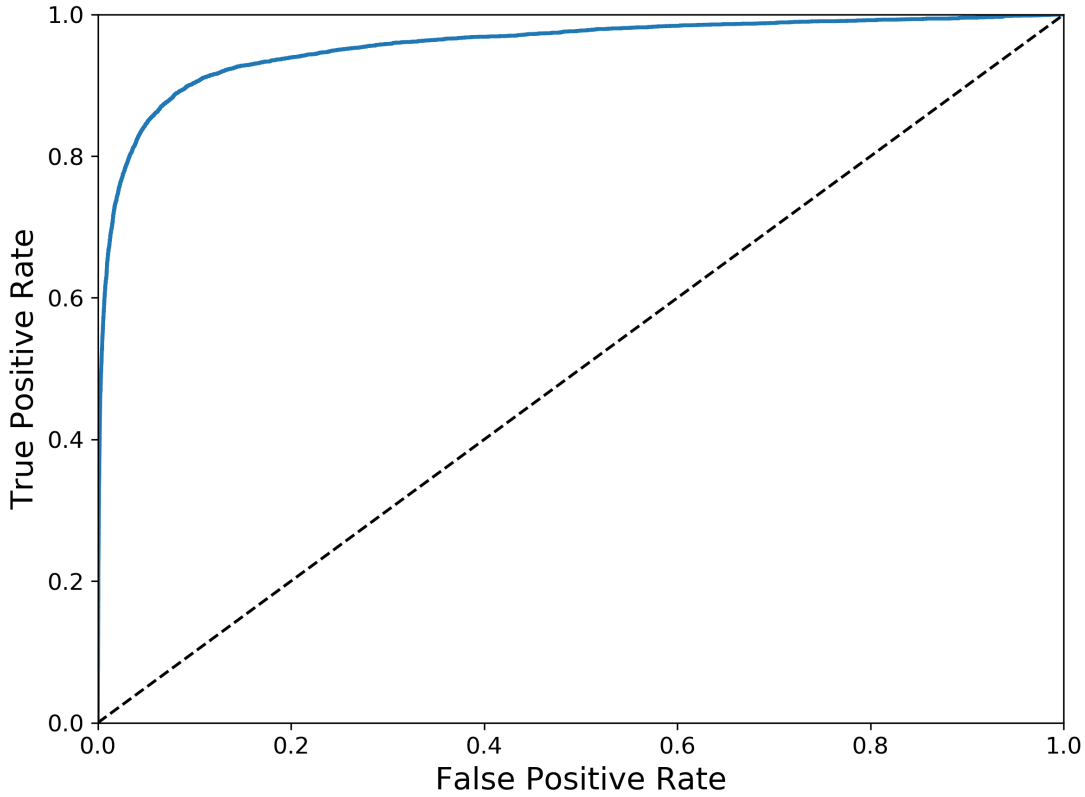
$$FPR = \frac{FP}{FP + TN}$$

whereas the *true negative rate* or specificity

$$TNR = \frac{TN}{FP + TN}$$

Note that $FPR + TNR = 1$, thus the ROC is a curve of the recall vs. 1-specificity, or FPR vs. TPR (if you want to keep everything positive!)

ROC plot



Normally, we calculate the "Area under the ROC curve" (AUROC) to compare our classification methods.

Note If you have ROC curve close to the dotted line, your classifier is not doing better than a random choice.

Multilabel and Multioutput Classification

In cases where you want to classify instances in multiple classes simultaneously, some classifiers (e.g., KNN) allow you to use that type of classification, and it is named (not surprisingly **multilabel classification**.)
Note that the confusion matrix can be extended to multiple classification, whereas some other metrics are for binary classification problems.
On the other hand the **multioutput classification** is similar to a multilabel but each of the labels can have more than two possible values (e.g., discrete output)

What have we learned?

- Linear discriminant functions and their use in classification.
- The definition of a confusion matrix.
- Precision and recall and their trade off.
- Receiver operating characteristic curves (ROC)

References

Materials and some of the pictures are from [2], [1], and [3]

[1] R. O. Duda, P. E. Hart, and D.G. Stork. *Pattern Classification*. 2nd. Edition. John Wiley & Sons, 2001. ISBN: 978-0-471-05669-0.

[2] J. Gareth et al. *An Introduction to Statistical Learning*. 1st edition. Springer, 2015. ISBN: 978-1-4614-7137-0.

[3] A. Géron. *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow. Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. Edition. O'Reilly, 2019. ISBN: 978-1-492-03264-9.

I have used some of the graphs by hacking TiKz code from StakExchange and other old tricks of TEX