

MATHEMATICAL TOOLS FOR DATA SCIENCE

Leila GHARSALLI (leila.gharsalli@ipsa.fr)

IPSA, AERO 4

2023-2024

GOALS OF THE COURSE

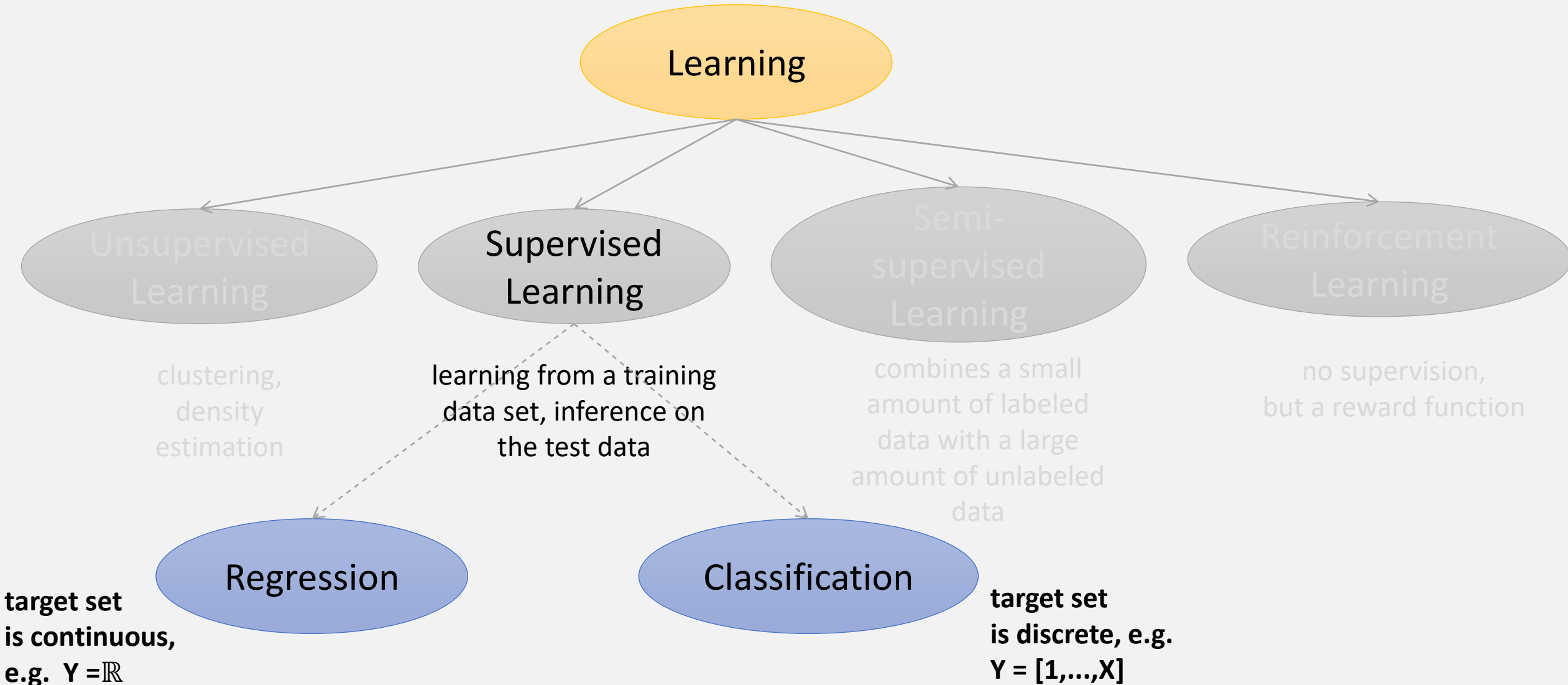
- *Instructor : Leila GHARSALLI*
- *mail to : leila.gharsalli@ipsa.fr*
- Provide a general introduction to data mining and data science.
- Introduce some important tools for solving problems in data science.
- Grading : Participation in class (practical work) as well as a final exam will be graded.
- Main background needed: basic notions in probabilities and statistics, programming skill.

CONTENT OF THE COURSE

1. Introduction
2. Linear regression
3. Sparse regression
4. Classification
5. Using trees for predictive analysis
6. Principal Component Analysis
7. Clustering
8. Density estimation

CLASSIFICATION

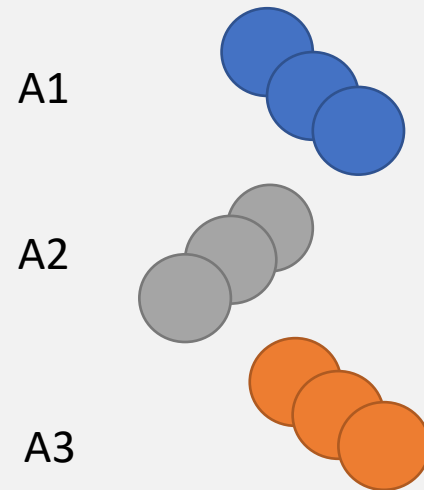
INTRODUCTION



INTRODUCTION

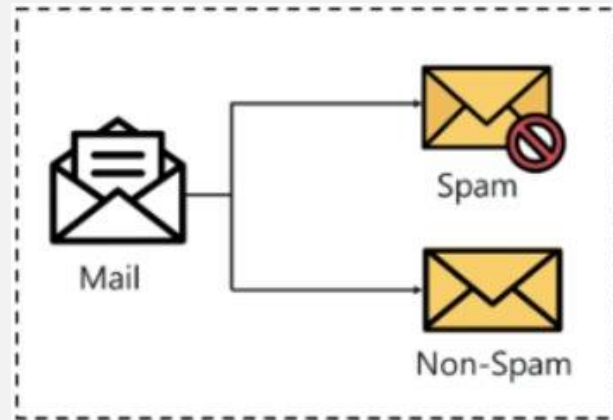
Supervised learning : knowledge of output : learning with the presence with an expert / teacher

- Data is labelled with a class or value
- Goal: predict a class or value label (Neural Network, Support Vector Machine, Decision Trees, Bayesian classifiers...)



INTRODUCTION

Classification



airplane



automobile



bird



cat



deer



dog



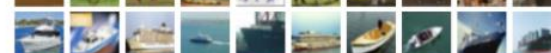
frog



horse



ship



truck



X: input, features, attributes..

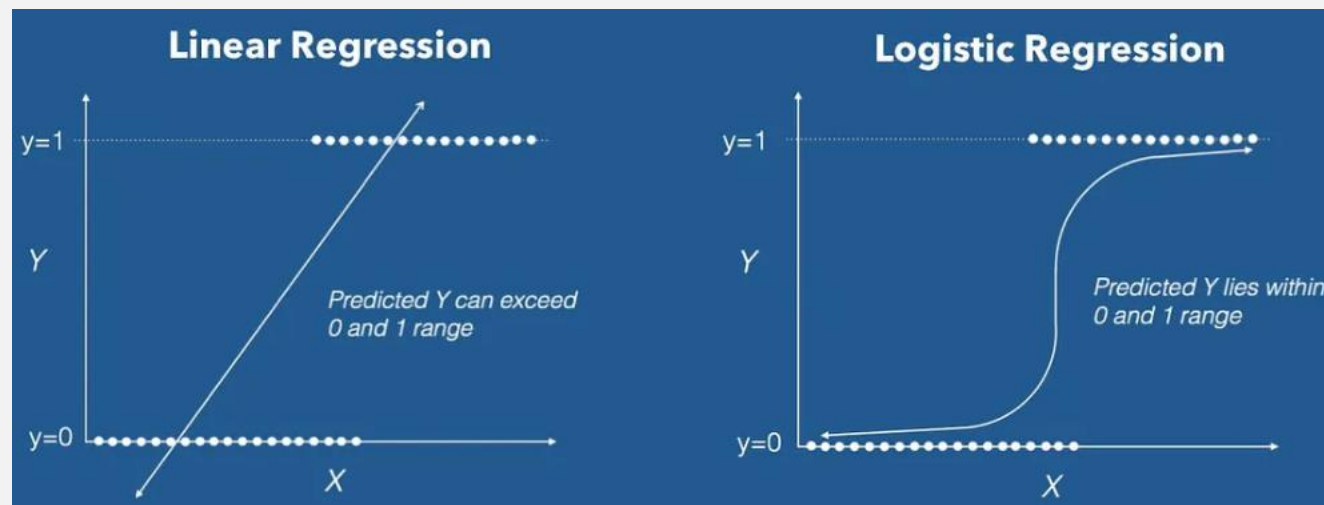
Y: output, labels, predictions..

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|-------------|---------|---------------|---------------|---------|------|--------------------------|-----|---------|
| 0 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 3 | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 4 | 0 | 137 | 40 | 35 | 188 | 43.1 | 2.288 | 33 | 1 |

LOGISTIC REGRESSION

LOGISTIC REGRESSION

- Binary logistic regression is a type of regression analysis where the dependent variable is a dummy variable: coded 0 (no) or 1 (yes).
- Logistic regression (also called logit regression) is commonly used to estimate the probability that an observation belongs to a particular class (for example the probability that an email is spam?).



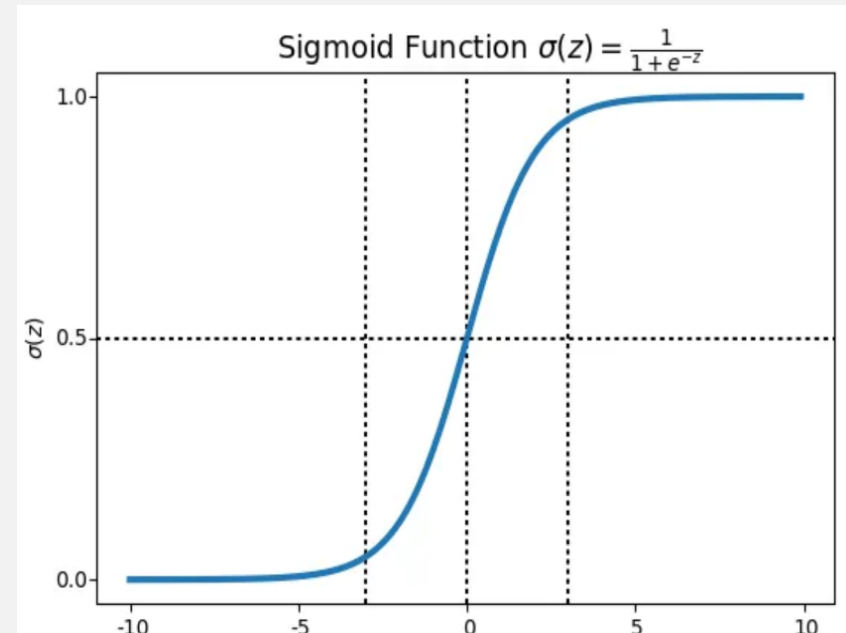
LOGISTIC REGRESSION

- A logistic regression model calculates a **weighted sum $\theta_0 + \sum_{i=1}^n \theta_i x_i$ of input characteristics (weighted sum of explanatory variables + a constant term)**. Instead of providing the prediction \hat{Y} from this weighted sum, logistic regression provides the logistics of the result:

$$\hat{p} = h_{\theta}(x) = \sigma(\theta_0 + \sum_{i=1}^n \theta_i x_i)$$

- With $x = (x_1, x_2, \dots, x_n)$ and σ is the logistic function which is a sigmoid function (in the form of S always between 0 and 1) and which is defined by:

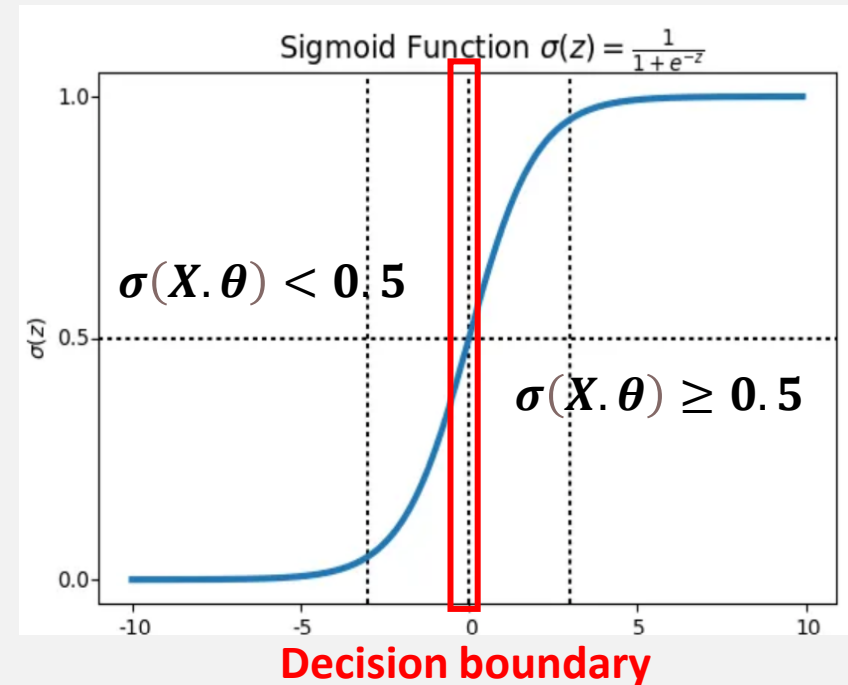
$$\sigma(t) = \frac{1}{1 + \exp(-t)} = \frac{\exp(t)}{1 + \exp(t)}$$



LOGISTIC REGRESSION

- From the sigmoid function, it is possible to define a decision boundary. We define a threshold at 0.5 such that:

$$\begin{cases} y = 0 & \text{si } \sigma(X.\theta) < 0.5 \\ y = 1 & \text{si } \sigma(X.\theta) \geq 0.5 \end{cases}$$



LOGISTIC REGRESSION

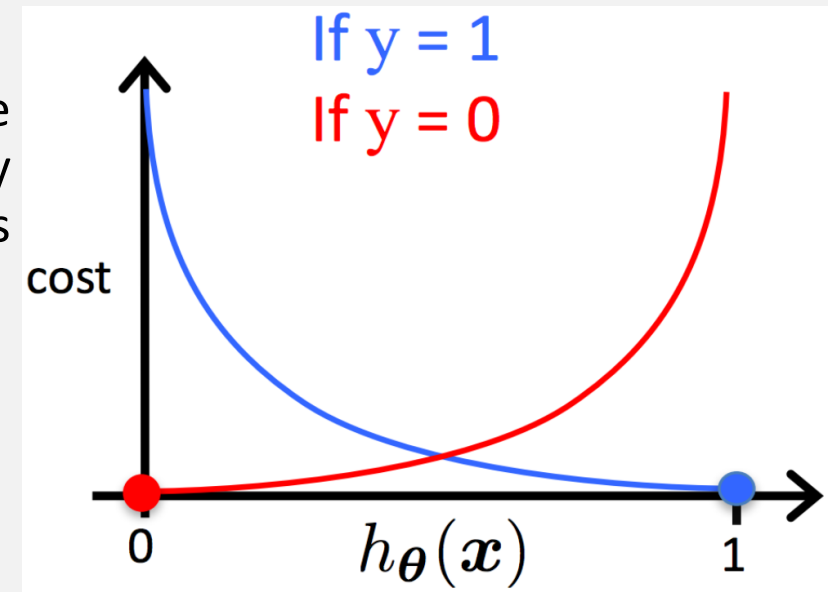
- We know how a logistic regression model estimates the probabilities and makes its predictions. But how is he trained? In other words, what is the mechanism for estimating the regression coefficients $\theta = (\theta_0, \theta_1, \dots, \theta_n)$ in the case of a logistic regression?
- The goal of training is to define the vector of parameters so that the model estimates high probabilities for positive observations ($y = 1$) and low probabilities for negative observations ($y = 0$).

LOGISTIC REGRESSION

- **Cost function:**

We use the logarithm function to transform the sigma function into a convex function by separating the cases where $y = 1$ from the cases where $y = 0$.

$$\begin{aligned} \text{cost} &= J(\theta) \\ &= -\frac{1}{n} \sum_{i=1}^n [Y_i \log(\hat{p}_i) + (1 - Y_i) \log(1 - \hat{p}_i)] \\ \hat{p} &= h_{\theta}(x) = \sigma(\theta_0 + \sum_{i=1}^n \theta_i x_i) \end{aligned}$$



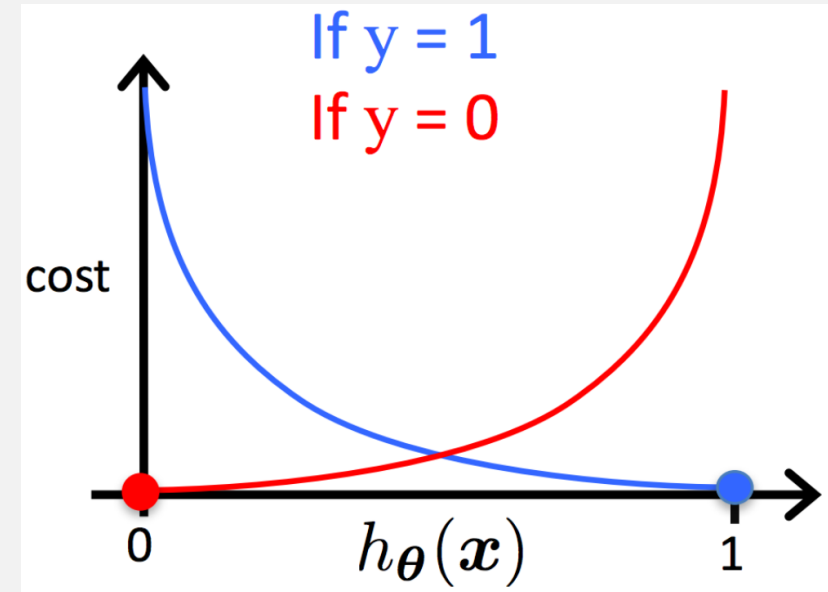
LOGISTIC REGRESSION

- **Cost function minimization:**

There is no known analytical solution to calculate the value of what minimizes the cost function as in the case of a simple / multiple linear regression → We then use the logarithm function to transform the sigma function into a convex function by separating cases where $y = 1$ and $y = 0$.

If $y = 1$; $J(\theta) = -\log(h_{\theta}(x))$

If $y = 0$; $J(\theta) = -\log(1 - h_{\theta}(x))$



LOGISTIC REGRESSION

- **Cost function minimization:** Thanks to the convexity of the cost function The gradient descent method makes it possible to find the minimum of the latter.
- Iteration 0: initialization of the vector $\theta = (\theta_0, \theta_1, \dots, \theta_n)$
- Iterate until convergence:

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \text{ pour } j = 0, \dots, n$$

The parameter that allows modulate the correction (α too much low; slow convergence, α too high; oscillation)

The "gradient", generalization multidimensional derivative [if only one parameter], Indicates the direction and importance of slope in the vicinity of θ_j

LOGISTIC REGRESSION

Model quality: Confusion matrix

- A privileged approach to evaluate the quality of the model would be to compare the predicted values with the true values taken by Y .
- This is the role of the confusion matrix (classification table). It always compares the observed values of the dependent variable with those which are predicted, then counts the good and bad predictions.
- The main advantage of this method is that it makes it possible to compare any classification method and thus select the one that proves to be the most efficient in the face of a given problem.

| | Actually Positive (1) | Actually Negative (0) |
|------------------------|-----------------------|-----------------------|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |

$$\text{Erreur} = \frac{F_p + F_n}{F_p + F_n + V_p + V_n}$$

```
from sklearn.metrics import confusion_matrix
```

[https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)

[learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)

LOGISTIC REGRESSION

Pros:

- Known to have estimated good probabilities.
- Linear separations.

Cons:

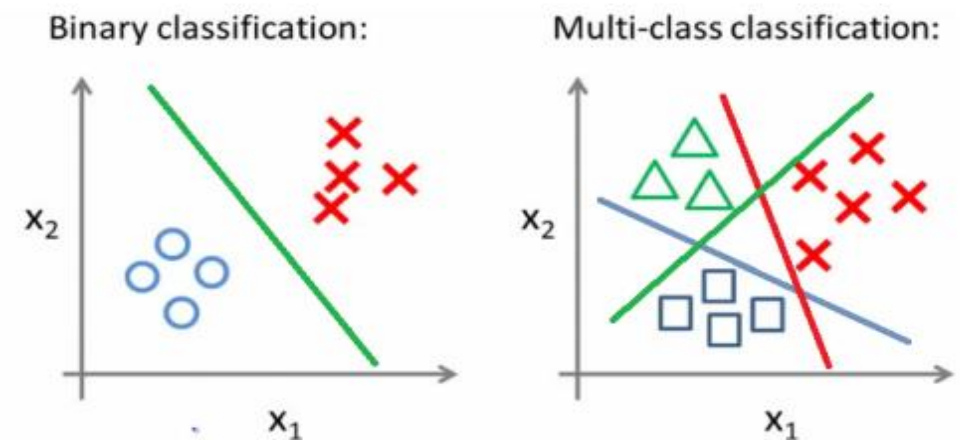
- Easier binary classification.
- More complex optimization problem (calculation time).
- In practice, the multi-class framework is sometimes managed by the technique of “one against all” and not by the logistical case ...

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

MULTI-CLASS CLASSIFICATION

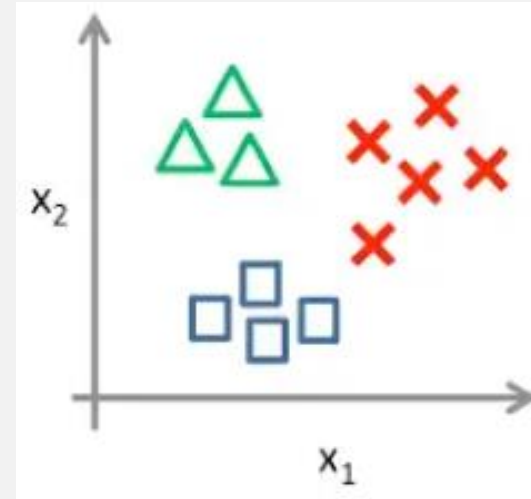
MULTI-CLASS CLASSIFICATION

- There is a variant of logistic regression called Multi-class classification which allows training a model from a vector \mathbf{Y} with \mathbf{I} modalities or classes.
- The **one-versus-all** algorithm, the principle of which is to transform the multi-class problem into several binary problems, is the most used in statistical learning.



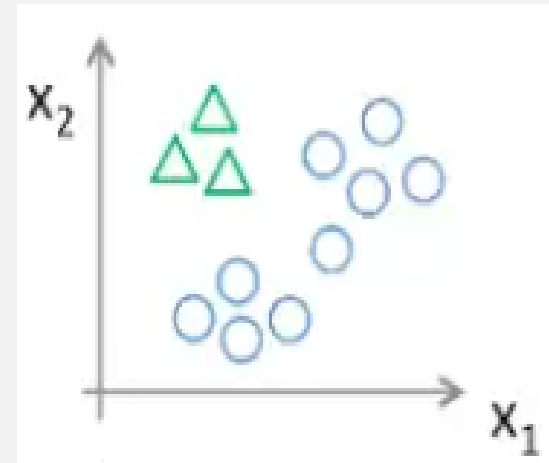
MULTI-CLASS CLASSIFICATION

- Suppose we want to sort the input data into 3 categories:
- class 1 (\triangle)
- class 2 (\square)
- class 3 (\times)
- We can turn this problem into 3 binary classification problems (i.e., where we only predict $y \in \{0,1\}$) to be able to use classifiers such as logistic regression.



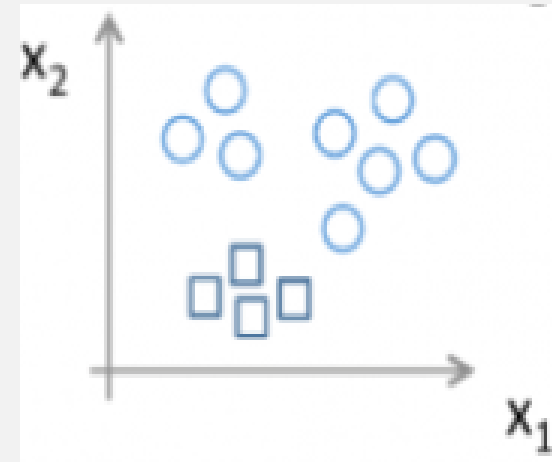
MULTI-CLASS CLASSIFICATION

- We take the values of one class and turn them into positive examples, and the rest of the classes into negatives.
- **Step 1:** the triangles are positive, and the rest are negative - and we run a classifier on them. We calculate $h_{\theta}^{(1)}(x)$.



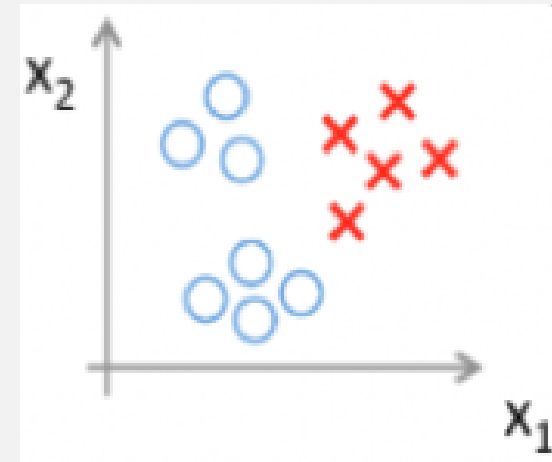
MULTI-CLASS CLASSIFICATION

- Then we do the same with the squares: we make them positive and the rest - negative
- **Step 2:** We calculate $h_{\theta}^{(2)}(x)$.

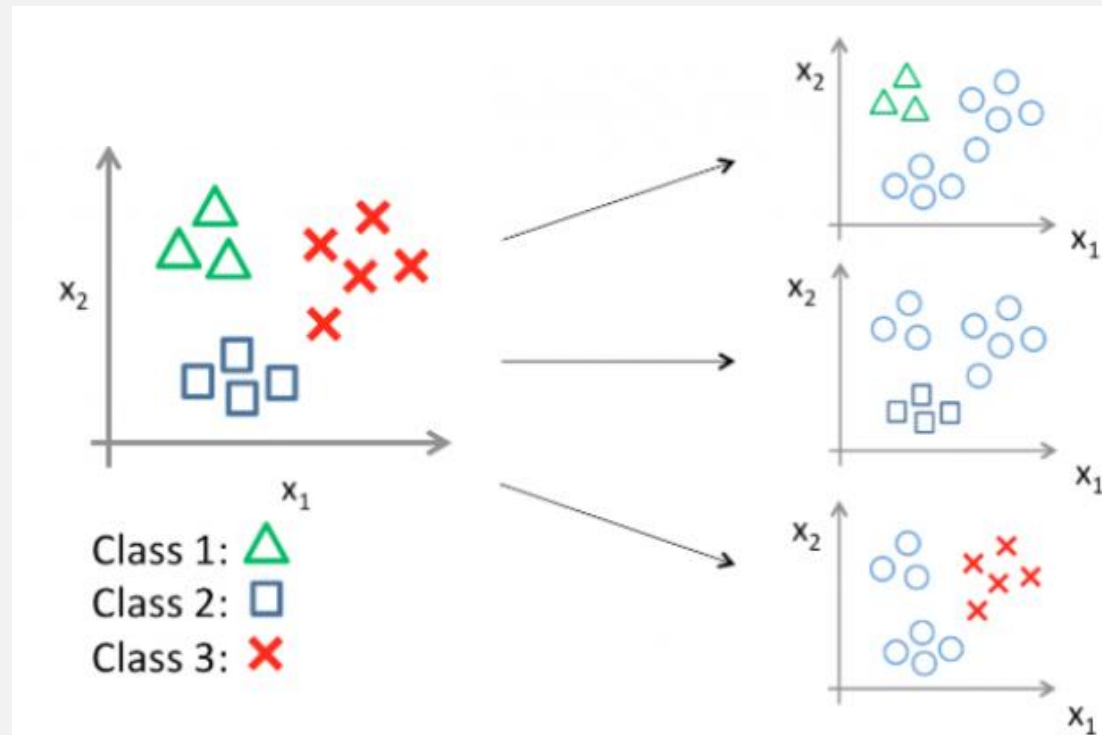


MULTI-CLASS CLASSIFICATION

- Finally, we make the class 3 (x) positive, and the rest of the classes are negative.
- **Step 3:** We calculate $h_{\theta}^{(3)}(x)$.



MULTI-CLASS CLASSIFICATION



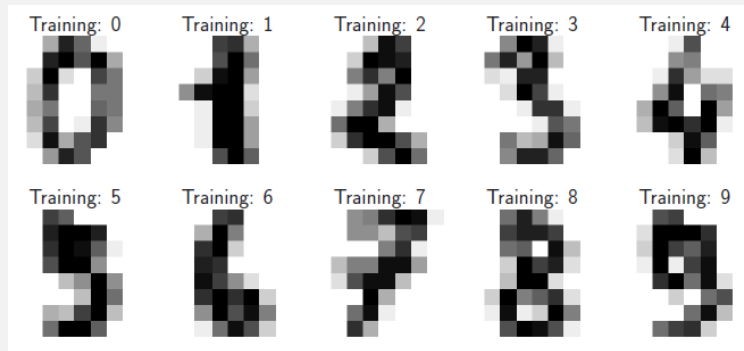
Thus, we have 3 classifications

$$\mathbf{h}_{\theta}^{(i)}(\mathbf{x}) = \mathbf{P}(\mathbf{y} = i | \mathbf{x}; \theta), i = 1, 2, 3.$$

And having calculated $\mathbf{h}_{\theta}(\mathbf{x}) = [\mathbf{h}_{\theta}^{(1)}(\mathbf{x}), \mathbf{h}_{\theta}^{(2)}(\mathbf{x}), \mathbf{h}_{\theta}^{(3)}(\mathbf{x})]$, we chose $\max_i \mathbf{h}_{\theta}^{(i)}(\mathbf{x})$.

MULTI-CLASS CLASSIFICATION

- Classify digitized figures / digits,

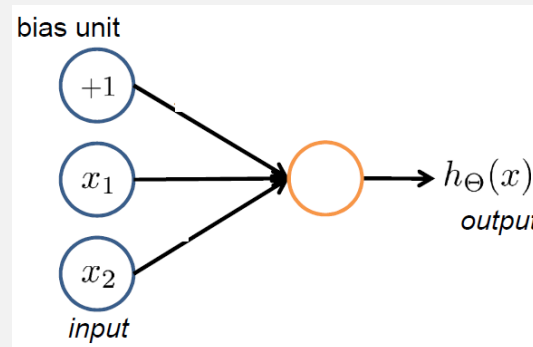


- Classify objects in images,
- Classify texts by theme,
- Classify animal / plant species.

SINGLE NEURONE PERCEPTRON

SINGLE NEURONE PERCEPTRON

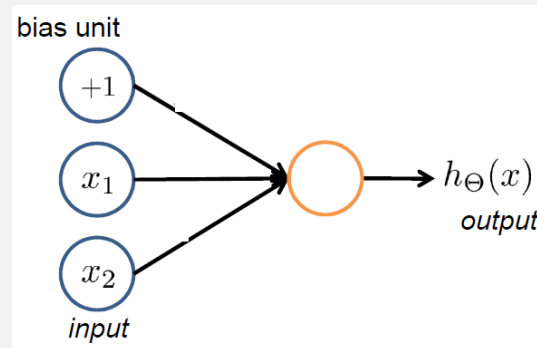
- The perceptron is a single-layer neural network.
- The perceptron is formed of a **first layer of units** (or neurons) which allow to **“read” the data**: each unit corresponds to one of the input variables.



- We can add a **bias unit** which is always activated (**it transmits 1 whatever the data**). These units are connected to a single output unit, which receives the sum of the units connected to it, weighted by connection weights.

SINGLE NEURONE PERCEPTRON

- These units are connected to a single output unit, which receives the sum of the units connected to it, weighted by connection weights.
- For p variables X_1, X_2, \dots, X_p , the output therefore receives:

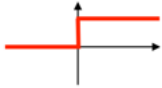
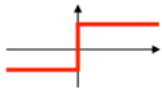

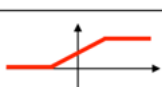
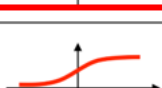

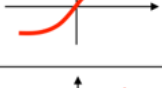



$$h_{\theta} \left(w_0 + \sum_{j=1}^p w_j x_j \right)$$

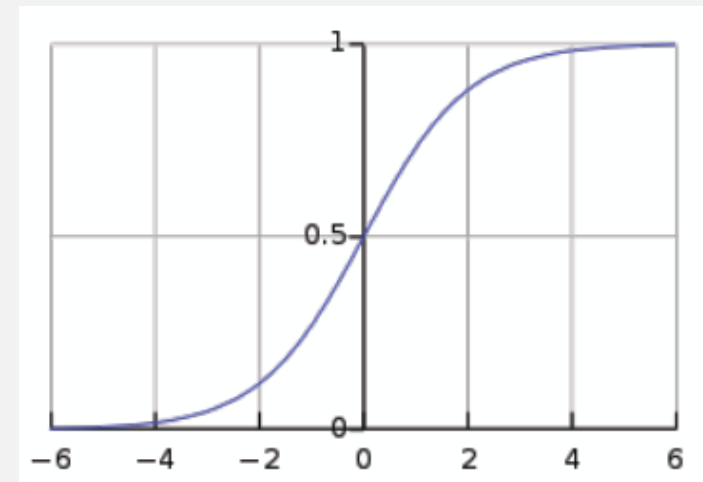
- The output unit then applies an activation function a to this output.

$$y = \text{activation} \left(w_0 + \sum_{j=1}^p w_j x_j \right) = h_{\theta} \left(w_0 + \sum_{j=1}^p w_j x_j \right)$$

SINGLE NEURONE PERCEPTRON

| Activation function | Equation | Example | 1D Graph |
|---|---|---|---|
| Unit step (Heaviside) | $\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Sign (Signum) | $\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Linear | $\phi(z) = z$ | Adaline, linear regression |  |
| Piece-wise linear | $\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$ | Support vector machine |  |
| Logistic (sigmoid) | $\phi(z) = \frac{1}{1 + e^{-z}}$ | Logistic regression, Multi-layer NN |  |
| Hyperbolic tangent | $\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ | Multi-layer Neural Networks |  |
| Rectifier, ReLU (Rectified Linear Unit) | $\phi(z) = \max(0, z)$ | Multi-layer Neural Networks |  |
| Rectifier, softplus | $\phi(z) = \ln(1 + e^z)$ | Multi-layer Neural Networks |  |

Copyright © Sebastian Raschka 2016
(<http://sebastianraschka.com>)

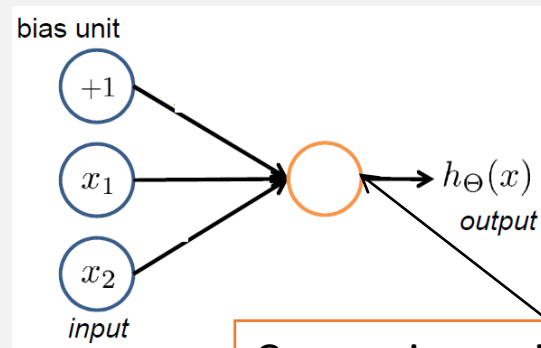


Logistic activation function

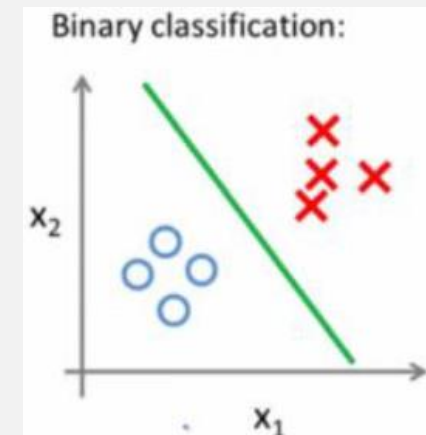
$$h_{\theta}(X) = \frac{1}{1 + e^{-\Theta X}}$$

SINGLE NEURONE PERCEPTRON

$$\text{Threshold } (w_0 + \sum_{j=1}^p w_j x_j) = \begin{cases} 0 & \text{if } (w_0 + \sum_{j=1}^p w_j x_j) \leq 0 \\ 1 & \text{otherwise} \end{cases} \rightarrow \text{One output unit (0 or 1)}$$



Comparison with a Threshold



SINGLE NEURONE PERCEPTRON

- Instead of using a single output unit, it will use as many as classes. Each of these units will be connected to all input units. We will therefore have $K \cdot (p + 1)$ connection weights, where K is the number of classes.

→ K output units e.g., $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

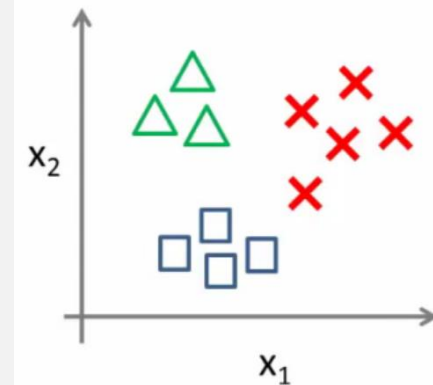
- The **SoftMax function** can then be used as the **activation function**. It is a **generalization of the sigmoid**, which can also be written:

$$\sigma(u) = \frac{e^u}{1 + e^u}$$

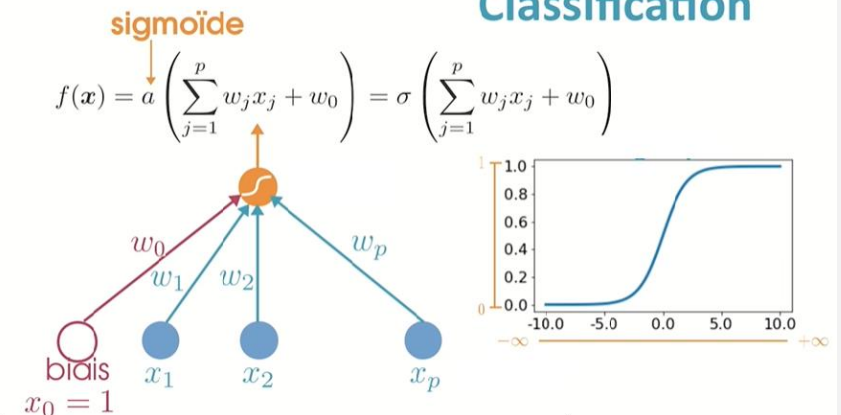
→ So, for K classe, we use **this activation function**:

$$\sigma_k(u_k) = \frac{e^{u_k}}{\sum_{l=1}^K e^{u_l}}$$

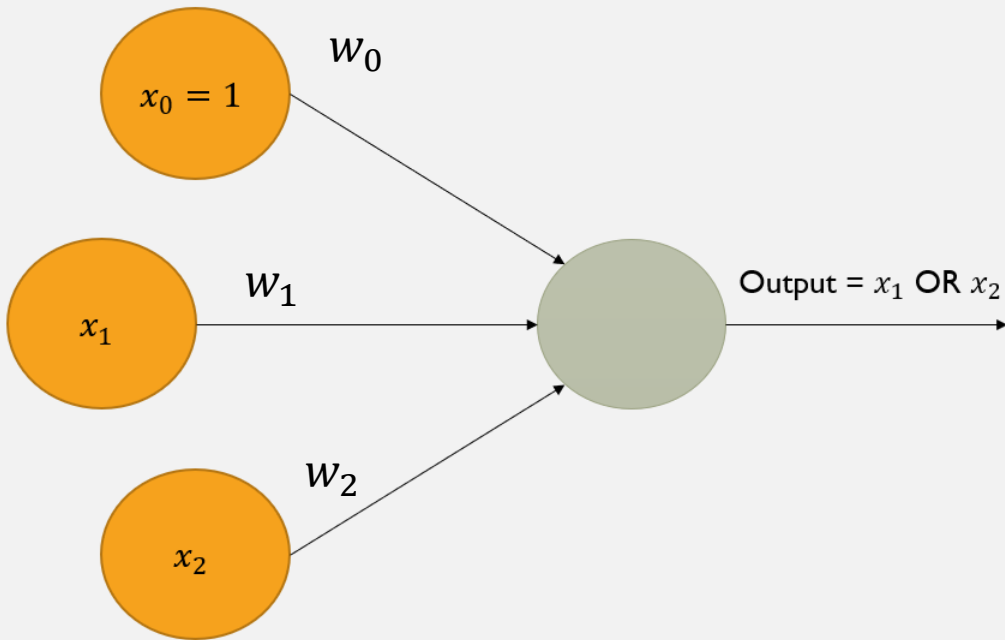
Multi-class classification:



Classification



SINGLE NEURONE PERCEPTRON



| x_1 | x_2 | <i>OR</i> |
|-------|-------|-----------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

The OR gate is 0 only if both inputs are 0.

What are the weights and bias for the OR perceptron?

SINGLE NEURONE PERCEPTRON

From $w_1x_1 + w_2x_2 + bias$, initializing w_1, w_2 as 1 and $bias$ as -1 . So we get:

$$x_1^{(1)} + x_2^{(1)} - 1$$

Passing the first row of the OR logic table ($x_1 = 0, x_2 = 0$), we get:

$$0 + 0 - 1 = -1$$

From the Perceptron rule, if $\left(\sum_{j=1}^p w_j x_j + w_0\right) \leq 0$, then $y = 0$. Therefore, this row is correct.

| x_1 | x_2 | <i>OR</i> |
|-------|-------|-----------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

SINGLE NEURONE PERCEPTRON

Passing the second row of the OR logic table ($x_1 = 0, x_2 = 1$), we get:

$$0 + 1 - 1 = 0 \text{ (} w_1x_1 + w_2x_2 + \text{bias)}$$

From the Perceptron rule, if $\left(\sum_{j=1}^p w_j x_j + w_0\right) \leq 0$, then $y = 0$. Therefore, the OR row is incorrect.

So we want values that will make inputs $x_1 = 0$ and $x_2 = 1$ give y a value of 1. If we change w_2 to 2, we have;

$$0 + 2 - 1 = 1$$

From the Perceptron rule, this is correct for both the row 1 and 2.

| x_1 | x_2 | <i>OR</i> |
|-------|-------|-----------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

SINGLE NEURONE PERCEPTRON

Passing the third row of the OR logic table ($x_1 = 1, x_2 = 0$), we get:

$$1 + 0 - 1 = 0 \quad (w_1x_1 + w_2x_2 + \text{bias})$$

From the Perceptron rule, if $\left(\sum_{j=1}^p w_j x_j + w_0\right) \leq 0$, then $y = 0$. Therefore, the OR row is incorrect.

Since it is similar to row 2, we can just change w_1 to 2, we have;

$$2 + 0 - 1 = 1$$

From the Perceptron rule, this is correct for both the row 1, 2 and 3.

| x_1 | x_2 | <i>OR</i> |
|-------|-------|-----------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

SINGLE NEURONE PERCEPTRON

Passing the fourth row of the OR logic table ($x_1 = 1, x_2 = 1$), we get:

$$2 + 2 - 1 = 3 \text{ (} w_1x_1 + w_2x_2 + \textit{bias} \text{)}$$

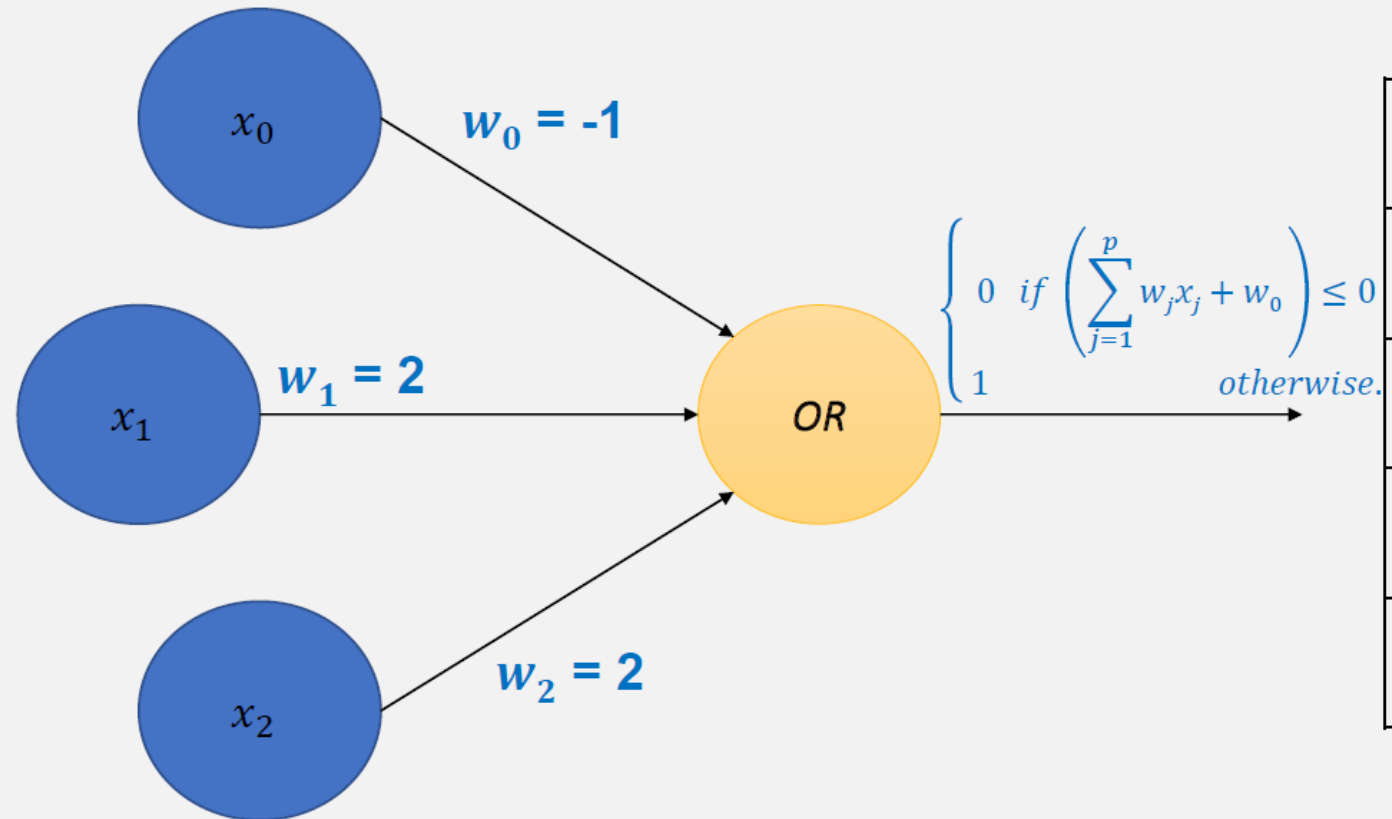
From the Perceptron rule, if $\left(\sum_{j=1}^p w_j x_j + w_0\right) > 0$, then $y = 1$. Therefore, the OR row is valid.

Therefore, we can conclude that the model to achieve an OR gate, using the Perceptron algorithm is:

$$2x_1 + 2x_2 - 1$$

| x_1 | x_2 | <i>OR</i> |
|-------|-------|-----------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

SINGLE NEURONE PERCEPTRON



| x_1 | x_2 | <i>OR</i> |
|-------|-------|-----------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

SINGLE NEURON PERCEPTRON

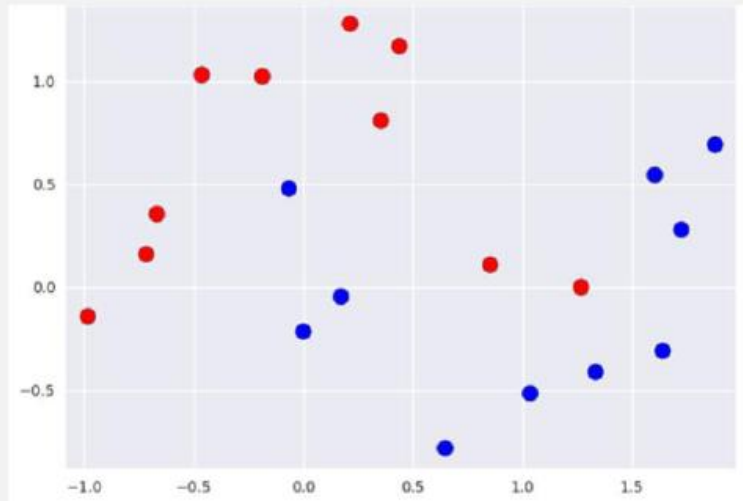
- Perceptron allows to learn parametric models based on a linear combination of variables.
- Perceptron makes it possible to learn **regression models** (the activation function is the identity), **binary classification** (the activation function is the logistic function) or **multi-class classification** (the activation function is the SoftMax function).
- The perceptron is trained by iterative updates of its weights thanks to the gradient algorithm. The same rule for updating weights applies in the case of regression, binary classification or multi-class classification.

K-NEAREST NEIGHBORS

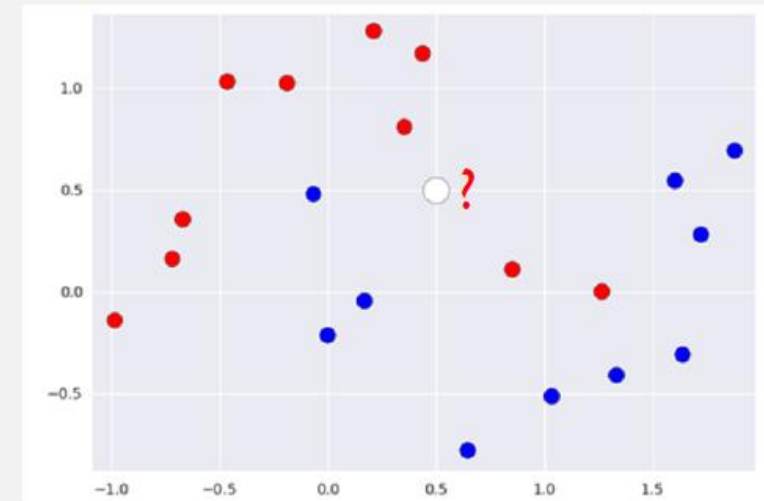
K-NEAREST NEIGHBORS

K-Nearest Neighbors algorithm (K-NN)

It is an algorithm that can be used for both **classification** and **regression**. It is nicknamed "nearest neighbors" because the principle of this model consists in choosing the k data closest to the point studied in order to predict its value.



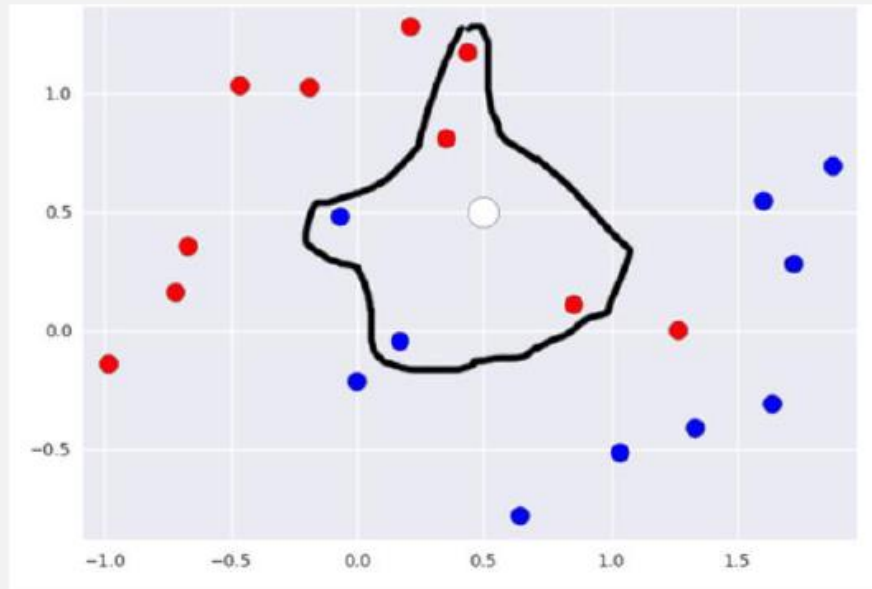
Set of test points



The white point is a new entry,
we want to predict his class

K-NEAREST NEIGHBORS

K-NN algorithm: a special type of algorithm that does not use a statistical model. It is "nonparametric", and it is based only on training data.



The 5 (k=5) closest points to the point we are trying to classify

Which class constitutes the majority of the neighbors?

Using the 5 NN, we can predict that the new data belongs to the **red class** since it has 3 reds and 2 blues around it

The Euclidean distance was used here as a measure of similarity :

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

K-NEAREST NEIGHBORS

K-NN algorithm

Let $(X_1, q_1), (X_2, q_2), \dots, (X_n, q_n)$ be given and $(Y, .)$ observed:

1. Initialize the K-NN with a distance d
2. For each element X_i of the sample, calculate the distance $d(X_i, Y)$
3. Insert if necessary X_i in the K-NN
4. Determine the most represented class C in the K-NN
5. if C is sufficiently represented in K-NN and if $d(Y, "C") < d_{max}$ then:
 - The result is acceptable.
 - Otherwise Y is not classable.

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

K-NEAREST NEIGHBORS

K-NN algorithm: how to chose K?

- K=1:

Very complex boundaries of classes,

Very sensitive to fluctuations in data (high variances),

Risk of over-adjustment,

Not very resistant to noise data.

- K=N:

Rigid boundary,

Less sensitive to noise,

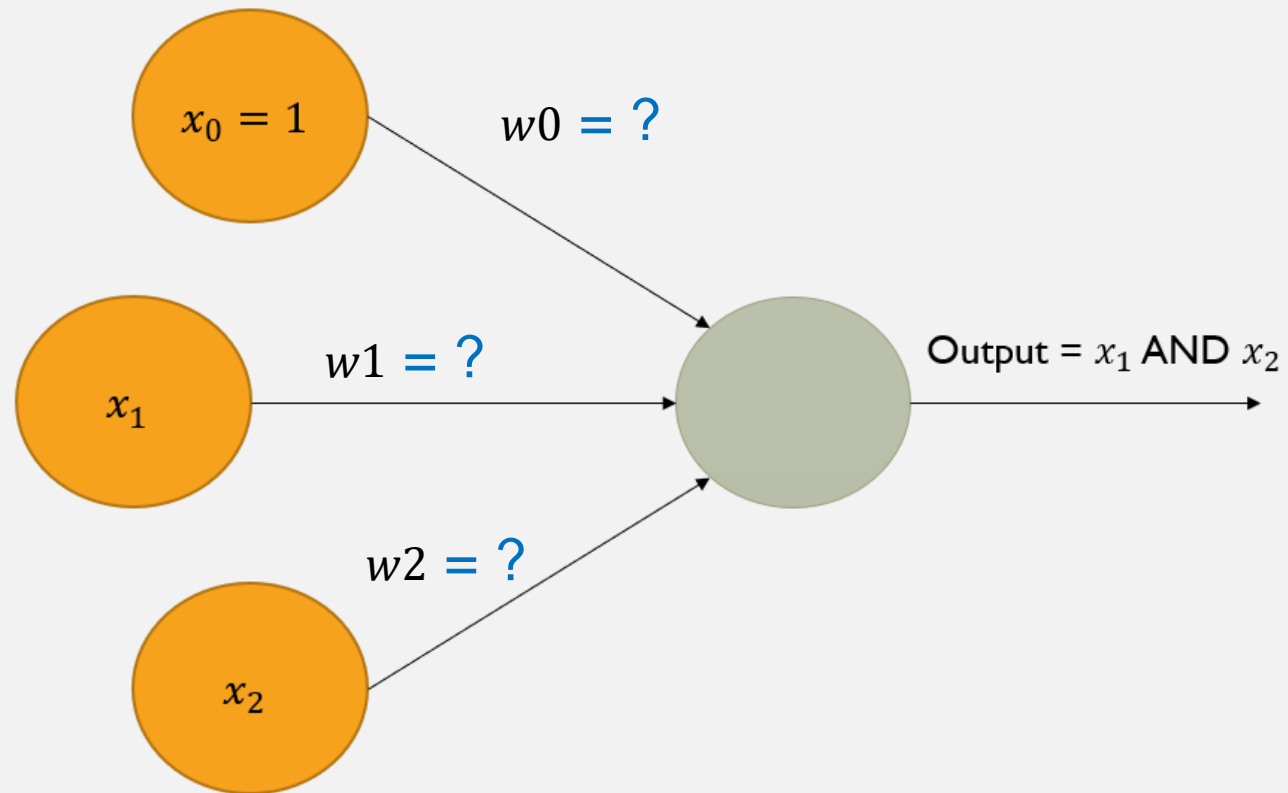
The greater the K-value the better the allocation result.

CONCLUSION

CONCLUSION

- Supervised learning - classification.
- Logistic regression is a very widespread classification algorithm, offering very strong explanatory power due to its linearity. The algorithm predicts the probability of an event occurring by fitting the data to a logistic function.
- The cost function of the logistic regression is based on the log loss, a very important metric that penalizes false positives and false negatives. By construction, logistic regression directly minimizes log loss.
- Relevance of other classification algorithms (perceptron, K nearest neighbors etc,.)

EXERCISE 1



| x_1 | x_2 | <i>AND</i> |
|-------|-------|------------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

EXERCISE 2

We consider a dataset composed of 2 quantitative explanatory variables (x_1, x_2) and a binary variable to predict y . There are 100 training data (data_train.txt) and 200 test data (data_test.txt). To predict the classes of the test data, we will apply the “k nearest neighbors” method.

1. Read the data, divide them between training and testing sets then visualize them with the two different classes.
2. Define the supervised classification model KNN from the *scikit-learn* package, train it, and make the predictions ($K = 30$) before proceeding to the evaluation of the used model.

The evaluation of the model can be done by calculating the error rate on the test set but also by using the score or the confusion matrix.

3. Display the learning error with the *knn.score* function of *scikit-learn* and estimate the confusion matrix.
4. Finally, discuss the complexity of this algorithm and identify the operation that takes the most time.