# Logistic Regression and Gradient Descent

Pattern Recognition and Machine Learning

Aniello Panariello, Lorenzo Bonicelli, Matteo Boschini

October 7th, 2022

University of Modena and Reggio Emilia

# Supervised learning setting

lo tupple degli elementi viene dato.

We are given a training set $\{X_i, Y_i\}_{i=1}^N$, with $X_i \in \mathbb{R}^m$ and $Y_i \in \{0, 1\}$ for each $i = 1, \ldots, N$.

- $N$ is the number of training examples;
- each example $X_i = \{x_i^{(1)}, \ldots, x_i^{(m)}\}$ is a vector of $m$ features;
- each label $Y_i$ is either 0 or 1.

$f: X \to Y$ t.c. funzioni sul training set.

We need to learn a function that maps X to Y such that "it works well on the training set".

PURPOSE $\Rightarrow$ voglio $f$ che presa $x$ sputi la prob. che $y \in X$

We need to learn the parameters $\mathbf{w}$ of a parametric function that maps X to Y such that “it works well on the training set”.

We need to learn the parameters **w** of a parametric function that maps X to Y such that **some error is as low as possible on the training set**.

- Definire MISURA ERRORE
- MINIMIZZARE ERRORE; $\rightarrow$ LOSS FUNCTION
- w si impara con il GRADIENT DESCENT

# Logistic regression classification rule

REGRESSIONE = PRENDE UN NUMERO

Vet. pl.

VOGLIO FUNZIONE che ci

mappi

The function for classification has the following form:

$$F(X_i, w) = \sigma(w^T \cdot X_i), \quad \text{where} \quad \sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

L. REGRESSION NON È MODELLO DI REGRESSIONE MA DI CLASSIFICAZIONE

REGR IN N DIM

$$y_N = \underbrace{\sum_i^N a_i}_{w^T} x_i + b = w^T x_i + b$$

$\rightarrow$ FUNZIONE DI REGRESSION CHE RESTITUISCE UNO SCALARE $M \in \mathbb{R}$ MA NOI VOGLIAMO ASSOCIARLO AD UNA PROBABILITÀ

# Logistic regression classification rule

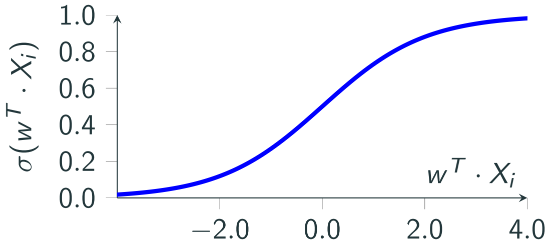
The function for classification has the following form:

RACCHIUDO LA RETTA N-DIM. IN UNA FUNZ. SIGMOIDE CHE RESTITUISCE (0÷1)

$$F(X_i, w) = \sigma(w^T \cdot X_i), \quad \text{where} \quad \sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

(FUNZIONE LOGISTICA)

- $\sigma(x)$ is called **sigmoid function**;

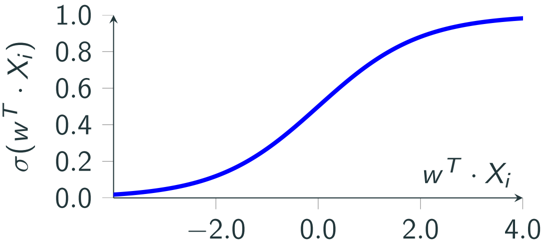


# Logistic regression classification rule

The function for classification has the following form:

$$F(X_i, w) = \sigma(w^T \cdot X_i), \quad \text{where} \quad \sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

- $\sigma(x)$ is called **sigmoid function**;
- $w$ is a vector in $\mathbb{R}^m$, and is called **weight vector**;

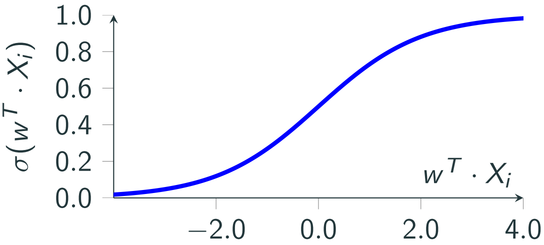


# Logistic regression classification rule

The function for classification has the following form:

$$F(X_i, w) = \sigma(w^T \cdot X_i), \quad \text{where} \quad \sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

- $\sigma(x)$ is called **sigmoid function**;
- $w$ is a vector in $\mathbb{R}^m$, and is called **weight vector**;
- $w$ is initialized randomly, but will improve as training goes.



# Binary crossentropy loss

LE LABEL $y_i$ IDENTIFICA DISTRIBUZIONE LABEL SU TRAINING SET.

During training, we want to **minimize** the following function:

è la mia predizione

$$\mathcal{L}(w) = -\frac{1}{N}\sum_{i=1}^{N}[Y_i \log(\overbrace{F(X_i, w)}) + (1 - Y_i)\log(1 - F(X_i, w))]$$

PROB CLASSE 1

Serve modo x trovare $w$ in MODO che $L(w)$ si minimizzi

$\Rightarrow$ GRADIENT DESCENT

$$L(\theta|D) = \sum^{N} y_i \log(P_{y=1}|x) + (1-y_i)\log(P_{y=0}|x)$$

$$P(y|x) = \frac{P(x,y)}{P(x)} \Longrightarrow \text{[illegible]}$$

$$W_i \to X_i \,, \quad W_i = \underbrace{\{ w_1 \ \dots \ w_m \}}$$

$$L(w)_{X_i} \Rightarrow \frac{\partial L}{\partial w_i} \Rightarrow w_i \quad \min L(w)_{X_i}$$

# Gradient Descent[1]

**Gradient descent** is an iterative optimization algorithm for finding the minimum of a function. How? Take step proportional to the negative of the gradient of the function at the current point.

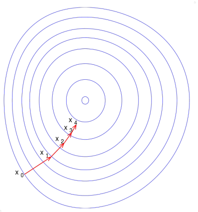



**Figure 1:** Gradient descent on a series of level sets

[1]Credits for this slide: Andrea Palazzi https://github.com/ndrplz/deep_learning_lectures

# Gradient Descent Update[1]



If we consider a function $f(\boldsymbol{\theta})$, the **gradient descent update** can be expressed as:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} f(\boldsymbol{\theta}) \tag{1}$$

$$\theta_{NEW} = \theta_{OLD} - \alpha \frac{\partial}{\partial \theta_{OLD}} f(\theta)$$

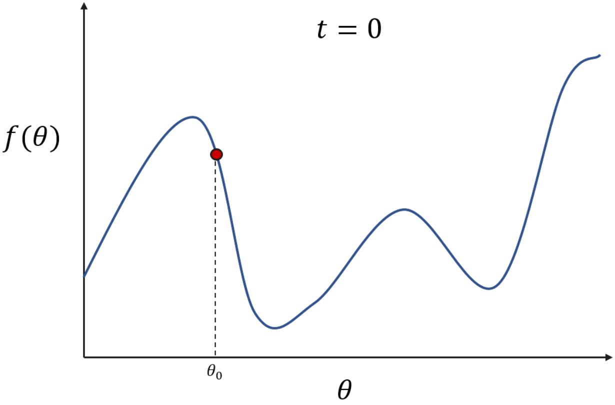
for each parameter $\theta_j$.

The size of the step is controlled by **learning rate** $\alpha$.

[1] Credits for this slide: Andrea Palazzi https://github.com/ndrplz/deep_learning_lectures

# Visualizing Gradient Descent[1]

Gradient Descent for 1-d function $f(\theta)$.



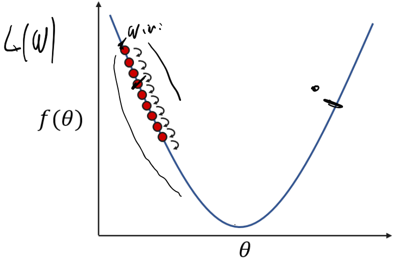

$$\frac{df}{d\theta} < 0$$

$$\theta_{new} = \theta_{old} - \alpha \frac{df}{d\theta}$$

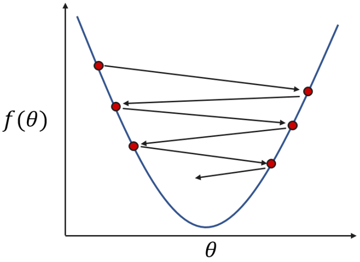
[1]Credits for this slide: Andrea Palazzi https://github.com/ndrplz/deep_learning_lectures

# Learning Rate[1]

Choosing the the right **learning rate** $\alpha$ is essential to correctly proceed towards the minimum. A step *too small* could lead to an extremely *slow* convergence. If the step is *too big* the optimizer could *overshoot* the minimum or even *diverge*.




Learning Rate too small




Learning Rate too big

[1] Credits for this slide: Andrea Palazzi https://github.com/ndrplz/deep_learning_lectures

# Simplify our loss

Back to our problem. We need to take the derivative of this function w.r.t. $w$:

$$
\begin{aligned}
\mathcal{L}(w) &= -\frac{1}{N}\sum_{i=1}^{N}[Y_i \log(F(X_i, w)) + (1 - Y_i)\log(1 - F(X_i, w))] \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[Y_i \log\left(\frac{e^{w^T \cdot X_i}}{1 + e^{w^T \cdot X_i}}\right) + (1 - Y_i)\log\left(\frac{1}{1 + e^{w^T \cdot X_i}}\right)\right] \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[Y_i(w^T \cdot X_i) - Y_i \log\left(1 + e^{w^T \cdot X_i}\right) + (Y_i - 1)\log\left(1 + e^{w^T \cdot X_i}\right)\right] \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[Y_i(w^T \cdot X_i) - \log\left(1 + e^{w^T \cdot X_i}\right)\right]
\end{aligned}
$$

# Derive the loss function



Back to our problem. We need to take the derivative of this function w.r.t. $w$:

$$\mathcal{L}(w) = -\frac{1}{N}\sum_{i=1}^{N}\left[Y_i(w^T \cdot X_i) - \log\left(1 + e^{w^T \cdot X_i}\right)\right]$$

SOMMA TUTTI
GLI ERRORI E
NO FA LA MEDIA
SE $W = 1$

$$W = \begin{bmatrix} 1_1 \\ \vdots \\ 1_m \end{bmatrix}$$

GRADIENTE X
IL PESO
J-ESIMO

$$\frac{\delta \mathcal{L}(w)}{\delta w_j} = -\frac{1}{N}\sum_{i=1}^{N}\left[Y_i x_i^{(j)} - \frac{e^{w^T \cdot X_i}}{1 + e^{w^T \cdot X_i}} x_i^{(j)}\right]$$

$$= -\frac{1}{N}\sum_{i=1}^{N}\left[Y_i - \frac{e^{w^T \cdot X_i}}{1 + e^{w^T \cdot X_i}}\right] x_i^{(j)}$$

$$= -\frac{1}{N}\sum_{i=1}^{N}\Big[Y_i - F(X_i, w)\Big] x_i^{(j)}$$

dove m è il
numero di
FEATURES

# Final gradient and update

Back to our problem. We need to take the derivative of this function w.r.t. $w$:

il numero di parametri w dipende dalle features degli N sample Xi

$$\frac{\delta \mathcal{L}(w)}{\delta w} = \begin{bmatrix} \frac{\delta \mathcal{L}(w)}{\delta w_1} \\ \frac{\delta \mathcal{L}(w)}{\delta w_2} \\ \vdots \\ \frac{\delta \mathcal{L}(w)}{\delta w_m} \end{bmatrix} = \begin{bmatrix} -\frac{1}{N}\sum_{i=1}^{N}(Y_i - F(X_i, w))x_i^{(1)} \\ -\frac{1}{N}\sum_{i=1}^{N}(Y_i - F(X_i, w))x_i^{(2)} \\ \vdots \\ -\frac{1}{N}\sum_{i=1}^{N}(Y_i - F(X_i, w))x_i^{(m)} \end{bmatrix} = -\frac{X^T \cdot (Y - F(X, w))}{N}$$

AVRÒ UN GRADIENTE X OGNUNO DEGLI m Wi $i: 1 \cdots M$

We will update the vector $w$ accordingly:

$$w \leftarrow w - \alpha \frac{\delta \mathcal{L}(w)}{\delta w}$$

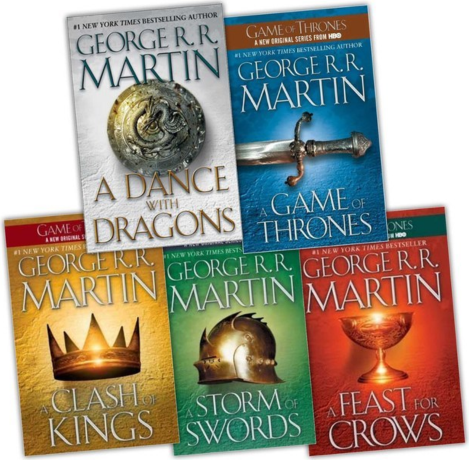
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

**Algorithm 1** pseudocode for training

$X, Y \leftarrow load\_training\_data()$
set learning rate $\alpha$ $\rightsquigarrow 10^{-3} \sim 10^{-4}$
initialize $w$ randomly
**for** $e = 1$ to $\underline{number\_of\_training\_steps}$ **do**
    compute the prediction according to the current weights $F(X, w) = \begin{bmatrix} F(x_1, w) \\ \vdots \\ F(x_N, w) \end{bmatrix}$
    compute the loss function $\mathcal{L}(w) = \Theta \;/\; w = 1$
    compute the derivative of the loss function w.r.t. weights $\frac{\delta \mathcal{L}(w)}{\delta w}$
    update the weight vector $w \leftarrow w - \alpha \frac{\delta \mathcal{L}(w)}{\delta w}$
**end for**

# Today's case study






- We want to predict if a character is alive or dead;
- (Some) of our features are:
  - male or female;
  - married or not;
  - number of deaths witnessed;
  - number of dead relatives;
  - ... and many more.