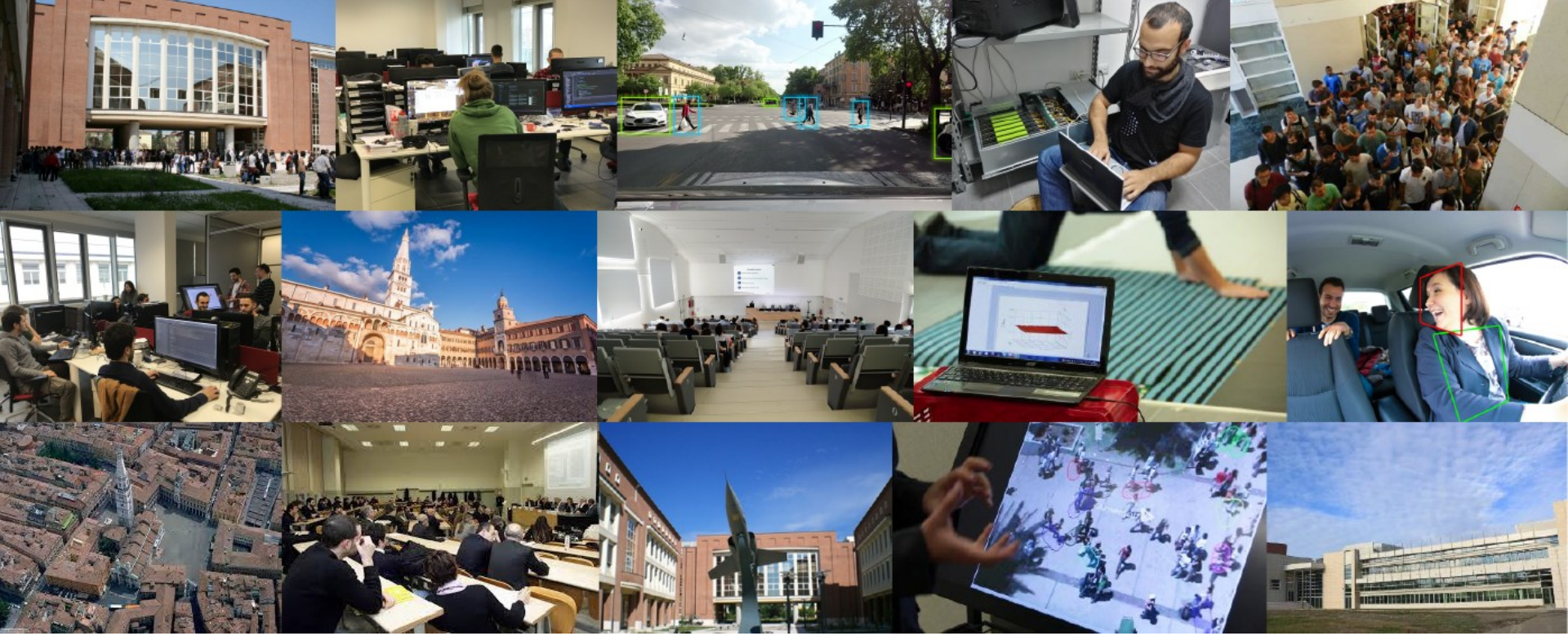


MACHINE LEARNING
Classificazione -

AImage Lab

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Notazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

x_j, j = 1, …, d → variabile indipendente/feature

$\mathbf{x} = [x_0, ..., x_d]$ → feature vector

y → variabile target

$(\mathbf{x}^{(i)}, y^{(i)})$, i = 1, …, n → training sample

θ → vettore o matrice dei parametri

Y → variabile aleatoria

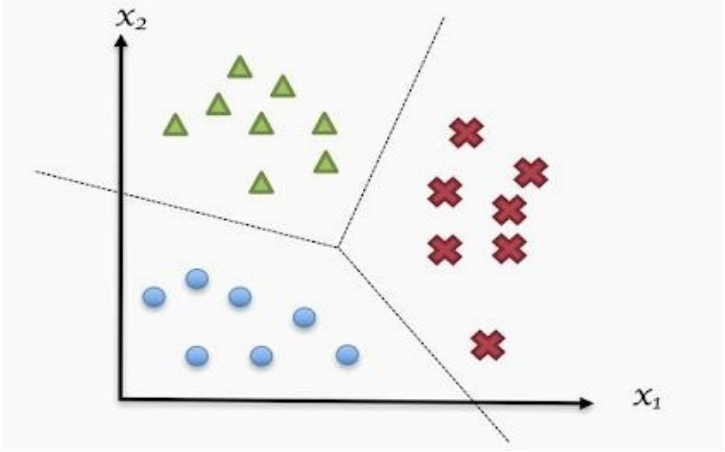
X → vettore di uno o più variabili aleatorie

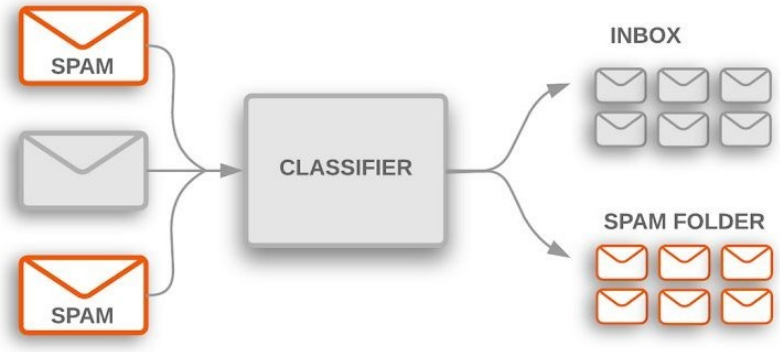
$P(Y = y \mid X = \boldsymbol{x}) = P(y \mid \boldsymbol{x})$ → probabilità condizionata che *Y* assuma il valore costante *y* data la conoscenza che X corrisponda al valore costante ***x***

Classificazione

Classificazione: Costruire un modello/funzione ipotesi che predica il valore di una varibile target di natura **categorica**

- Esempi:
  - Classificare le email come spam/non spam
  - Classificare un'immagine in base all'oggetto principale che contiene (e.g., un cane, un gatto, un'auto…)
- Tipicamente, la variabile target (y), categorica, viene rappresentata in maniera numerica discreta





Classificazione Binaria

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

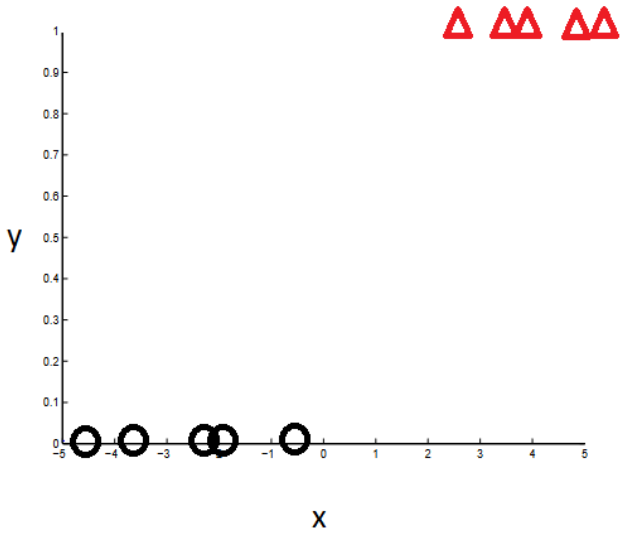
Consideriamo anzitutto il caso della **classificazione binaria**, in cui $y \in \{0,1\}$

Ad esempio, $y = 1$ indica una mail di spam, e $y = 0$ una mail non-spam

Ora che y assume valori numerici, possiamo usare la Linear Regression per predirne il valore?

Esempio 1D

Un semplice esempio di dataset di training, con una sola variabile indipendente (*x*), è mostrato nella figura qui sotto, dove cerchi e triangoli rappresentano le due classi e dove ho (arbitrariamente) scelto di usare *y = 1* per i triangoli e *y = 0* per i cerchi



Esempio 1D

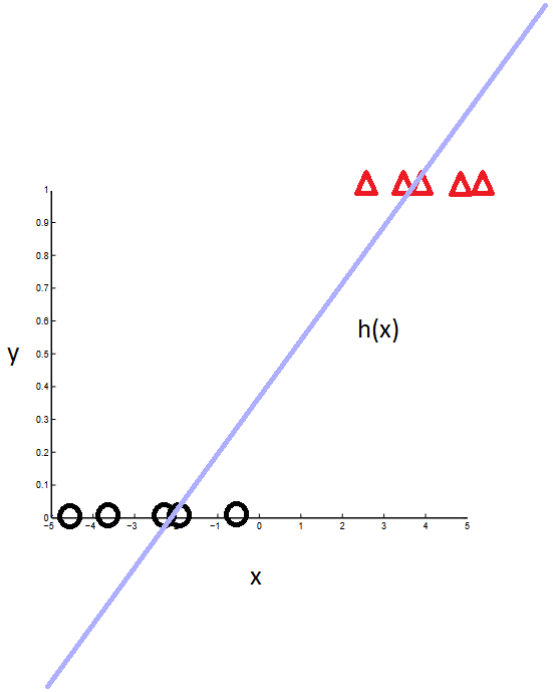
Se usassimo la Linear Regression, il risultato (cioè, la funzione di predizione *h(x)*) sarebbe qualcosa di simile alla figura accanto

h(x) assume valori in *R*, mentre la ground truth può assumere solo due valori binari ($y \in \{0,1\}$)

In particolare, per un dato *x* potremmo avere *h(x) > 1* oppure *h(x) < 0*

Se vogliamo invece che *h(x)* rappresenti un valore di *probabilità* dobbiamo limitarla superiormente e inferiormente

Più precisamente, vorremmo che la funzione di predizione rappresenti la probabilità condizionale: *0 <= P(Y = 1| X = x) <= 1*



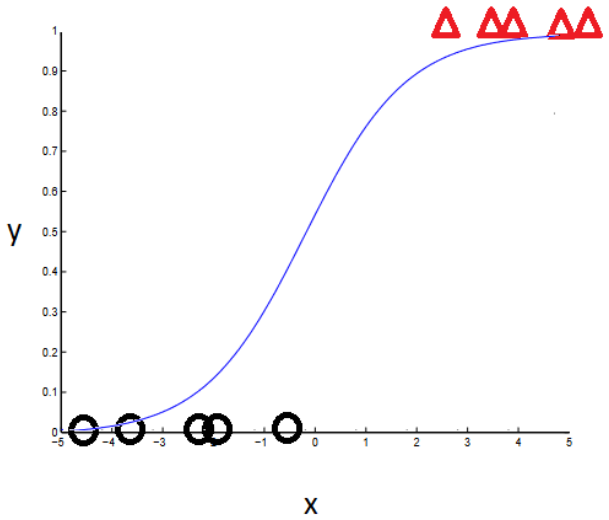
Esempio 1D

Il nostro obiettivo è, quindi, costruire un modello *h()* tale che: $h(x) = P(Y = 1 | X = x)$ e perciò il codominio di *h()* deve essere [0, 1]

Per stimare la probabilità dell'altra classe, ovvero $P(Y = 0 | X = x)$, posso semplicemente prendere il complementare di *h(x)*:

$$P(Y = 0 | X = x) = 1 - h(x)$$

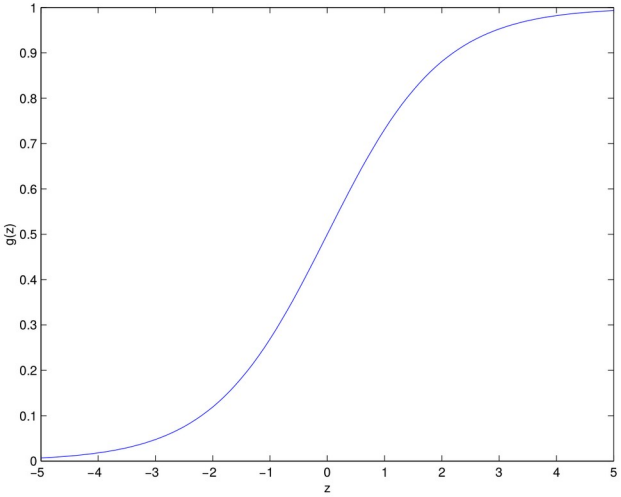
Perchè far sì che il codominio di *h()* sia [0, 1], devo "piegare" la funzione ipotesi per ottenere qualcosa di simile alla figura accanto



Logistic function

Una possibile funzione che ha «forma ad S» come quella in figura è la cosiddetta «funzione **logistica»** (caso speciale di **sigmoide**), definita come segue:

$$g(z) = \frac{1}{1 + e^{-z}}$$



N.B. Spesso «sigmoide» e «funzione logistica» vengono usati come sinonimi, anche se, in realtà, anche altre funzioni, come, e.g., *tanh(z)* appartengono alla famiglia delle sigmoidi (i.e., «forma ad S»)

Definizione del modello

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Mettiamo ora insieme le due cose (Linear Regression e Logistic Function) e definiamo il nostro modello predittivo

Anzitutto, la *parte parametrica* è identica alla Linear Regression, ovvero si basa su una combinazione lineare delle feature:

$$f_\theta(\boldsymbol{x}) = \theta^T \boldsymbol{x}$$

Ora uso una variabile intermedia (*z*) e ho che *z = f(**x**)*

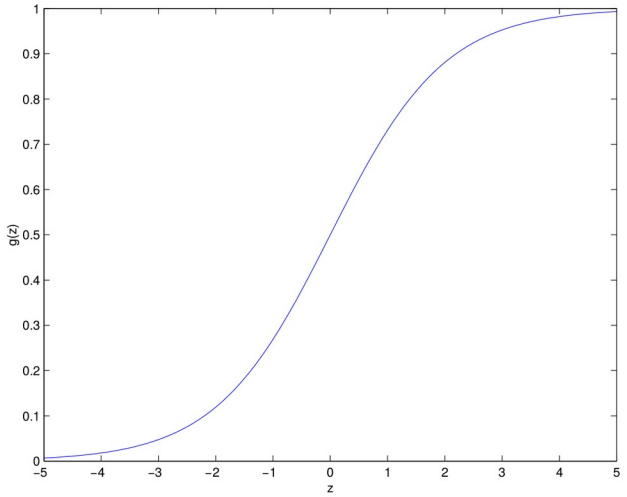
Infine, uso la logistic function per “piegare” l’output di *f()*, ottenendo una funzione composta *h(**x**) = g(z) = g(f(**x**))* che rappresenta una (stima della) probabilità, ovvero un valore compreso in [0, 1]:

$$h_\theta = g(f_\theta) : \mathbb{R}^d \to [0, 1]$$

Logistic regression: modello completo

$$h_\theta(\boldsymbol{x}) = g(\theta^T \boldsymbol{x}) = \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}}$$

$$g(z) = \frac{1}{1 + e^{-z}}$$



- Il termine «logistic regression» deriva dal fatto che abbiamo ricondotto il task di classificazione ad un task di regressione (*non lineare*) di valori di probabilità, quest'ultimi ottenuti tramite la funzione logistica
- Il modello completo è, infatti, una funzione non lineare
- La parte parametrica, però, è ancora lineare…

Logistic regression: esempi 1D

- Ovviamente, e come al solito, valori diversi del vettore dei parametri θ portano ad istanze del modello predittivo diverso, cioè portano a funzioni di classificazione $h_\theta()$ diverse
- La cosa si capisce meglio “srotolando” la sommatoria pesata della parte parametrica e osservando, ad esempio, il caso monodimensionale, in cui la formula diventa:

$$h_{[\theta_0,\theta_1]}(x) = \frac{1}{1 + e^{-\theta_1 x - \theta_0}}$$

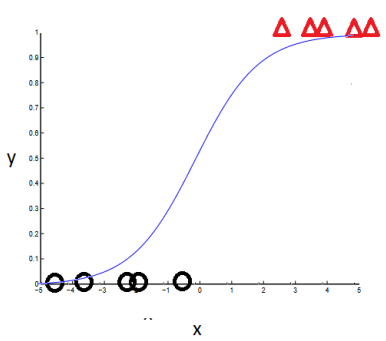
- θ_0 è il parametro di “shift” (traslazione), mentre θ_1 è il parametro di scalamento
- In particolare, $h_\theta(x) = 0.5$ quando $\theta_1 x = -\theta_0$

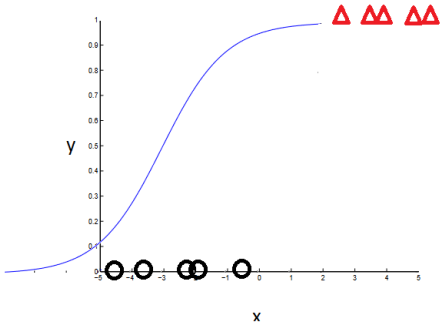
Logistic regression: esempi 1D

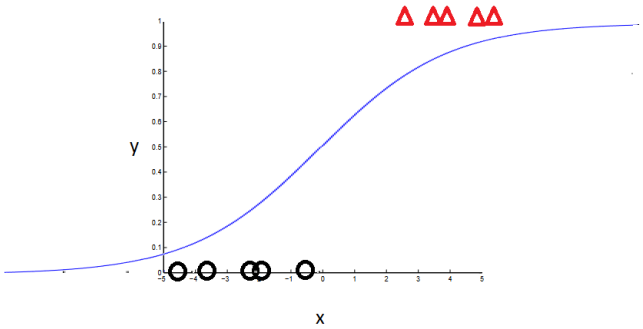
AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

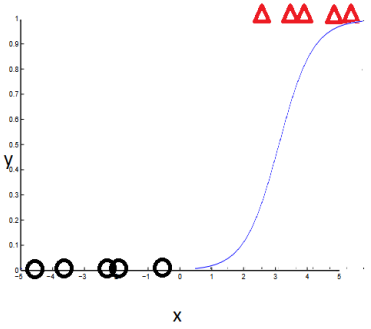
$$h_{[\theta_0,\theta_1]}(x) = \frac{1}{1 + e^{-\theta_1 x - \theta_0}}$$

Gli esempi a fianco corrispondono a valori diversi del vettore θ









Ottimizzazione

Ottimizzazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dobbiamo ora trovare un modo per stimare i parametri θ

Nel caso della regressione siamo partiti da un modello statistico del tipo:

$y = h(\boldsymbol{x}; \theta) + e$

e abbiamo definito una Loss Function ($J(\theta)$) che quantifica l'errore (e) su tutto il dataset di training T

$J(\theta)$ è stata usata come funzione obiettivo di un problema di ottimizzazione ("minimizzazione", per essere più specifici) il cui scopo è quello di trovare la funzione ipotesi il cui modello statistico associato minimizza l'errore rispetto ai dati in T

Ottimizzazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Nel caso della classificazione il modello statistico può essere espresso come segue.

Dato un generico esempio di training $(\boldsymbol{x}, y) \in T$ e la funzione ipotesi parametrica $h(\boldsymbol{x}; \theta)$, il modello statistico che "spiega" $(\boldsymbol{x}, y)$ basandosi su $h(\boldsymbol{x}; \theta)$ è definito tramite:

- $h(\boldsymbol{x}; \theta) = P(Y = y_1 | \boldsymbol{x})$ se $y = y_1$
- $1 - h(\boldsymbol{x}; \theta) = P(Y = y_2 | \boldsymbol{x})$ se $y = y_2$

Useremo questo modello statistico per definire una funzione obiettivo (dipendente da θ) e, analogamente al caso della regressione, impostare un problema di ottimizzazione, che infine risolveremo rispetto a θ

Come vedremo, in questo caso si tratterà di un problema di *massimizzazione* (anziché di minimizzazione), detto Maximum Likelihood Estimation (MLE)

Maximum Likelihood Estimation

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Da un punto di vista intuitivo, la Maximum Likelihood Estimation (MLE) ci permette di misurare in maniera quantitativa la capacità di «spiegazione dei dati» di un modello statistico descritto in maniera probabilistica come quello precedente

e poi di ottimizzare questa stima in maniera simile a quanto fatto in precedenza usando una Loss Function

La MLE può essere applicata in vari ambiti del ML (non solo con la logistic regression)

Per cui ora studieremo la MLE in forma generale e poi torneremo al caso particolare della logistic regression

Likelihood function: idea intuitiva

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dato un dataset di training $T = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)})\}$, una Loss Function $J(\theta)$ descrive l'errore cumulativo dello specifico modello predittivo $h_\theta()$ su T

Analogamente, ora definiremo una *Likelihood Function*, la quale, però, anzichè stimare un errore, stimerà la *probabilità* di ottenere tutti gli esempi contenuti in T usando $h_\theta()$

Premessa: dati i.i.d.

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Un insieme di variabili aleatorie si dice “independent and identically distributed” (*i.i.d.*) se sono mutuamente indipendenti e hanno tutte la stessa distribuzione di probabilità

Ad ese., in ML, di solito si assume che i samples (sia di training che di testing) siano i.i.d., ovvero che sono stati “estratti” (selezionati) utilizzando la stessa distribuzione (identically distributed) e indipendentemente gli uni dagli altri (independent).

	Features						Label
Samples	Temperature - Zone 1	Temperature - Zone 2	Voltage - Engine 1	Voltage - Engine 2	Accelerometer 1	Accelerometer 2	Energy consumption
	25.4	26.8	12.2	14.4	0.01	0.05	25.6
	21.3	19.7	11.8	14.7	0.033	0.045	22.8
	22.2	33.6	11.9	14.9	0.012	0.067	27.8
	24.3	32.1	12.0	14.0	0.098	0.105	21.2
	22.8	27.9	11.0	13.5	0.001	0.003	18.8

Likelihood function

Dato un dataset (di training) $T = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)})\}$, dove ogni $(\boldsymbol{x}^{(i)}, y^{(i)})$ è i.i.d., l'idea base è calcolare la probabilità di ottenere ogni $(\boldsymbol{x}^{(i)}, y^{(i)})$ *utilizzando il modello predittivo* $h_\theta(\boldsymbol{x}^{(i)})$, la cui forma parametrica si assume nota (ovvero conosciamo la classe di funzioni a cui $h_\theta()$ appartiene)

Ad esempio, $h_\theta()$ appartiene alla classe delle funzioni lineari nel caso della linear regression, oppure ha la forma vista prima della logistic regression, ecc.

Quindi, per un dato θ, e uno specifico $(\boldsymbol{x}^{(i)}, y^{(i)})$ in T, voglio calcolare $P(Y = y^{(i)} | X = \boldsymbol{x}^{(i)})$

Per sottolineare che questa stima dipende da θ, scrivo: $P(Y = y^{(i)} | X = \boldsymbol{x}^{(i)}, \theta)$

Matematicamente, θ *è ora diventato anch'esso una variabile aleatoria*

Per essere più precisi, θ è un vettore di variabili aleatorie: $\theta = [\theta_0, \ldots, \theta_d]$

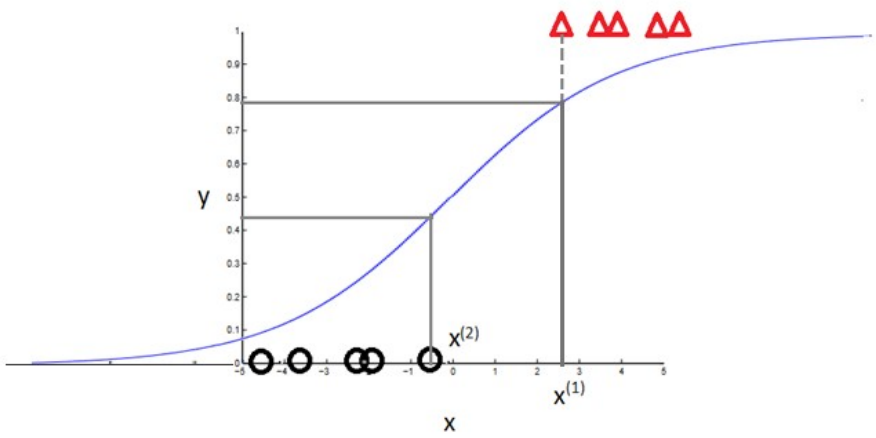
Ad ese., nel caso della classificazione binaria, voglio calcolare:

- $P(Y = 1 | X = \boldsymbol{x}^{(i)}, \theta)$ se $(\boldsymbol{x}^{(i)}, 1)$ in T, oppure
- $P(Y = 0 | X = \boldsymbol{x}^{(i)}, \theta)$ se $(\boldsymbol{x}^{(i)}, 0)$ in T

Dato che, per definizione del mio modello parametrico, ho che $h_\theta(\boldsymbol{x}^{(i)}) = P(Y = 1 | X = \boldsymbol{x}^{(i)}, \theta)$, avrò (nel caso di classificazione binaria):

- $P(Y = 1 | X = \boldsymbol{x}^{(i)}, \theta) = h_\theta(\boldsymbol{x}^{(i)})$
- $P(Y = 0 | X = \boldsymbol{x}^{(i)}, \theta) = 1 - h_\theta(\boldsymbol{x}^{(i)})$

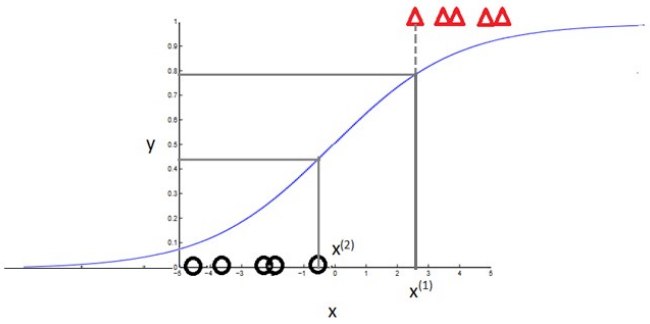
Esempio (1D)



Dato $T = \{(x^{(1)},1), (x^{(2)},0), \ldots\}$, con $x^{(1)} = 2.6$ e $x^{(2)} = -0.5$, secondo lo specifico (istanza di) modello rappresentato qui sopra (e *dipendente da un θ specifico*), ho che:

- $P(Y = 1 | X = x^{(1)}, \theta) = h_\theta(x^{(1)}) = 0.78$
- $P(Y = 0 | X = x^{(2)}, \theta) = 1 - h_\theta(x^{(2)}) = 1 - 0.43 = 0.57$

Esempio (1D)



$$h_{[\theta_0,\theta_1]}(x) = \frac{1}{1+e^{-\theta_1 x-\theta_0}}$$

$T = \{(x^{(1)},1), (x^{(2)},0), \ldots\}$, $x^{(1)} = 2.6$, $x^{(2)} = -0.5$, …

Attenzione però: in fase di training il valore di θ non lo conosco: è la mia «incognita»
Per cui, per un *generico* θ, e lo stesso T di prima, avrò:

$$P(Y=1|X=2.6,\theta) = \frac{1}{1+e^{-\theta_1 2.6-\theta_0}}$$

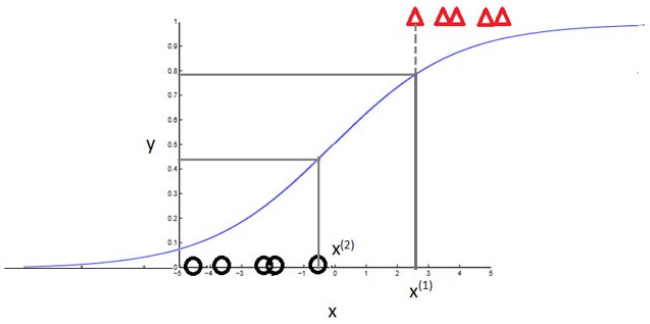
$$P(Y=0|X=-0.5,\theta) = 1 - \frac{1}{1+e^{+\theta_1 0.5-\theta_0}}$$

Likelihood function

A questo punto estendiamo questa stima di probabilità a tutto T, calcolando $P(T|\theta)$
$P(T|\theta)$ può essere letto come: la «probabilità di avere T dato θ»
Usando l'assunzione che i dati sono i.i.d., ho che:
$P(T|\theta) = P(y^{(1)}|\boldsymbol{x}^{(1)}, \theta) * P(y^{(2)}|\boldsymbol{x}^{(2)}, \theta) * \ldots * P(y^{(n)}|\boldsymbol{x}^{(n)}, \theta)$
$P(T|\theta)$ è chiamata *likelihood function*, ed è una vera e propria funzione (di θ), che esprime la probabilità di ottenere *quello specifico* insieme di dati (T) dato il modello parametrico $h_\theta()$

In sostanza, $P(T|\theta)$ esprime la «verosomiglianza» dei dati (T) rispetto ad un determinato vettore di parametri θ e corrispondente funzione ipotesi $h_\theta()$

Esempio (1D)



$$h_{[\theta_0,\theta_1]}(x) = \frac{1}{1+e^{-\theta_1 x-\theta_0}}$$

$T = \{(x^{(1)},1), (x^{(2)},0), \ldots\}$, $x^{(1)} = 2.6$, $x^{(2)} = -0.5$,

$$P(Y=1|X=2.6,\theta) = \frac{1}{1+e^{-\theta_1 2.6-\theta_0}}$$

$$P(Y=0|X=-0.5,\theta) = 1 - \frac{1}{1+e^{+\theta_1 0.5-\theta_0}}$$

$$P(T|\theta) = \frac{1}{1+e^{-\theta_1 2.6-\theta_0}} * [1 - \frac{1}{1+e^{+\theta_1 0.5-\theta_0}}] * \ldots$$

Maximum likelihood estimation

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Il «miglior» vettore di parametri è quello che *massimizza* la likelihood function
Maximum Likelihood Estimation (MLE) significa risolvere:

$$\arg\max_{\theta} P(T|\theta)$$

Maximum likelihood estimation

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Notate l'analogia tra una loss function e una likelihood function

- La prima esprime l'*errore* che si commette, utilizzando $h_\theta()$, nel predire i dati di training in *T*
- La seconda, invece, ha un'interpretazione probabilistica, ed esprime la *probabilità* («verosomiglianza») nel predire i dati di training in *T* usando il modello $h_\theta()$

Avendo una loss function, userò il Gradient Descent per trovare un *minimo* locale nello spazio dei parametri
Avendo una likelihood function, userò il Gradient *Ascent* per trovare un *massimo* locale nello spazio dei parametri

Il Gradient Ascent è identico al Gradient Descent, con l'unica differenza che, ad ogni iterazione, il gradiente viene *aggiunto* al valore attuale di θ anziché sottratto

A seconda dei casi, può essere più facile/conveniente definire una loss function o una likelihood function

Torniamo ora alla logistic regression e definiamo (nei dettagli) una likelihood function

Likelihood function per la classificazione binaria

Come visto prima, per la classificazione binaria abbiamo che, dato un generico *(**x**, y)* in *T*:

- $P(Y = 1 | X = \boldsymbol{x}, \theta) = h_\theta(\boldsymbol{x})$
- $P(Y = 0 | X = \boldsymbol{x}, \theta) = 1 - h_\theta(\boldsymbol{x})$

Dato che $y \in \{0,1\}$, posso scrivere le due formule di sopra in maniera succinta come:

$$P(Y = y | X = \boldsymbol{x}, \theta) = (h_\theta(\boldsymbol{x}))^y (1 - h_\theta(\boldsymbol{x}))^{1-y}$$

Likelihood function per la classificazione binaria

$$L(\theta) = P(T|\theta) = \quad = \prod_{i=1}^{n} P(y^{(i)}|\boldsymbol{x}^{(i)}, \theta)$$
$$= \prod_{i=1}^{n} (h_\theta(\boldsymbol{x}^{(i)}))^{y^{(i)}} (1 - h_\theta(\boldsymbol{x}^{(i)}))^{1-y^{(i)}}$$

Maximum log-likelihood Estimation

$$L(\theta) = P(T|\theta) = = \prod_{i=1}^{n} P(y^{(i)}|\boldsymbol{x}^{(i)}, \theta)$$
$$= \prod_{i=1}^{n} (h_\theta(\boldsymbol{x}^{(i)}))^{y^{(i)}} (1 - h_\theta(\boldsymbol{x}^{(i)}))^{1-y^{(i)}}$$

Di solito, invece di usare $L(\theta)$, come definita sopra, si usa la funzione $l(\theta)$ definita qui sotto, ovvero il *logaritmo* della likelihood (detto «log-likelihood»)

$$l(\theta) = \log L(\theta)$$
$$= \sum_{i=1}^{n} y^{(i)} \log(h_\theta(\boldsymbol{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(\boldsymbol{x}^{(i)}))$$

Il motivo è la volontà di semplificare i calcoli e renderli numericamente più stabili. Ad esempio, in questo caso il logaritmo ci pemette di trasformare il prodotto in una somma

Risolvere il problema di ottimizzazione

La Maximum (log) Likelihood Estimation consiste nel massimizzare $l(\theta)$ rispetto a θ:

$$\arg\max_{\theta} l(\theta)$$

La parte non lineare di *h()* (i.e., *g()*) non ci permette di arrivare ad un sistema di equazioni lineari, ovvero ad una soluzione diretta («closed form solution»). Per cui useremo il Gradient Ascent

Come al solito, dobbiamo calcolarci il gradiente di $l(\theta)$ rispetto a θ e poi usarlo nell'algoritmo del Gradient Descent, ricordandoci però di sommarlo anziché sottrarlo (=> Gradient Ascent)

Eviteremo di descrivere questa parte perché non aggiunge nulla di nuovo

Relazione tra la Maximum log-Likelihood Estimation e la Binary Cross Entropy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

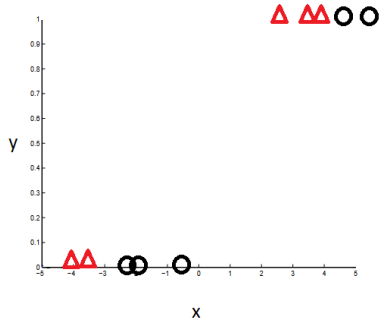
$$\begin{aligned} l(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{n} y^{(i)} \log(h_\theta(\boldsymbol{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(\boldsymbol{x}^{(i)})) \end{aligned}$$

La funzione qui sopra, col segno meno davanti (-l(θ)), è nota come Binary Cross Entropy, che è una *loss function* molto usata in classificazione binaria (usata non solo con la logistic regression), e può essere ottenuta utilizzando il concetto di “entropia dell’informazione” (che vedremo in seguito) senza usare una likelihood function

Ovviamente, massimizzare $l(\theta)$ e minimizzare - $l(\theta)$ è equivalente:

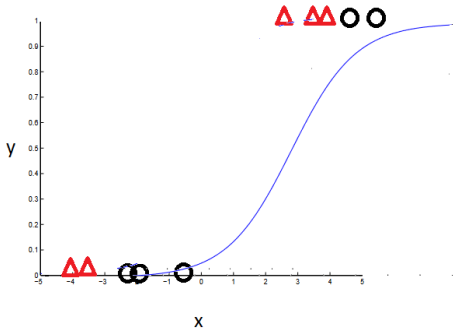
$$\arg\max_\theta l(\theta) = \arg\min_\theta -l(\theta)$$

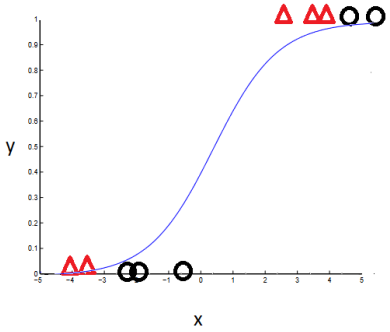
Classi non (linearmente) separabili

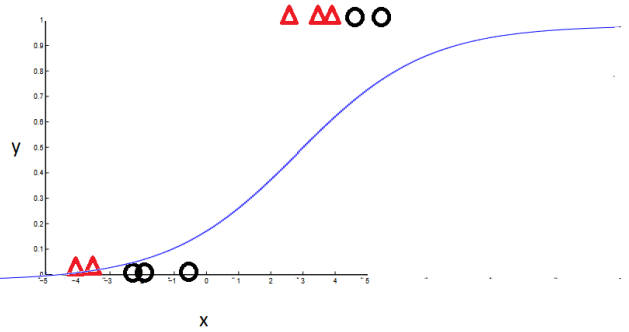


Attenzione: se le gli esempi in *T* non sono separabili, come nell'esempio qui a fianco, allora qualsiasi θ porterà ad un underfitting (v. esempi qui sotto)

Ciò vuol dire che la parte parametrica lineare non è sufficiente







Classificazione con più classi

Classificazione con più classi

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Supponiamo ora che $y \in \{1, \ldots, k\}$ ed estendiamo quanto detto ad un task multi-classe

Continueremo a basarci su una parte parametrica lineare e vedremo classificatori con parte parametrica non lineare nelle prossime lezioni

L'estensione della logistic regression ad un task multi-classe prende il nome di «multinomial logistic regression» o «softmax regression»

Definizione del modello: parte parametrica

Per la parte parametrica usiamo *k differenti funzioni lineari*, ognuna dedicata ad una classe, dove ogni funzione è definita da un vettore dei parametri θ_j *specifico per la classe j-esima*:

$$f_1(\boldsymbol{x};\theta_1) = \theta_1^T \boldsymbol{x}$$
$$\dots$$
$$f_j(\boldsymbol{x};\theta_j) = \theta_j^T \boldsymbol{x}$$
$$\dots$$
$$f_k(\boldsymbol{x};\theta_k) = \theta_k^T \boldsymbol{x}$$

Le $f_j(\boldsymbol{x})$ sono dette “score function” perchè assegnano un “punteggio” (score) ad $\boldsymbol{x}$ che stima l’appartenenza di $\boldsymbol{x}$ alla classe j-esima (N.B.: questo punteggio non è ancora una probabilità)

Definizione del modello: parte parametrica

$$f_1(\boldsymbol{x};\theta_1) = \theta_1^T \boldsymbol{x}$$
$$\dots$$
$$f_j(\boldsymbol{x};\theta_j) = \theta_j^T \boldsymbol{x}$$
$$\dots$$
$$f_k(\boldsymbol{x};\theta_k) = \theta_k^T \boldsymbol{x}$$

Analogamente al caso binario, dobbiamo trasformare questi “score” in probabilità, tenendo conto che, per un dato ***x***, la somma delle probabilità di tutte le classi deve fare 1

N.B.: Ora abbiamo *k* vettori di parametri anzichè uno solo: θ_1, …, θ_k e dovremo ottimizzarli tutti e *k*

Softmax Regression

Una possibile funzione non lineare che si può usare per trasformare gli score in probabilità è la cosiddetta funzione **softmax**, che, combinata con le score function, porta al modello completo della Softmax Regression, dato da:

$$h_\theta(\boldsymbol{x}, j) = P(Y = j | X = \boldsymbol{x}, \theta) = \frac{\exp\left(\theta_j^T \boldsymbol{x}\right)}{\sum_{q=1}^{k} \exp\left(\theta_q^T \boldsymbol{x}\right)}$$

dove θ ora è una ***matrice*** contenente i *k* vettori dei parametri: $\theta = \begin{bmatrix} \theta_1^T \\ \dots \\ \theta_k^T \end{bmatrix}$

e, per ogni *1<= j <= k,* la funzione ipotesi $h_\theta(\boldsymbol{x}, j)$ restituisce la probabilità che ***x*** appartenga alla classe j-esima

Softmax Regression

Per trovare i valori di θ faremo come abbiamo fatto per il caso binario, usando la Maximum log-Likelihood Estimation. Ovvero:

- Definiamo una log-likelihood function per il caso multi-classe
- Massimizziamo la log-likelihood usando il Gradient Ascent

Log-likelihood per la Softmax Regression: premessa

Se ho un dataset $T = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)})\}$, con $y \in \{1, \ldots, k\}$, per semplificare la notazione (e non solo), il valore delle label (y) può essere rappresentato utilizzando la codifica «one-hot», ottenuta come segue:

Se $(\boldsymbol{x}^{(i)}, j) \in T$, allora $y^{(i)} = [0, 0, \ldots, 1, 0, \ldots, 0]^T$

è la «one-hot» representation della label j, ed è formato da un vettore di tutti 0 eccetto la j-esima componente (uguale ad 1), dove j è la classe di ground truth corrispondente a $\boldsymbol{x}^{(i)}$

Esempio:

- $j = 2, k = 4$
- $y^{(i)} = [0, 1, 0, 0]^T$

Maximum log-likelihood per la Softmax Regression

Utilizzando le notazioni precedenti, la log-likelihood per un task di classificazione multi-classe può essere espressa come segue:

$$\ell(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{k} y_j^{(i)} log(h_\theta(\boldsymbol{x}^{(i)}, j))$$

dove $y_j^{(i)}$ è la j-esima componente del one hot vector $y^{(i)}$

La funzione qui sopra è un'estensione di quella che abbiamo visto per il caso binario

Analogamente al caso binario, può essere derivata anche senza usare una likelihood, basandosi sul concetto di entropia ed arrivando a formulare una loss function. In questo caso si ottiene la stessa espressione ma col segno meno davanti (- $l(\theta)$), e si chiama Cross Entropy

Si ottimizza col Gradient Ascent (o Descent se si usa la Cross Entropy)

Inference time e Valutazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Inference time e Decision rules

Dato che la funzione ipotesi, sia nel caso binario che in quelllo multi-classe, restituisce una stima di probabilità (detta anche prediction «confidence»), per avere un classificatore completo *C()* c'è bisogno di fare una scelta, ovvero c'è bisogno di una *decision rule* che trasforma le probabilità predette nella scelta finale della classe (ritenuta) più probabile

N.B.: la decision rule viene usata solo a inference time

Decision rules: caso binario

Una semplice decision rule per un classificatore binario potrebbe essere:

- *Scelgo la classe y = 1* se $h_\theta(\boldsymbol{x}) > threshold = 0.5$
- *Scelgo la classe y = 0 altrimenti*

threshold = 0.5 è la soglia di classificazione. Vedremo tra poco che a volte può essere conveniente usare valori diversi da 0.5

Decision rules: caso multi classe

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Una semplice decision rule per un classificatore multiclasse potrebbe essere:

- *Scelgo* $y = arg\ max_{j \in \{1, \ldots, k\}} h_\theta(\boldsymbol{x}, j)$

Ovvero, scelgo la classe con la probabilità più alta (ricordiamoci che $h_\theta(\boldsymbol{x}, j) = P(Y = j \mid X = \boldsymbol{x})$)

Attenzione: è una decision rule, per cui quell' «arg max» è solo un modo conciso di scrivere la procedura di decisione e non ha nulla a che fare con l'ottimizzazione dei parametri nello spazio dei parametri!

Valutazione: metriche

Esaminiamo le metriche più comuni per il caso della classificazione binaria

La maggior parte di quelle che vedremo possono essere estese al caso non binario

Notazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

P	→	Positive (ovvero y = 1)
N	→	Negative (ovvero y = 0)
T	→	True
F	→	False

Confusion Matrix

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

	Actual P	**Actual N**
Predicted P	TP	FP
Predicted N	FN	TN

TP = True Positive (il classificatore predice 1 e la ground truth è 1)
FP = False Positive (il classificatore predice 1 ma la ground truth è 0)
FN = False Negative (il classificatore predice 0 ma la ground truth è 1)
TN = True Negative (il classificatore predice 0 e la ground truth è 0)

Metriche per la classificazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Le metriche che seguono, ricavate dalla confusion matrix, assumono tutte valori tra 0 ed 1

Accuracy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

	Actual P	Actual N
Predicted P	TP	FP
Predicted N	FN	TN

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

	Actual P	Actual N
Predicted P	TP	FP
Predicted N	FN	TN

$$Precision = \frac{TP}{TP+FP}$$

Recall/Sensitivity/True Positive Rate (TPR)

	Actual P	Actual N
Predicted P	TP	FP
Predicted N	FN	TN

$$Recall = \frac{TP}{TP+FN}$$

False Positive Rate (FPR)

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

	Actual P	Actual N
Predicted P	TP	FP
Predicted N	FN	TN

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

F1-Score

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

	Actual P	Actual N
Predicted P	TP	FP
Predicted N	FN	TN

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

La F1-score è 1 solo quando sia la Precision che la Recall sono 1

Dipendenza delle metriche dalla decision rule

Variando la soglia di classificazione nella decision rule, cambia il numero totale di TP, FP, ecc., e, di conseguenza, si ottengono valori diversi per le metriche viste finora

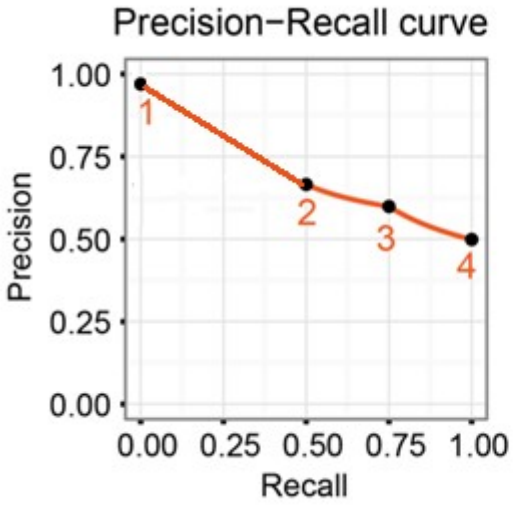
Variare la soglia di classificazione può essere importante in alcuni ambiti applicativi

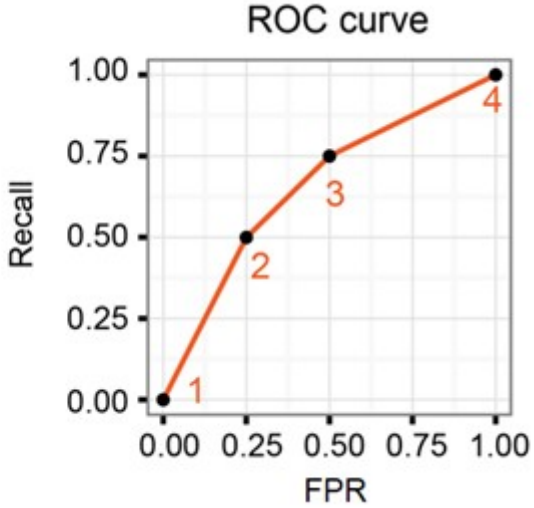
Ad esempio, in ambito medico, potrebbe essere più importante minimizzare i FN a discapito dei FP (o viceversa)

Curve P-R e ROC, AUC

Le variazioni delle metriche ottenute con valori di soglia diversi possono essere cumulativamente rappresentare in grafici come la curva **Precision-Recall** e la curva ROC (**Receiver Operating Characteristic**).

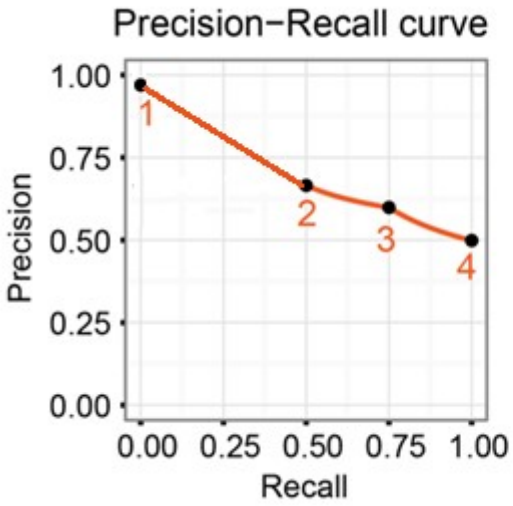
Nelle figure a fianco i numeri 1-4 indicano il risultato di 4 diverse soglie di decisione

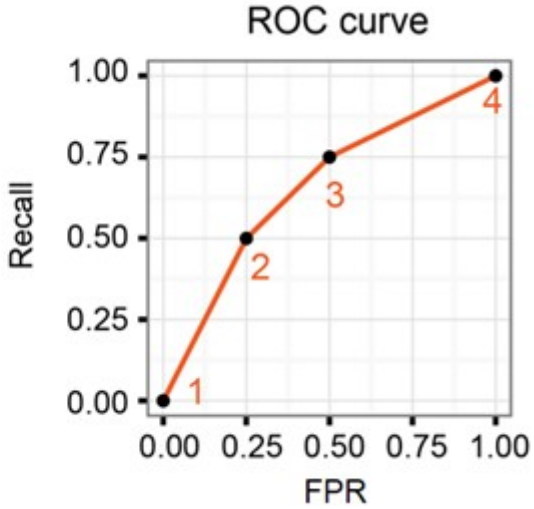




Curve P-R e ROC, AUC

Dalla curva ROC, si ottiene un'altra metrica, l'**Area Under the Curve** (AUC), che misura l'area (l'integrale) sotto la curva ROC
L'AUC non dipende da una specifica soglia di classificazione





Esercizi

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

In un task di classificazione binaria, assumendo un dataset di testing in cui le due classi sono bilanciate (ovvero $P(Y=0) = P(Y=1) = 0.5$), calcolare l'Accuracy, la Recall e il FPR dei seguenti "finti" classificatori:

1. Un classificatore che risponde sempre 1

2. Un classificatore che, indipendentemente dall'input, in maniera random, col 50% di probabilità risponde 1 e col 50% risponde 0 (il cosiddetto "chance level")

Riferimenti

- https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf
- **A tu per tu col Machine Learning. L'incredibile viaggio di un developer nel favoloso mondo della Data Science**, Alessandro Cucci, The Dot Company, 2017 [cap.5]
- **Pattern Classification, second edition,** R. O. Duda, P. E. Hart, D. G. Stork, Wiley-Interscience, 2000
- **An introduction to statistical learning**, Gareth M. James, Daniela Witten, Trevor Hastie, Robert Tibshirani, New York: Springer, 2013 [cap. 4]