

# Classificazione discriminativa

Vincenzo Bonifaci

IN550 – Machine Learning

# Classificazione generativa vs. discriminativa

## Approccio generativo

- Stima  $\Pr(x, y)$  per poi dedurre  $\Pr(y|x)$
- Confronta le  $\Pr(y|x)$  per trovare la classe più verosimile

Esempi: QDA, LDA, Naive Bayes

## Approccio discriminativo

- Stima  $\Pr(y|x)$
- Confronta le  $\Pr(y|x)$  per trovare la classe più verosimile

Oppure, evitando completamente le probabilità,

- Costruisci direttamente una funzione da  $\mathcal{X}$  a  $\mathcal{Y}$

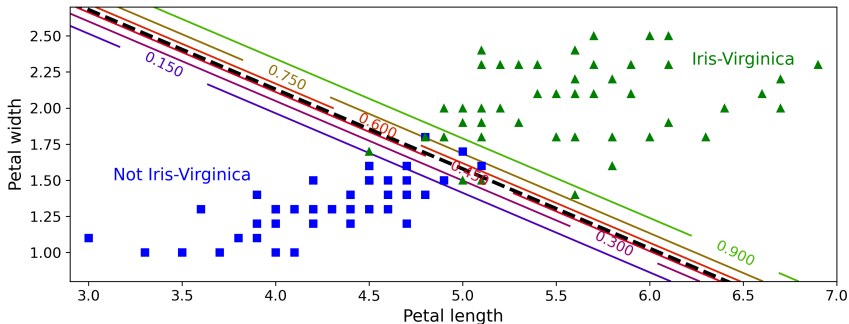
Esempi: Regressione logistica, Percettrone, Support Vector Machines

# Stima di probabilità per etichette binarie

## Stima della probabilità condizionata (etichette binarie)

**Dato:** un insieme di esempi  $(x, y)$  con  $x \in \mathbb{R}^{d+1}$  e  $y \in \{0, 1\}$

**Trova:** una funzione  $h : \mathcal{X} \rightarrow [0, 1]$  con  $h(x) \approx \Pr(y = 1|x)$



# Un modello lineare per la stima di probabilità?

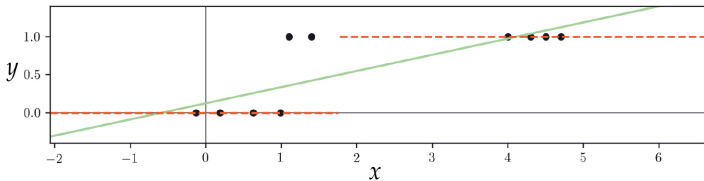
Dato  $x$ , vogliamo stimare  $\Pr(y = 1|x)$  attraverso una funzione lineare

$$w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d = w^\top x$$

Vorremmo che  $\Pr(y = 1|x)$ :

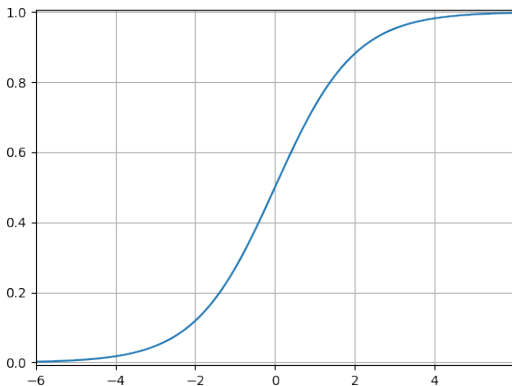
- aumenti quando la funzione lineare aumenta
- sia 50% quando la funzione lineare vale zero

Come convertire  $w^\top x$  in una probabilità?



# La funzione sigmoide (sigmoide logistica)

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad \in [0, 1]$$



# Alcune proprietà della sigmoide

Se  $\sigma(z) = (1 + \exp(-z))^{-1}$ , allora per ogni  $z \in \mathbb{R}$ ,

$$1 - \sigma(z) = \sigma(-z)$$

Se  $\sigma(z) = (1 + \exp(-z))^{-1}$ , allora per ogni  $z \in \mathbb{R}$ ,

$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$$

# Regressione logistica binaria (etichette 0/1)

Assumiamo che:

$$\Pr(y = 1|x) = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)}$$

Ne consegue che:

$$\Pr(y = 0|x) = 1 - \sigma(w^\top x) = \sigma(-w^\top x) = \frac{1}{1 + \exp(w^\top x)}$$

Qualunque sia  $y \in \{0, 1\}$ , possiamo quindi scrivere

$$\Pr(y|x) = (h(x))^y (1 - h(x))^{1-y}$$

dove  $h(x) = \sigma(w^\top x)$

# La classe di ipotesi della regressione logistica binaria

Nella *regressione logistica*, l'insieme delle ipotesi è l'insieme  $\mathcal{H}_{sig}$  delle funzioni ottenute componendo la sigmoide con una funzione lineare da  $\mathcal{X} \subseteq \mathbb{R}^{d+1}$  a  $\mathbb{R}$ :

$$h \in \mathcal{H}_{sig} \quad \Leftrightarrow \quad h(x) = \sigma(w^\top x) \quad \text{per qualche } w \in \mathbb{R}^{d+1}$$



# MLE nella regressione logistica (etichette 0/1)

## Principio di massima verosimiglianza

Dati gli esempi  $S = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ , scegli  $w \in \mathbb{R}^{d+1}$  che massimizza la funzione di *verosimiglianza* [*likelihood*]:

$$\mathcal{L}(w) = \prod_{i=1}^m \Pr(y^{(i)} | x^{(i)}; w)$$

Massimizzare  $\mathcal{L}(w)$  equivale a minimizzare la *cross-entropia*:

$$\sum_{i=1}^m \left[ -y^{(i)} \log h(x^{(i)}) - (1 - y^{(i)}) \log(1 - h(x^{(i)})) \right]$$

## Funzione costo nella regressione logistica (etichette 0/1)

In altre parole stiamo assumendo la seguente funzione costo:

Funzione costo *cross-entropia* (etichette 0/1)

$$\ell(h, (x, y)) = \begin{cases} -\log h(x) & \text{se } y = 1 \\ -\log(1 - h(x)) & \text{se } y = 0 \end{cases}$$

È una funzione **convessa** nel vettore  $w$  dei parametri

## Rischio empirico nella regressione logistica (etichette 0/1)

Il principio MLE in questo caso è quindi equivalente al principio Empirical Risk Minimization con la funzione obiettivo cross-entropia

## ERM nella regressione logistica (etichette 0/1)

Dati gli esempi  $S = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ ,  
scegli  $w \in \mathbb{R}^{d+1}$  che minimizza

$$\begin{aligned} L_S(w) &= - \sum_{i=1}^m \log \Pr(y^{(i)} | x^{(i)}; w) \\ &= \sum_{i=1}^m \left[ -y^{(i)} \log h(x^{(i)}) - (1 - y^{(i)}) \log(1 - h(x^{(i)})) \right] \end{aligned}$$

# SGD per la regressione logistica (etichette 0/1)

Possiamo minimizzare il costo con i metodi gradiente

Per esempio con SGD:  $w \leftarrow w - \eta \nabla \ell(h_w)$

Calcolando  $\nabla \ell(h_w)$  e sfruttando  $\sigma'(z) = \sigma(z)(1 - \sigma(z))$  otteniamo

$$\nabla \ell(h_w) = (h(x) - y)x$$

## Regola SGD per la regressione logistica (0/1)

$$w \leftarrow w - \eta \cdot (h(x) - y) \cdot x$$

NB. La regola ha la stessa struttura della regola Least Mean Squares (LMS), ma il significato di  $h(x)$  è differente

# Regressione logistica binaria (etichette $\pm 1$ )

Assumiamo che:

$$\Pr(y = +1|x) = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)}$$

Ne consegue che:

$$\Pr(y = -1|x) = \sigma(-w^\top x) = \frac{1}{1 + \exp(w^\top x)}$$

In generale, per  $y \in \{-1, +1\}$ , possiamo scrivere

$$\Pr(y|x) = \sigma(y \cdot w^\top x) = \frac{1}{1 + \exp(-y \cdot w^\top x)}$$

# Funzione costo nella regressione logistica (etichette $\pm 1$ )

La funzione di costo diventa la seguente:

Funzione costo *log-loss* o *softmax* (etichette  $\pm 1$ )

$$\begin{aligned}\ell(h, (x, y)) &= -\log \Pr(y|x) \\ &= \log(1 + \exp(-y \cdot w^\top x)) \\ &= \log [\exp(0) + \exp(-y \cdot w^\top x)] \\ &= \text{softmax}(0, -y \cdot w^\top x)\end{aligned}$$

dove

$$\text{softmax}(a, b) \stackrel{\text{def}}{=} \log [\exp(a) + \exp(b)]$$

# Funzione softmax

## Funzione *softmax*

Per  $a_1, a_2, \dots, a_K \in \mathbb{R}$ ,

$$\text{softmax}(a_1, a_2, \dots, a_K) \stackrel{\text{def}}{=} \frac{\exp(a_i)}{\exp(a_1) + \exp(a_2) + \dots + \exp(a_K)}$$

Ha analogie con la funzione  $\max(a_1, \dots, a_K)$ :

- Quando  $a_i \gg a_j$  per ogni  $j \neq i$ ,  $\text{softmax}(a_1, \dots, a_K) \approx a_i$
- Si ha sempre  $\max(a_1, \dots, a_K) \leq \text{softmax}(a_1, \dots, a_K) \leq (\log K) + \max(a_1, \dots, a_K)$
- È una funzione **convessa** del vettore  $a$
- Ma (diversamente da  $\max$ ) è **differenziabile** ovunque

# MLE nella regressione logistica

## Principio di massima verosimiglianza

Dati gli esempi  $S = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ , scegli  $w \in \mathbb{R}^{d+1}$  che massimizza la funzione di *verosimiglianza* [*likelihood*]:

$$\prod_{i=1}^m \Pr(y^{(i)} | x^{(i)}; w)$$



# Rischio empirico nella regressione logistica (etichette $\pm 1$ )

Equivalentemente, passando al logaritmo, vogliamo *minimizzare* il **rischio empirico**, in linea col principio **ERM**:

## ERM nella regressione logistica (etichette $\pm 1$ )

Dati gli esempi  $S = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ , scegli  $w \in \mathbb{R}^d$  che minimizza

$$L_S(w) = - \sum_{i=1}^m \log \Pr(y^{(i)} | x^{(i)}; w) = \sum_{i=1}^m \log(1 + \exp(-y^{(i)} \cdot w^\top x^{(i)}))$$

# Minimizzazione del rischio empirico nella regressione logistica

## Rischio empirico nella regressione logistica (etichette $\pm 1$ )

$$L_S(w) = \sum_{i=1}^m \log(1 + \exp(-y^{(i)} \cdot w^\top x^{(i)}))$$

- Il minimizzatore  $w^*$  non è esprimibile in forma chiusa
- Ma  $L_S(w)$  è una funzione **convessa** in  $w$

⇒ Il problema di ottimizzazione corrispondente è convesso

⇒ Possiamo trovare  $w^*$  attraverso Gradient Descent o le sue varianti

Si possono usare anche metodi del secondo ordine (la funzione è sia convessa che ovunque differenziabile)

# SGD per la regressione logistica (etichette $\pm 1$ )

Per derivare la regola di aggiornamento di SGD ricapitoliamo le assunzioni:

- Ipotesi in  $\mathcal{H}_{sig}$ :  $h_w(x) = \sigma(w^\top x)$
- Costo logistico:  $\ell(h_w) = \log(1 + \exp(-y \cdot w^\top x))$

Prendendo le derivate parziali di  $\ell$ , otteniamo

$$\begin{aligned}
 \frac{\partial \ell}{\partial w_j}(h_w) &= \frac{1}{1 + \exp(-yw^\top x)} \frac{\partial}{\partial w_j} \left[ 1 + \exp(-yw^\top x) \right] \\
 &= \frac{\exp(-yw^\top x)}{1 + \exp(-yw^\top x)} \frac{\partial}{\partial w_j} \left[ -yw^\top x \right] \\
 &= -\frac{1}{1 + \exp(+yw^\top x)} \cdot y \cdot x_j \\
 &= -\Pr(-y|x; w) \cdot y \cdot x_j
 \end{aligned}$$

# SGD per la regressione logistica (etichette $\pm 1$ )

In forma vettoriale,

$$\nabla \ell(h_w) = -\Pr(-y|x; w) \cdot y \cdot x$$

La regola di aggiornamento SGD (con etichette  $\pm 1$ ) è quindi

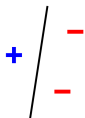
$$w \leftarrow w + \eta \cdot \Pr(-y|x; w) \cdot y \cdot x$$

# Classificazione binaria e separabilità lineare

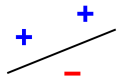
## Separabilità lineare

Un insieme di esempi  $(x, y)$  con etichette di due tipi (+ e -) è *linearmente separabile* se esiste  $w \in \mathbb{R}^{d+1}$  tale che:

- $w^\top x > 0$  ogniqualvolta  $x$  è un esempio di tipo +
- $w^\top x < 0$  ogniqualvolta  $x$  è un esempio di tipo -



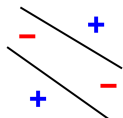
separabile



separabile



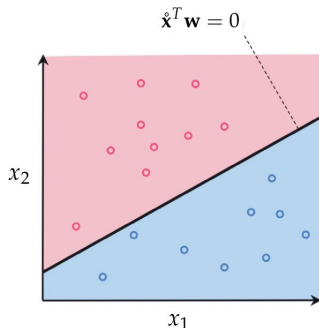
separabile



non separabile

# Valori $y \cdot w^\top x$ e separabilità lineare

Supponiamo che il vettore  $w$  separi **perfettamente** gli esempi positivi e negativi:



Questo significa che per ogni  $(x, y)$ ,

- $y = +1$  e  $w^\top x > 0$ , oppure
- $y = -1$  e  $w^\top x < 0$

In altre parole,  $-y \cdot w^\top x$  è sempre  $< 0$

# Costo softmax e separabilità lineare

Se  $w$  separa **perfettamente** gli esempi positivi e negativi:

$$-y \cdot w^\top x < 0 \quad \text{per ogni } (x, y)$$

allora qualunque sia il costo softmax

$$\log \left[ 1 + \exp(-y \cdot w^\top x) \right] > 0$$

possiamo **diminuirlo** semplicemente usando  $2w$  invece di  $w$ :

$$\log \left[ 1 + \exp(-y \cdot 2w^\top x) \right] < \log \left[ 1 + \exp(-y \cdot w^\top x) \right]$$

Questo significa che il minimo della funzione si ha quando  $\|w\| \rightarrow \infty$ !

La frontiera di decisione definita da  $w$  e  $2w$  è la stessa, ma il divergere di  $w$  può causare instabilità numerica negli algoritmi

## Regressione logistica regolarizzata

Per evitare il divergere di  $w$ , si può considerare il problema di ottimizzazione vincolata

$$\begin{aligned} & \underset{w \in \mathbb{R}^{d+1}}{\text{minimize}} \quad \frac{1}{m} \sum_{i=1}^m \log \left[ 1 + \exp(-y^{(i)} \cdot w^\top x^{(i)}) \right] \\ & \text{subject to} \quad \|\omega\|_2^2 \leq 1 \end{aligned}$$

che (per qualche  $\lambda > 0$ ) equivale alla seguente formulazione

### Regressione logistica con regolarizzazione $\ell_2$

$$\underset{w \in \mathbb{R}^{d+1}}{\text{minimize}} \quad \frac{1}{m} \sum_{i=1}^m \log \left[ 1 + \exp(-y^{(i)} \cdot w^\top x^{(i)}) \right] + \lambda \|\omega\|_2^2$$



# Regressione logistica in scikit-learn

Regressione logistica	Iperparametri	Interfaccia scikit-learn
Non regolarizzata	nessuno	<code>LogisticRegression(penalty='none')</code>
Regolarizzata	$C (= \frac{1}{2\lambda})$	<code>LogisticRegression(penalty='l2', C)</code>

Con il parametro `solver` si possono inoltre specificare vari tipi di risolutori:

- Metodi del primo ordine (`sag`, `saga`)
- Metodi del secondo ordine (`newton-cg`, `lbfgs`)  
(più efficienti per dataset piccoli)

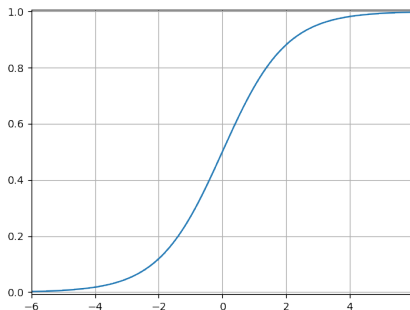
Oppure si può invocare Stochastic Gradient Descent con funzione costo *log*:

Regressione logistica ( $\pm 1$ )	Iperparametri	Interfaccia scikit-learn
Regolarizzata	$\alpha (= \lambda), \eta, T$	<code>SGDClassifier(loss='log', alpha, learning_rate, max_iter)</code>

# Percettroni e Support Vector Machines

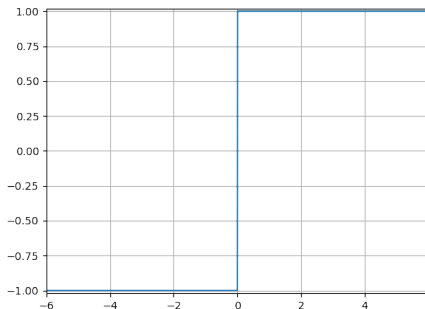
# Il regressore logistico

- $x = (1, x_1, \dots, x_d)$ ,  $p \in [0, 1]$  (probabilità di  $y = 1|x$ )
- $w = (w_0, \dots, w_d)$
- $\hat{p} = h(x) = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)}$



# Il Percettrone

- $x = (1, x_1, \dots, x_d)$ ,  $y \in \{-1, +1\}$  (etichetta)
- $w = (w_0, \dots, w_d)$
- $\hat{y} = h(x) = \text{sign}(w^\top x)$



Se la vera etichetta è  $y$ , la predizione è corretta  $\Leftrightarrow y \cdot w^\top x > 0$

# La classe di ipotesi del Perceptrone

Nel *Perceptrone*, l'insieme delle ipotesi è l'insieme  $\mathcal{H}_P$  delle funzioni ottenute componendo la funzione `sign` con una funzione lineare da  $\mathcal{X}$  a  $\mathbb{R}$ :

$$h \in \mathcal{H}_P \Leftrightarrow h(x) = \text{sign}(w^\top x) \text{ per qualche } w \in \mathbb{R}^{d+1}$$

# Una funzione di costo per il Percettrone

La funzione più naturale (costo 0/1) purtroppo è **ardua** da ottimizzare! (si ha un problema NP-arduo)

Cerchiamo di costruire un *surrogato* della funzione costo:

- Se  $y \cdot w^\top x > 0$ , poniamo costo = 0 (la predizione è corretta)
- Se  $y \cdot w^\top x \leq 0$ , poniamo costo =  $-y \cdot w^\top x$

*Hinge Loss* (per etichette  $\pm 1$ )

$$\ell(h, (x, y)) \stackrel{\text{def}}{=} \max(-y \cdot w^\top x, 0)$$

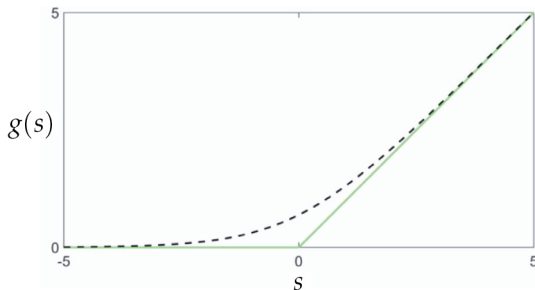
Per ogni  $(x, y)$ , questa funzione costo è **convessa**, quindi possiamo usarla con i metodi gradiente!

Notare la similarità formale con la funzione costo softmax:  $\ell = \text{softmax}(-y \cdot w^\top x, 0)$

# Funzione costo: Percettrone vs. regressione logistica

$$\ell_{\text{perceptrone}}(h, (x, y)) = \max(-y \cdot w^\top x, 0)$$

$$\ell_{\text{logistica}}(h, (x, y)) = \text{softmax}(-y \cdot w^\top x, 0)$$



$g(s) = \max(s, 0)$  è chiamata **ReLU** (Rectified Linear Unit) o **hinge loss**

- È convessa
- È differenziabile ovunque tranne in  $s = 0$

# Minimizzazione del rischio empirico nel perceptrone

## Rischio empirico nel perceptrone

$$L_S(w) = \sum_{i=1}^m \max(-y^{(i)} \cdot w^\top x^{(i)}, 0)$$

- $L_S(w)$  è una funzione **convessa** in  $w$

⇒ Il problema di ottimizzazione corrispondente è convesso

⇒ Possiamo trovare  $w^*$  attraverso Gradient Descent o le sue varianti

Non è ottimizzabile con metodi del secondo ordine



# SGD per il Percettrone

- Se  $y \cdot w^\top x > 0$ , allora  $\ell = 0$  (la predizione è corretta)
- Se  $y \cdot w^\top x \leq 0$ , allora  $\ell = -y \cdot w^\top x$

Calcolando il gradiente del costo, otteniamo:

- Nel primo caso,  $\nabla \ell = 0$
- Nel secondo caso,  $\nabla \ell = -y \cdot x$

## Regola di aggiornamento SGD per il Percettrone

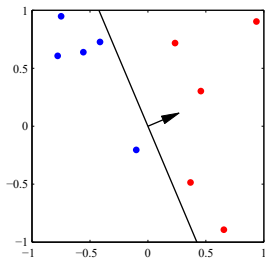
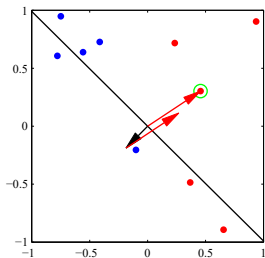
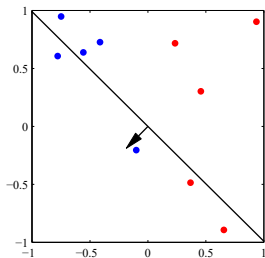
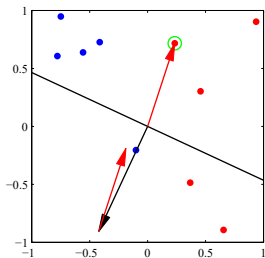
- Se  $y \cdot w^\top x > 0$ , poni  $w^{(t+1)} \leftarrow w^{(t)} - \eta \cdot 0$
- Se  $y \cdot w^\top x \leq 0$ , poni  $w^{(t+1)} \leftarrow w^{(t)} + \eta \cdot y \cdot x$

# Algoritmo del Perceptrone

## Algoritmo del Perceptrone

- 1 Inizializza  $w^{(1)} = 0_{(d+1) \times 1}$
- 2 Per  $t = 1, 2, \dots$ , considera ciclicamente ogni esempio  $(x, y)$ :
  - Se  $y \cdot w^{(t)\top} x \leq 0$ , aggiorna  $w^{(t+1)} \leftarrow w^{(t)} + \eta y \cdot x$

# Algoritmo del Perceptrone



# Convergenza del Percettrone

## Teorema (Convergenza del Percettrone)

Se il training set è **linearmente separabile**:

- L'algoritmo del Percettrone trova un'ipotesi con rischio empirico pari a **zero** (= separa perfettamente gli esempi di training)
- L'algoritmo converge in un numero **finito** di passi

# Forma duale del Perceptrone

## Algoritmo del Perceptrone

- 1 Inizializza  $w = 0_{(d+1) \times 1}$
- 2 Per  $t = 1, 2, \dots$ , considera ciclicamente ogni esempio  $(x, y)$ :
  - Se  $(x, y)$  è misclassificato, aggiorna  $w \leftarrow w + \eta y \cdot x$

Quindi l'output dell'algoritmo avrà la forma

$$w = \eta \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$$

dove  $\alpha_i$  è il # di volte che l'esempio  $i$ -esimo ha causato un aggiornamento

⇒ Possiamo rappresentare  $w$  tramite  $\alpha = (\alpha_1, \dots, \alpha_m)$  (variabili *duali*)

# Forma duale del perceptrone

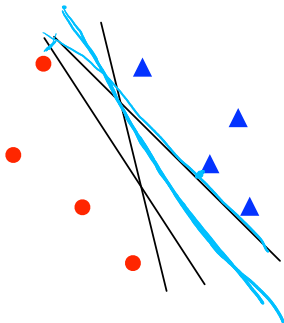
## Algoritmo del Perceptrone (forma primale)

- 1 Inizializza  $w = 0_{(d+1) \times 1}$
- 2 Per  $t = 1, 2, \dots$ , considera ciclicamente ogni esempio  $(x^{(i)}, y^{(i)})$ :
  - Se  $(x^{(i)}, y^{(i)})$  è misclassificato, aggiorna  $w \leftarrow w + \eta y^{(i)} \cdot x^{(i)}$

## Algoritmo del Perceptrone (forma duale)

- 1 Inizializza  $\alpha = 0_{m \times 1}$
- 2 Per  $t = 1, 2, \dots$ , considera ciclicamente ogni esempio  $(x^{(i)}, y^{(i)})$ :
  - Se  $(x^{(i)}, y^{(i)})$  è misclassificato, poni  $\alpha_i \leftarrow \alpha_i + 1$
- 3 Restituisci  $w = \eta \sum_i \alpha_i y^{(i)} x^{(i)}$

# Oltre il Percettrone



**Domanda:** Possiamo selezionare il separatore più “robusto”?

## Verso una separazione robusta

**Dati:**  $m$  esempi  $(x, y) \in \mathbb{R}^{d+1} \times \{-1, +1\}$

**Trova:**  $w \in \mathbb{R}^{d+1}$  tale che

$$y \cdot w^\top x > 0 \quad \text{per ogni esempio } (x, y)$$

$+1 \cdot \text{😊} > 0$   
 $-1 \cdot \text{☹️} < 0 \Rightarrow \delta > 0$

Scalando  $w$ , è equivalente a richiedere

$$y \cdot w^\top x \geq 1 \quad \text{per ogni esempio } (x, y)$$

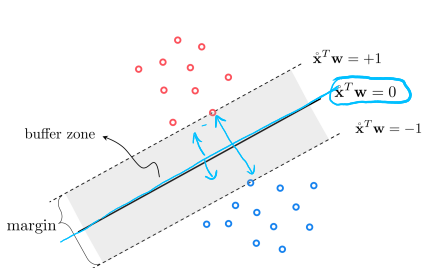
$w' \rightsquigarrow w/\delta$



# Margine di un separatore lineare

## Margine di un separatore

Il *margin* di  $w \in \mathbb{R}^{d+1}$  è  $2 / \|(w_1, \dots, w_d)\| = 2 / \|\omega\|$



$\min \|\omega\|_2$

massimizzare il margine  $\equiv$  minimizzare  $\sqrt{w_1^2 + w_2^2 + \dots + w_d^2}$

# Separazione a massimo margine

## *Hard-margin Support Vector Machine* (Hard SVM)

$$\begin{aligned} & \underset{w \in \mathbb{R}^{d+1}}{\text{minimize}} \quad w_1^2 + w_2^2 + \dots + w_d^2 \\ & \text{subject to} \quad \underline{y \cdot w^\top x} \geq 1 \quad \leftarrow \quad \text{per ogni esempio } (x, y) \end{aligned}$$

È un problema di minimizzazione convessa vincolata:

- funzione obiettivo convessa
- vincoli lineari

⇒ è risolvibile in modo efficiente

**Importante:** ha soluzione **solo** se gli esempi sono linearmente separabili

## Hard SVM: formulazione alternativa

$$w = (w_1, \dots, w_d)$$

$$\|w\| = \sqrt{w_1^2 + \dots + w_d^2}$$

$$\text{minimize}_{w \in \mathbb{R}^{d+1}} \|w\|_2^2$$

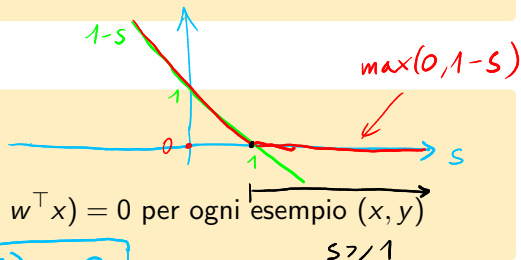
$$\text{subject to } \underbrace{y \cdot w^T x}_s \geq 1$$

per ogni esempio  $(x, y)$ 

è equivalente a

$$\text{minimize}_{w \in \mathbb{R}^{d+1}} \|w\|_2^2$$

$$\text{subject to } \max(0, 1 - y \cdot w^T x) = 0 \text{ per ogni esempio } (x, y)$$



$$\max(0, 1-s) = 0$$

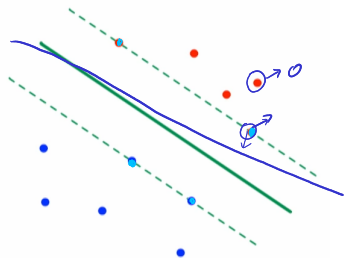
ReLUHinge

# Vettori di supporto

Si può dimostrare che la soluzione ottima ha la forma

$$w^* = \sum_{i=1}^m \alpha_i \cdot y^{(i)} \cdot x^{(i)}$$

con  $\alpha_i \neq 0$  solo per gli  $x^{(i)}$  giacenti sul margine (*vettori di supporto*)



Ma cosa fare se il dataset **non** è linearmente separabile?

# Il caso non separabile

Introduciamo delle variabili di *slack* (“allentamento”):

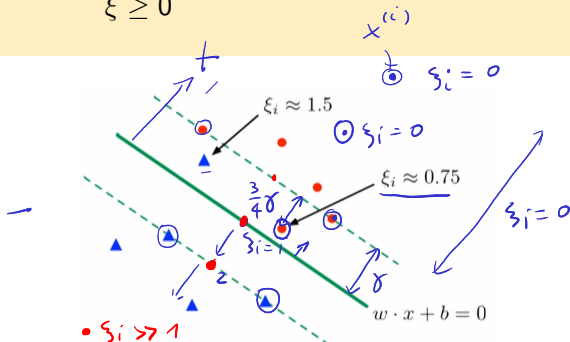
## Soft-Margin Support Vector Machine (Soft SVM)

$$\underset{w}{\text{minimize}} \quad \|\omega\|_2^2 + C \sum_{i=1}^m \xi_i$$

iperparametro

subject to  $y^{(i)} \cdot w^\top x^{(i)} \geq 1 - \xi_i$  per  $i = 1, 2, \dots, m$

$$\xi \geq 0$$



# Compromesso tra margine e slack

Che ruolo gioca l'iperparametro  $C$ ?

## Soft-Margin Support Vector Machine

$$\begin{aligned} & \underset{w}{\text{minimize}} \quad \|\omega\|_2^2 + C \sum_{i=1}^m \xi_i \\ & \text{subject to} \quad y^{(i)} \cdot w^\top x^{(i)} \geq 1 - \xi_i \text{ per } i = 1, 2, \dots, m \\ & \quad \quad \quad \xi \geq 0 \end{aligned}$$

$C = 0$ : restituisce  $w^* = 0$  (gli allentamenti non sono penalizzati)

$C \rightarrow \infty$ : equivalente a Hard-margin SVM

## Soft-SVM: formulazione alternativa

$$\underset{w \in \mathbb{R}^{d+1}}{\text{minimize}} \quad \|\omega\|_2^2 + C \sum_{i=1}^m \xi_i$$

$$\text{subject to } y^{(i)} \cdot w^\top x^{(i)} \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

per ogni esempio  $(x^{(i)}, y^{(i)})$

$$\rightarrow \begin{cases} \xi_i \geq 1 - y^{(i)} \cdot w^\top x^{(i)} \\ \xi_i \geq 0 \end{cases}$$

$$\xi_i = \max(0, 1 - y^{(i)} w^\top x^{(i)})$$

è equivalente a

$$\underset{w \in \mathbb{R}^{d+1}}{\text{minimize}} \quad \|\omega\|_2^2 + C \sum_{i=1}^m \xi_i$$

$$\text{subject to } \max(0, 1 - y^{(i)} \cdot w^\top x^{(i)}) = \xi_i \text{ per ogni esempio } (x^{(i)}, y^{(i)})$$

Esiste anche la variante con softmax (log loss) al posto di max (hinge loss)

# Percettrone e SVM in scikit-learn

Approccio	Iperparametri	Interfaccia scikit-learn
Percettrone	$T$ $T; \underline{\eta} = 1$	Perceptron(max_iter) SGDClassifier(loss='perceptron', max_iter, penalty=None, learning_rate='constant', eta0=1)
Soft SVM (hinge loss)	$C$ $\alpha (= \frac{1}{C}), T, \underline{\eta}$	LinearSVC(C) oppure SVC(C) SGDClassifier(loss='hinge', max_iter, <i>alpha</i> , penalty='l2', learning_rate)
Soft SVM (log loss)	$\alpha (= \frac{1}{C}), T, \underline{\eta}$	SGDClassifier(loss='log', max_iter, alpha, penalty='l2', learning_rate)

$\max(0, -y \cdot w^T x)$   
 ("hinge")  
 "perceptron"

$$\text{softmax}(0, 1 - y \cdot w^T x)$$

↓  
log loss

$$\max(0, 1 - y \cdot w^T x)$$

↓  
hinge loss



## Classificazione binaria: metriche di qualità

## Doppio ruolo delle funzioni di costo

La funzione di costo per **misurare la qualità** delle predizioni nei problemi di classificazione è in genere la funzione costo 0-1:

$$\ell(h, (x, y)) = \begin{cases} 0 & \text{se } h(x) = y \\ 1 & \text{se } h(x) \neq y \end{cases}$$

⇒ **Accuratezza**: frazione di nuovi esempi correttamente classificati

La funzione di costo per **apprendere il modello** è un suo **surrogato**, per motivi computazionali:

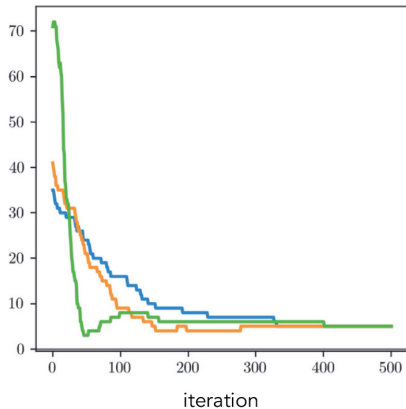
- Softmax/log-loss (regressione logistica)
- Hinge loss (percettrone)
- ...

Il modello va **valutato** sulla funzione originale, non sul surrogato!

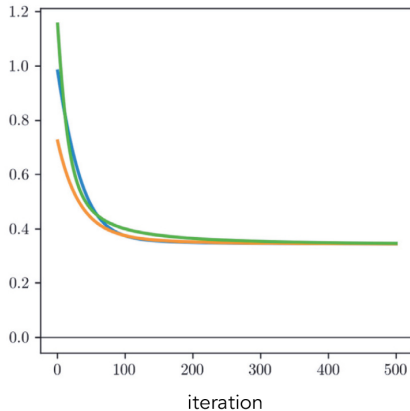
⇒ utilizziamo l'**accuratezza** come metrica di qualità

# Esempio

number of misclassifications



Softmax cost value



# Inconvenienti dell'accuratezza come metrica

L'(in)accuratezza a volte può presentare inconvenienti

- Quando diversi tipi di misclassificazione hanno costo molto diverso
- Se c'è uno **sbilanciamento** tra le classi, in cui i casi positivi (o quelli negativi) sono estremamente rari

Esempio: diagnosi di una malattia rara ( $<1$  per mille della popolazione)  
Un classificatore che restituisce sempre **NO** ha accuratezza 99.9%, ma è del tutto inutile

# Matrice di confusione

Per problemi con forte sbilanciamento, è utile separare i tipi di errore attraverso una *matrice di confusione*

	$y = 1$	$y = 0$
	<i>Veri Positivi</i>	<i>Falsi Positivi</i>
$\hat{y} = 1$	Abbiamo urlato al lupo! Abbiamo salvato il villaggio.	Errore: il lupo non c'era. Abbiamo innervosito tutti.
	<i>Falsi Negativi</i>	<i>Veri Negativi</i>
$\hat{y} = 0$	C'era un lupo, ma non lo abbiamo stanato. Ha mangiato tutto il pollame.	Nessun lupo, nessun allarme. Tutto tranquillo.

# Accuratezza e matrice di confusione

Accuratezza [accuracy]

$$\mathcal{A} \stackrel{\text{def}}{=} \frac{VP + VN}{VP + VN + FP + FN}$$

# Sensibilità, specificità e precisione

## *Sensibilità* [sensitivity, recall]

$$\text{sensibilità} \stackrel{\text{def}}{=} \frac{VP}{VP + FN} = \mathcal{A}_{y=1}$$

## *Specificità* [specificity]

$$\text{specificità} \stackrel{\text{def}}{=} \frac{VN}{VN + FP} = \mathcal{A}_{y=0}$$

## *Precisione* [precision]

$$\text{precisione} \stackrel{\text{def}}{=} \frac{VP}{VP + FP} = \mathcal{A}_{\hat{y}=1}$$

# Accuratezza bilanciata

## Accuratezza bilanciata

$$\begin{aligned}
 \mathcal{A}_{\text{balanced}} &\stackrel{\text{def}}{=} \frac{\mathcal{A}_{y=1} + \mathcal{A}_{y=0}}{2} \\
 &= \frac{1}{2} \text{sensibilità} + \frac{1}{2} \text{specificità} \\
 &= \frac{1}{2} \frac{VP}{VP + FN} + \frac{1}{2} \frac{VN}{VN + FP}
 \end{aligned}$$

**Attenzione.** La formula nel libro di testo (Watt et al.) non è corretta (confonde la sensibilità con la precisione)



# Esempio (Settembre 2020)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJEMB.2020.3026928, IEEE Open Journal of Engineering in Medicine and Biology

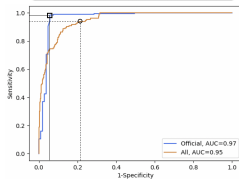
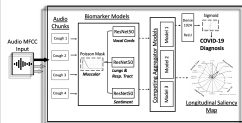
## COVID-19 Artificial Intelligence Diagnosis using only Cough Recordings

### COVID-19 Cough Test



-  Non-invasive
-  Essentially free
-  Unlimited throughput
-  Real-time results
-  Longitudinally monitor




### AI Discrimination Model



### Performance

- 98.5% sensitivity - 94.2% specificity on PCR/serology confirmed subjects
- 100% Asymptomatic detection rate

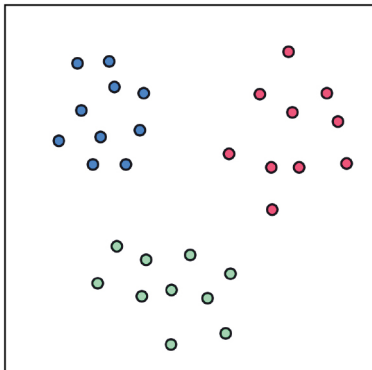
### Use-Cases

-  Daily Country-Wide Screening
-  Outbreak Monitoring
-  Test Pooling Candidate Selection

# Classificazione multiclasse

# Regressione logistica multiclasse

Come trattare il caso di  $K$  classi con  $K > 2$ ?

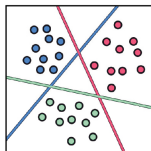
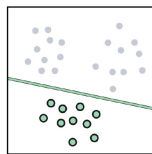
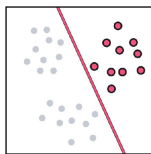
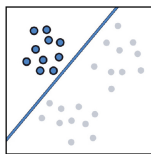


- 1 Approccio *one vs. rest* (applichiamo  $K$  volte un classificatore binario)
- 2 Approccio *multinomiale* (vera generalizzazione a  $K$  categorie)

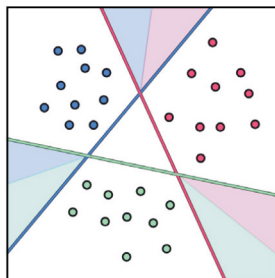
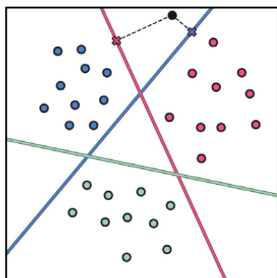
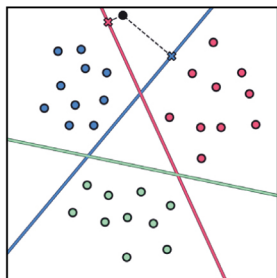
# One vs. rest

## Approccio *One vs. rest*

- 1 Apprendi  $K$  ipotesi  $h^{(1)}, \dots, h^{(K)}$ , dove la  $j$ -esima ipotesi distingue la classe  $j$  dalle altre  $K - 1$  classi
- 2 Dato  $x$ , restituisci la classe  $j$  che massimizza  $h^{(j)}(x)$



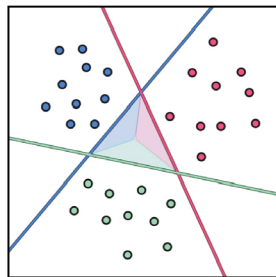
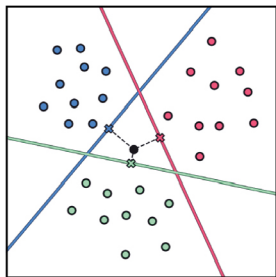
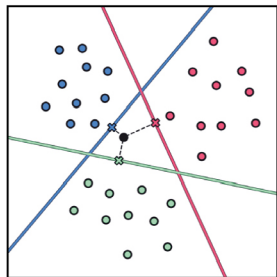
## One vs. rest



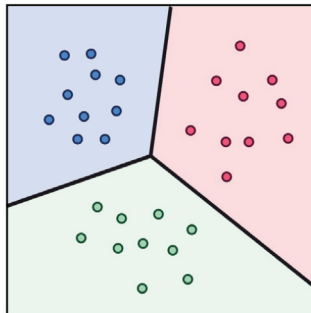
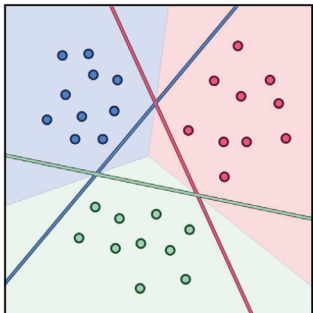
$$\operatorname{argmax}_{j=1}^K h^{(j)}(x) = \operatorname{argmax}_{j=1}^K \sigma(w^{(j)\top} x) = \operatorname{argmax}_{j=1}^K w^{(j)\top} x$$

**Importante:** assume che le componenti di  $w^{(j)}$  (eccetto  $w_0^{(j)}$ ) siano state normalizzate, in modo che  $\|w^{(j)}\| = 1$  per ogni  $j$

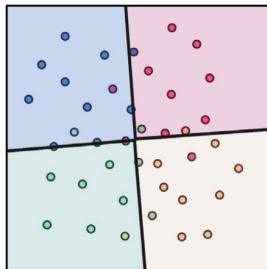
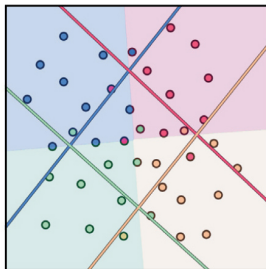
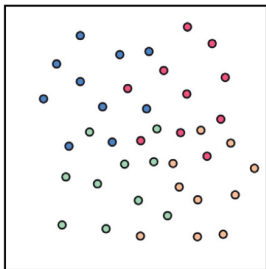
## One vs. rest



# One vs. rest



# One vs. rest





# Approccio multinomiale

## Approccio multinomiale (*Softmax multiclasse*)

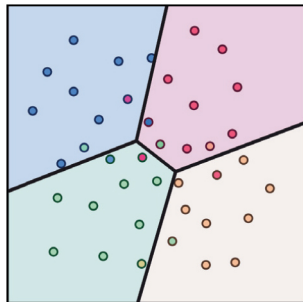
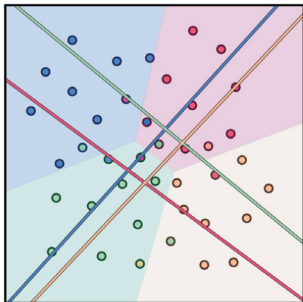
### 1 Assumi

$$\Pr(y = j|x) = \frac{\exp(w^{(j)\top} x)}{\sum_k \exp(w^{(k)\top} x)} = \frac{\exp(w^{(j)\top} x)}{\exp(\text{softmax}(w^{(1)\top} x, \dots, w^{(K)\top} x))}$$

2 Ottimizza i vettori  $w^{(1)}, \dots, w^{(K-1)}$  per massimizzare la likelihood (senza perdita di generalità,  $w^{(K)} = 0$ )

3 Dato  $x$ , restituisci la classe  $j$  che massimizza  $\Pr(y = j|x)$

# Approccio multinomiale



# One vs. rest o multinomiale? Confronto

