



SAPIENZA
UNIVERSITÀ DI ROMA

METODI DI INTELLIGENZA ARTIFICIALE E MACHINE LEARNING IN FISICA

S. Giagu - AA 2019/2020
Lezione 5: 11.3.2020

SUGGERIMENTO PER LA SOLUZIONE DEL PROBLEMA DEL COMMESO VIAGGIATORE CON L'ALGORITMO GENETICO

- Due passi fondamentali quando si applica un algoritmo genetico ad un problema che non è di natura genetica:
 - opportuna codifica delle soluzioni
 - opportuna definizione e vincoli associati sulle variazioni genetiche nelle soluzioni
- devono garantire una descrizione corretta e la più semplice del problema:
- devono garantire che tutti i vincoli del problema siano correttamente implementati
- una scelta semplice ed efficace per la codifica della soluzione nel problema specifico:
 - $s = [\text{indice_prima_città_visitata}, \text{indice_seconda_città_visitata}, \dots, \text{indice_D-esima_città_visitata}] = [1, 2, 3, 4, 5, \dots, D]$
 - con $D = \text{numero di città da visitare}$
 - in cui è imposto il vincolo che gli indici devono essere tutti diversi (non sono permessi indici ripetuti nello stesso vettore soluzione)
- Mutazioni genetiche permesse: due alternative:
 - o tutte, eliminando a posteriori le soluzioni che non rispettano il vincolo che ogni città possa essere visitata una sola volta
 - consentendo solo mutazioni che conservano il vincolo:
 - MUTAZIONI -> solo SWAP mutations: $[1, 2, 3, 4, 5, 6] \rightarrow [1, 5, 3, 4, 2, 6]$
 - CROSS-OVER -> solo ORDERED x-over:

$$S_1 = [1, 2, 3, 4, 5, 6]$$

$$S_2 = [6, 4, 1, 5, 2, 3]$$

parent
solutions

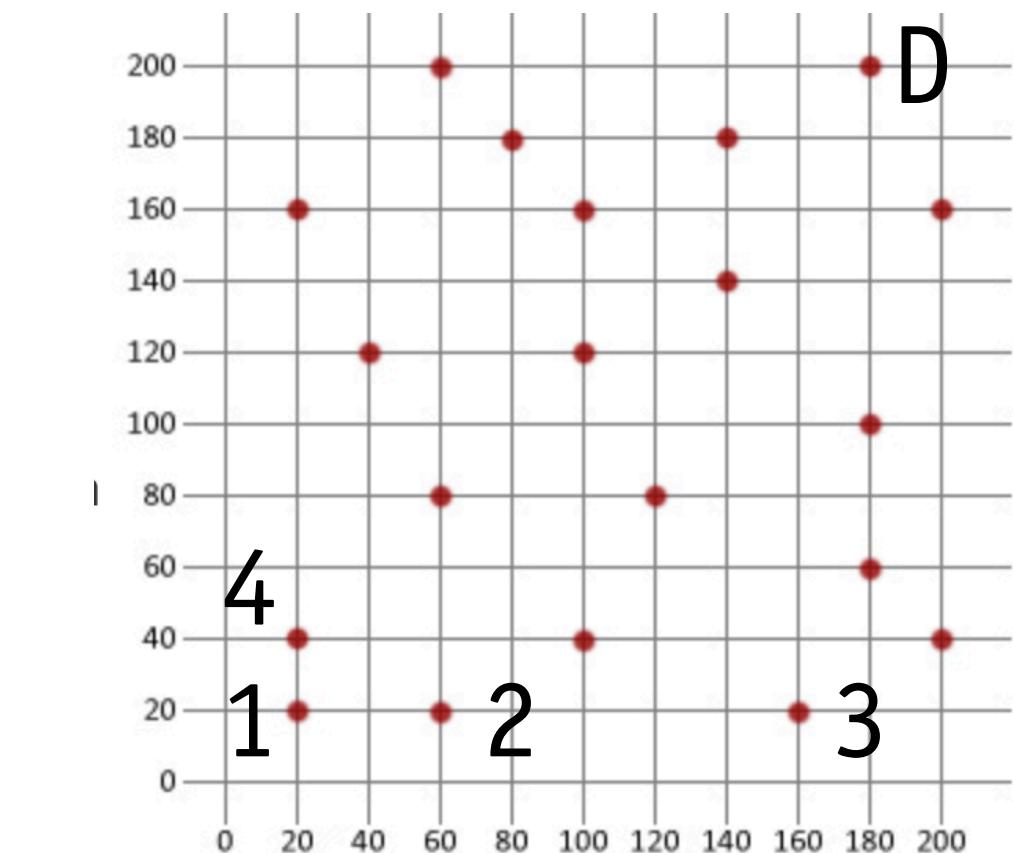
$$S_1 = [1, 2, 3, 4, 5, 6]$$

$$S_2 = [6, 4, 1, 5, 2, 3]$$

6, 4 e 1 non presenti nel
segmento selezionato di S_1

$$S' = [6, 1, 3, 4, 5, 2]$$

offspring
solution



provate ad implementare
le due alternative
e confrontate i risultati

METODI GRADIENT-BASED

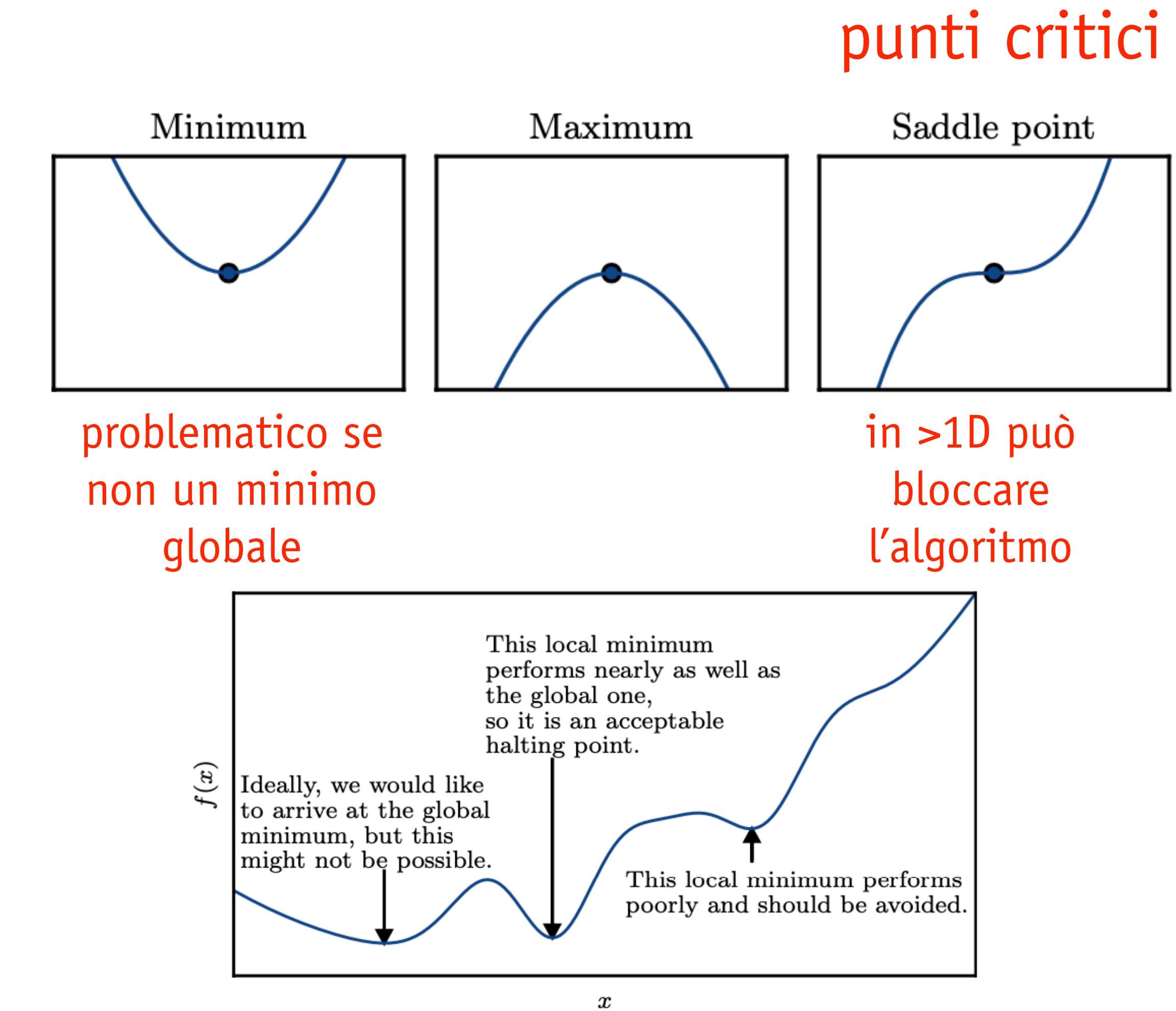
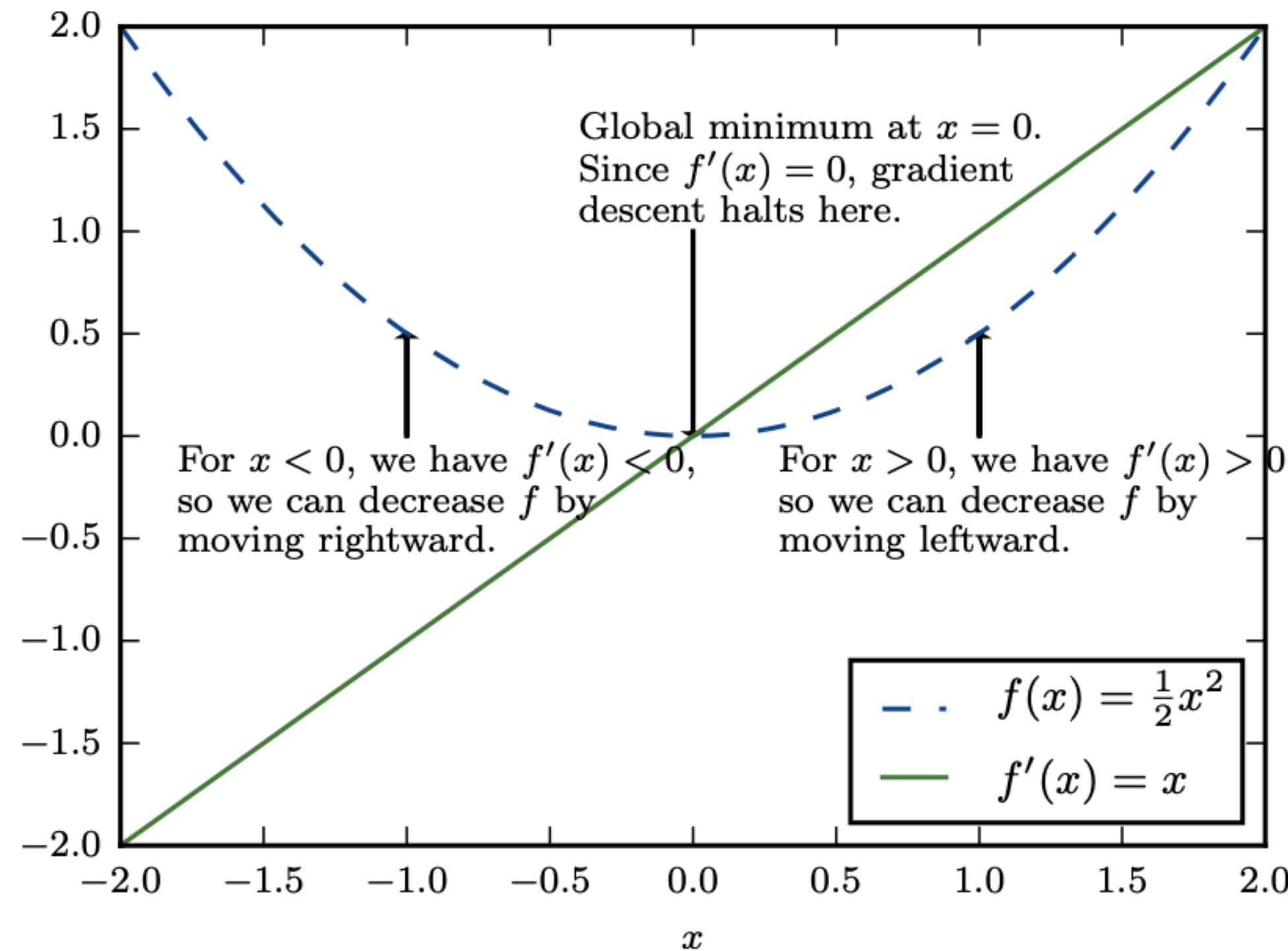
- Gradient descent:
 - algoritmo iterativo: per raggiungere il punto di minimo del criterio, a partire da una configurazione iniziale del vettore dei pesi, si modificano i valori dei pesi nella direzione in cui il criterio decresce più rapidamente
 - non sempre converge: può restare intrappolato in minimi locali
 - può risultare molto lento per grandi campioni e spazi ad alta dimensione (punti di sella spesso presenti)
- Stochastic Gradient Descent:
 - versione più efficiente e veloce del Gradient Descent
 - funziona da auto-regolarizzatore del modello



DISCESA LUNGO IL GRADIENTE (GD)

- goal: trovare il vettore \mathbf{w}^* che minimizza una funzione obiettivo (loss function o cost function) $L(\mathbf{w})$:

$$\mathbf{w}^* = \arg \min L(\mathbf{w})$$

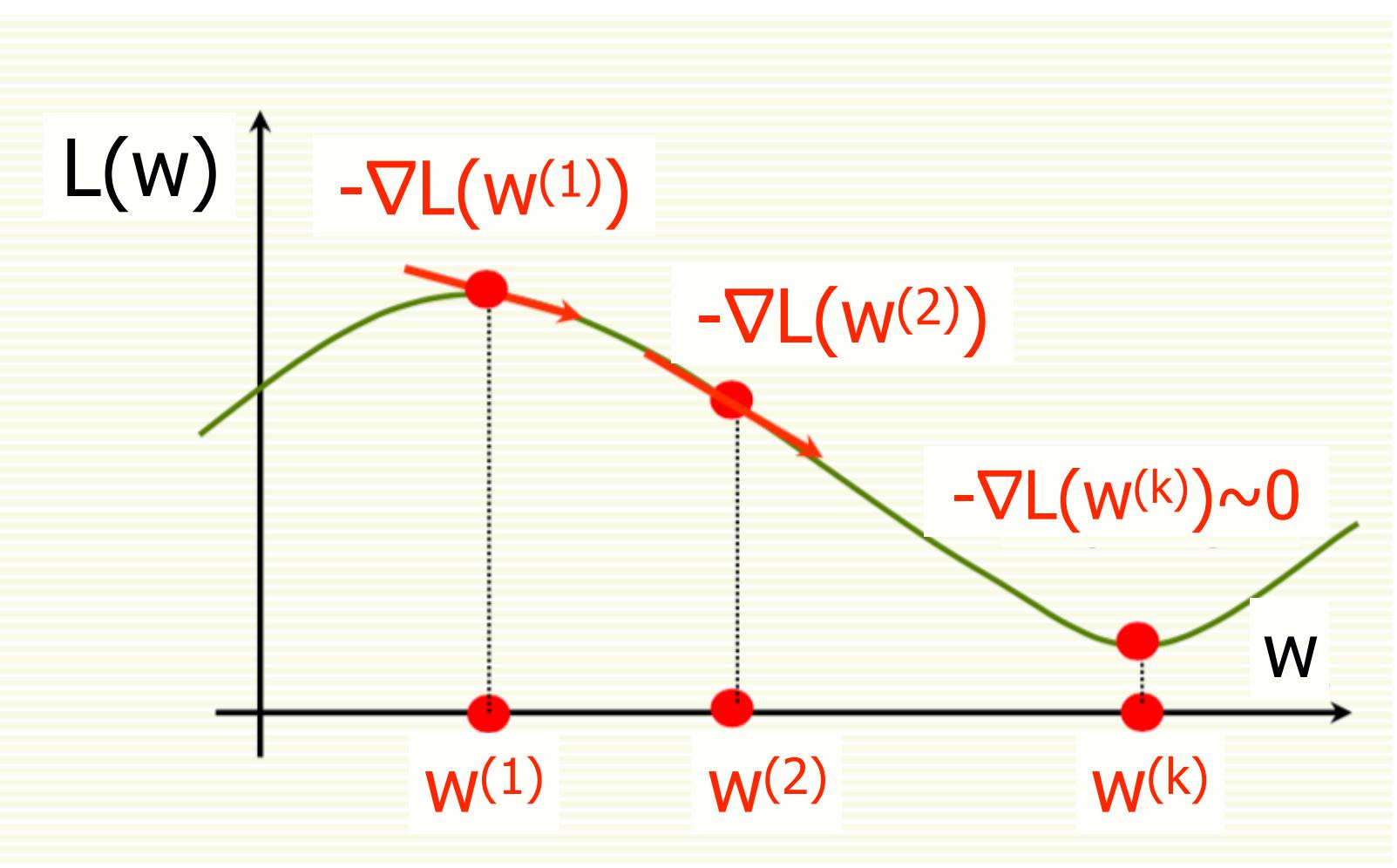


DISCESA LUNGO IL GRADIENTE

- In pratica per trovare \mathbf{w}^* che minimizza $L(\mathbf{w})$ si deve trovare il punto in cui si azzera il gradiente di $L(\mathbf{w})$:

\mathbf{w}^* tale che $\nabla_{\mathbf{w}} L(\mathbf{w})|_{\mathbf{w}^*} = 0$

```
k = 1  
w(1) = any initial guess  
choose η, ε  
while η ||∇L(w(k))|| > ε  
    w(k+1) = w(k) - η · ∇L(w(k))  
    k = k + 1
```



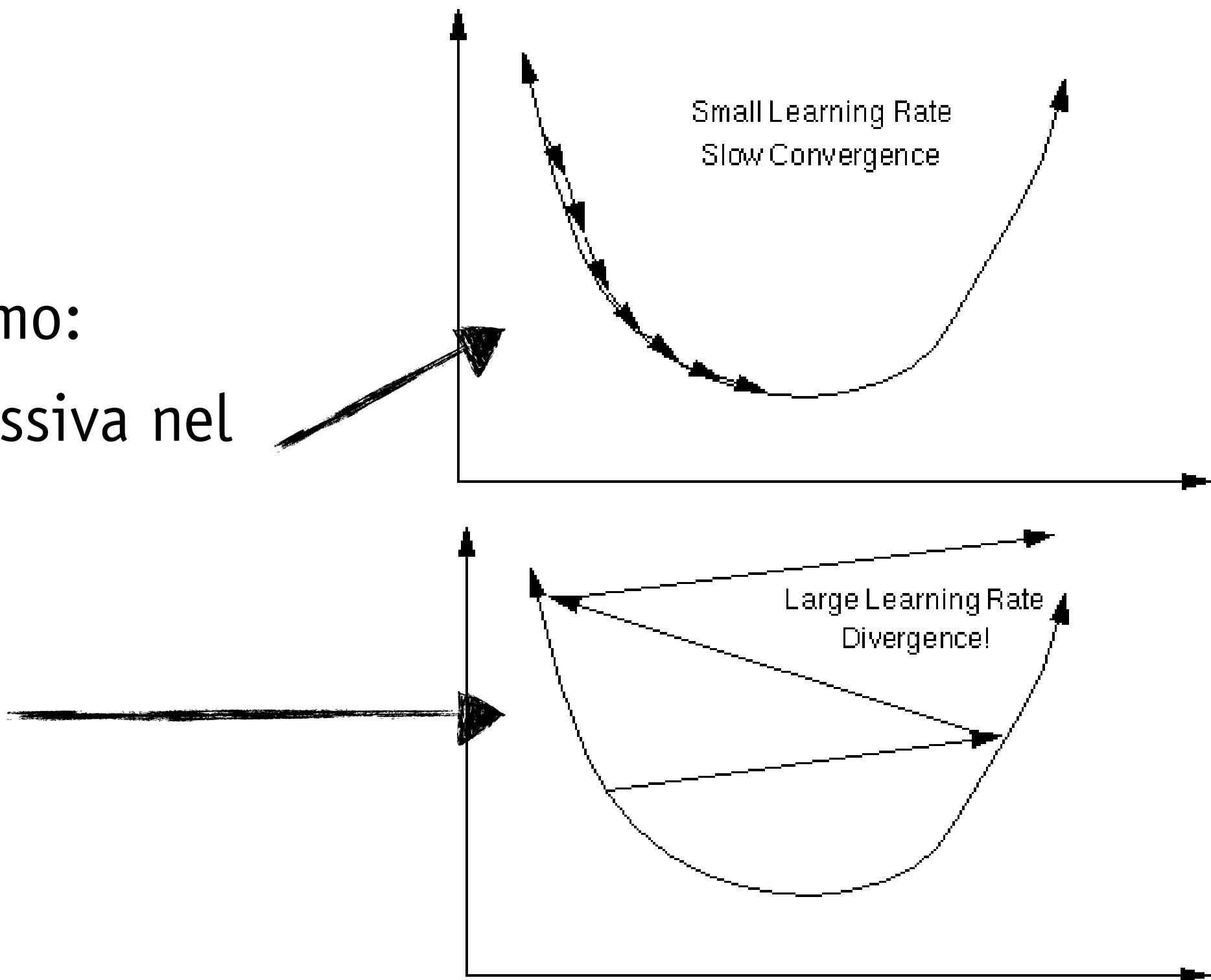
- Se alla generica iterazione k abbiamo un vettore dei pesi $\mathbf{w}(k)$ e il gradiente di L per quel vettore è $\nabla L(\mathbf{w}(k))$:
- il nuovo valore aggiornato per \mathbf{w} sarà:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta \cdot \nabla L(\mathbf{w}(k))$$

in cui η è una costante definita **tasso di apprendimento** che definisce l'ampiezza della modifica del vettore
per funzioni convesse smooth con unico minimo w^* dopo k -step si ha accuracy $L(\mathbf{w}(k)) - L(\mathbf{w}^*) = O(1/k)$

DISCESA LUNGO IL GRADIENTE

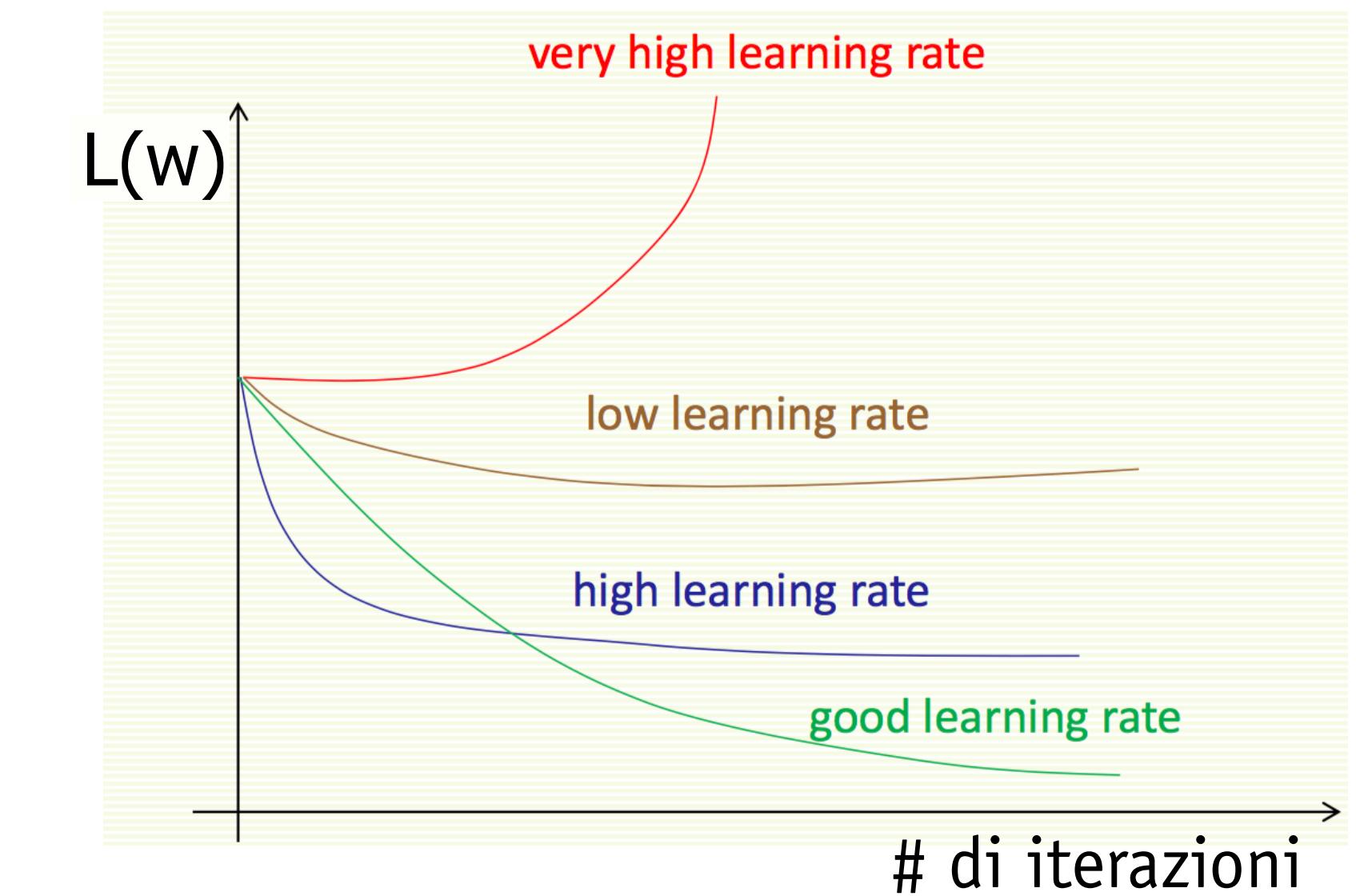
- il valore di η influenza sulla rapidità di convergenza dell'algoritmo:
 - un valore troppo piccolo può risultare in una lentezza eccessiva nel raggiungere il minimo e un aumento della probabilità di essere intrappolato in mini locali
 - un valore troppo grande può far divergere l'algoritmo ...



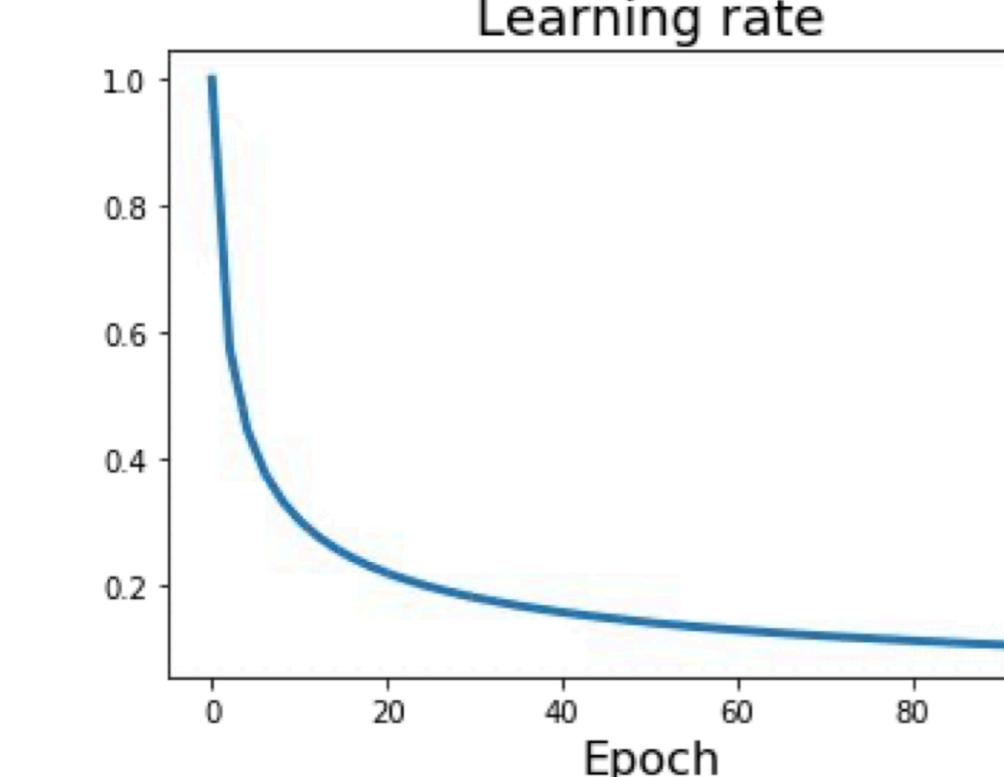
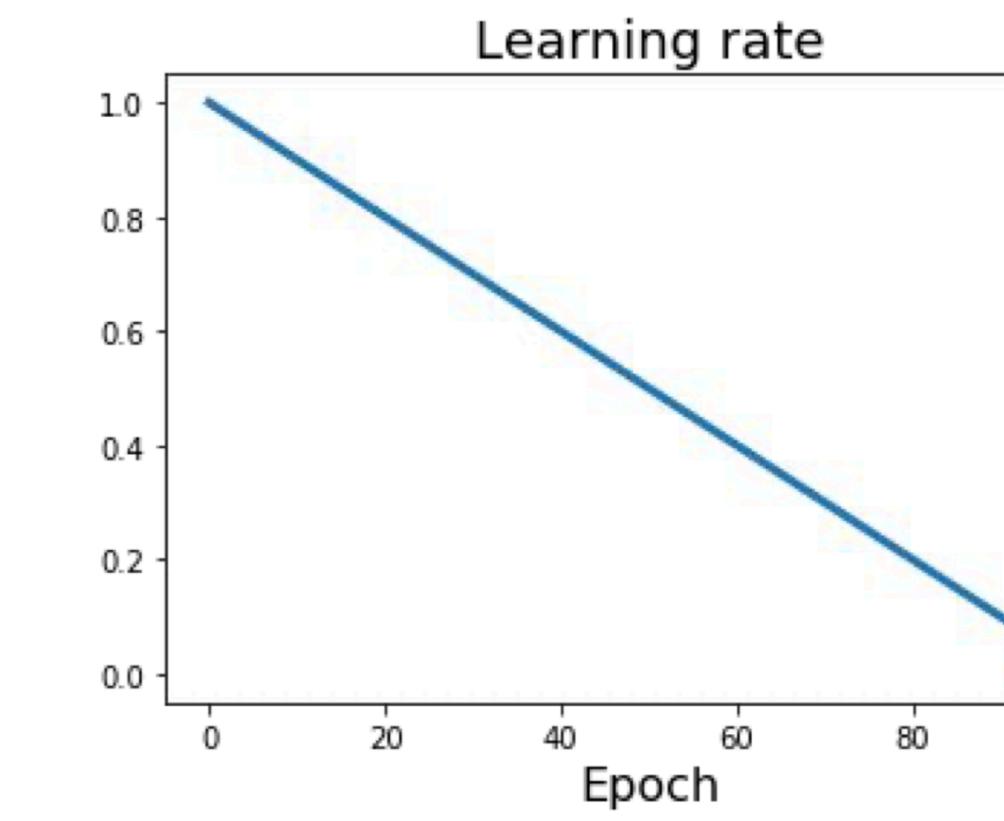
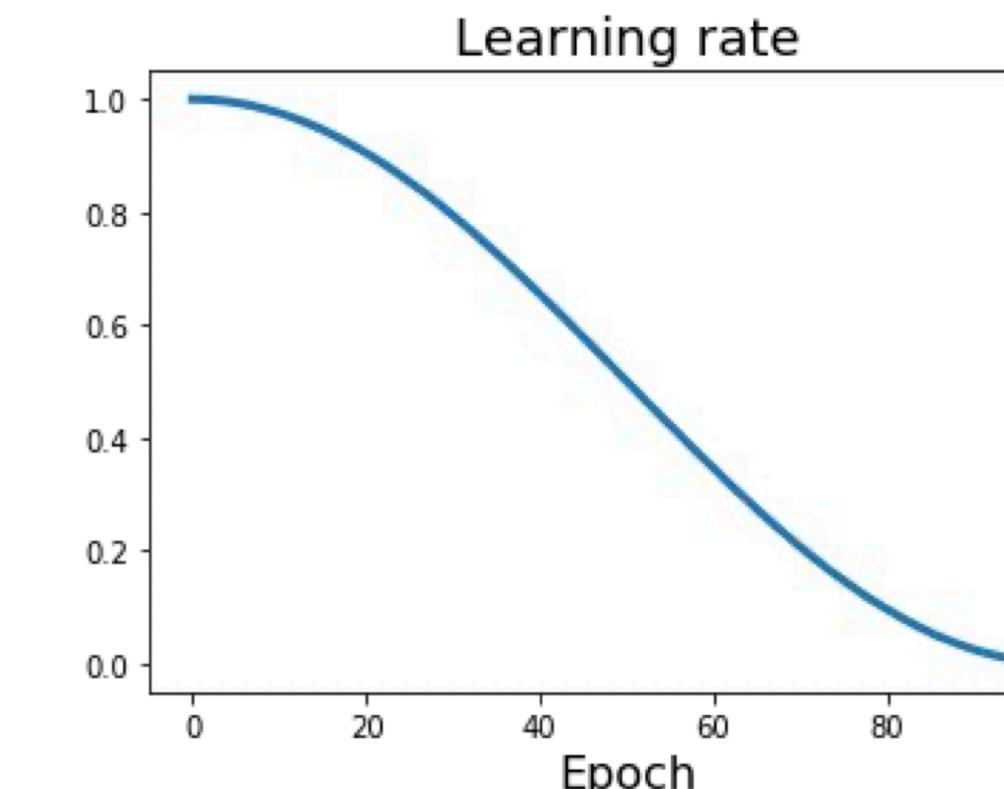
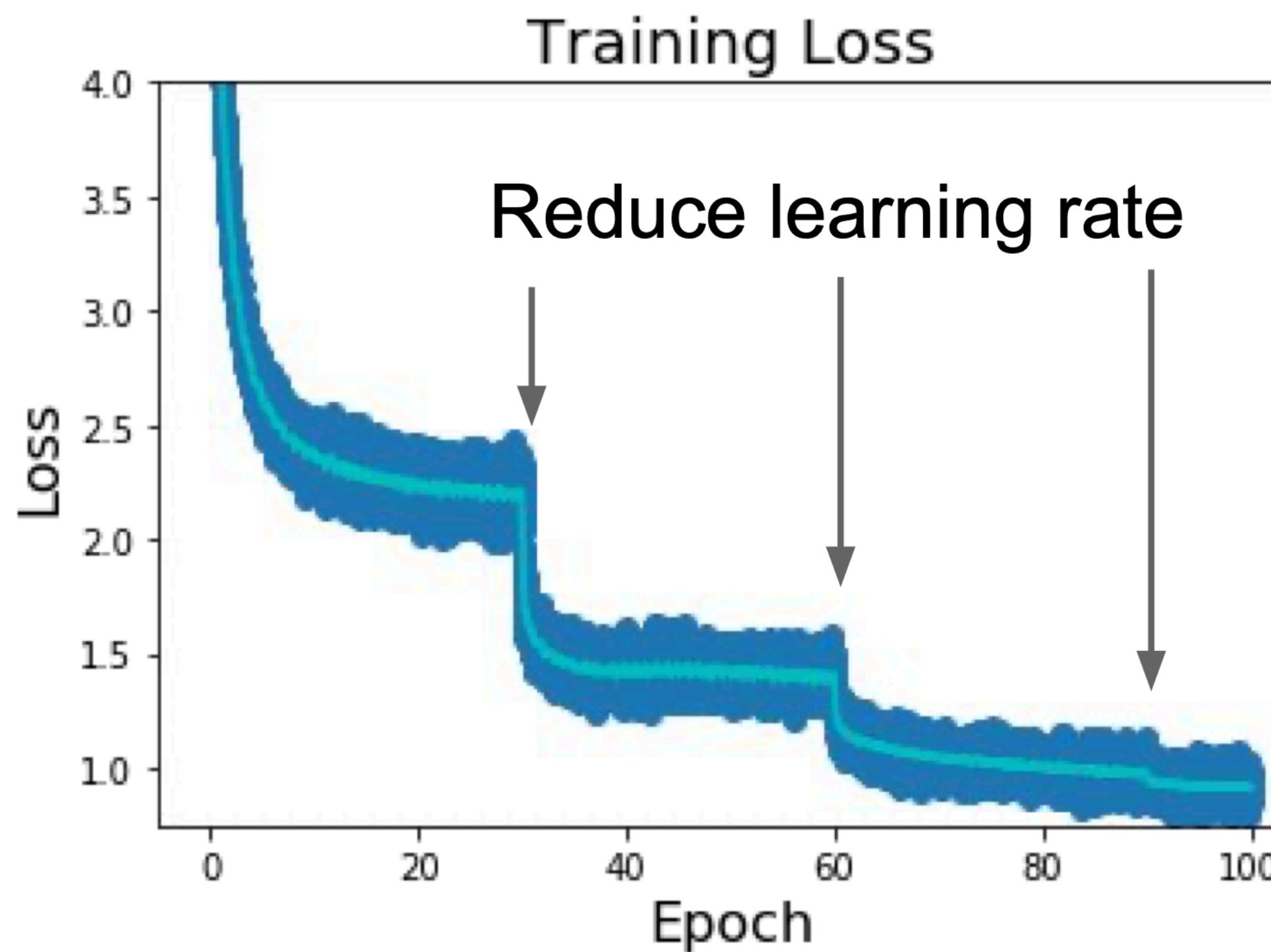
- soluzione: **Variable Learning Rate**

- durante le iterazioni il tasso di apprendimento viene monitorato guardando a quanto velocemente diminuisce la funzione obiettivo

```
k = 1  
w(1) = any initial guess  
choose ε  
while η || ∇L(w(k)) || > ε  
    choose η(k)  
    w(k+1) = w(k) - η(k) ∇L(w(k))  
    k = k + 1
```



LEARNING RATE DECAY



Cosine: $\alpha_t = \frac{1}{2}\alpha_0 (1 + \cos(t\pi/T))$

Linear: $\alpha_t = \alpha_0(1 - t/T)$

Inverse sqrt: $\alpha_t = \alpha_0/\sqrt{t}$

α_0 : Initial learning rate

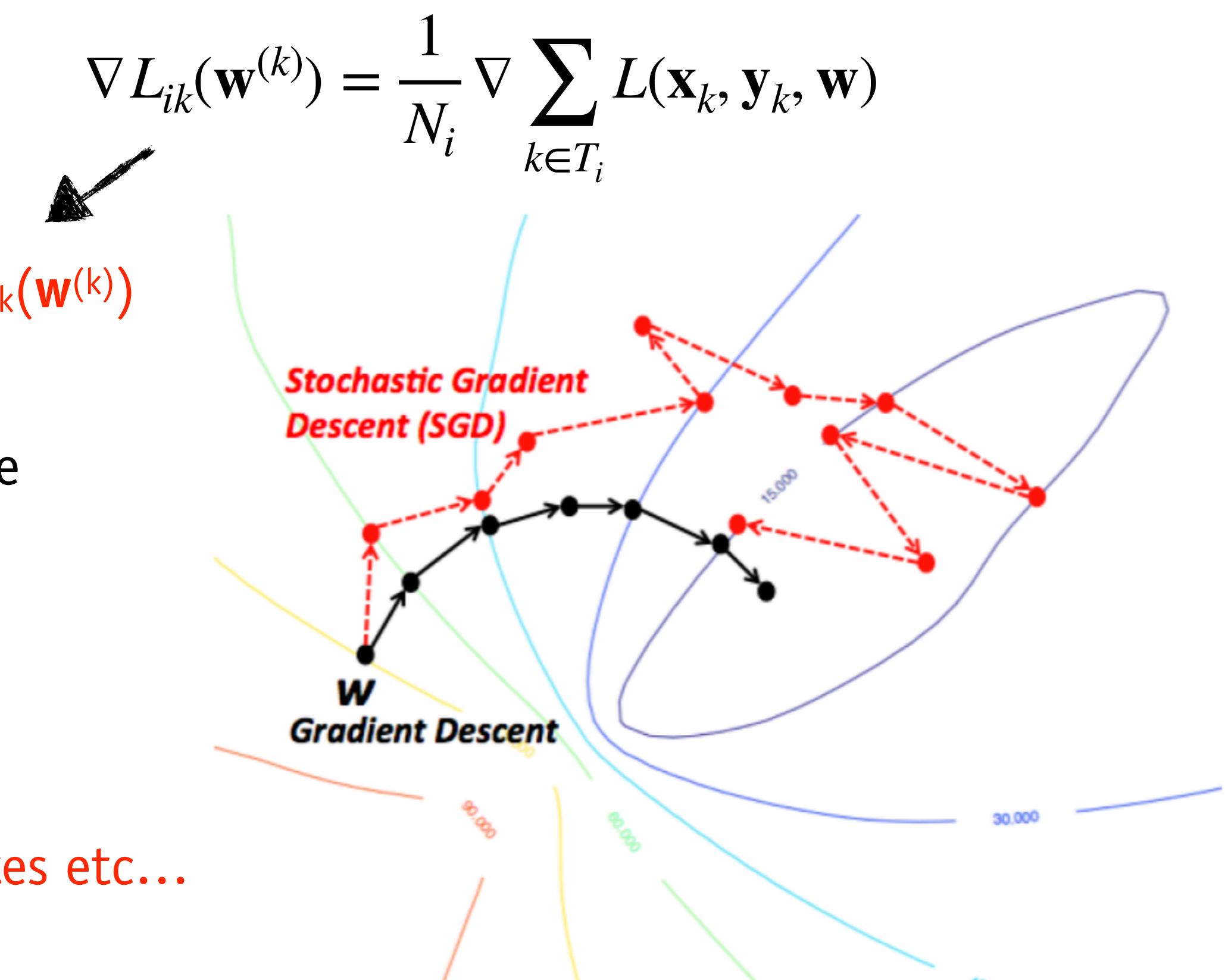
α_t : Learning rate at epoch t

T : Total number of epochs



STOCHASTIC GRADIENT DESCENT (SGD)

- per campioni di grande dimensione la procedura di aggiornamento ad ogni step dei pesi risulta computazionalmente molto inefficiente
- soluzione (**batch optimization**): nel calcolo del gradiente ad ogni iterazione viene usato un sottocampione diverso, al limite ciascun singolo evento del campione T:
 - si divide T in m sottocampioni $T_1 \dots T_m$
 - si sceglie in modo random uno di essi: $T_i \in \{T_1 \dots T_m\}$
 - si aggiornano i pesi con tale campione $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \cdot \nabla L_{ik}(\mathbf{w}^{(k)})$
- per funzioni convesse smooth con unico minimo w^* dopo k-step e utilizzando sotto-campioni fatti da singoli eventi si ha accuracy $(L(\mathbf{w}(k)) - L(w^*)) = O(1/\sqrt{k})$
- è l'algoritmo più usato in assoluto nel ML
- per migliorarne la convergenza: **momentum, adaptive learning rates etc...**



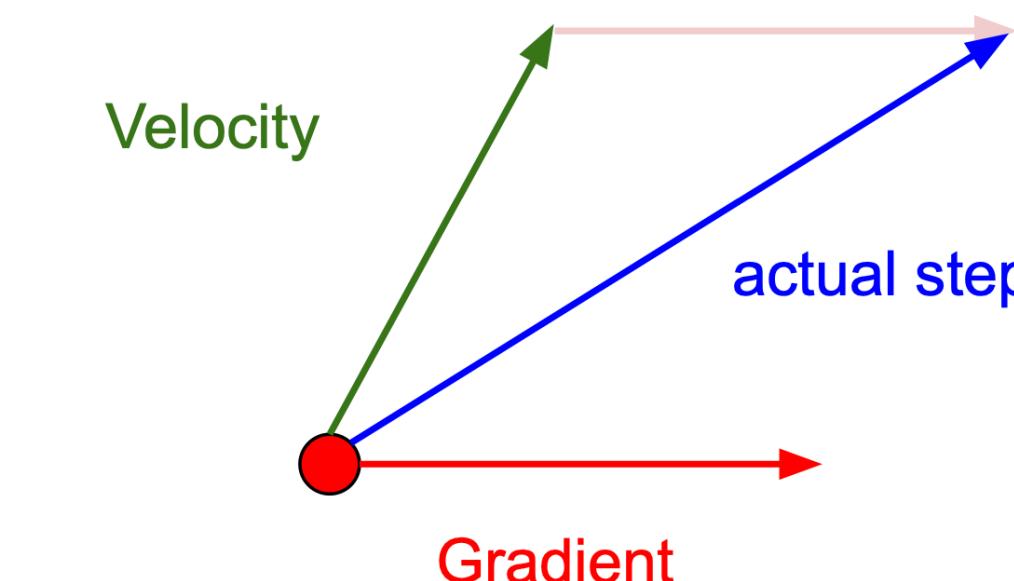
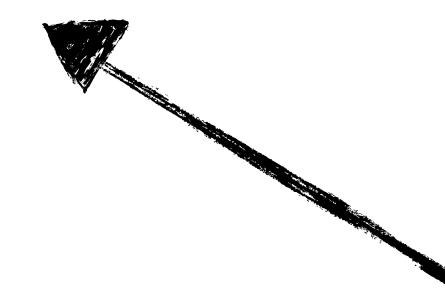
MOMENTUM

- la tecnica del “Momentum” è un metodo molto popolare per velocizzare la convergenza di GD e SGD in presenza di loss function a grande curvatura, punti di sella e regioni di plateau, e gradienti rumorosi
- durante l’aggiornamento del peso una frazione della direzione è basata sul risultato del precedente aggiornamento:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \mathbf{v}^{(k+1)}$$
$$\mathbf{v}^{(k+1)} = \alpha \mathbf{v}^{(k)} + (1 - \alpha) \nabla L(\mathbf{w}^{(k)})$$

direzione
precedente

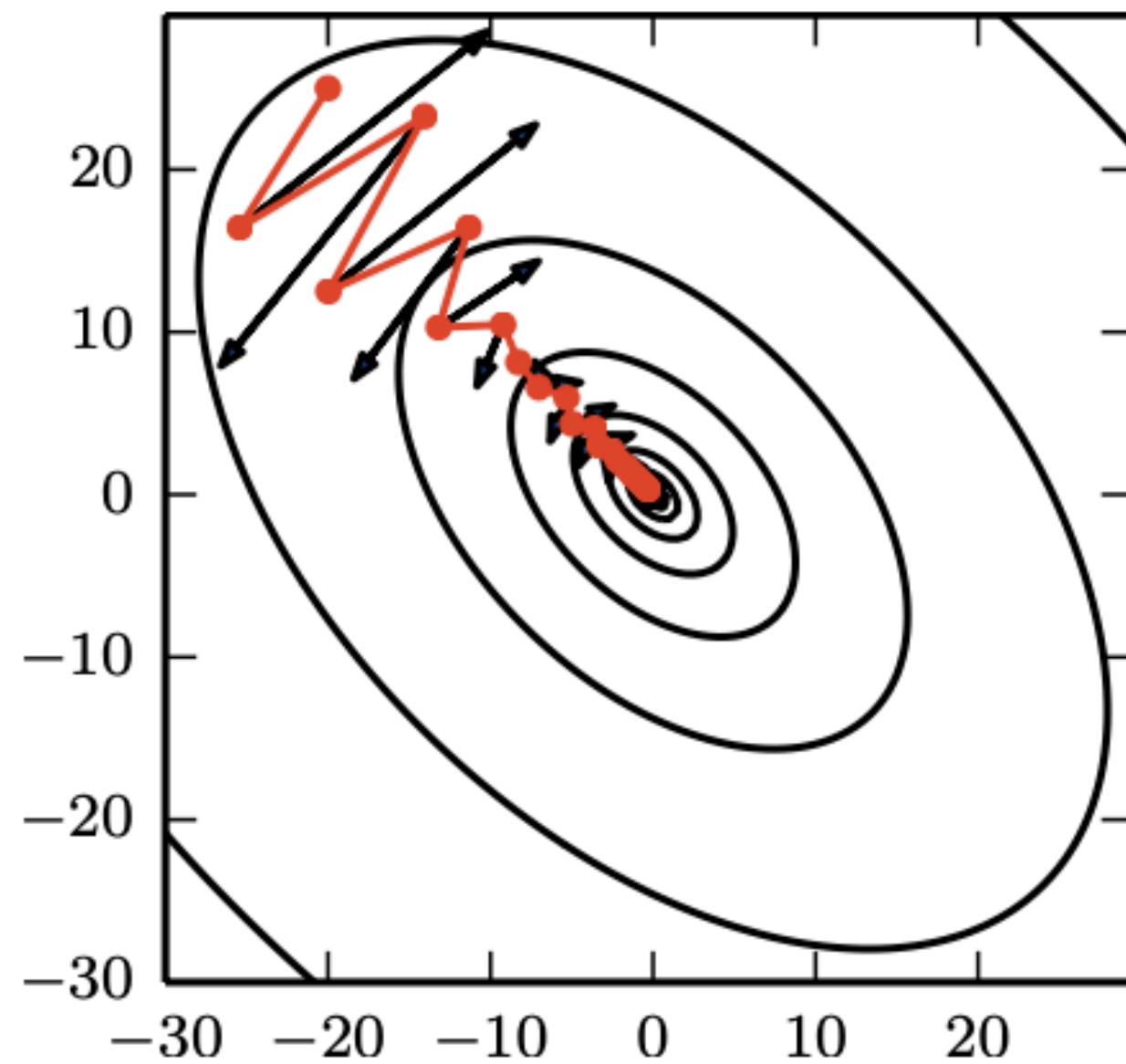
direzione del
gradiente



- v:** gioca il ruolo di una velocità
- velocità e direzione con cui parametri si muovono nello spazio dei parametri
 - costruita come running average dei gradienti

- α gioca il ruolo di una sorta di attrito
- per $\alpha=0$ GD/SGD classico
- for $\alpha=1$ la discesa lungo il gradiente viene ignorata e l’aggiornamento dei pesi avviene nella direzione dello step precedente (momentum)
- tipicamente: $\alpha \sim 0.9-0.99$

MOMENTUM



momentum risulta estremamente utile in caso di poor conditioning (i.e. cambiamenti rapidi rispetto a piccoli cambiamenti dell'input) della matrice hessiana

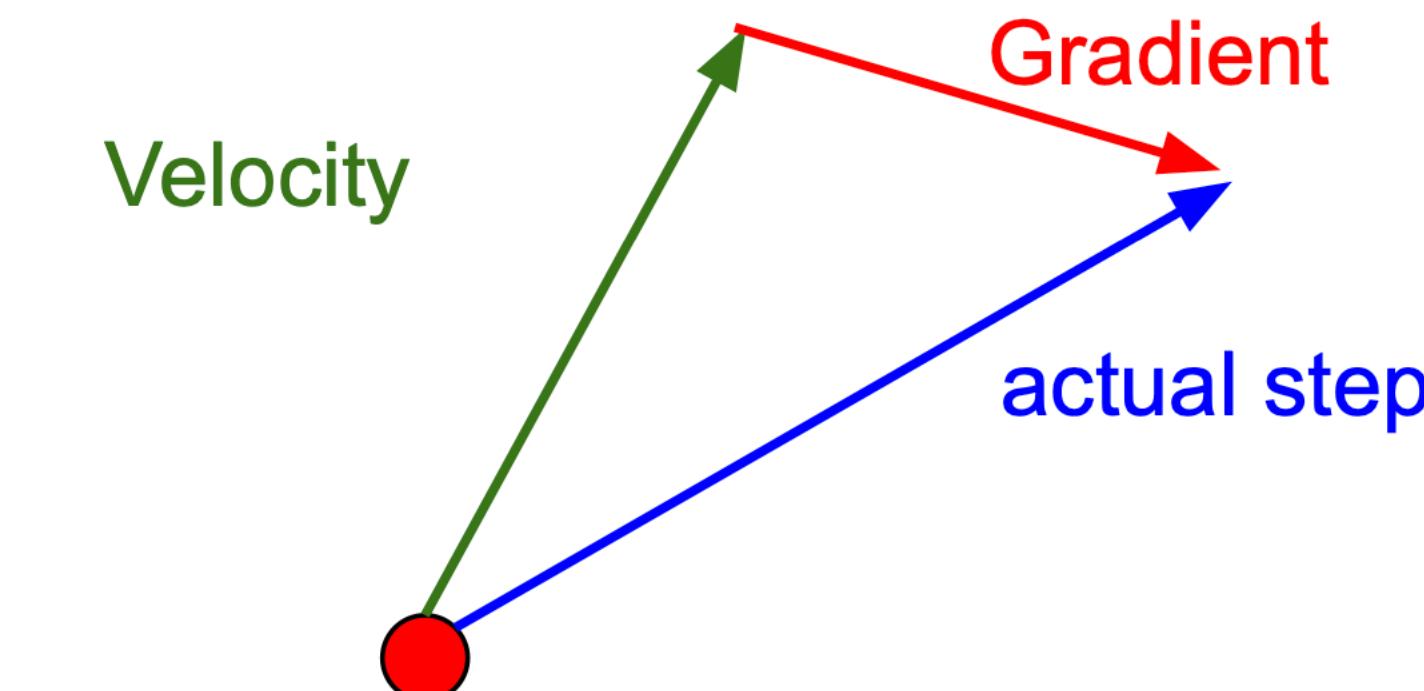
nell'esempio è riportata una funzione di perdita quadratica (MSE) con una matrice hessiana problematica

- frecce nere: direzione del gradiente ad ogni step → la funzione si comporta come una valle con pareti strette in cui il gradiente perde tempo salendo e scendendo dalle pareti
- cammino rosso: SDG con momentum → attraversa la valle correttamente

- **Variazione del metodo: NESTEROV MOMENTUM**

l'algoritmo “guarda avanti” al punto in cui ci porterebbe un update basato solo sulla velocità, calcola il gradiente in tale punto e lo mescola con la velocità per calcolare l'update dei pesi reale

in principio accelera la convergenza ma in pratica risulta utile solo in alcuni casi specifici (non con SGD)



LEARNING RATE ADATTIVO

- **Tecnica molto utile per velocizzare la convergenza (usata ovunque nei DNN)**

ADA grad: adatta il learning rate individualmente per ogni parametro:

- ▶ parameteri associati a features che compaiono con alta frequenza: riduce η
 - ▶ parameteri associati a features che compaiono con bassa frequenza: aumenta η
- in pratica il learning rate associato ad ogni parametro è scalato individualmente in modo inversamente proporzionale alla radice della somma storica dei quadrati dei gradienti per quel parametro
- un parametro con derivate parziali più grandi avrà learning rate più piccolo

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \frac{\partial L}{\partial w}$$

$$G_t = \sum_{\tau=1}^t \left[\frac{\partial L}{\partial w} \right]^2$$

Algorithm 8.4 The AdaGrad algorithm

Require: Global learning rate ϵ

Require: Initial parameter θ

Require: Small constant δ , perhaps 10^{-7} , for numerical stability

Initialize gradient accumulation variable $r = \mathbf{0}$

while stopping criterion not met **do**

 Sample a minibatch of m examples from the training set $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.

 Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$.

 Accumulate squared gradient: $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{g} \odot \mathbf{g}$.

 Compute update: $\Delta\theta \leftarrow -\frac{\epsilon}{\delta + \sqrt{\mathbf{r}}} \odot \mathbf{g}$. (Division and square root applied element-wise)

 Apply update: $\theta \leftarrow \theta + \Delta\theta$.

end while



LEARNING RATE ADATTIVO

RMSProp/Adadelta: variazione su ADA grad utile per mitigare il comportamento troppo aggressivo di ADA grad (riduce troppo velocemente il learning rate): restringe la finestra di gradienti accumulati per pesare il learning rate

RMSProp calcola un running average con pesi esponenzialmente decrescenti:

```
grad_squared = 0
while True:
    dx = compute_gradient(x)
    grad_squared += dx * dx
    x -= learning_rate * dx / (np.sqrt(grad_squared) + 1e-7)
```



```
grad_squared = 0
while True:
    dx = compute_gradient(x)
    grad_squared = decay_rate * grad_squared + (1 - decay_rate) * dx * dx
    x -= learning_rate * dx / (np.sqrt(grad_squared) + 1e-7)
```

Adam: RMSProp + momentum

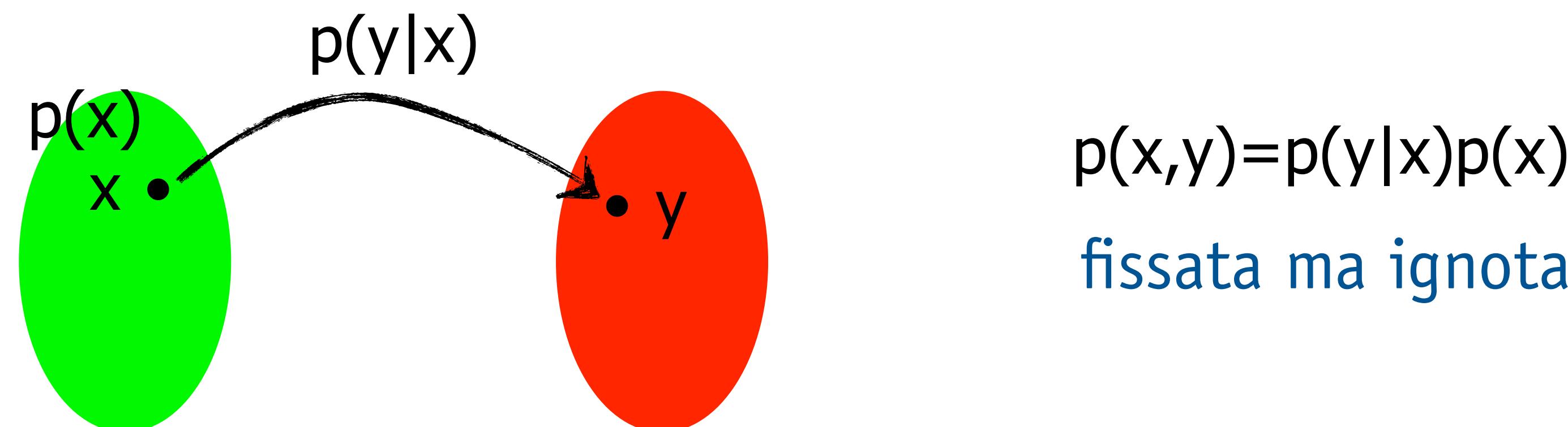
In pratica:

- Adam è la scelta di default migliore nella maggior parte dei casi
- SGD + momentum + cosine decay learning rate è in grado di fornire migliori prestazioni finali rispetto ad Adam ma si paga in termine di molto lavoro di tuning dei parametri della LR-schedule



APPRENDIMENTO STATISTICO

- inquadriamo ora da un punto di vista statistico il problema dell'apprendimento di un classificatore (*supervised learning*)
- il training set $T=\{(x_1, y_1), \dots, (x_n, y_n)\}$ può essere descritto come costituito da un insieme di coppie input-output (x, y) (eventi), estratte in modo indipendente e identicamente distribuito dall'insieme $X \times Y$ secondo una distribuzione di probabilità fissata (ma ignota)



- apprendimento supervisionato: usare T per costruire una funzione f_T che, applicata ad un valore di x non analizzato precedentemente, fornisca una “buona” predizione \hat{y} del vero output y corrispondente:

$$\hat{y} = f_T(x)$$

APPRENDIMENTO STATISTICO

- per scegliere tra tutte le possibili funzioni f_T è necessario definire un criterio di qualità da ottimizzare
- a tale scopo è naturale scegliere una funzione di costo (loss function) $L(y, f_T(x)) = L(x, y, T)$, che misuri la discrepanza tra il valore previsto ed il valore effettivo y :

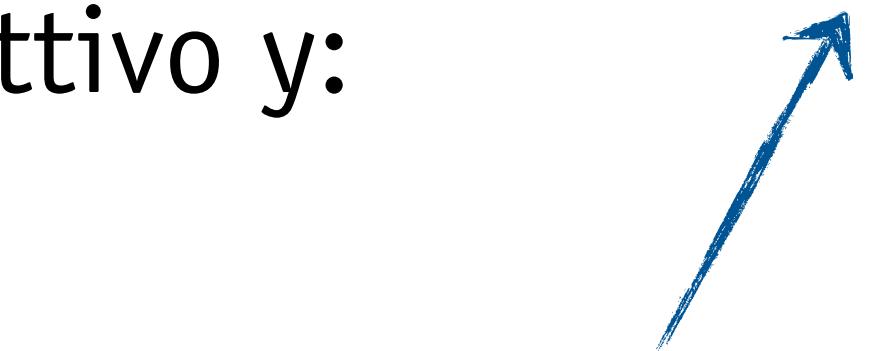
$$L(y, f_T(x)) : R^n \rightarrow [-1, 1] \quad y \in [-1, 1]$$

$$L(y, f_T(x)) = \begin{cases} 0 & \text{if } f_T(x) = y \\ 1 & \text{if } f_T(x) \neq y \end{cases}$$

Esempi:

$$L(y, f_T(x)) = \frac{1}{2} |f_T(x) - y|$$

$$L(y, f_T(x)) = \frac{1}{2} |f_T(x) - y|^2$$



L misura il costo di apprendere cose sbagliate



MAE e MSE (quadratic loss): usate tipicamente in NN, BDT, rule-based systems, random forests ...

è naturale usare come “criterio di qualità” per scegliere la funzione f_T per esempio il valore atteso dell’errore dovuto alla scelta di una particolare funzione di costo:

$$R(f_T) = \mathbb{E}[L(y, f_T(x))] = \int L(y, f_T(x)) dp(x, y)$$

RISCHIO STATISTICO



APPRENDIMENTO COME APPROXIMAZIONE

- l'obiettivo è quello di minimizzare $R(f_T)$
- il problema è che R non si può minimizzare in pratica perché $p(x,y)$ è sconosciuta
- scelta ragionevole: utilizzare una funzione che **approssimi** R e che richieda per essere calcolata solo l'uso dei dati disponibili:

$$R_{\text{emp}}(f_T) = \frac{1}{N} \sum L(y_i, f_T(x_i))$$

RISCHIO EMPIRICO

- dipende solo dai dati e dalla funzione f_T
- scopo dell'algoritmo di apprendimento è quello di individuare quella f_T che minimizza R_{emp} : $\min[R_{\text{emp}}(f_T)]$
- NOTA: la scelta potrà essere fatta solo tra l'insieme di funzioni che l'algoritmo di apprendimento è capace di implementare
→ possibilità di errore:
 - **errore di approssimazione:**
 - conseguenza della limitatezza dell'insieme di funzioni implementabili
 - **errore di stima (o di training):**
 - conseguenza della procedura di apprendimento che, scegliendo la funzione f_T sulla base del rischio empirico, non individua la funzione ottimale

←
←
errori di
generalizzazione



ELEMENTI DI TEORIA BAYESIANA DELLA DECISIONE (BAYESIAN LEARNING)

- La teoria bayesiana della decisione è un dei possibili approcci statistici al problema della classificazione
- obiettivo: confrontare quantitativamente diverse decisioni di classificazione utilizzando la probabilità ed i costi che accompagnano tali decisioni
- assunzione: sono noti i valori di tutte le probabilità rilevanti per il problema (non vera nella pratica reale)
- Consideriamo un generico problema a C classi (categorie) di appartenenza, con etichette ω_j {j=1,...,C}.
 - siano α_i {i=1,2,...,a} le decisioni che è possibile prendere
 - supponiamo di conoscere:
 - la probabilità $P(\omega_j)$ che un campione appartenga ad una certa classe (prob. a priori)
 - la funzione di costo $\lambda(\alpha_i|\omega_j)$ che descrive il costo indotto dall'aver preso la decisione α_i quando il campione appartiene alla classe ω_j



ELEMENTI DI TEORIA BAYESIANA DELLA DECISIONE

- se non avessimo altre informazioni, la regola di decisione dovrebbe essere basata interamente sulle $P(\omega_j)$
 - Per esempio scelgo la classe con probabilità a priori più alta:

$$\alpha = \operatorname{argmax}_{j \in C} P(\omega_j)$$

NOTA (argmax and argmin):

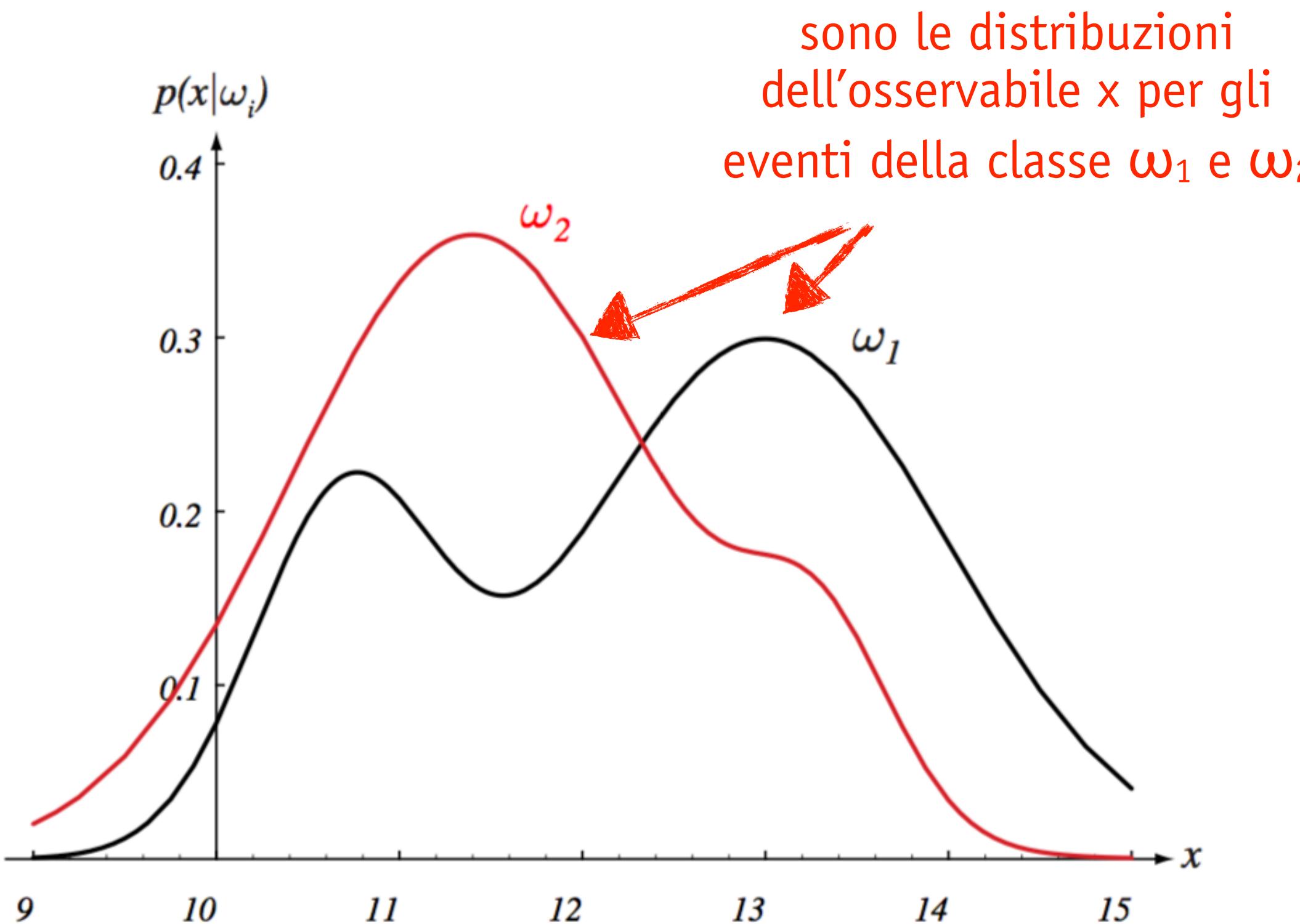
$$\arg \max_x f(x) := \{x \mid \forall y : f(y) \leq f(x)\}$$

- supponiamo, invece, di poter utilizzare anche un feature vector x , formalizzabile come una variabile aleatoria n-dimensionale
- supponiamo inoltre di conoscere la funzione di densità di probabilità condizionata alla classe: $p(x|\omega_j)$
- a partire dalle conoscenze descritte possiamo stabilire la probabilità a posteriori $p(\omega_j|x)$ che il campione descritto da x appartenga alla classe ω_j tramite il teorema di Bayes:

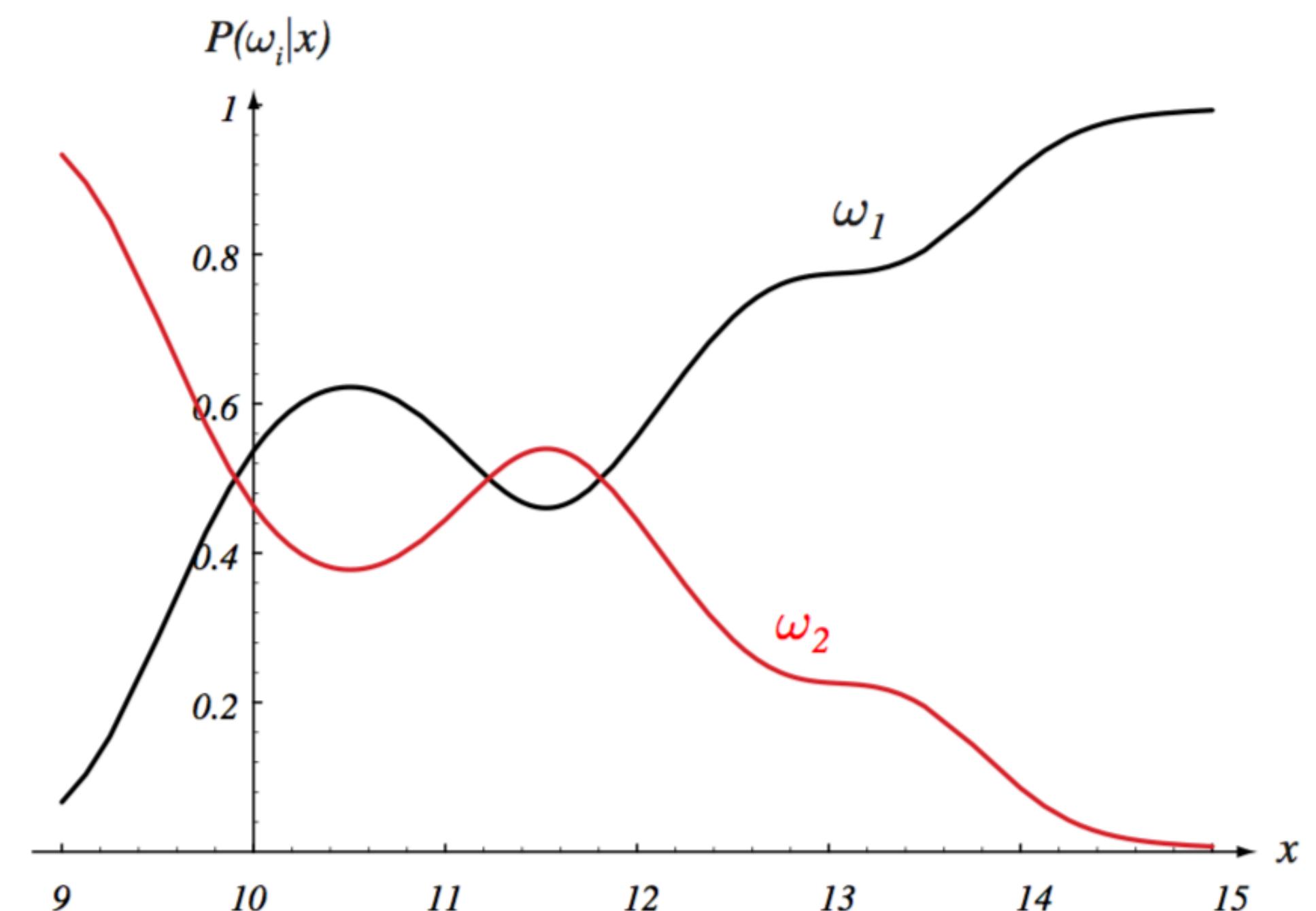
$$P(\omega_j|x) = \frac{p(x|\omega_j) \cdot P(\omega_j)}{p(x)} \text{ dove } p(x) = \sum_{j=1}^c p(x|\omega_j) \cdot P(\omega_j)$$



$$P(\omega_j|x) = \frac{p(x|\omega_j) \cdot P(\omega_j)}{p(x)} \text{ dove } p(x) = \sum_{j=1}^C p(x|\omega_j) \cdot P(\omega_j)$$



esempio di densità di probabilità
condizionata alle classi con $C=2$



probabilità a posteriori, assumendo come
esempio $P(\omega_1)=2/3$ e $P(\omega_2)=1/3$

CLASSIFICATORE BAYESIANO

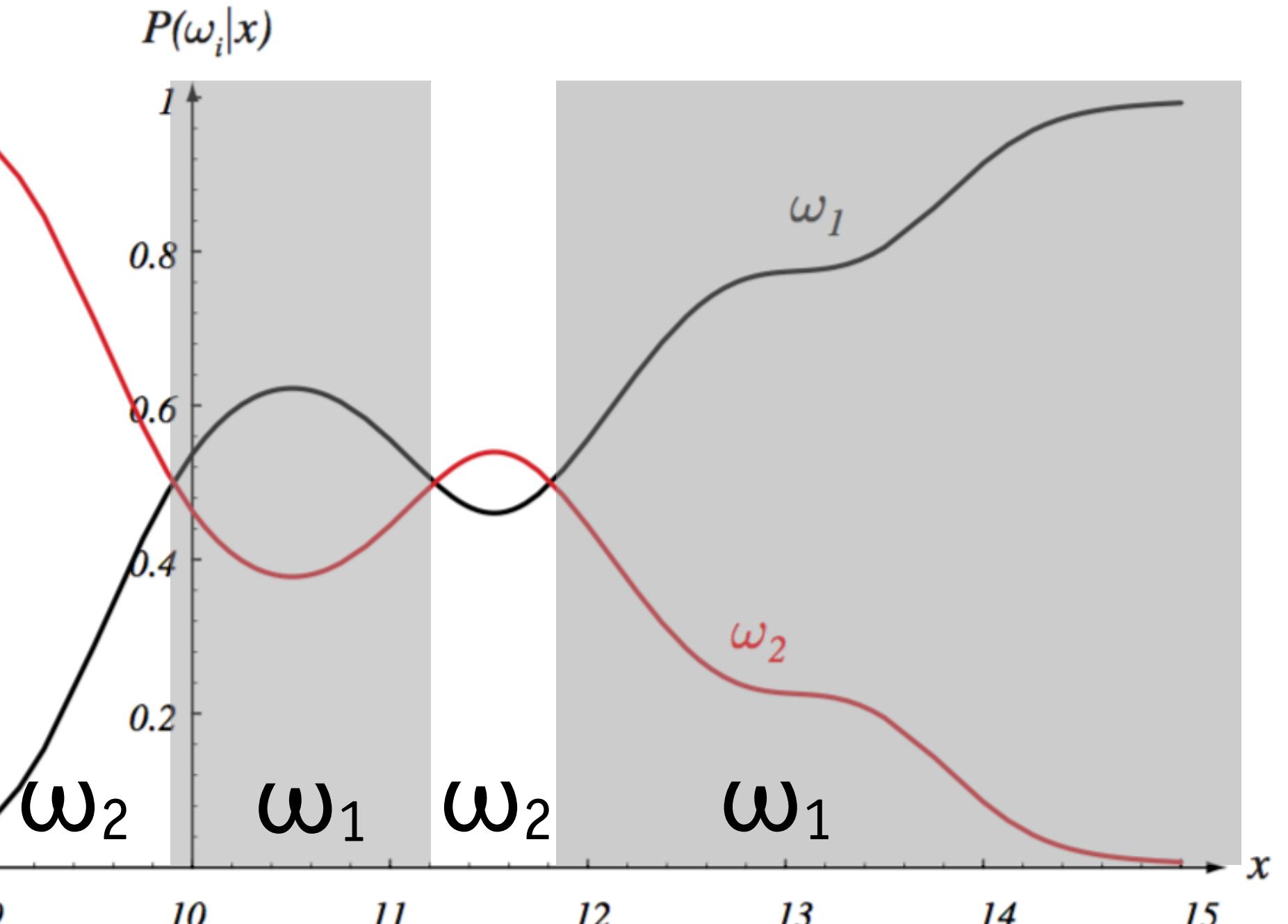
- la decisione tende naturalmente verso la classe cui compete la probabilità a posteriori maggiore:

$\alpha(x)$:

- $p(\omega_1|x) > p(\omega_2|x) \rightarrow$ decido ω_1
- altrimenti \rightarrow decido ω_2

- questa regola di fatto minimizza la probabilità di errore:

$$P(\text{error}|x) = \begin{cases} P(\omega_1|x) & \text{if we decide } \omega_2 \\ P(\omega_2|x) & \text{if we decide } \omega_1 \end{cases} \Rightarrow P(\text{error}|x) = \min P(\omega_1|x), P(\omega_2|x) \rightarrow$$



- da un punto di vista operativo $p(x)$ non entra in gioco nella decisione che si riduce quindi a:

$\alpha(x)$:

- $p(x|\omega_1)P(\omega_1) > p(x|\omega_2)P(\omega_2) \rightarrow$ decido ω_1
- altrimenti \rightarrow decido ω_2

$p(x)$ è solo un fattore di normalizzazione che ci dice con quale frequenza verrà misurato un dato pattern x

GENERALIZZAZIONE: MINIMIZZAZIONE DEL COSTO

- supponiamo ora che alcuni errori siano più costosi di altri. A fronte dell'osservazione di un feature vector x , il costo che dobbiamo aspettarci nel prendere la decisione α_i è il rischio statistico condizionale:

$$R(\alpha_i|x) = \sum_{j=1}^C \lambda(\alpha_i|\omega_j) \cdot P(\omega_j|x)$$

- la regola di decisione può ancora essere definita come una funzione $\alpha(x)$ che indica quale azione intraprendere per ogni possibile valore di x osservato
- la regola di decisione ottimale diventa quella per cui si ha il minimo rischio condizionale:

$$\alpha(x) = \operatorname{argmin}_{1 \leq j \leq C} R(\alpha_i|x)$$



PROBLEMI A 2 CLASSI: LIKELIHOOD RATIO TEST

- Nel caso particolare di problemi a due classi ($i=1,2$), definendo con $\lambda_{ij} = \lambda(\alpha_i | \omega_j)$ \Rightarrow i rischi condizionali possono essere scritti esplicitamente come:

$$R(\alpha_1 | x) = \lambda_{11} P(\omega_1 | x) + \lambda_{12} P(\omega_2 | x)$$
$$R(\alpha_2 | x) = \lambda_{21} P(\omega_1 | x) + \lambda_{22} P(\omega_2 | x)$$

- scegliamo ω_1 se $R(\alpha_1 | x) < R(\alpha_2 | x)$, cioè se:

$$\lambda_{11} P(\omega_1 | x) + \lambda_{12} P(\omega_2 | x) < \lambda_{21} P(\omega_1 | x) + \lambda_{22} P(\omega_2 | x), \text{ che equivale a:}$$

$$(\lambda_{11} - \lambda_{21}) P(\omega_1 | x) < (\lambda_{22} - \lambda_{12}) P(\omega_2 | x)$$

- essendo $(\lambda_{11} - \lambda_{21}) < 0$ e $(\lambda_{22} - \lambda_{12}) < 0$ (la decisione sbagliata costa più di quella corretta), possiamo moltiplicare i due membri per -1 e cambiare verso alla disequazione: $(\lambda_{21} - \lambda_{11}) P(\omega_1 | x) > (\lambda_{12} - \lambda_{22}) P(\omega_2 | x)$

- da cui:

$$\frac{P(\omega_1 | x)}{P(\omega_2 | x)} \stackrel{\omega_1}{>} \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}}$$



$$\frac{p(x|\omega_1)}{p(x|\omega_2)} \stackrel{\omega_1}{>} \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{P(\omega_2)}{P(\omega_1)}$$

Likelihood Ratio
Test

indipendente
dall'osservazione $x!$

il LRT rappresenta la regola di decisione statisticamente ottimale



ZERO-ONE LOSS FUNCTION

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & \text{se } i = j \quad (\text{decisione corretta}) \\ 1 & \text{altrimenti} \quad (\text{decisione sbagliata}) \end{cases}$$

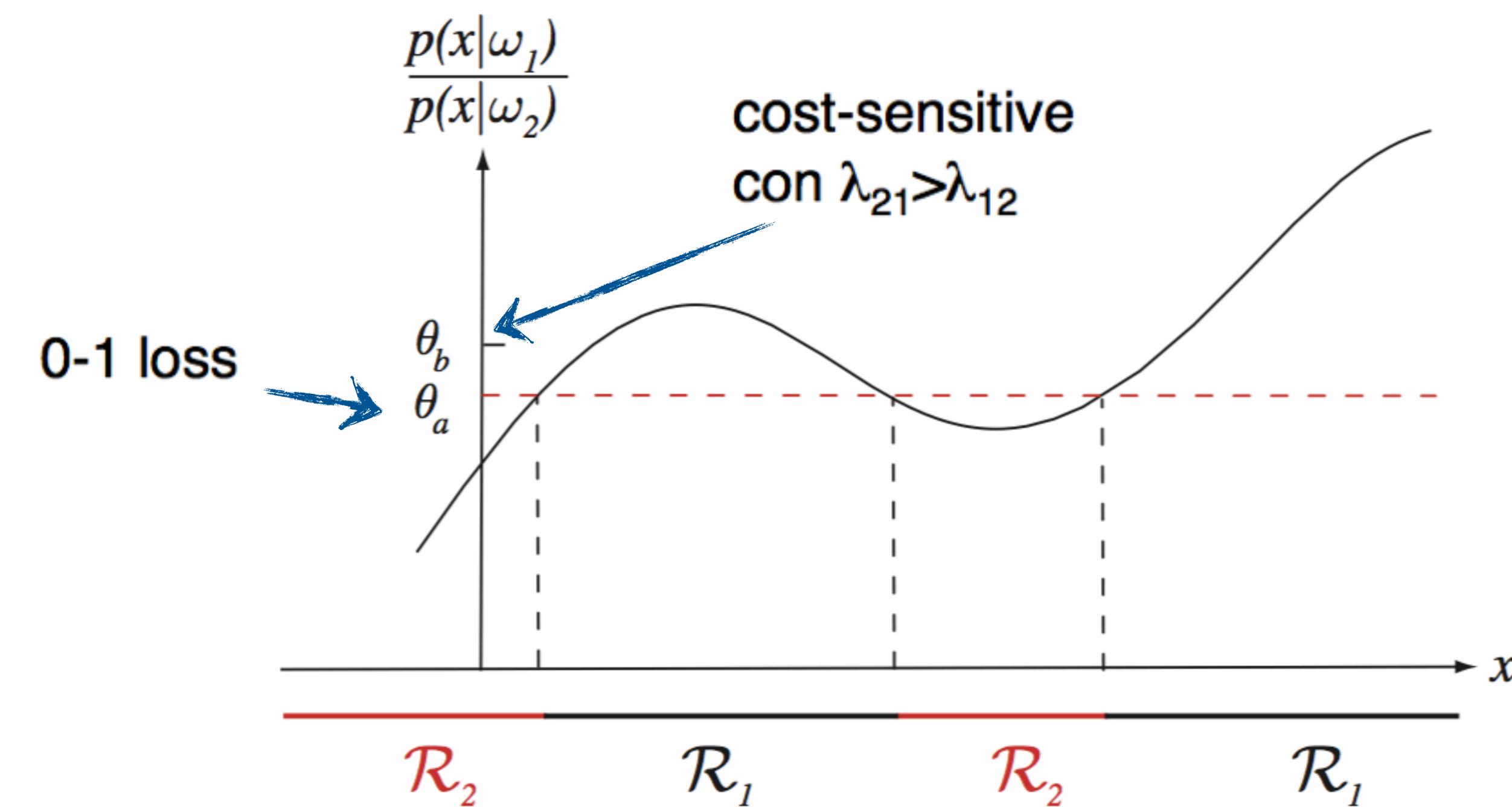
$$\begin{aligned} R(\alpha_i|x) &= \sum_{j=1}^c \lambda(\alpha_i|\omega_j) P(\omega_j|x) \\ &= \sum_{j \neq i} P(\omega_j|x) = 1 - P(\omega_i|x) \end{aligned}$$

probabilità condizionale che la scelta α_i sia corretta

- utilizzata quando si vuole minimizzare la probabilità di errore (cioè la rate di errori)
- questo caso è incluso nel LRT, a patto di porre $\lambda_{21}=\lambda_{12}=1$ e $\lambda_{11}=\lambda_{22}=0$ (funzione di costo simmetrica o zero-one loss):

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} \stackrel{\omega_1}{>} \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{P(\omega_2)}{P(\omega_1)} \stackrel{\omega_1}{>} \frac{P(\omega_2)}{P(\omega_1)}$$

1-0/1-0 = 1



VISUALIZZAZIONE DELLA REGOLA DI BAYES

Possiamo capire meglio il modo di operare di un classificatore di Bayes guardando più in dettaglio alle sorgenti di errore nella classificazione

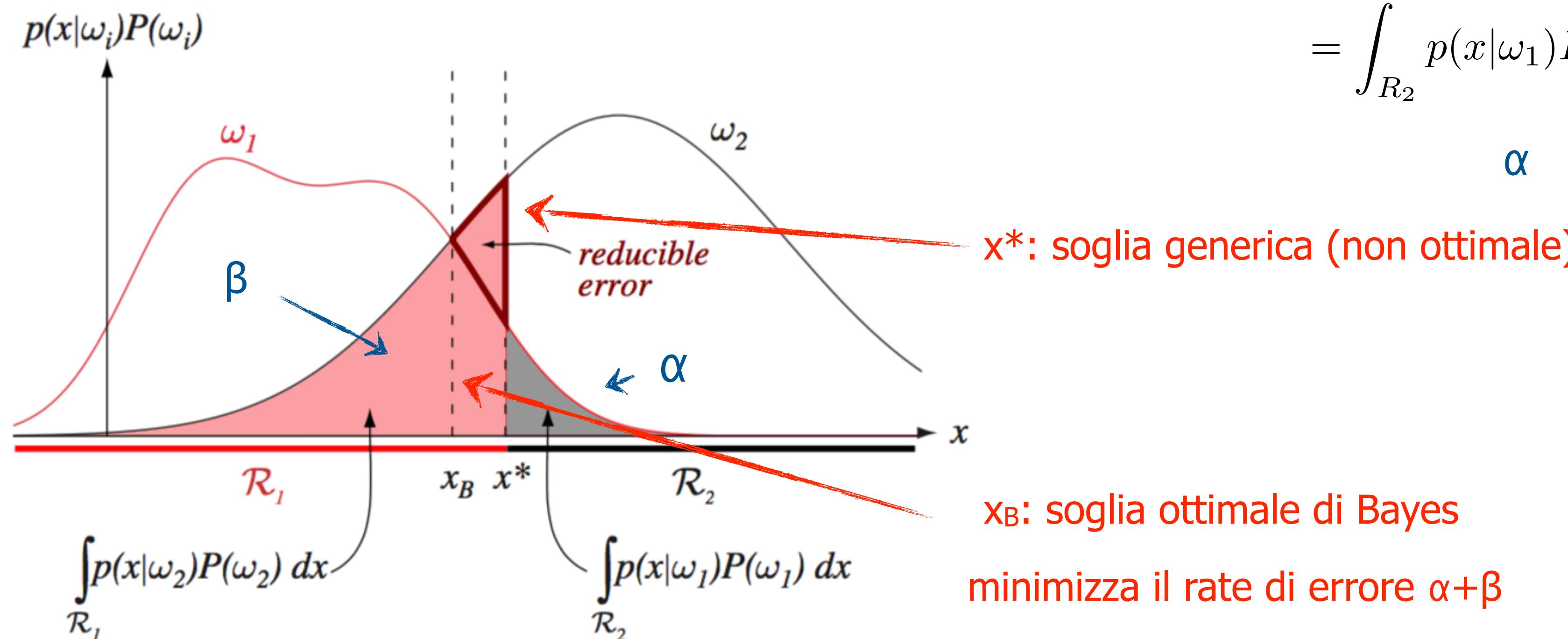
Supponiamo di scegliere le regioni di decisione R_1 e R_2 in modo non ottimale → esistono due modi in cui si può generare un errore di classificazione:

- l'osservazione x cade in R_1 mentre l'evento vero appartiene a ω_2
- l'osservazione x cade in R_2 mentre l'evento vero appartiene a ω_1

gli eventi sono mutualmente esclusivi, quindi avremo:

$$\begin{aligned} P(\text{error}) &= p(x \in R_2, X \in \omega_1) + P(x \in R_1, X \in \omega_2) \\ &= p(x \in R_2 | \omega_1)P(\omega_1) + p(x \in R_1 | \omega_2)P(\omega_2) \\ &= \int_{R_2} p(x|\omega_1)P(\omega_1)dx + \int_{R_1} p(x|\omega_2)P(\omega_2)dx \end{aligned}$$

α + β



x^* : soglia generica (non ottimale)

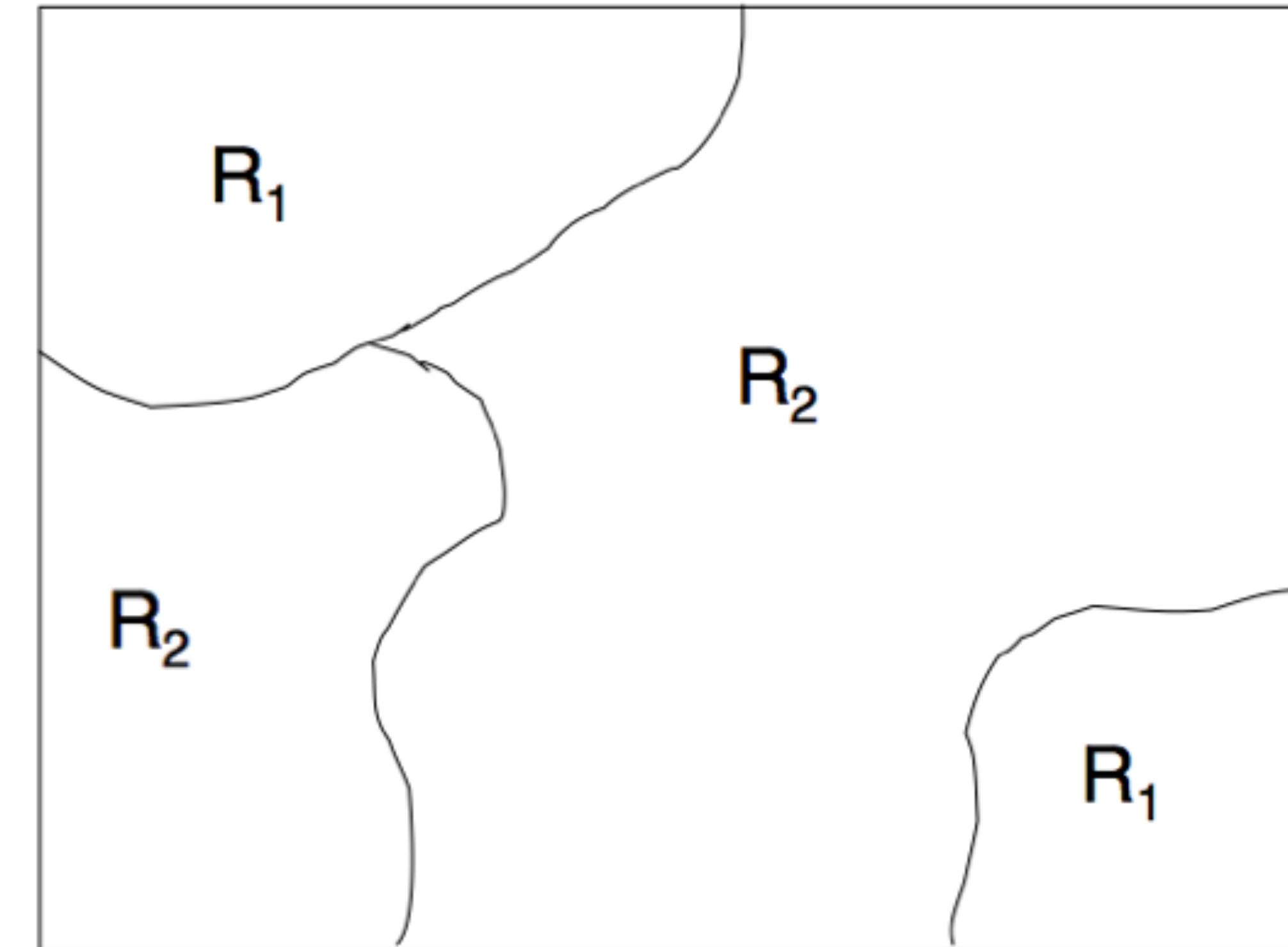
x_B : soglia ottimale di Bayes
minimizza il rate di errore $\alpha+\beta$

REGIONI DI DECISIONE

- la regola di decisione induce nello spazio delle features un insieme di **regioni di decisione**:

$$x \in R_1 \Leftrightarrow p(x|\omega_1)/p(x|\omega_2) > \gamma$$

$$x \in R_i \Leftrightarrow \alpha(x) = \alpha_i$$



PROBABILITÀ MINIMA DI ERRORE NEL TLR: DIMOSTRAZIONE FORMALE

- vogliamo dimostrare esplicitamente come la regola di decisione del LRT risulti ottimale (i.e. fornisca il miglior risultato raggiungibile), una volta scelto opportunamente il valore di soglia
- consideriamo sempre il problema a 2 classi, che indichiamo con ω_1 e ω_2 , e sia x il vettore di feature
- siano R_1 e R_2 le due regioni di decisione di x per le quali si decida $x \in \omega_1$ e $x \in \omega_2$ e sia $R_1 \cup R_2$ il dominio di x

siano λ_{ij} i costi associati ad
ogni decisione

λ_{11} decido $x \in \omega_1$ quando $x \in \omega_1$
 λ_{12} decido $x \in \omega_1$ quando $x \in \omega_2$
 λ_{21} decido $x \in \omega_2$ quando $x \in \omega_1$
 λ_{22} decido $x \in \omega_2$ quando $x \in \omega_2$

$\lambda_{21} > \lambda_{11}$
 $\lambda_{12} > \lambda_{22}$

la decisione sbagliata
costa di più di quella corretta

Valore aspettato del costo condizionale:

$$\begin{aligned} R = \mathbb{E}[\text{costo}] &= \int_{R_1} \lambda_{11} p(\omega_1|x) p(x) dx + \int_{R_1} \lambda_{12} p(\omega_2|x) p(x) dx + \\ &+ \int_{R_2} \lambda_{21} p(\omega_1|x) p(x) dx + \int_{R_2} \lambda_{22} p(\omega_2|x) p(x) dx = \\ &= \int_{R_1} [\lambda_{11} p(x|\omega_1) p(\omega_1) + \lambda_{12} p(x|\omega_2) p(\omega_2)] dx + \\ &+ \int_{R_2} [\lambda_{21} p(x|\omega_1) p(\omega_1) + \lambda_{22} p(x|\omega_2) p(\omega_2)] dx \end{aligned}$$

vogliamo scegliere R_1 , R_2 in modo da rendere minimo il valore
aspettato del costo condizionale

sfruttiamo il fatto che R_1 e R_2 non si
intersecano e $R_1 \cup R_2 = 1$:

$$\int_{R_1} p(x|\omega) dx = 1 - \int_{R_2} p(x|\omega) dx \Rightarrow$$



$$\begin{aligned}
R &= \int_{R_1} [\lambda_{11}p(x|\omega_1)p(\omega_1) + \lambda_{12}p(x|\omega_2)p(\omega_2)]dx + \int_{R_2} [\lambda_{21}p(x|\omega_1)p(\omega_1) + \lambda_{22}p(x|\omega_2)p(\omega_2)]dx = \\
&= \lambda_{11}p(\omega_1) \int_{R_1} p(x|\omega_1)dx + \lambda_{12}p(\omega_2) \int_{R_1} p(x|\omega_2)dx + \\
&\quad + \lambda_{21}p(\omega_1) \left[1 - \int_{R_1} p(x|\omega_1)dx \right] + \lambda_{22}p(\omega_2) \left[1 - \int_{R_1} p(x|\omega_2)dx \right] = \\
&= \lambda_{21}p(\omega_1) + \lambda_{22}p(\omega_2) + \int_{R_1} [-(\lambda_{21} - \lambda_{11})p(\omega_1)p(x|\omega_1) + (\lambda_{12} - \lambda_{22})p(\omega_2)p(x|\omega_2)]dx
\end{aligned}$$

abbiamo trasformato il problema nello scegliere R_1 in modo tale che R sia minimo

supponiamo che per un dato x l'integrandi in R sia negativo, possiamo allora diminuire R assegnando x a R_1 , nel caso contrario possiamo ridurre r assegnando x a R_2

formalmente: $(\lambda_{12} - \lambda_{22})p(\omega_2)p(x|\omega_2) < (\lambda_{21} - \lambda_{11})p(\omega_1)p(x|\omega_1) \rightarrow x \in R_1$

$(\lambda_{12} - \lambda_{22})p(\omega_2)p(x|\omega_2) > (\lambda_{21} - \lambda_{11})p(\omega_1)p(x|\omega_1) \rightarrow x \in R_2$

$$\lambda_{21} - \lambda_{11} > 0$$

$$\lambda_{12} - \lambda_{22} > 0$$

che può essere scritta come:

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{p(\omega_2)}{p(\omega_1)} \rightarrow x \in \omega_1$$

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} < \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{p(\omega_2)}{p(\omega_1)} \rightarrow x \in \omega_2$$

è il LR di Bayes per una funzione di costo generale, se si pone
 $\lambda_{21} - \lambda_{11} = \lambda_{12} - \lambda_{22}$ il costo diventa la probabilità di errore ed
 otteniamo la dim. per il TLR simmetrico.



CASO MULTICLASSE

- in maniera analoga si può dimostrare che anche nei problemi multiclassi, il classificatore Bayesiano risulta il classificatore ottimo

- poiché:

$$P_{err} = 1 - P(correct) = 1 - \sum_{i=1}^C \int_{R_i} p(x|\omega_i)P(\omega_i)dx$$

- avremo:

$$\sum_{i=1}^C \int_{R_i} p(x|\omega_i)P(\omega_i)dx \leq \int_T \max_{1 \leq i \leq C} [p(x|\omega_i)P(\omega_i)]dx$$

$$P_{err} \leq 1 - \int_T \max_{1 \leq i \leq C} [p(x|\omega_i)P(\omega_i)]dx$$

- e quindi la probabilità minima verrà raggiunta con la regola di decisione bayesiana:

$$\alpha(x) = \arg \max_{1 \leq i \leq C} [P(\omega_i|x)]$$



ALTRI CRITERI DI DECISIONE

- Il LRT è il criterio di decisione ottimale in quanto minimizza il rischio condizionale
- esistono però criteri alternativi utili in casi speciali:

- **Minimax:**

- in alcune circostanze, potrebbe essere necessario progettare un sistema di classificazione che debba garantire buone prestazioni non per un valore determinato della probabilità a priori, ma su un intervallo di valori possibili
 - esempio: non conosciamo esattamente la prob. a priori ma sappiamo che varia in un certo intervallo, oppure le prob. a priori possono cambiare in modo imprevedibile nella fase operativa del classificatore
- in questo caso, un approccio ragionevole è quello di **progettare il classificatore in modo da minimizzare i danni nel caso peggiore**
- ciò equivale a minimizzare il valore massimo del rischio al variare delle probabilità a priori

- **Neyman-Person:**

- mentre il criterio di Bayes ottimizza il rischio condizionale complessivo, si potrebbe avere l'esigenza di limitare la probabilità di errore all'interno di una classe
 - ex. è necessario che la probabilità di errore sui campioni della classe ω_2 sia minore di ξ e che sia inoltre la minima possibile su ω_1
- l'idea è quella di usare i moltiplicatori di lagrange per porre vincoli sulla funzione obiettivo imponendo che siano soddisfatti i vincoli sugli errori e calcolando in questo modo il valore di soglia

$$L = R_{\text{emp}}(f_E) = \frac{1}{N} \sum L(y_i, f_E(x_i)) + C \leftarrow \text{vincoli}$$

