Apprendimento Automatico

Introduzione

Argomenti del corso di Apprendimento Automatico

- Apprendimento automatico: studio di algoritmi per la creazione automatica di basi di conoscenza
- Aree applicative: robotica, agenti intelligenti, data & text mining
- Argomenti delle lezioni:
  - Apprendimento da esempi: alberi di decisione, apprendimento probabilistico, reti neurali, support vector machines
  - Apprendimento per rinforzo: Q-learning, algoritmi genetici
  - Progetto (nel passato: riconoscimento di volti, anti-spam, utilizzo del toolkit WEKA)

Notizie sul corso

- 4 crediti
- 11 settimane (48 ore)
- Prova intermedia (scritta) + Esame orale + progetto

Perché è importante l'Apprendimento Automatico

- **"Bottleneck" della conoscenza**: è più facile apprendere automaticamente che codificare conoscenza a mano!
- Progressi nell'area algoritmi e modelli teorici
- Crescente disponibilità di **dati** on-line
- Macchine più potenti
- Applicazioni industriali rilevanti

Due importanti "nicchie" di applicazione per l'AA:

1. **Data Mining**: scoprire regolarità e patterns in dati multi-dimensionali e complessi

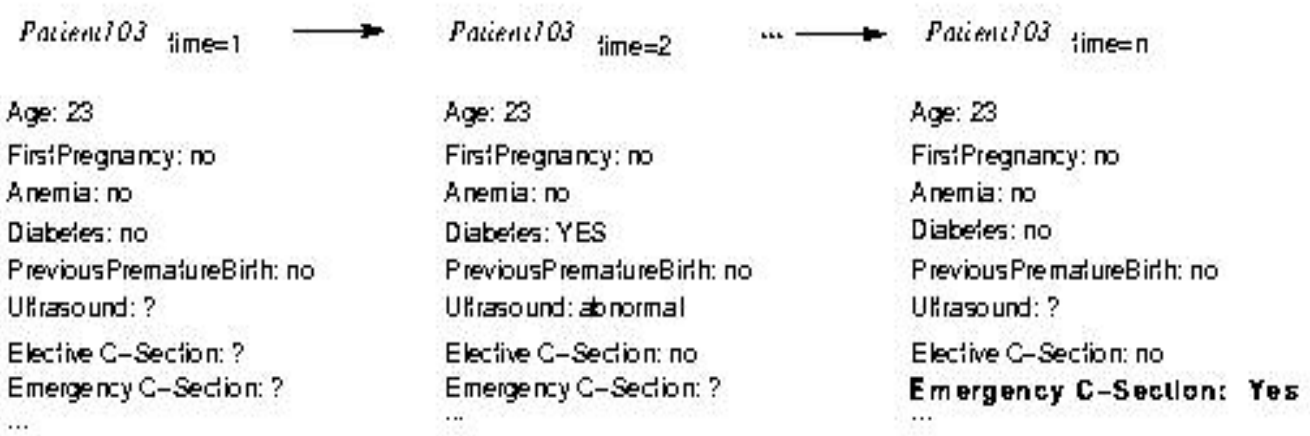
– Cartelle cliniche → conoscenza medica

– Dati di vendita → strategie di marketing

– Denuncie dei redditi → politiche di verifica fiscale.

2. **Applicazioni software difficili da programmare perchè:**
   - Non esistono esperti umani (es. *Analisi DNA*)
   - Gli umani sanno eseguire un compito, ma non sanno come (es. *speech recognition*, *comprensione di immagini*)
   - Ogni utente ha esigenze specifiche (es. *filtraggio di notizie dal web*)

1. Data Mining (esempio)



A partire da:

9714 cartelle cliniche , ognuna delle quali descrive una gestazione ed una nascita. Ogni cartella contiene 215 caratteristiche (*features*)

Impara a fare predizioni su:

- categorie di pazienti ad alto rischio per il reparto Parti Cesarei

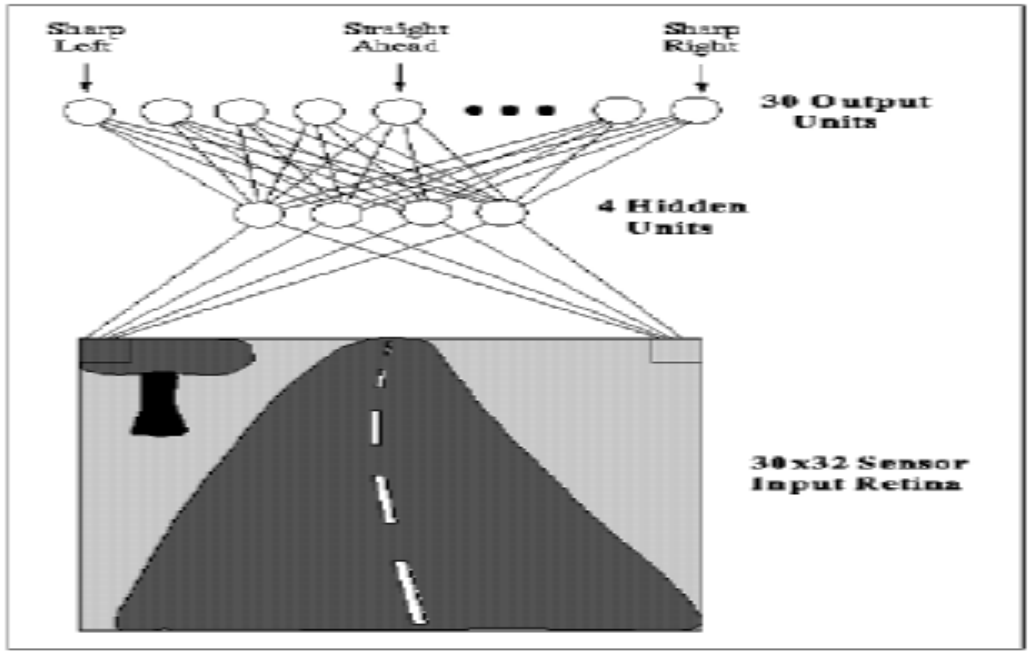
Ex: ***If*** *No previous non-cesarean delivery,* ***and*** *Abnormal 2nd Trimester Ultrasound,* ***and*** *Malpresentation ad admission* ***Then*** *Probability Emergency C-Section is 0.6*

Altre applicazioni di data mining

- Fraud Detection (evasione fiscale, comportamenti fraudolenti, concessione di fidi bancari)
- Market Analysis (profili di consumatori, disposizione di merci, politiche di mercato)
- User Modeling (es.usando i log di utente)

2. Problemi troppo complessi per una programmazione manuale

- Pilota automatico ALVINN (Carnagie Mellon)
  http://www.ri.cmu.edu/projects/project_160.html
- Apprende strategie di guida usando un modello di reti neurali. Guida a 70 m/h sulle autostrade



Software adattabili all'utente

Web Agents

Letizia: suggerisce pagine interessanti imparando gli interessi dell'utente sulla base di dati ottenuti tracciando il suo comportamento

http://lieber.www.media.mit.edu/people/lieber/Lieberary/Letizia/Letizia.html

Aree metodologiche correlate

- Intelligenza Artificiale
- Teoria della Complessità computazionale
- Teoria dell'Informazione
- Psicologia e neurobiologia
- Statistica e metodi Bayesiani
- …

Il problema dell'apprendimento

- Apprendere = migliorare la capacità di esecuzione di un certo compito, attraverso l'esperienza
  - Migliorare nel *task* T
  - Rispetto ad una misura di prestazione P
  - Basandosi sull'esperienza E
- **E** =esperienza

E=

- un **insieme di esempi già classificati** da un "istruttore" umano (es: cartelle cliniche contenenti la diagnosi, o l'esito del ricovero)
- un **sistema di "ricompense e "punizioni"** che rinforzi i comportamenti utili e penalizzi quelli non utili per il task T (es: se T=partita a scacchi, premiare mosse che facciano mangiare pedine importanti)

Scelte nella definizione di un sistema di apprendimento

1. **Modalità di training**: supervisionato, guidato da obiettivi
2. **Scelta di una rappresentazione** per gli oggetti del mondo nel dominio di interesse
3. **Scelta e rappresentazione** di una "funzione obiettivo" che rappresenti il compito
4. Scelta di un **algoritmo di apprendimento**

Modalità di Training

- Da cosa impara il sistema?
- Nei modelli supervisionati un “trainer” fornisce a priori esempi di comportamenti corretti.
- Nei sistemi non supervisionati ci deve essere un qualche metodo di “rinforzo” implicito (es. l’assegnazione di un “premio” se il sistema fa “la cosa giusta”)

Rappresentare gli oggetti del dominio

- Un sistema di apprendimento deve:
  - **classificare** oggetti del mondo reale (ad esempio cartelle cliniche, denunce dei redditi, ilamenti di DNA..), oppure
  - **eseguire compiti** in un ambiente reale (automobile che si muove lungo una strada, schacchiera, catena di montaggio..)

Rappresentare gli oggetti del dominio (2)

- Rappresentare gli oggetti da classificare:
  - Vettori di caratteristiche o feature vectors (punti in uno spazio n-dimensionale)
  - Grafi (non solo features (*vertici*) ma anche relazioni fra features (*archi*)) o rappreentazioni strutturate equivalenti (*frames*)
- Rappresentare l'ambiente:
  - Matrici n x m (es. caselle della schacchiera)
  - Rappresentazioni grafiche più complesse (contorni, textures..)

Un semplice esempio

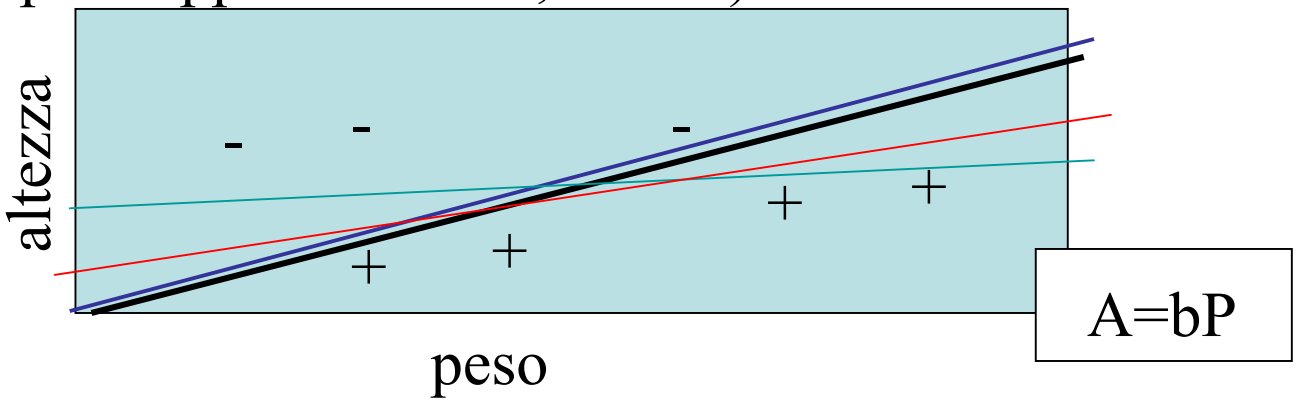
- Voglio imparare a classificare individui come normopesi o obesi
- Gli oggetti del mondo: esseri umani
- Rispetto al problema di interesse, quali sono le caratteristiche o features che interessa rappresentare?
  - Peso, altezza, sesso, età
  - Vettore: V(p,a,s,e) dove: $p,a,e \in R$, $s \in B$

Scelta della funzione obiettivo

- Funzione obiettivo: espressione formale della conoscenza appresa. Viene usata per determinare le prestazioni del programma di apprendimento.
  - Esempi di funzioni obiettivo: polinomi, alberi di decisione, reti neurali, insiemi di regole….

Esempio

- Semplificando il problema precedente, supponiamo di considerare solo vettori a 2 dimensioni v(Peso,Altezza)
- La funzione obiettivo può essere ad es. una retta R: A=bP tale per cui, se "-" sono gli obesi e "+" i normopesi, R "separa" i + dai - (i punti dello spazio rappresentano gli esempi di apprendimento, cioè E)



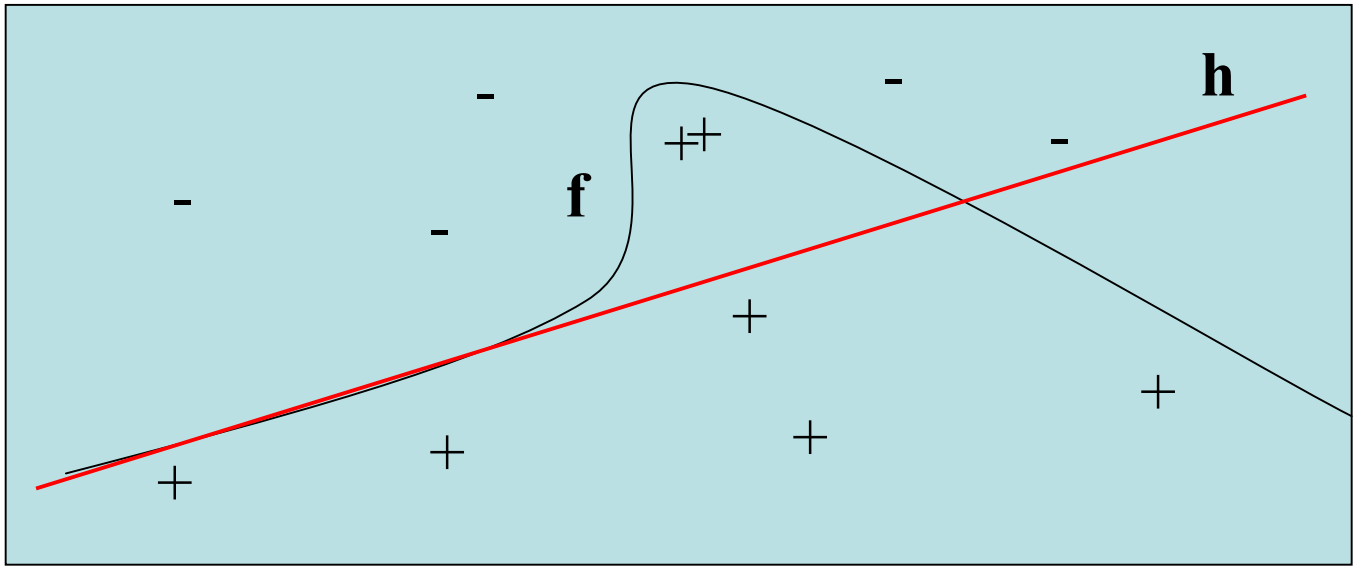
Osservate che esistono infinite soluzioni!!

Funzione obiettivo e funzione appresa

- Nota: Difficilmente un sistema riesce ad apprendere *perfettamente* una funzione obbiettivo *f* (l'esempio della funzione altezza-peso è un caso semplice ma i prolemi reali sono molto complessi).
- In genere viene appresa un funzione, avente la *forma* prescelta (polinomio, regole,..), che rappresenta una stima, o ipotesi (indicata con *h*, o $\hat{f}$), per la funzione obiettivo.
- L'obiettivo è di apprendere una *h* che approssimi *f* "il meglio possibile"

Esempio

- In questo caso una retta (h) approssima la funzione reale (f)



Scelta di un algoritmo di apprendimento

Scelta dell'algoritmo e della funzione obiettivo ovviamente correlate

- Metodi statistici (la f è una funzione probabilistica)
  - Bayesian learning
  - Markov models
- Metodi algebrici (f è una funzione algebrica, es. polinomio)
  - Gradient descent
  - Support Vector Machines
- Metodi knowledge-based (f: espressioni logiche)
  - Alberi di decisione
  - Regole di associazione

Esempio: giocare a scacchi

- Task T = giocare a scacchi
- Misura della prestazione P = % partite vinte
- Esperienza E = partite già giocate (quindi sono note le sequenze delle mosse dei due giocatori e l'esito di alcune partite)

Come rappresentare E?

- **E** in questo caso è l'ambiente nel quale l'apprendista si muove, cioè la scacchiera
- Potremmo rappresentare lo stato della scacchiera come una matrice, ogni casella contiene il codice della pedina che vi si trova in t
- In realtà è troppo complicato, basta rappresentare un numero minore e più significativo di elementi, ad esempio:
  - *bp* numero dei pezzi neri
  - *rp* numero dei pezzi bianchi
  - *bk* numero dei reali neri
  - *rk* numero dei reali bianchi
  - *bt* numero dei pezzi bianchi che, al prossimo turno, sono mangiabili dai neri
  - *rt* idem per I neri
- In tal caso ogni stato della scacchiera è un vettore a 6 argomenti
- b(t)=(bp,rp,bk,rk,bt,rt)

Funzione obiettivo

Una possibile definizione per la funzione obiettivo *f*:

b è uno "stato" della scacchiera (il vettore a 6 attributi: b(t)),

f: V(b) è una funzione che assegna un valore allo stato.

1. Se b è uno stato finale, ed è una partita vinta, V(b)=100
2. Se b è uno stato finale, ed è una partita persa, V(b)=-100
3. Se b è uno stato finale, ed è una "patta", V(b)=0
4. Se b non è uno stato finale, allora V(b)=V(b') dove:

b' è il miglior stato finale raggiungibile da b e giocando in modo ottimale fino alla fine del gioco

L'ultima regola non è utilizzabile!!!

Però è "approssimabile", vediamo come.

Giocare a scacchi

- **una scelta migliore per V(b)**
- Scegliere una rappresentazione per la funzione obiettivo:

$$V(b) = w0 + w1 \cdot bp(b) + w2 \cdot rp(b) + w3 \cdot bk(b) +$$
$$w4 \cdot rk(b) + w5 \cdot bt(b) + w6 \cdot rt(b)$$

La funzione obiettivo è una funzione algebrica, precisamente un polinomio di primo grado (funzione lineare nei parametri bp, rp, ecc)

Giocare a scacchi

- L'obiettivo di apprendimento è trovare dei valori per i coefficenti del polinomio (= stimare la V)
- Per stimare i valori dei coefficenti, si utilizzano **partite giocate**, di cui è dunque noto l'esito (E). La funzione V(b) non è nota, ma sono noti i suoi valori per alcuni $b \in B$
- Es (per uno stato in cui i neri vincono):

<<x1=3,x2=0,x3=1,x4=0,x5=0,x6=0>, +100>

Giocare a scacchi

- Dall'esame di alcuni stati della scacchiera in partite giocate, è dunque noto il valore V(b) per alcuni $b \in B$
- Supponiamo di partire da una certa stima $\hat{V}$ per V.
- $\hat{V}(b)$ è il **valore,** calcolato per un certo b, della **funzione** appresa (cioè una stima dei coefficienti), che deve approssimare la funzione obiettivo V(b), mentre $V_{train}(b)$ è un **valore** ottenuto dagli esempi di addestramento (*training set*), cioè il valore calcolato dalla funzione V "reale".

Algoritmo di apprendimento per gli scacchi

- **Metodo del gradiente** (*gradient descent*)
  - Inizializza i coefficienti di $\hat{V}$ (a caso, o ponendoli=1)
  - Seleziona a caso un esempio V(b) per il quale si noto il valore della funzione
  - Calcola l'errore $e(b) = V_{train}(b) - \hat{V}(b)$
  - **Per ogni** caratteristica *fi* della scacchiera (*bp,rp*..), aggiorna il relativo peso come segue:

$$w_i \leftarrow w_i + c_i \cdot f_i \cdot e(b)$$

  - Dove c è una piccola costante, es c=0,1
  - **Ripeti** finchè l'errore non scende al di sotto di una soglia ε

Esempio

- Supponiamo di inizializzare la funzione $\hat{V}$ con $w_1=w_2=...=1$
- Consideriamo un punto della funzione per il quale è noto il valore rale di V, es:

  V(b(3,0,1,0,0,0))=100

  e=V-V^=100-(1+1x3+1x0+1x1+1x0+1x0+1x0)=95

  w1=1+0,1x3x95=29,5

  w3=1+0,1x1x95=10,5

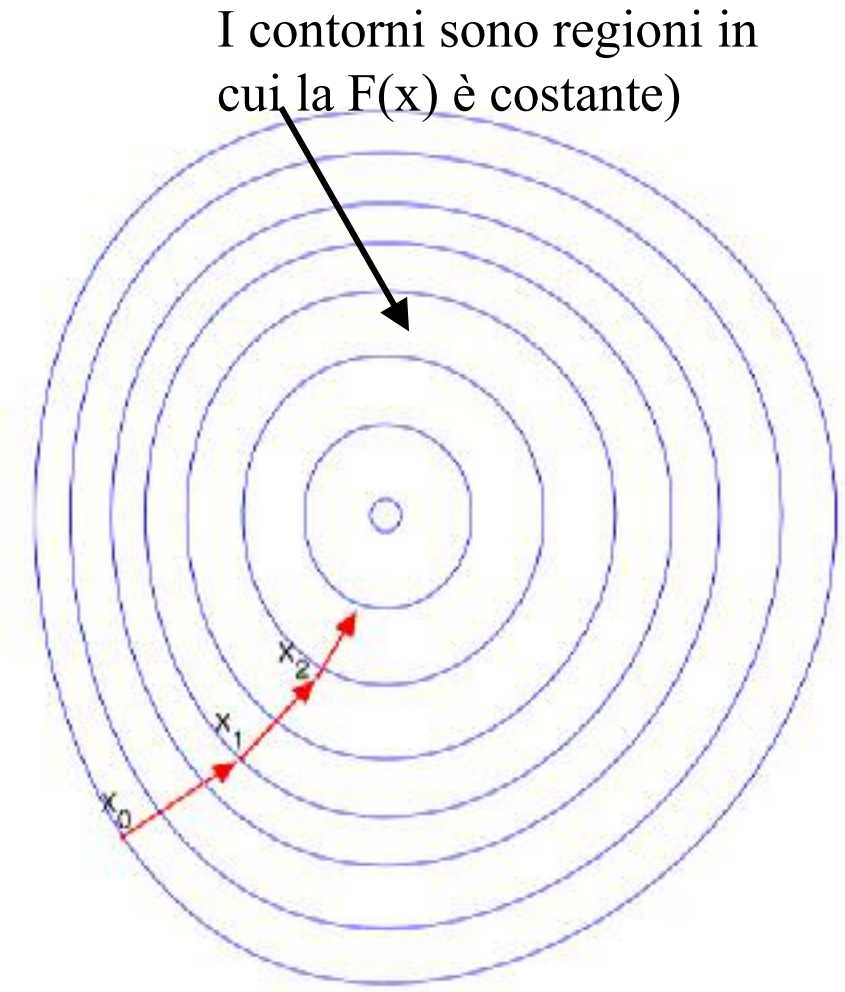
  Calcolo errore

  Aggiorn

  Aggiorno w3

- Già dopo la prima iterazione i valori dei w_i crescono velocemente, riducendo l'errore!

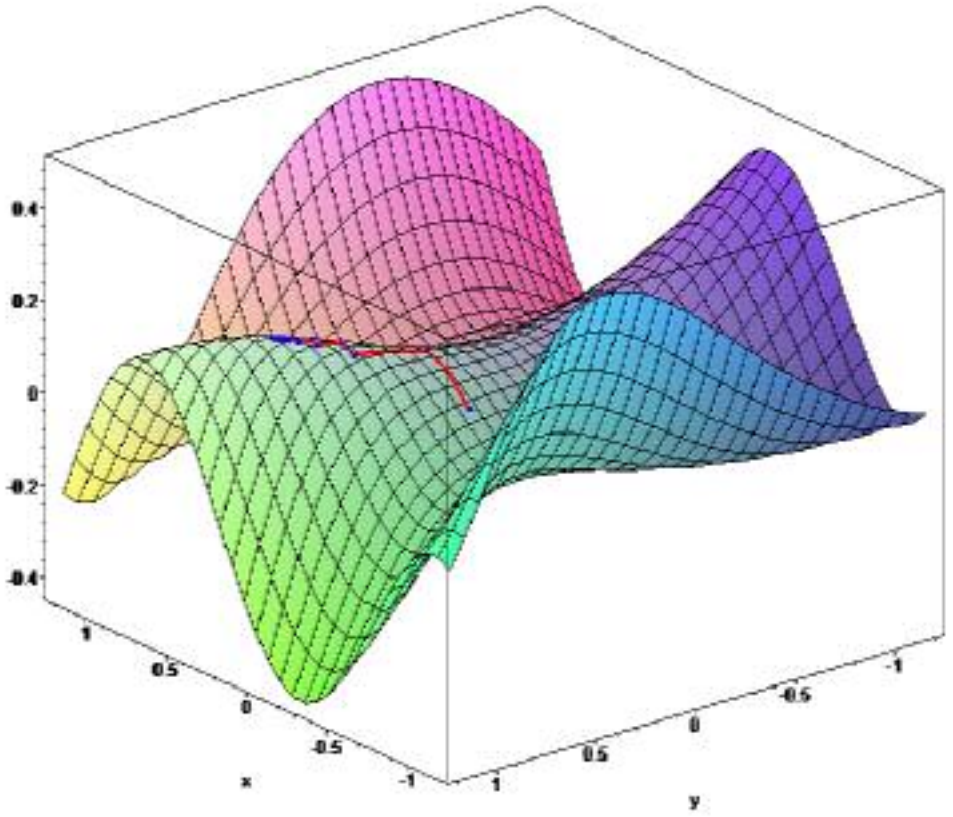
Discesa del gradiente



- F(x) funzione differenziable (nel nostro caso, la funzione di errore $F(x) = V_{train}(x) - \widehat{V}(x)$)
- Vogliamo convergere verso un minimo (possibilmente un minimo locale)
- F(x) decresce più velocemente se ci si muove nella direzione di pendenza massima (la derivata di F(x)

$$x_{n+1} = x_n + \gamma \Delta F(x_n)$$

Funzione F(x) qualsiasi, vista tridimensionale



Scelte progettuali nella definizione di un sistema di AA