Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

ELEMENTI DI PROBABILITÁ PER IL MACHINE LEARNING

Relatore:
Chiar.mo Prof.
Andrea Pascucci

Presentata da:
Elia Saltarelli

Sessione I
Anno Accademico 2019/2020

A me stesso,
alla mia famiglia,
ai miei amici piú cari.

Indice

Introduzione

Il machine learning o apprendimento automatico é quella branca dell'intelligenza artificiale che si pone l'obiettivo di far apprendere in modo automatico a delle macchine che funzionano imitando il comportamento del cervello umano.
Fu proprio la possibilitá che le macchine imparassero dai dati ad interessare negli anni '40 il matematico Marvin Minsky e i ricercatori Arthur Samuel e Frank Rosenblatt, sfruttando teorie e concetti probabilistici.
Fu lo stesso Minsky a costruire la prima rete neurale stocastica, lo *SNARC* (Stochastic neural analog reinforcement calculator) che comprendeva 40 neuroni artificiali interconnessi.
Nel '58, Frank Rosenblatt pubblicó il primo modello di percettrone, una rete neurale che in un numero finito di cicli di addestramento poteva apprendere qualsiasi tipo di compito, nei limiti delle sue dimensioni. Per ulteriori informazioni si veda [6].
Oggi, le tecniche di machine learning alimentano molti settori della societá moderna:
Ingegneria e automazione: realizzare sistemi automatizzati in grado di apprendere dal mutare delle condizioni, di ridurre gli sprechi e di imparare dai propri errori.
Informatica: le ricerche sul web utilizzano motori di ricerca, i quali attraverso una o piú parole chiave, restituiscono liste di risultati attinenti alla ricerca effettuata. Nello stesso modo avviene la selezione dei contenuti sui social network per abbinare notizie, post e pubblicitá agli interessi degli utenti [1].
Finanza: si utilizzano le reti neurali come strumento di predizione dei movimenti futuri di un dato titolo.
Medicina: la possibilitá di usare macchine di Boltzmann per l'analisi di immagini mediche, per la refertazione in radiologia ecc. Per esempio si veda [4].
Genetica: nel sequenziamento, per identificare specifici pattern, all'interno di una grande quantità di dati genetici, con l'obiettivo di aiutare a predire le probabilità di un paziente di sviluppare una data patologia o di progettarne un potenziale percorso di cura.
Sicurezza: tramite sistemi di prevenzione dalle frodi, ad esempio i filtri anti-spam delle e-mail basati su sistemi di machine learning in grado di riconoscere messaggi di posta fraudolenti. Per la clonazione di carte di credito, gli algoritmi mettono in relazione eventi ed abitudini del soggetto in modo da individuare comportamenti anomali. Fino ad arri-

vare al riconoscimento facciale e a quello vocale per poter accedere ai nostri smartphone.
Industria automobilistica: il machine learning é alla base delle auto a guida autonoma, consentendo loro di riconoscere l'ambiente circostante, distinguere un pedone da un lampione oppure rilevare segnali, strisce pedonali e corsie. Si veda per esempio [5].
L'obiettivo di questo elaborato é quello di definire le reti neurali e studiare alcune tecniche di machine learning da un punto di vista matematico.

Capitolo 1

Preliminari

1.1 Nozioni probabilistiche

Per le definizioni e i teoremi di questo capitolo si veda [11].

Definizione 1.1. Spazio Misurabile
Uno spazio misurabile é una coppia (Ω, F) dove:

- Ω é un insieme non vuoto;
- F é una σ-algebra su Ω cioé una famiglia non vuota di sottoinsiemi di Ω che soddisfa le seguenti proprietá:
  1. se $A \in F$ allora $A^c \in F$;
  2. l'unione numerabile di elementi di F appartiene ad F.

Definizione 1.2. Misura
Una misura sullo spazio misurabile (Ω, F) é una funzione

$$\mu : F \to [0, +\infty] \tag{1.1}$$

tale che:

1. $\mu(\varnothing) = 0$
2. μ é σ-additiva su F, ossia per ogni successione $(A_n)_{n\in\mathbb{N}}$ di elementi disgiunti di F vale:

$$\mu\left(\bigsqcup_{n=1}^{+\infty} A_n\right) = \sum_{n=1}^{+\infty} \mu(A_n) \tag{1.2}$$

Definizione 1.3. Spazio di probabilitá
Uno spazio con misura (Ω, F, μ) in cui $\mu(\Omega) = 1$ é detto spazio di probabilitá. In particolare $\mu = P$ é chiamata *misura di probabilitá* o semplicemente probabilitá. Ogni elemento $\omega \in \Omega$ é detto *esito*; ogni $A \in F$ é detto *evento* e il numero $P(A)$ é la *probabilitá di A*.

Puó essere utile, come di seguito, introdurre il concetto di probabilitá condizionata per studiare la dipendenza tra eventi.

Definizione 1.4. Probabilitá condizionata
In uno spazio di probabilitá (Ω, F, P) sia B un evento *non trascurabile*, cioé $P(B) > 0$, la probabilitá di A condizionata a B é definita da:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \qquad A \in F \tag{1.3}$$

Teorema 1.1. Formula della probabilitá totale
Per ogni evento B tale che $0 < P(B) < 1$ vale

$$P(A) = P(A|B)\,P(B) + P\left(A|B^C\right)(1 - P(B))$$

Piú in generale, se $(B_i)_{i \in I}$ é una partizione finita o numerabile di Ω, con $P(B_i) > 0$ per ogni $i \in I$, vale

$$P(A) = \sum_{i \in I} P(A \cap B_i) = \sum_{i \in I} P(A|B_i)\,P(B_i) \tag{1.4}$$

dimostrazione.
La dimostrazione deriva direttamente dalla $\sigma-$additivitá della misura P.

Teorema 1.2. Formula di Bayes
Siano A, B eventi non trascurabili. Allora vale:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \tag{1.5}$$

dimostrazione.
Il risultato viene direttamente dalla definizione di probabilitá condizionata, considerando

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \qquad P(B|A) = \frac{P(B \cap A)}{P(A)}$$

Notando che $P(A \cap B)$ e $P(B \cap A)$ sono equivalenti, otteniamo

$$P(A|B)\,P(B) = P(B|A)\,P(A)\,.$$

Da qui il risultato.

Definizione 1.5. Indipendenza di eventi
In uno spazio di probabilitá (Ω, F, P) diciamo che due eventi A e B sono *indipendenti in* P se

$$P(A \cap B) = P(A)\,P(B) \tag{1.6}$$

Questo significa che la misura dell'intersezione tra i due eventi é il prodotto delle misure dei due eventi.
Il concetto di indipendenza tra eventi é relativo alla misura di probabilitá che sto considerando. Essere eventi indipendenti in P sta a significare che l'accadere dell'evento B non influenza la probabilitá di A.

Le nozioni e gli esempi svolti in questo elaborato si riferiscono a spazi di probabilitá discreti, ossia quando Ω é finito o al piú numerabile, in quanto é sufficiente rivolgersi al caso discreto per introdurci allo studio del machine learning.
Capiremo, in seguito, come ci si puó occupare di apprendimento automatico utilizzando spazi di probabilitá continui.
Consideriamo (Ω, F, P) e fissiamo $d \in \mathbb{N}$. dati $H \subseteq \mathbb{R}^d$ e una funzione $X : \Omega \to \mathbb{R}^d$, indichiamo con

$$(X \in H) := \{\omega \in \Omega | X(\omega) \in H\} = X^{-1}(H)$$

la contro-immagine di H mediante X. Intuitivamente $(X \in H)$ rappresenta l'insieme degli esiti ω, ossia i possibili stati del fenomeno aleatorio, tali che $X(\omega) \in H$.

Definizione 1.6. Variabile aleatoria
Una *variabile aleatoria* su (Ω, F, P) a valori in $\mathbb{R}^d$ é una funzione

$$X : \Omega \to \mathbb{R}^d$$

tale che $(X \in H) \in F$ per ogni $H \in \mathbb{B}_d$: scriviamo che $X \in mF$ e diciamo che X é F-misurabile.

Esempio 1.1. Distribuzione di Bernoulli
Sia p $\in [0,1]$. La *distribuzione di Bernoulli di parametro p* si indica con Be_p ed è definita come combinazione di due delta di Dirac:

$$Be_p \approx p\delta_1 + (1-p)\delta_0$$

dove

$$\delta_a(H) = \begin{cases} 1 & se\ \ a \in H \\ 0 & se\ \ a \notin H \end{cases}$$

In particolare, per $H \in B_d$ si ha

$$Be_p(H) = \begin{cases} 0 & se\ \ 0,1 \notin H \\ 1 & se\ \ 0,1 \in H \\ p & se\ \ 1 \in H\ \ e\ \ 0 \notin H \\ 1-p & se\ \ 0 \in H\ \ e\ \ 1 \notin H \end{cases}$$

Diciamo che una variabile aleatoria T ha distribuzione di Bernoulli, $T \approx Be_p$ con $p \in [0,1]$, se

$$P(X=i) = p^i(1-p)^{1-i} \qquad per\ \ i = 0,1.$$

In teoria della probabilitá uno dei concetti piú importanti relativi ad una variabile aleatoria discreta é il suo *valore atteso*, altro non é che la media dei possibili stati della variabile aleatoria pesata secondo le rispettive probabilitá.

Definizione 1.7. Valore atteso
In uno spazio di probabilitá (Ω, F, P) discreto, sia X una variabile aleatoria con distribuzione discreta finita

$$X \approx \sum_{k=1}^{n} p_k \delta_{x_k}$$

dove $p_k = P(X = x_k)$ per $k = 1, ..., n$, il valore atteso di X é

$$E[X] := \sum_{k=1}^{n} X_k p_k \tag{1.7}$$

$$\tag{1.8}$$

Nel caso in cui X sia una v.a. con distribuzione discreta non finita é sufficiente fare delle ipotesi sulla convergenza della serie.
Un risultato importante é

Teorema 1.3. Del calcolo della media per v.a. discrete
Siano

$$X : \Omega \to \mathbb{R}^d \qquad e \qquad f : \mathbb{R}^d \to \mathbb{R}^N$$

rispettivamente una v.a. discreta su (Ω, F, P) con legge $\mu_X = \sum_{k=1}^{+\infty} p_k \delta_{X_k}$ e f una funzione $\mathbb{B}_d - misurabile$ allora se $\sum_{k=1}^{+\infty} |x_k(t)| < +\infty$ vale

$$E[f(X)] = \sum_{k=1}^{+\infty} f(X_k) p_k \tag{1.9}$$

$$\tag{1.10}$$

Definizione 1.8. Varianza
Data una v.a. discreta reale tale che $\sum_{k=1}^{+\infty} |x|^2 < +\infty$. Si definisce *varianza di* X il numero reale non negativo

$$var(X) := E\left[(X - E[X])^2\right] = E\left[X^2\right] - E[X]^2 \tag{1.11}$$

La radice quadrata della varianza $\sqrt{var(X)}$ é detta *deviazione standard*.

Per assimilare in modo corretto le definizioni date possiamo pensare al valore atteso come la media scolastica di uno studente, e la deviazione standard come una misura della costanza dello studente, con una relazione proporzionalmente inversa, all'aumentare del valore di $\sqrt{var(X)}$ diminuisce la perseveranza dello studente.

La positivitá di $var(X)$ ci viene garantita dal seguente risultato

Teorema 1.4. Disuguaglianza di Jensen
Siano $-\infty < a < b < +\infty$ e

$$X : \Omega \to]a, b[\qquad e \qquad f :]a, b[\to \mathbb{R}$$

rispettivamente una v.a. sullo spazio (Ω, F, P) e una funzione convessa. Se $\sum_{k=1}^{+\infty} |X| < +\infty$ e $\sum_{k=1}^{+\infty} |f(X)| < +\infty$ allora si ha:

$$f(E[X]) \leq E[f(X)] \tag{1.12}$$

dimostrazione.
Se f é convessa vale che per ogni $z \in]a, b[$ esiste $m \in \mathbb{R}$ tale che

$$f(w) \geq f(z) + m(w - z), \qquad w \in]a, b[$$

Posto ora $z = E[X]$ ($E[X] \in]a, b[$ per ipotesi) si ha

$$f(X(w)) \geq f(E[X]) + m(X(w) - E[X]), \qquad w \in \Omega$$

da cui, prendendone il valore atteso e utilizzando la sua proprietá di monotonia ottengo

$$E[f(X)] \geq E[f(E[X]) + m(X(w) - E[X])] =$$

(utilizzando la proprietá di linearitá del valore atteso e il fatto che $E[c] = c$)

$$= f(E[X]) + m[X - E[X]] = f(E[X]).$$

Definizione 1.9. Covarianza
La covarianza di due variabili aleatorie é il numero reale

$$cov(X, Y) := E[(X - E[X])(Y - E[Y])]$$

Per la disuguaglianza di Cauchy-Schwartz si ha che

$$|cov(X, Y)| \leq \sqrt{var(X)var(Y)} \tag{1.13}$$

Definizione 1.10. Correlazione
Siano X,Y due variabili aleatorie reali tali che var(X) e var(Y) siano > 0. Il coefficiente di correlazione tra X e Y é

$$\rho_{XY} = \frac{cov(X, Y)}{var(X)var(Y)} \tag{1.14}$$

per la (1.14) abbiamo che $\rho \in [-1, 1]$ e in particolare se ρ é negativo le v.a. si dicono *negativamente correlate*, se ρ é positivo le v.a. si dicono positivamente correlate. Inoltre $|\rho| = 1$ se e soltanto se c'é una dipendenza lineare del tipo $Y = aX + b$.

1.2 Nozioni di Teoria dei Grafi

Per le nozioni di questo paragrafo si veda [7]

Definizione 1.11. Grafo
Un grafo G é una coppia (S, T) composta da un insieme finito di elementi detti nodi (o vertici) S e da un insieme di *connessioni* (o archi) T che connettono coppie di nodi.

Due nodi connessi da un arco sono detti *adiacenti*.

Definizione 1.12. Grafo orientato
Un grafo $G = (S, T)$ é un grafo orientato se $T \subset SxS$ é formato da coppie ordinate di nodi. L'arco (s_i, s_j) é un arco da s_i a s_j, si definisce s_j come la *testa* dell'arco e s_i come la *coda* dell'arco.

Se invece la relazione tra i nodi fosse simmetrica, il grafo si direbbe *non orientato*.

Definizione 1.13. Grafo pesato
Dati S, T e una funzione $\omega : T \to \mathbb{R}$ che associa un valore (o *peso*) ad ogni arco, il grafo $G = (S, T, \omega)$ é un grafo pesato.

Un grafo non pesato é un grafo pesato con ω costante uguale ad 1.

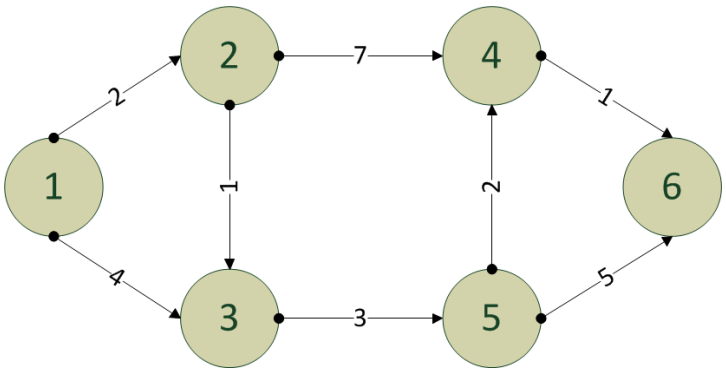


Figura 1.1: Esempio di grafo orientato pesato

Definizione 1.14. Grafo multipartito
Un grafo G é multipartito se l'insieme dei nodi S puó essere partizionato in k sottoinsiemi $S_1, ..., S_k$ tali che ciascun arco del grafo ha una delle due estremitá in S_l e l'altra in S_h con $l \neq h$, cioé se (s_i, s_j) é un arco del grafo allora s_i e s_j non possono appartenere allo stesso sottoinsieme di vertici.

Capitolo 2

Macchine di Boltzmann

Le macchine di Boltzmann vennero introdotte per la prima volta nel 1985 da Geoffrey Hinton e da Terry Sejnowski [10]. Vogliamo arrivare a questa struttura costruendola in modo rigoroso utilizzando dei concetti matematici.

2.1 Neurone formale

Nel 1943 il neurofisiologo Mc Cullog e il matematico Pitts introdussero il concetto di neurone formale.
Consideriamo il seguente spazio campionario

$$\Omega = \mathbb{R}$$

con la σ-algebra degli eventi data da

$$F = \left\{\emptyset, \Omega, A, A^C\right\}, \qquad A = \left(f(\sum_{i=1}^{N} \omega_{i,1} s_i + b) > m \right)$$

dove:

- $b \in \mathbb{R}$ é un valore numerico detto *bias*
- $\{s_i\}_{i=1}^N$ é una famiglia di v.a. (*ingressi*) indipendenti con distribuzione di Bernoulli che possono assumere i valori 0 o 1
- $\omega_{i,1} \in \mathbb{R}$ $i = 1, ..., N$ é detto il *peso* relativo ad ogni s_i
- $f : \mathbb{R} \longrightarrow \mathbb{R}$ é chiamata *funzione di attivazione*

- $m \in \mathbb{R}$ é un *valore di soglia* fissato

Una misura di probabilitá P su (Ω, F) puó essere introdotta assegnando una probabilitá all'evento A.

Definizione 2.1. Neurone formale
Un neurone formale é una variabile aleatoria dello spazio (Ω, F, P) tale che

$$X = \mathbb{1}_A = \begin{cases} 1 & se\ A \\ 0 & altrimenti \end{cases} \tag{2.1}$$

X ha distribuzione di Bernoulli, $X \approx Be_p$.

Osservazione 2.1. La definizione di neurone formale prevede l'assunzione di due soli possibili stati (uscite del neurone), uno attivo e l'altro inattivo, determinati dal valore di soglia prescelto.
La probabilitá di attivazione di un neurone formale cioé $P(X = 1) = p$ é determinata dalla funzione di attivazione considerata.

2.2 Rete neurale stocastica

In seguito, un neurone verrá sempre inteso come neurone formale secondo la definizione data.

Definizione 2.2. Rete neurale stocastica
Sia (S, T, ω) un grafo orientato pesato in cui S é un insieme di neuroni formali, sia $b : S \longrightarrow \mathbb{R}$ la funzione che associa ad ogni s_i il proprio bias b_i. Allora (S, T, ω, b) é una rete neurale stocastica e la funzione

$$\omega : \ T \longrightarrow \mathbb{R}$$

$$(s_i, s_j) \mapsto \omega_{ij}$$

é detta *peso delle connessioni*.
Inoltre, se $\omega_{ij} > 0$ la connessione é *eccitatoria*, se $\omega_{ij} < 0$ la connessione é *inibitoria*.

Perció una rete neurale stocastica é costituita da un insieme di neuroni formali, ognuno con un certo numero di connessioni in ingresso e/o un certo numero di connessioni in uscita. Ciascuna di queste unitá é una variabile aleatoria che dipende dalle proprie connessioni in ingresso e influenza tutte le altre unitá connesse tramite le connessioni in uscita.

Osservazione 2.2. In una rete neurale stocastica (S, T, ω), le connessioni $(s_i, s_j) \in T$ non sono neutre, infatti se consideriamo il neurone s_j, la sua funzione di attivazione riceverá come input il valore delle v.a. $s_i \; \forall s_i | (s_i, s_j) \in T$ moltiplicato per il peso relativo alla connessione ω_{ij}.
Il peso ω_{ij} si riferisce alla connessione che va dal neurone s_i al neurone s_j e non viceversa.

Definizione 2.3. Macchina di Boltzmann
Una rete neurale stocastica (S, T, ω, b) che soddisfa la seguente proprietá di simmetria

$$Se \;\; (s_i, s_j) \in E \;\; e \;\; (s_j, s_i) \in E \Rightarrow \omega_{ij} = \omega_{ji}$$

é una macchina di Boltzmann.
In una macchina di Boltzmann definiamo ω_{ij} la covarianza tra le due variabili aleatorie s_i e s_j.

Osservazione 2.3. Allo stesso modo potremmo definire una macchina di Boltzmann con $\omega : T \longrightarrow [-1, 1]$ in modo che ω_{ij} rappresenti la correlazione tra s_i e s_j.

2.3 Macchine di Boltzmann ristrette

Definizione 2.4. Macchine di Boltzmann ristrette
Una macchina di Boltzmann ristretta o RBM é una macchina di Boltzmann (S, T, ω, b) multipartita in k insiemi di neuroni $S_1, ..., S_k$ detti *strati* con la proprietá che $\exists \, (s_i, s_j) \in T$ $\forall s_i \in S_l$ e $s_j \in S_{l+1} \; \forall l = 1, ..., k-1$. Questo vale a dire che le connessioni tra i neuroni sono consentite con tutti i neuroni degli strati adiacenti.

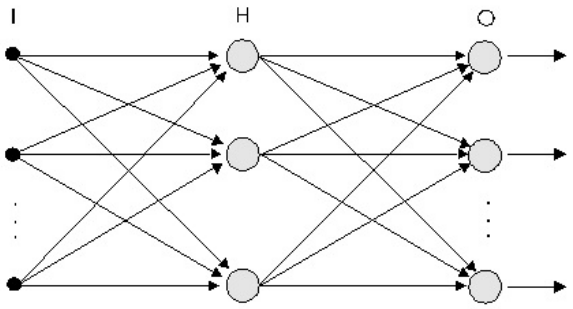


Figura 2.1: Rappresentazione di una RBM

Osservazione 2.4. Una macchina di Boltzmann non ha vincoli su numero di connessioni o numero di neuroni perció la quantitá di parametri da determinare durante un processo

di addestramento (definito in seguito) é molto elevata, questa puó essere ridotta imponendo dei limiti sulle connessioni (si veda ad esempio [9]), ed é per questo che abbiamo introdotto il concetto di RBM.

Osservazione 2.5. In seguito considereremo soltanto RBM a due strati (bipartite) in cui

- $V = \{0,1\}^D$ é lo spazio dei *vettori visibili* detto *strato visibile*
- $H = \{0,1\}^K$ é lo spazio dei *vettori nascosti* detto *strato nascosto*

Notazione 2.1. Indicheremo con $v \in V$ il vettore che ha come componenti gli stati dei neuroni visibili (v.a. dello strato visibile) e con $h \in H$ il vettore che ha come componenti gli stati dei neuroni nascosti (v.a. dello strato nascosto).

Definizione 2.5. Energia di una configurazione
Sia (S, T, ω, b) una RBM a due strati, l'energia di una configurazione congiunta dei vettori v ed h é definita come

$$E(v,h) = v^T b + h^T c + v^T W h$$

dove

- $b \in \mathbb{R}^D$ é il vettore dei bias relativi ai neuroni dello strato visibile
- $c \in \mathbb{R}^K$ é il vettore dei bias relativi ai neuroni dello strato nascosto
- $W \in \mathbb{R}^{DxK}$ é la matrice dei pesi ω_{ij} che mettono in relazione l'i-esimo neurone dello strato visibile con il j-esimo neurone dello strato nascosto.

Osservazione 2.6. Una RBM definisce la *distribuzione di Boltzmann*, in cui la probabilitá di una certa configurazione congiunta del vettore visibile e di quello nascosto é determinata solamente dall'energia della configurazione considerata rispetto all'energia di tutte le configurazioni possibili

$$p(v,h) = \frac{e^{E(v,h)}}{\sum_{v \in V} \sum_{h \in H} e^{E(v,h)}}$$

Prop 2.1. Per una RBM a due strati valgono le seguenti relazioni

$$P(h|v) = \prod_{i=1}^{K} P(h_i|v) \qquad P(v|h) = \prod_{j=1}^{D} P(v_i|h)$$

dimostrazione.
É possibile dimostrare che se X,Y sono due v.a. con distribuzione rispettivamente μ_X e μ_Y, il vettore aleatorio (X,Y) ha distribuzione $\mu_{(X,Y)} = \mu_X \mu_Y \Leftrightarrow$ X e Y sono indipendenti.

In una RBM i neuroni dello stesso strato non sono connessi tra loro cioé sono indipendenti. Infatti se s_i, $s_j \in V$ si ha $\omega_{ij} = cov(s_i, s_j) = 0 \; \forall i \neq j$. Vista l'indipendenza

$$P(h|v) = \frac{P(h \cap v)}{P(v)} = \prod_{i=1}^{K} \frac{P(h_i \cap v)}{P(v)} = \prod_{i=1}^{K} P(h_i|v)$$

e analogamente

$$P(v|h) = \frac{P(v \cap h)}{P(h)} = \prod_{i=1}^{K} \frac{P(v_i \cap h)}{P(h)} = \prod_{i=1}^{K} P(v_i|h)$$

2.4 Metodo della massima verosimiglianza

Definizione 2.6. Processo di addestramento
Per processo di addestramento di una rete neurale stocastica (S, T, ω, b) si intende un procedimento per la determinazione dei pesi ω_{ij} e dei bias relativi ai neuroni in modo da rendere noti i parametri della distribuzione di probabilitá che la caratterizza.

Sia $x = (x_1, ..., x_n)$ un *campione* costituito da n eventi indipendenti, la cui probabilitá dipende da un vettore di parametri θ.

Definizione 2.7. Funzione di verosimiglianza
Dato x campione, la funzione

$$L(x, \theta) = \prod_{i=1}^{n} P_\theta(x_i)$$

é la funzione di verosimiglianza associata al campione.

Osservazione 2.7. La funzione di verosimiglianza associata ad un campione rappresenta la funzione di probabilitá $P_\theta(x)$ del campione stesso, vista l'indipendenza degli eventi che lo costituiscono.

Il metodo della massima verosimiglianza come processo di addestramento di una rete neurale stocastica consiste nella determinazione del vettore dei parametri θ relativi ad una distribuzione di probabilitá che massimizzi la funzione di verosimiglianza.

Definizione 2.8. Stimatore di massima verosimiglianza
Data una funzione di verosimiglianza $L(x, \theta)$

$$\theta^* = arg \max L(x, \theta)$$

é detto stimatore di massima verosimiglianza.

Osservazione 2.8. Per facilitarne il calcolo si introduce la funzione di *log-verosimiglianza*

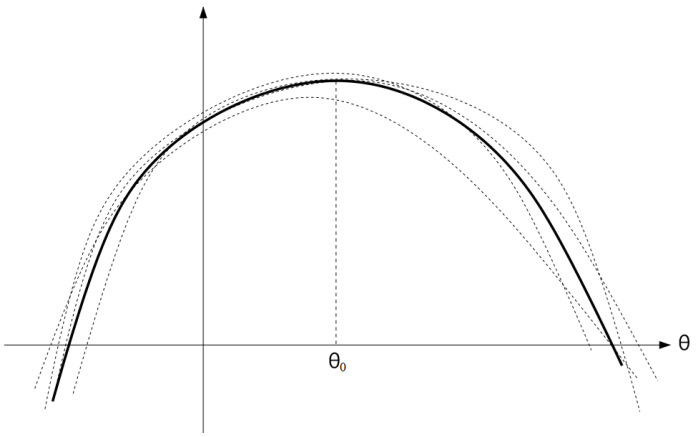
$$l(x,\theta) := \log L(x,\theta)$$

Il logaritmo é una funzione monotona crescente che non va a modificare le caratteristiche della funzione $L(x,\theta)$ in termini di crescenza e decrescenza, ma soprattutto si ottiene una forma piú semplice da trattare. In effetti, nel caso di variabili i.i.d, come nel nostro campione, si ottiene

$$l(x,\theta) = \log \prod_{i=1}^{n} P_\theta(x_i) = \sum_{i=1}^{n} P_\theta(x_i)$$

Teorema 2.1. del valore atteso della log-verosimiglianza
La funzione $E[l(x,\theta)]$ assume un massimo in corrispondenza di θ_0 cioé del vero valore della funzione di probabilitá di x.

La figura seguente rappresenta l'andamento della funzione di log-verosimiglianza dei campioni (linee tratteggiate) relativamente a quello di $E[l(x,\theta)]$ (linea continua).



dimostrazione.
Consideriamo $A[\theta] = E\left[\frac{L(x,\theta)}{L(x,\theta_0)}\right]$
Ricordiamo che stiamo considerando il caso discreto e che $L(x,\theta)$ é una funzione di probabilitá di x per ogni valore di θ vale

$$A[\theta] = \sum_x \frac{L(x,\theta)}{L(x,\theta_0)} L(x,\theta_0) = \sum_x L(x,\theta) = 1$$

Da ció segue che $\log A[\theta] = 0$. Siccome la funzione log é concava, possiamo applicare la disuguaglianza di Jensen (1.14)

$$E\left[\log \frac{L(x,\theta)}{L(x,\theta_0)}\right] \leq \log E\left[\frac{L(x,\theta)}{L(x,\theta_0)}\right] = 0$$

$$E\left[l\left(x,\theta\right)-l\left(x,\theta_0\right)\right]\leq 0$$

Utilizzando la linearitá del valore atteso

$$E\left[l\left(x,\theta\right)\right]\leq E\left[l\left(x,\theta_0\right)\right]$$

Questa relazione é vera per ogni θ!

Notazione 2.2. In seguito, per provare la consistenza del metodo della massima verosimiglianza, indicheremo la funzione di log-verosimiglianza con $l_n\left(x,\theta\right)$ e il suo stimatore di massima verosimiglianza con θ_n^* per esplicitare la dimensione n del campione.

Osservazione 2.9. La relazione derivata nel teorema precedente ci mette in evidenza che se $\theta\neq\theta_0$, per ogni $\epsilon>0$, per ogni $\delta\in\left]0,1\right[$ $\exists n_0\in\mathbb{N}$ tale che:

$$P\left(l_n\left(x,\theta_0\right)-l_n\left(x,\theta_j\right)>\epsilon\ \ per\ \ ogni\ \ n>n_0\right)\geq 1-\delta$$

perché quando n diverge la funzione di verosimiglianza in θ_0 tende a diventare molto piú grande di ogni altro valore di θ.

Definizione 2.9. Stimatore consistente
Sia θ_n^* uno stimatore per la funzione di log-verosimiglianza e θ_0 il valore di θ corrispondente alla vera distribuzione di probabilitá del campione x. Diremo che θ_n^* é uno stimatore consistente del parametro se

$$\lim_{n\to\infty}P\left(\left|\theta_n^*-\theta_0\right|>\epsilon\right)=0\quad\ \ per\ \ ogni\ \ \epsilon>0$$

Teorema 2.2. Consistenza dello stimatore di massima verosimiglianza
Data $l_n\left(x,\theta\right)$ funzione di log-verosimiglianza del campione x, lo stimatore di massima verosimiglianza θ^* é uno stimatore consistente del parametro θ.

dimostrazione.
Per la dimostrazione ci restringiamo al caso in cui θ possa assumere un numero finito di valori θ_j con $j=0,...,m$ (vale analogamente anche nel caso in cui $\theta\in\mathbb{R}$).
Definiamo

$$A_j=\left\{l_n\left(x,\theta_0\right)-l_n\left(x,\theta_j\right)>\epsilon\ \ per\ \ ogni\ \ n>n_0\right\}\quad\ \ per\ \ j=1,...,m.$$

la cui probabilitá, per quanto visto nel teorema (2.1), puó essere resa maggiore di $1-\delta$, scegliendo un certo n_0 sufficientemente elevato. Perció

$$\begin{aligned} P\left(\bigcap_{j=1}^{m} A_j\right) &= 1 - P\left(\bigcap_{j=1}^{m} A_j^C\right) \\ &\geq 1 - \sum_{j=1}^{m} P\left(A_j^C\right) \\ &\geq 1 - m(1-(1-\delta)) \\ &= 1 - m\delta \end{aligned} \tag{2.2}$$

e cosí abbiamo mostrato che

$$P\left(l_n\left(x,\theta_0\right) - l_n\left(x,\theta_j\right) > \epsilon \quad per\ \ ogni\ \ j \neq 0\ \ per\ \ ogni\ \ n > n_0\right) \geq 1 - m\delta$$

con δ arbitrario. Ció é un risultato ancora piú forte della consistenza definita per uno stimatore!

Osservazione 2.10. Per massimizzare la funzione di log-verosimiglianza é necessario porre

$$\frac{\partial l\left(x,\theta\right)}{\partial \theta} = 0$$

Naturalmente, se θ fosse un vettore di parametri, la formula qui sopra verrá intesa come derivata rispetto ad ogni componente del vettore considerando costanti le altre.
Se il campione é composto da n eventi indipendenti possiamo porre

$$\frac{\partial l\left(x,\theta\right)}{\partial \theta} = \sum_{i=1}^{n} \frac{\partial \log P_\theta(x_i)}{\partial \theta} = 0$$

Esempio 2.1. Stimatore di una distribuzione di Bernoulli
Diciamo che una variabile aleatoria X ha distribuzione di Bernoulli se

$$P\left(X = i\right) = p^i(1-p)^{1-i} \qquad per \quad i = 0, 1.$$

Dato un campione costituito da una famiglia $\{x_i\}_{i=1}^n$ di eventi indipendenti, uno stimatore per il parametro p incognito (in questo caso θ =p) possiamo trovarlo applicando il metodo della massima verosimiglianza. La funzione di verosimiglianza é

$$L\left(x,p\right) = \prod_{i=1}^{n} p^{x_i}\left(1-p\right)^{1-x_i} = p^{\xi}\left(1-p\right)^{n-\xi}$$

dove $\xi = \sum_{i=1}^{n} x_i$
La funzione di log-verosimiglianza é data da

$$l\left(x,p\right) = \log p \sum_{i=1}^{n} x_i + \log\left(1-p\right) \sum_{i=1}^{n} \left(1-x_i\right)$$

$$l\left(x,p\right) = \xi \log p + \left(n-\xi\right) \log 1 - p$$

Derivando la funzione di log-verosimiglianza

$$\frac{\partial l\left(x,p\right)}{\partial p} = \frac{\xi}{p} - \frac{n-\xi}{1-p}$$

e ponendo

$$\frac{\partial l\left(x,p\right)}{\partial p} = \frac{\xi}{p} - \frac{n-\xi}{1-p} = 0$$

otteniamo uno stimatore per p

$$p^* = \frac{1}{n}\xi = \frac{1}{n}\sum_{i=1}^{n} x_i$$

cioé lo stimatore p per una variabile con distribuzione di Bernoulli é la media dei dati del campione.

2.5 Addestramento di una RBM

Il caso piú semplice che possiamo trattare é quello in cui la nostra RBM bipartita sia costituita da due soli neuroni che hanno dimensione 1 cioé due variabili aleatorie che possono assumere il valore 0 o 1. Indichiamo con b il bias relativo al neurone visibile, con c il bias relativo al neurone nascosto e siano inoltre $V = H = \{0,1\}$.
In questo caso il campione é rappresentato da

$$\phi_1 = \left(v^{(1)}, h^{(1)}\right)$$

$$\phi_2 = \left(v^{(2)}, h^{(2)}\right)$$

$$\ldots$$

$$\phi_N = \left(v^{(N)}, h^{(N)}\right)$$

La distribuzione di probabilitá é

$$P_\theta(v,h) = \frac{e^{bv+ch+\omega vh}}{\sum_{v\in V}\sum_{h\in H} e^{bv+ch+\omega vh}}$$

dove ω é il paramentro di interazione tra i due neuroni e $\theta = (b, c, \omega)$ é il vettore dei parametri da determinare durante il processo di addestramento.
Costruiamo la funzione di log-verosimiglianza per calcolare b, c e W

$$l\left(b, c, \omega\right) = \log \prod_{i=1}^{N} P_\theta \left(v^{(i)}, h^{(i)}\right)$$

Per comoditá pongo $Z = \sum_{v \in V} \sum_{h \in H} e^{bv + ch + \omega vh}$

$$\begin{aligned} l\left(b, c, \omega\right) &= \log \prod_{i=1}^{N} \frac{e^{bv^{(i)} + ch^{(i)} + \omega v^{(i)} h^{(i)}}}{Z} \\ &= \log \frac{e^{\sum_{i=1}^{N} bv^{(i)} + ch^{(i)} + \omega v^{(i)} h^{(i)}}}{Z^N} \\ &= \log e^{\sum_{i=1}^{N} bv^{(i)} + ch^{(i)} + \omega v^{(i)} h^{(i)}} - \log Z^N \\ &= \sum_{i=1}^{N} bv^{(i)} + ch^{(i)} + \omega v^{(i)} h^{(i)} - N \log Z \end{aligned} \tag{2.3}$$

Cerchiamo b, c, ω che massimizzino la funzione di log-verosimiglianza, perció poniamo

$$\frac{\partial l}{\partial b} = 0, \qquad \frac{\partial l}{\partial c} = 0, \qquad \frac{\partial l}{\partial \omega} = 0.$$

Svolgendo i calcoli

$$\frac{\partial l}{\partial b} = \sum_{i=1}^{N} v^{(i)} - N \frac{\sum_{v \in V} \sum_{h \in H} v e^{bv + ch + \omega vh}}{Z} = 0$$

$$\frac{\sum_{v \in V} \sum_{h \in H} v e^{bv + ch + \omega vh}}{Z} = \frac{\sum_{i=1}^{N} v^{(i)}}{N}$$

$$\sum_{v \in V} \sum_{h \in H} p(v, h) v = \frac{\sum_{i=1}^{N} v^{(i)}}{N} \tag{2.4}$$

Analogamente rispetto c e ω si ha

$$\sum_{v \in V} \sum_{h \in H} p(v, h) h = \frac{\sum_{i=1}^{N} h^{(i)}}{N} \tag{2.5}$$

$$\sum_{v \in V} \sum_{h \in H} p(v,h) vh = \frac{\sum_{i=1}^{N} v^{(i)} h^{(i)}}{N} \tag{2.6}$$

Si noti che le espressioni in (2.3), (2.4) e (2.5) sono equivalenze tra il valore atteso della distribuzione congiunta e la media aritmetica dei dati del campione.

Capitolo 3

Algoritmo

In questo capitolo verrá presentato un algoritmo di machine learning in Python per la classificazione di oggetti. Dato un fiore di iris in input vogliamo ottenere la sua categoria di appartenenza in output.

Primo passo: é necessario scaricare tutto quello di cui avremo bisogno per eseguire l'algoritmo.
Partiamo con il download di *Anaconda*, una piattaforma per il Data Science con Python (linguaggio di programmazione), una volta completato il download dobbiamo installare nel seguente ordine:

- *Numpy*, una libreria opensource che contiene funzioni matematiche utili per lavorare con matrici e vettori multidimensionali;
- *Sklearn*, una libreria opensource di apprendimento automatico che contiene algoritmi di classificazione e machine learning;
- *Tensorflow*, una liberia opensource che fornisce modelli sperimentati e ottimizzati per il machine learning.

Secondo passo: per il processo di addestramento di un modello occorre fornire all'algoritmo di apprendimento i dati di addestramento da cui possa apprendere.

```
1      from sklearn import datasets
2      iris_dataset = datasets.load_iris()
3      print(iris_dataset['DESCR'])
```

Il dataset di iris é un insieme di addestramento composto da esempi che misurano la lunghezza e la larghezza del petalo e del sepalo di Iris-Setosa, Iris-Versicolor e Iris-Virginica. L'ultimo comando ci fornisce la descrizione del dataset di Iris:

```
:Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Class Distribution: 33.3\% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (July, 1988)
```

I dati sono giá separati all'interno del dataset in data, che comprende la matrice con le varie caratteristiche per ogni esempio, e target dove c'é l'output.

```
4       X = iris_dataset.data
5       y = iris_dataset.target
```

In seguito vediamo la struttura matriciale del dataset, la riga k-esima della matrice degli input iris_dataset.data é associata al k-esimo elemento del vettore degli output iris_dataset.target che puó assumere i valori 0, 1 o 2 relativi alle tre classi di appartenenza:

```
>> iris_dataset.data

array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [...])

>>iris_dataset.target

array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,...])
```

Abbiamo poi utilizzato il metodo train_test_split che divide il dataset in dati di addestramento e dati di test, in modo che i dati di test, utilizzati a fine addestramento per capire se il modello ha appreso tanto da riuscire a "generalizzare", non siano uguali ai dati di addestramento, altrimenti testeremo il modello solo sugli esempi che gli sono stati forniti.

```
6       from sklearn.model_selection import train_test_split
7       X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)
```

Il parametro test_size in train_test_split indica la percentuale di esempi per l'insieme di test (30%).

Passo 3: consiste nel scegliere e addestrare il modello sui dati di training:

```
8       from sklearn.linear_model import Perceptron
9       model= Perceptron(max_iter=100)
10      model.fit(X_train, y_train)
```

Perceptron (percettrone) é uno dei primi algoritmi di classificazione della storia, a livello strutturale é una macchina di Boltzmann ristretta.
L'attributo max_iter fissa il numero massimo di iterazioni, altri parametri sono predefiniti.
Arrivati a questo punto il modello é addestrato, dobbiamo capire se i dati scelti sono consistenti con il metodo (ovvero permettono "di generalizzare").

Passo 4: facciamo predire al modello sui dati di training che ha giá appreso (per vedere se restituisce gli stessi output che gli abbiamo dato) e sui dati di test (per verificare la capacitá di classificare un oggetto del quale non ha precedentemente appreso la classe di appartenenza).

```
11      predicted_train = model.predict(X_train)
12      predicted_test  = model.predict(X_test)
```

Siamo in grado di ottenere l'accuratezza del modello, sui dati di training e sui dati di test:

```
13      from sklearn.metrics import accuracy_score
14      print('Train accuracy')
15      print( accuracy_score(y_train, predicted_train) )
16      print('Test accuracy')
17      print( accuracy_score(y_test, predicted_test) )
```

Facendo un primo tentativo:

```
Train accuracy
0.9047619047619048
Test accuracy
0.8444444444444444
```

cioé il modello ha un'accuratezza del 90,5% sui dati di training e del 84,4% sui dati di test.

Esistono molti altri algoritmi per il machine learning per la classificazione, anche molto piú efficienti di quello visto.
Nello specifico con la libreria tensorflow é possibile addestrare modelli RBM scegliendo il numero di strati nascosti e per ognuno di questi il numero di neuroni:

```
1       import tensorflow as tf
2       columns = [tf.contrib.layers.real_valued_column("", dimension=4)]
3       classifier = tf.contrib.learn.DNNClassifier(feature_columns=columns,
                     hidden_units=[10, 20, 10], n_classes=3,
                     model_dir=iris_dataset)
4       classifier.fit(iris_dataset.data, iris_dataset.target)
```

Bibliografia

[1] Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

[2] Domingos, P. (2012). *A few useful things to know about machine learning*. Communications of the ACM, 55(10), 78-87.

[3] Benjamin, A., Chartrand, G., Zhang, P. (2017). *The fascinating world of graph theory*. Princeton University Press.

[4] Rajkomar, A., Dean, J., Kohane, I. (2019). *Machine learning in medicine*. New England Journal of Medicine, 380(14), 1347-1358.

[5] Li, Q., Zheng, N., Cheng, H. (2004). Springrobot: A prototype autonomous vehicle and its algorithms for lane detection. *IEEE Transactions on Intelligent Transportation Systems*, 5(4), 300-308.

[6] Minsky, M., Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. MIT press.

[7] West, D. B. (2001). *Introduction to graph theory* (Vol. 2). Upper Saddle River: Prentice hall.

[8] Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons—from backpropagation to adaptive learning algorithms. Computer Standards Interfaces, 16(3), 265-278.

[9] Larochelle, H., Mandel, M., Pascanu, R., Bengio, Y. (2012). *Learning algorithms for the classification restricted boltzmann machine*. Journal of Machine Learning Research, 13(Mar), 643-669.

[10] Hinton, G. E., Sejnowski, T. J., Ackley, D. H. (1984). *Boltzmann machines: Constraint satisfaction networks that learn*. Pittsburgh: Carnegie-Mellon University, Department of Computer Science.

[11] Pascucci, Andrea. (2020). *Teoria della probabilitá*. Springer-Verlag Mailand.

[12] Minini, Andrea. http://www.andreaminini.com/python/sklearn/come-fare-machine-learning-con-scikit-learn-di-python.

[13] Minini, Andrea. http://www.andreaminini.com/ai/tensorflow/esempio-tutorial-tensorflow.