MACHINE LEARNING

APPUNTI A CURA DI: RICCARDO LO IACONO

Università degli studi di Palermo

a.a. 2023-2024

Indice.

– 1 – Classificatori Bayesiani.

Sia $X = (x_1, \dots, x_n)^T$ un vettore di features. Siano $\omega = (\omega_1, \dots, \omega_m)$ classi distinte e sia $\Pr(\omega_i|X)$ la probabilità che ω_i sia la classe di appartenenza di X. Quel che si fa con i classificatori bayesiani, è massimizzare tale probabilità. Per far ciò si definisce la funzione di rischio

$$r = r_1 \Pr(\omega_1) + \cdots + r_n \Pr(\omega_n)$$

ove

$$r_i = \sum_{j=1}^{m} \lambda_{ij} \int_{R_j} \Pr(X|\omega_i)\, \mathrm{d}X$$

con R_j j-esima *superficie decisionale* (si veda la sezione a seguire), λ_{ij} penalità per aver assegnato X a ω_i quando la classe corretta è ω_j.

– 1.1 – Superfici decisionali.

Si parta dal considerare il caso bidimensionale. Sia $X = (x_1, \dots, x_n)^T$ un vettore di features, e siano ω_1, ω_2 le due possibili classi. Allora se rappresentato X su di un piano, è possibile identificare due regioni, siano queste R_1, R_2, tali che

$$X \in \begin{cases} R_1 \iff \Pr(\omega_1|X) > \Pr(\omega_2|X), \\ R_2 \iff \Pr(\omega_2|X) > \Pr(\omega_1|X). \end{cases}$$

Si definisce superficie decisionale una funzione $g(x)$ tale che $\Pr(\omega_1|X) - \Pr(\omega_2|X) = 0$.

Più in generale, supposto $X = (x_1, \dots, x_n)^T$ un vettore di features e $\omega = (\omega_1, \dots, \omega_m)$ possibili classi, una superficie decisionale è una funzione $g_{ij}(x)$ tale che $\Pr(\omega_i|X) - \Pr\left(\omega_j\middle|X\right) = 0$, $\forall i, j \in \{1, \dots, m\}, i \neq j$.

– 1.2 – Stima della densità di probabilità.

Noto come calcolare la probabilità per ogni classe, resta il problema di come identificare la distribuzione di probabilità dei dati. Si distinguono in questo contesto due approcci:

- *approccio parametrico:* è nota la forma funzionale dei dati, da cui è facile ricavare la distribuzione di probabilità;
- *approccio non-parametrico:* sono noti i valori di alcune features, si può allora stimare la forma funzionale.

Nello specifico a seguito ci si concentra suglia approcci funzionali, in particolare saranno trattati i criteri di *massima verosimiglianza* e *massima probabilità a posteriori*.

– 1.2.1 – Massima verosimiglianza.

Sia supposto $X = (x_1, \dots, x_n)^T$ un vettore di features, con x_i stocasticamente indipendente da $x_j, \forall i \neq j$. Sia inoltre $\Pr(X)$ nota, unicamente dipendente da un qualche parametro ignoto θ; cioè

$$\Pr(X) = \Pr(X|\theta) = \prod_{i=1}^{n} \Pr(x_i|\theta)$$

Si definisce $\Pr(X|\theta)$ verosimiglianza di θ ad X. Segue banalmente

$$\theta_{ML} = \arg\max_{\theta} \left\{ \prod_{i=1}^{n} \Pr(x_i|\theta) \right\}$$

– 1.2.2 – Massima probabilità a posteriori.

Il criterio di massimizza verosimiglianza non sempre è applicabile, si procede in questi casi ad applicare il criterio di massima probabilità a posteriori. Per esso, noto $X = (x_1, \dots, x_n)^T$ vettore di features, si deve calcolare θ_{MAP} tale da massimizzare $\Pr(\theta|X)$. Dal *teorema di Bayes* si ha

$$\Pr(\theta|X) = \frac{\Pr(\theta)\Pr(X|\theta)}{\Pr(X)}$$

da cui segue che

$$\begin{aligned}\theta_{MAP} &= \arg\max_{\theta} \{\Pr(\theta|X)\} \\ &= \arg\max_{\theta} \left\{ \frac{\Pr(\theta)\Pr(X|\theta)}{\Pr(X)} \right\}\end{aligned}$$

– 1.3 – Classificatori naive.

Siano $X \in \mathbb{R}^n$ un vettore di features, $\omega = (\omega_1, \dots, \omega_m)$, e si supponga di dover stabilire $\Pr(X|\omega_i)$, per $i \in \{1, \dots, m\}$. In generale, affinché si possa avere una buona stima della funzione di densita sarebbero necessari n^m punti. Se si assume però che x_i e x_j sono stocasticamente indipendenti per ogni $i, j \in \{1, \dots, n\}, i \neq j$, allora

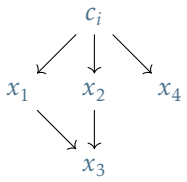
$$\Pr(X|\omega_i) = \prod_{j=1}^{n} \Pr\left(x_j \middle| \omega_i\right)$$

caso in cui $n \cdot m$ punti risultano sufficienti. Si dimostra che anche nei casi in cui tale indipendenza non sia rispettata, un classificare bayesiano da risultati soddisfacenti.

– 1.4 – Reti Bayesiane.

Come detto in generale i classificatori bayesiani operano bene in molti casi. Vi sono casi però in cui è necessario calcolare le probabilità congiunte in maniera esatta, nascono per tale ragione le reti Bayesiane. Queste procedono come semplici classificatori bayesiani, ma all'occorrenza calcolano opportunamente le probabilità congiunte.

Esempio: sia supposto $X = (x_1, \dots, x_4)$ secondo le relazioni rappresentate dal grafo a seguito riportato.



e siano $c = (c_1, \dots, c_3)$ classi, allora

$$\Pr(c_i|X) = \Pr(c_i)\Pr(x_3|x_2x_1)\Pr(x_4)$$

Più in generale, una rete (o network) bayesiana è un grafo diretto e aciclico i cui nodi rappresentano le variabile.

– 2 – Classificatori context-dependent.

I classificatori discussi sinora, sono utilizzabili in casi in cui i dati sono simultaneamente presenti, ma soprattutto se le classi sono unicamente dipendenti dai valori assunte dalle stesse e da quello delle features. Esistono però scenari in cui tale situazione non si verifica; è pertanto necessario poter definire modelli che apprendono dinamicamente.

– 2.1 – Classificatori Bayesiani context-dependent.

Sia supposto $X = (x_1, \dots, x_n)$ un vettore di features e siano $\omega = (\omega_1, \dots, \omega_m)$ classi. Per quanto detto sinora X è assegnato ad $\omega_i \iff \Pr(\omega_i|X) > \Pr\left(\omega_j\middle|X\right), \forall i \neq j$. Come detto però, ciò è limitato ai casi in cui vi è una sorta di indipendenza tra le classi. Considerando il caso in cui invece tale indipendenza viene meno, sia

$$\Omega_i = \left\{\omega_{i_j}\right\}_{j\in\{1,\dots,n\}}$$

Da ciò la regola di classificazione Bayesiana può essere riscritta come

$$X \to \Omega_i \iff \Pr(\Omega_i|X) > \Pr\left(\Omega_j\middle|X\right), \forall i \neq j$$

Ora, affinche il modello possa essere definito context-dependent, è necessario che esso tenga traccia degli stati precedenti del classificatore; per farlo, tra le altre possibilità, vi sono le catene di Markov, per le quali

$$\Pr\left(\omega_{i_k}\middle|\omega_{i_{k-1}}, \dots, \omega_{i_1}\right) = \Pr\left(\omega_{i_k}\middle|\omega_{i_{k-1}}\right) \tag{1}$$

cioè la dipendenza è ristretta all'ultimo stato della classe.

Definizione: un processo statistico tale da soddisfare l'*Equazione* (1) è detto *processo di Markov*.

– 2.1.1 – Equazioni di Chapman-Kolmogorov.

Vantaggio principale dei modelli basati sulle catene di Markov, è che, attraverso quelle che sono note come *equazioni di Chapman-Kolmogorov*, è possibile determinare lo stato in cui si troverà in futuro (si veda l'esempio a seguire). Nello specifico, partendo dal definire le *probabilità transitorie ad un passo* come

$$p_{ij}(k) = \Pr(X_{k+1} = j|X_k = i)$$

ove con X_k si intendo lo stato del classificatore allo stato k, e tali che

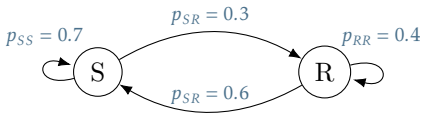
$$\sum_{j=1}^{N} p_{ij}(k) = 1, \forall i, k \in \{1, \dots, M\}.$$

Sfruttando la legge della probabilità totale, l'equazione di Chapman-Kolmogorov permette di definire la probabilità transitoria a n passi, come

$$p_{ij}(k) = \sum_{r=1}^{R} p_{ir}(k, u) p_{rj}(u, k+n), \qquad n \leq u \leq k+n \tag{2}$$

ove $p_{ij}(k, u) = \Pr(X_u = j|X_k = i)$.

Esempio: si supponga un modello meteorologico, come quello a seguire.



Si supponga di voler calcolare la probabilità che tra due giorni piova, supposto che oggi vi sia il sole. Dall'*Equazione* (2) segue

$$p_{SR}(d_0, d_2) = p_{SS}(d_0, d_1)p_{SR}(d_1, d_2) + p_{SR}(d_0, d_1)p_{RR}(d_1, d_2) = \cdots = 0.33$$

ove $d_i, i = 0, 1, 2$ indica il numero di giorni da quello attuale.

– 2.1.2 – Matrici di transizione.

Sia considerata *Equazione 2*, si osserva che questa rappresenta il prodotto riga-colonna di una qualche matrice. Sia definita allora

$$H(k, k+n) = \left[p_{ij}(k, k+n)\right]$$

come la matrice di transizione a n passi. Con tale formulazione, l'equazione di Chapman-Kolmogorov, può essere riscritta equivalentemente come

$$H(k, k+n) = H(k, u)H(u, k+n)$$

da cui scegliendo opportunamente u è possibile definire

- *l'equazione di Chapman-Kolmogorov in avanti* se si pone $u = k + n - 1$, da cui

  $$\begin{aligned} H(k, k+n) &= H(k, k+n-1)H(H+n-1, k+n) \\ &= H(k, k+n-1)P(k+n-1) \end{aligned}$$

  oppure;

- *l'equazione di Chapman-Kolmogorov all'indietro* se si pone $u = k + 1$, da cui

  $$\begin{aligned} H(k, k+n) &= H(k, k+1)H(H+1, k+n) \\ &= P(k)H(k+1, k+n) \end{aligned}$$

  con P *matrice di transizione*.

Inoltre, se $P(k) = P$, ossia

$$p_{ij}(k) = \Pr(X_{k+1} = j | X_k = i) = \Pr(X_k = j | X_{k-1} = i)$$

allora il processo di Markov sarà detto *omogeneo*.

– 2.1.3 – Probabilità di stato.

Una quantità spesso utile quando si lavora con le catene di Markov, è quella relativa la probabilità di trovare la catena in un certo stato. Sia allora definita

$$\begin{aligned}\pi_i(k) &= \Pr(X_k = i) \\ &= \sum_j \Pr(X_k = i, X_{k-1} = j)\Pr(X_{k-1} = j) \\ &= \sum_j p_{ij}(k)\pi_{j(k-1)}\end{aligned}$$

Volendo inoltre definire la probabilità dipendente dal tempo, questa diventa

$$\pi_j^n = \Pr(X_n = j)$$

Posti $\pi(k) = \begin{pmatrix}\pi_0(k) & \pi_1(k) & \cdots\end{pmatrix}$ e $\pi^n(k) = \begin{pmatrix}\pi_0^n(k) & \pi_1^n(k) & \cdots\end{pmatrix}$, se

$$\lim_{n\to\infty} \pi^n(k) = \pi(k)$$

si definisce $\pi(k)$ *distribuzione limite.*

– 2.2 – Classificatori Bayesiani pt.2.

Per quanto detto sinora, è noto che

$$\begin{aligned}\Pr(\Omega_i) &= \Pr\left(\omega_{i_1}, \cdots, \omega_{i_n}\right) \\ &= \Pr\left(\omega_{i_n}\middle|\omega_{i_{n-1}}, \cdots, \omega_{i_1}\right)\Pr\left(\omega_{i_{n-1}}\middle|\omega_{i_{n-2}}, \cdots, \omega_{i_1}\right)\cdots\Pr\left(\omega_{i_2}\middle|\omega_{i_1}\right)\Pr(\omega_i) \\ &= \left(\prod_{k=2}^{n}\Pr\left(\omega_{i_k}\middle|\omega_{i_{k-1}}\right)\right)\Pr\left(\omega_{i_1}\right)\end{aligned}$$

Se si considera ora X vettore di feature, le cui componenti $x_i, i \in \{1, \ldots, n\}$, sono tra loro stocasticamente indipendenti, allora la funzione di densità di probabilita di ogni classe è indipendenti dalle altre. Si ha quindi

$$\Pr(X|\Omega_i) = \prod_{k=1}^{n}\Pr\left(X_k\middle|\omega_{i_k}\right)$$

che nel caso di modelli basati su catene di Markov diventa

$$\Pr(X|\Omega_i)\Pr(\Omega_i) = \Pr\left(\omega_{i_1}\right)\prod_{k=2}^{n}\Pr\left(\omega_{i_k}\middle|\omega_{i_{k-1}}\right)\Pr\left(x_k\middle|\omega_{i_k}\right)$$

che dovendo essere massimizzata, risulta tale se e solo se ciascun termine è massimizzato.
Osservazione. In termini computazionali, assunte N misurazioni e M classi distinte, l'operazione di massimizzazione di cui sopra richiederebbe tempo $\mathcal{O}\left(NM^N\right)$.

– 2.3 – Addestramento e riconoscimento attraverso HMM.

Sia considerato un modello i cui stati non sono direttamente osservabili, ma si può unicamente risalire ad essi solo dai dati di addestramento: tale tipologia di modelli sono detti *hidden Markov models.*

Tali modelli sono in genere applicati per una serie di problemi di natura bio-informatica. Più in generale, gli HMM sono descritti dalla seguente quadrupla

$$S = (\Pr(i|j), \Pr(X|j), \Pr(i), k)$$

con

- $\Pr(i|j)$ insieme delle probabilità di transizione;
- $\Pr(X|i)$ insieme delle probabilità a priori;
- $\Pr(i)$ insieme delle probabilità di stato iniziale;
- k numero degli stati.

Considerando ora un'applicazione di tali modelli, si descrive a seguito il *pattern recognition.*

– 2.3.1 – Pattern recognition.

Siano dati m pattern di riferimento, ognuno descritto da un HMM e di questi si trovi S. Sia ora p un nuovo pattern; riconoscere p equivale a ricercare quale tra gli m pattern sia più simile a p stesso.

Più in generale, sia M un insieme di modelli noti, e sia $X = (x_1, \dots, x_n)$ sequenza di osservazioni. Il riconoscimento di X coincide con il determinare S_X tale che

$$S_X = \arg\max_S \{\Pr(S|X)\}$$

che, nel caso ciascuno degli modelli in M è equiprobabile, diventa

$$S_X = \arg\max_S \{\Pr(X|S)\}$$

Si osserva però che per ognuno dei modelli esiste una sequenza di stati di transizione Ω_i, da cui

$$\begin{aligned}\Pr(X|S) &= \sum_i \Pr(X, \Omega_i|S) \\ &= \sum_i \Pr(X|\Omega_i, S)\end{aligned}$$

che se posto per un qualche i_k al passo k

$$\begin{aligned}a(i_{k+1}) &= \Pr(x_1, \dots, x_{k+1}, i_{k+1}|S) \\ &= \sum_{i_k} a(i_k) \Pr(i_{k+1}|i_k) \Pr(x_{k+1}|i_{k+1})\end{aligned}$$

può essere reso computazionalmente più efficiente.

– 2.4 – Processi di decisione di Markov e reinforced learning.

Si parta dal considerare i diversi stili di apprendimento umano, questi risultano essere: *procedurali,* di *classificazione* e di *memorizzazione.* Nel caso del machine learning, i tipi di apprendimento di interesse risultano essere quelli procedurali e di classificazione.

Partendo dal considerare quest'ultimi, obbiettivo è quello di definire un modello che, basandosi sui dati di addestramento, sia capace di classificare correttamente dati nuovi. Considerando invece modelli di tipo procedurale, a partire da "ambienti" contenti "ricompense", definisco un modello che sia capace di prendere decisioni. In quest'ultimo caso si distinguono due scenari

- gli "ambienti" sono noti, allora si può utilizzare la programmazione dinamica per valutare l'iterazione e agire di conseguenza,
- gli "ambienti" sono ignoti, si utilizza il reinforced learning, agendo e osservando l'ambiente.

– 2.4.1 – Apprendimento rinforzato.

Tipologia particolare di apprendimento supervisionato in cui il carattere/comportamento del modello è determinato dei rinforzi. Nello specifico, ogni azione può cambiare lo stato del modello e, conseguentemente attivare un meccanismo di ricompense. Da ciò segue che il corretto modo di agire è dettato dall'esperienza acquisita. Considerando ora i diversi tipi di apprendimento rinforzato, questi si distinguono in

- apprendimento *basato su ricerca*;
- apprendimento *basato su modelli*;
- apprendimento *model-free*, con cui si determina cosa fare indipendentemente dal modello. Di questi si distinguono
  - l'*apprendimento actor-critic*;
  - il *Q-learning*.

– 2.4.2 – Processi decisionali di Markov.

Rappresentano la formulazione matematica di un problema di apprendimento rinforzato. Essi godono per tale ragione della proprietà di Markov (lo stato corrente caratterizza l'intero sistema). In generale è espresso dalla seguente quintupla

$$(S, A, R, P, \gamma)$$

con

- S insieme dei possibili stati;
- A insieme delle possibili azioni;
- R distribuzione delle ricompense, date azione e stato;
- P probabilità di transizione;
- γ fattore di sconto.

– 3 – Classificatori lineari.

I classificatori sinora descritti, risultano ottimi per quel che riguarda la minimizzazione dell'errore; presentano però un problema: sono strettamente dipendenti dalla distribuzione di probabilità che può non essere nota, o difficile da calcolare.

Si considera ora una tipologia di classificatori che, sebbene non ottimali[1], risultano meno restrittivi. Nello specifico tali classificatori, detti *lineari*, per ogni coppia di classi determinano un iper-piano tale da non causare ambiguita nella classificazione.

Definizione: sia $X = (x_1, \dots, x_k) \in \mathbb{R}^n$. Sia inoltre

$$g(X) = w_0 + w_1 x_1 + \cdots + w_k x_k$$

una combinazione lineare di X. Siano ω_i, ω_j le soli classi, per il classificatore lineare se $g(X) > 0$, allora X sarà classificato appartenente a ω_i, viceversa ad ω_j.

[1] Ci si riferisce alla possibilità in cui l'errore di classificazione può non essere minimizzato.

In particolare, il luogo dei punti tali che $g_{ij}(x) = 0$ è detto iper-piano di decisione tra le classi ω_i, ω_j.

Siano x_1, x_2 punti dell'iper-piano, posto $W^T = (w_1, w_2)^T$ segue

$$w_0 + w^T x_1 = w_0 + w^T x_2$$

da cui allora

$$w^T(x_1 - x_2) = 0, \qquad \forall x_1, x_2$$

cioè w^T è perpendicolare all'iper-piano.

– 3.1 – Algoritmo del percettrone.

Sia $X = (x_1, \dots, x_k)$ un vettore di features, resta il problema di come calcolare w. Tra le soluzioni quella a seguire. Siano ω_i, ω_j classi e sia supposto che le due siano linearmente separabili. Cioè

$$\exists w : x \in w_i \iff w^T x + w_0 > 0 \land x \in w_j \iff w^T x + w_0 < 0$$

da cui posti $w' = \begin{pmatrix} w & w_0 \end{pmatrix}$ e $X' = \begin{pmatrix} X & 1 \end{pmatrix}$, si ha

$$g(x) = w'X' = 0$$

Ultimo paso è quello di scegliere una *funzione di costo* e un algoritmo di minimizzazione della stessa.

– 3.1.1 – Funzioni di costo.

Sia $D = \{(X_1, C_1), \cdots, (X_k, C_k)\}$ dataset di addestramento, ove ogni (X_i, C_i) rappresenta la coppia vettore di addestramento e corretta classificazione dello stesso. Sia $E = \{X_i : C_i \neq \omega_i\}$, ossia l'insieme dei vettori non classificati correttamente. Posta

$$\delta(w) = \begin{cases} 1, \text{ se } w^T X > 0; \\ 0, \text{ se } w^T X < 0 \end{cases}$$

si definisce funzione di costo

$$J(w) = \sum_{X \in E} \delta(X) w^T X$$

Considerando il calcolo di w, questi è in generale effettuati utilizzando la *discesa del gradiente*. Per essa, si ha

$$w^{k+1} = w^k - \Delta w, k = 0, 1, \dots$$

con

$$\Delta w = \frac{\partial J(w)}{\partial w} = \cdots = \sum_{x \in E} \delta(w) X$$

Si dimostra che l'algoritmo converge dopo un numero finito di passi.

– 3.2 – Funzione logistica e regressore logistico.

Un regressore logistico è un regressore che tenta di minimizzare $\left| y - \sigma g(X) \right|$. Più in generale, una funzione logistica è una qualunque funzione del tipo

$$\sigma : \mathbb{R}^n \to [0, 1]$$

e tale che se $g(X) \to +\infty$, allora $\sigma \to 1^-$, viceversa se $g(X) \to -\infty$, si ha $\sigma \to 0^+$. Si può dunque pensare a σ come una funzione di probabilità. Segue che la più semplice rappresentazione è

$$\sigma = \frac{1}{1 - e^{-g(X)}}$$

Parlando del classificatore logistico, i dati di addestramento sono del tipo $(X, \Pr(y))$; in questo caso, il classificatore assegna X alla classe y se e solo se $\Pr(y) \geq 0.5$.

– 3.2.1 – Funzione di perdita logistica.

Sia assunto che $\Pr(y = 1 \big| X, w) = \sigma(w^T X)$ e che $\Pr(y = 0 \big| X, w) = 1 - \sigma(w^T X)$. Posto di avere N osservazioni, la funzione di perdita logistica $J(w)$ è definita come

$$J(w) = \sum_{i=1}^{N} y_i \ln\Big(\sigma(w^T X)\Big) + (1 - y_i) \ln\Big(1 - \sigma(w^T X)\Big)$$

che risulta minimizzata quando

$$w = \min_{w \in \mathbb{R}^n} \left\{ \sum_{i=1}^{N} y_i \ln\Big(\sigma(w^T X)\Big) + (1 - y_i) \ln\Big(1 - \sigma(w^T X)\Big) \right\}$$

– 3.3 – Support vector machine.

Siano ω_i, ω_j classi linearmente separabili; obiettivo di una SVM è quello di definire un iper-piano

$$g(x) = w^T X + w_0$$

tale che la distanza tra l'iper-piano e le due classi sia massima. Geometricamente, w rappresenta la direzione dell'iper-piano e w_0 la posizione iniziale. Come detto con SVM si massimizza la distanza tra l'iper-piano e un punto X, distanza definita come

$$d_X = \frac{g(x)}{\|w\|}$$

Inoltre, in generale, per semplificare i calcoli w, w_0 sono riscalati così che

$$g(X) = \begin{cases} 1, \text{ se } X \in \omega_i \\ -1, \text{ se } X \in \omega_j \end{cases}$$

Circa $J(w)$, si dimostra che questa risulta essere

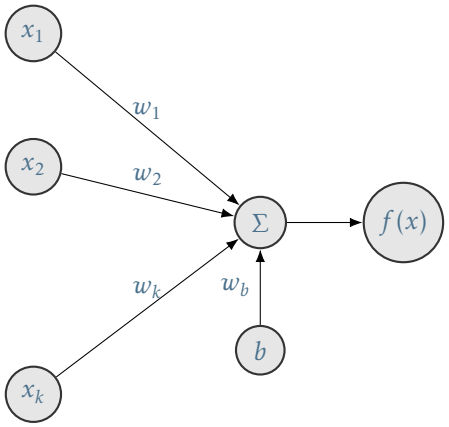
$$J(w) = \frac{\|w\|^2}{2}$$

che, essendo una quadratica, può essere minimizzata calcolando gli zeri della seguente lagrangiana

$$\mathcal{L}(w, w_0, \lambda) = \frac{1}{2} \sum_{i=1}^{N} \lambda_i [y_i (w^T x_i + w_0) - 1]$$

– 4 – Classificatori non lineari.

Nella precedente sezione si è discusso l'algoritmo del percettrone, alla base del quale vi sono i *percettroni*. Quest'ultimi, nel seguito rappresentati con la seguente struttura, identificano una qualsiasi struttura tale che, sfruttando l'algoritmo del percettrone, sia capace di apprendere.



Considerando il percettrone in se, questi è dipendente da una funzione di attivazione, che nel caso più semplice è definita come

$$f(X) = \begin{cases} 1, \text{ se } X \geq 0; \\ 0, \text{ altrimenti } . \end{cases} \tag{3}$$

In molti casi però la funzione di attivazione definita in (3), può risultare restrittiva: è in questi casi che si fa uso di una funzione di attivazione sigmoidale. Questa definita come

$$f(X) = \frac{1}{1 + \exp(-z)}$$

con

$$z = \sum_i (w_i x_i) - Bias$$

permette di definire un classificatore più flessibile.

– 4.1 – Reti neurali.

Vantaggio principale di utilizzare i percettroni è dovuto al fatto che, combinandone due e più, è possibile realizzare modelli più potenti: le *reti neurali*. Tra le diverse applicazioni vi sono

- la realizzazione di porte logiche;
- la creazione di modelli universali.

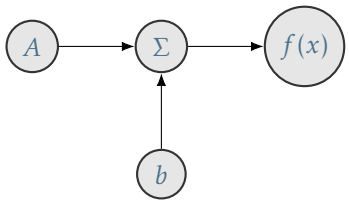
– 4.1.1 – Multi-layered-perceptron com porte logiche.

Come anticipato, le MLP possono essere impiegate per realizzare una qualsiasi porta logica.

Esempio: si supponga di dover realizzare una porta NOT, la cui tavola di verità si ricorda essere la seguente.

A	Y
0	1
1	0

Segue banalmente che una rappresentazione tramite MLP, è descritta dalla seguente struttura Qui assunto A l'input, resta da definire b bias e $f(x)$. Segue però da una semplice

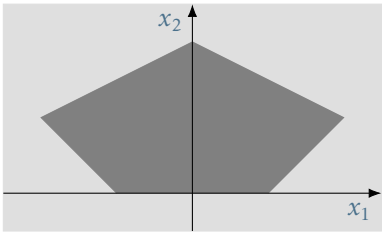


osservazione che, una soluzione valida è $b = 1, f(x) = -A + b$.

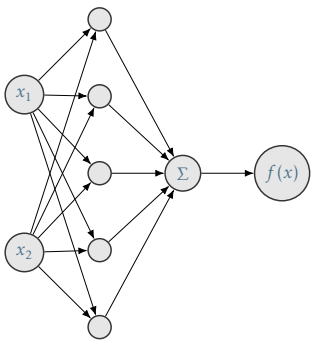
– 4.1.2 – MLP in spazi n-dimensionali.

Da quanto detto, i MLP permettono di effettuare riconoscimenti in spazi bi-dimensionali. Ci si chiede se è possibile utilizzarli in spazi n-dimensionali. Si dimostra che tale generalizzazione è possibile se e solo se si definisce opportunamente il MLP.

Esempio: Sia supposto di dover definire un MLP, tale che questi permetta di stabilire se un dato punti sia nella regione pentagonale nella figura a seguire. Osservando che quanto



richiesto può essere espresso come un funzione booleana a cinque variabili, questa può essere approssimata da un MLP a due livelli come nella figura a seguire.



– 4.2 – Retro propagazione.

Potrebbe capitare che il calcolo delle regioni decisionali non sia possibile, o che ciò risulti complesso; è dunque necessario un algoritmo che proceda a calcolarle automaticamente: una soluzione è l'algoritmo di retro propagazione. Tale algoritmo computa autonomamente, in maniera iterativa, i pesi di ogni sinapsi, cosicché la funzione di costo scelta sia minimizzata.
Osservazione. Poiché il calcolo dei pesi comporta il calcolo di derivate, la funzione di attivazione definita in (3) non può essere utilizzata, in quanto discontinua. Per ovviare a ciò, nel seguito si assume che la funzione di attivazione sia la funzione logistica.

Considerando l'algoritmo in se: dopo aver definito la funzione di costo da ottimizzare e il metodo di minimizzazione, l'algoritmo procede a ottimizzare w vettore di pesi. Assunto che il metodo scelto sia il gradient descent, si ha quanto segue

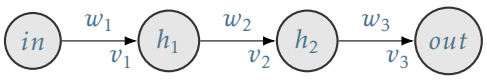
$$w^{(k+1)} = w^{(k)} - \Delta w^{(k)}$$

con

$$\Delta w^{(k)} = -\frac{\partial J(w)}{\partial w^{(k)}}$$

ove $J(w)$ è la funzione di costo.

Esempio: sia supposta la seguente rete



Sia f la funzione di attivazione e siano y_1, y_2 output di h_1, h_2 rispettivamente. Segue che $out = f(w_3 h_2), h_2 = f(w_2 h_1), h_1 = f(w_1 in)$. Cioè

$$out = f(w_3 f(w_2 f(w_1 in)))$$

Da ciò, passando alle derivate, si ha

$$\frac{\partial E}{\partial w_3} = \frac{\partial E}{\partial v_3}\frac{\partial v_3}{\partial w_3} = -(y - f(v_3))f(v_3)y_2$$
$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial v_2}\frac{\partial v_2}{\partial w_2} = \frac{\partial E}{\partial v_3}\frac{\partial v_3}{\partial v_2}\frac{\partial v_2}{\partial w_2}$$
$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial v_1}\frac{\partial v_1}{\partial w_1} = \frac{\partial E}{\partial v_2}\frac{\partial v_2}{\partial v_1}\frac{\partial v_1}{\partial w_1}$$

da cui calcolando il gradiente

$$E = (y - f(w_3 y_2))^2$$