DISPENSA DI INTELLIGENZA ARTIFICIALE

ANDREA CASINI & MAURO TEMPESTA

Appunti del corso di Intelligenza Artificiale

Laurea Specialistica in Informatica
Università Ca' Foscari di Venezia

5 settembre 2014 – versione 1.5

INDICE

ELENCO DELLE FIGURE

Parte I

RETI NEURALI

1

MODELLO NEURALE

Le reti neurali artificiali sono state create per riprodurre attività tipiche del cervello umano quali la percezione di immagini, il riconoscimento di forme, la comprensione del linguaggio e il coordinamento senso-motorio. A tale scopo sono state studiate le caratteristiche del cervello umano.

Nel sistema nervoso esistono miliardi di neuroni (cellule nervose). Nel modello biologico un **neurone** è caratterizzato da:

- **corpo cellulare (soma)**: l'unità di calcolo del neurone (5/10 micron);
- **assone**: meccanismo di output di un neurone;
- **dendriti**: ricevono segnali in input da altri assoni tramite le **sinapsi**.

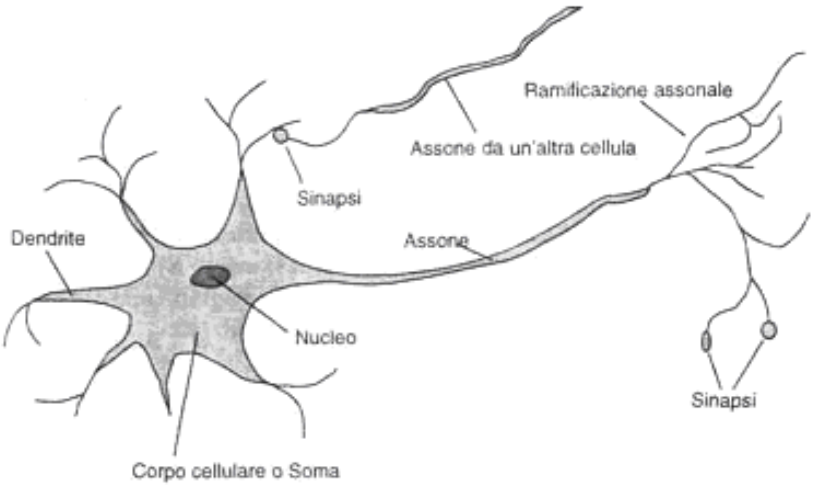


Figura 1: Un neurone biologico.

All'estremità l'assone si ramifica formando terminali attraverso i quali i segnali elettrici vengono trasmessi ad altre cellule (ad esempio ai dendriti di altri neuroni). Tra un terminale di un assone e la cellula ricevente esiste uno spazio: i segnali superano questo spazio per mezzo di sostanze chimiche dette *neurotrasmettitori*. Il punto di connessione tra terminale e dendrite è detto **sinapsi**.

La trasmissione di un segnale nella corteccia cerebrale è un processo complesso. In maniera molto semplificata, è composto delle seguenti fasi:

1. il corpo cellulare esegue una somma pesata dei segnali in ingresso;

2. se il risultato supera un certo valore soglia allora si produce un **potenziale d'azione**, ovvero una scarica di impulsi elettrici che viene inviata all'assone. In questo caso si dice che il neurone ha *sparato*, altrimenti è rimasto inattivo;
3. quando il segnale elettrico raggiunge la sinapsi, viene rilasciato chimicamente un *neurotrasmettitore* che si combinerà con i *recettori* nella membrana post-sinaptica;
4. i recettori post-sinaptici provocano una diffusione del segnale elettrico nel neurone post-sinaptico.

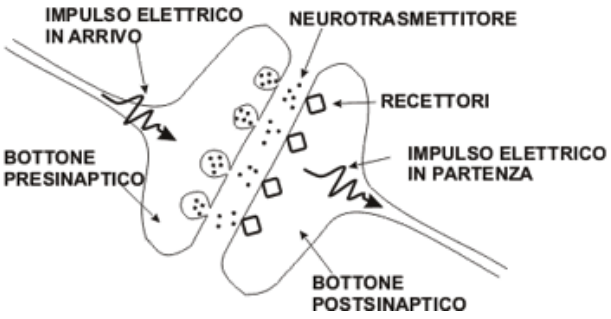


Figura 2: La sinapsi.

L'**apprendimento** si attua modificando la cosiddetta **efficacia sinaptica**, ovvero l'ammontare di corrente che entra nel neurone post-sinaptico rispetto al potenziale d'azione del neurone pre-sinaptico. In questo contesto le sinapsi possono essere **eccitatorie**, nel senso che favoriscono la generazione di potenziale d'azione nel neurone post-sinaptico, oppure **inibitorie**, se ne limitano la generazione.

Nel cervello non esiste un controllo centralizzato, nel senso che le varie zone del cervello funzionano insieme, influenzandosi reciprocamente e contribuendo alla realizzazione di uno specifico compito.

Infine, il cervello è *fault tolerant*, ovvero se un neurone o una delle sue connessioni sono danneggiati, il cervello continua a funzionare, anche se con prestazioni leggermente degradate. In particolare, le prestazioni del processo cerebrale degradano gradualmente man mano che si distruggono sempre più neuroni (*graceful degradation*).

Per riprodurre artificialmente il cervello umano occorre realizzare una rete di elementi molto semplici che sia una struttura **distribuita**, massicciamente **parallela**, capace di **apprendere** e quindi di **generalizzare** (cioè produrre uscite in corrispondenza di ingressi non incontrati durante l'addestramento).

Il metodo più usato per addestrare una rete neurale consiste nel presentare in ingresso alla rete un insieme di esempi (training set). La risposta fornita dalla rete per ogni esempio viene confrontata con la risposta desiderata, si valuta la

differenza (errore) fra le due e, in base ad essa, si aggiustano i pesi. Questo processo viene ripetuto sull'intero training set finché le uscite della rete producono un errore al di sotto di una soglia prestabilita.

Anche se di recente introduzione, le reti neurali trovano valida applicazione in settori quali predizione, classificazione, riconoscimento e controllo, portando spesso contributi significativi alla soluzione di problemi difficilmente trattabili con metodologie classiche.

1.1 MODELLO DI MCCULLOCH & PITTS

Nel 1943, W. McCulloch e W. Pitts hanno presentato un modello artificiale del neurone biologico, avente molti ingressi ed una sola uscita: ciascun ingresso ha associato un peso, che determina la conducibilità del canale di ingresso. L'attivazione del neurone è una funzione della somma pesata degli ingressi.

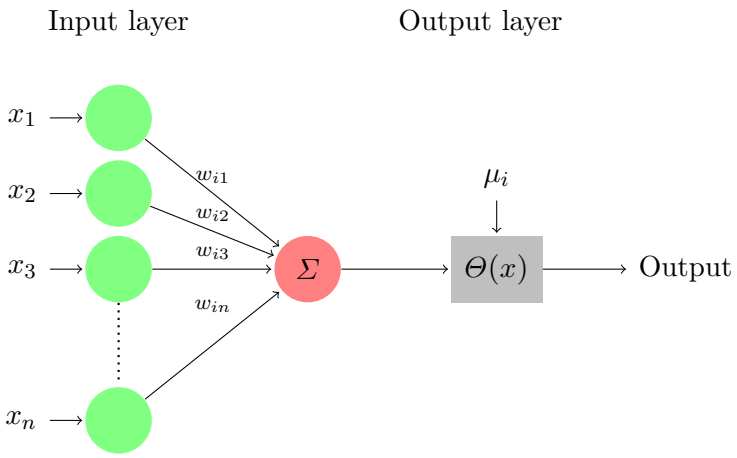


Figura 3: Neurone artificiale nel modello McCulloch & Pitts.

Un insieme di neuroni artificiali forma una **rete neurale**: essa può essere rappresentata mediante un grafo diretto pesato $N = (V, E, w)$ dove V è l'insieme delle unità (o neuroni), $E \subseteq V \times V$ è l'insieme di connessioni e $w : E \to \mathbb{R}$ è una funzione che assegna un peso con valore reale $w(i, j)$ per ogni connessione $(i, j) \in E$. I pesi rappresentano l'efficacia sinaptica: pesi positivi amplificano il segnale, mentre quelli negativi lo inibiscono. Come notazione sarà utilizzata w_{ji} anziché $w(i, j)$.

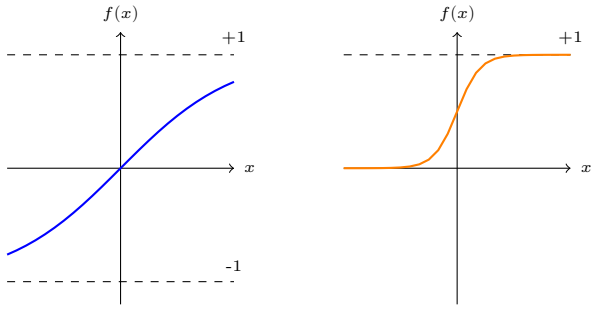
Un neurone si attiva se la somma pesata degli input supera un certo valore soglia μ_i

$$x_i = \Theta\left(\sum_j w_{ij} x_j - \mu_i\right)$$

dove $\Theta(x)$ è la **funzione di attivazione** e può essere *discreta*, come

$$x_i = \begin{cases} 1 & \text{se } \sum_j w_{ij} x_j \geq \mu_i \\ 0 & \text{altrimenti} \end{cases}$$

oppure *continua*, in modo tale da rendere il sistema più realistico: le funzioni continue più utilizzate sono la *sigmoidea* e la *tangente iperbolica*, in quanto permettono di ridurre la varianza in output di un neurone mappando il risultato della somma pesata entro un intervallo ridotto, $[0, 1]$ nel caso della prima e $[-1, 1]$ nel caso della seconda.



(a) Tangente iperbolica (b) Sigmoidea

Figura 4: Le funzioni di attivazione continue più utilizzate.

È possibile sbarazzarsi dei valori soglia associati ai neuroni aggiungendo una nuova unità extra permanentemente bloccata a -1. In questo modo i valori soglia diventano pesi e possono essere opportunamente aggiustati durante l'apprendimento.

Combinando opportunamente i neuroni di McCulloch & Pitts si possono costruire reti in grado di realizzare qualsiasi operazione del calcolo proposizionale. È importante notare che in questo modello **non** avviene ancora alcun tipo di apprendimento: i pesi sono fissi, mentre l'apprendimento si realizza attraverso la variazione dei pesi sinaptici.

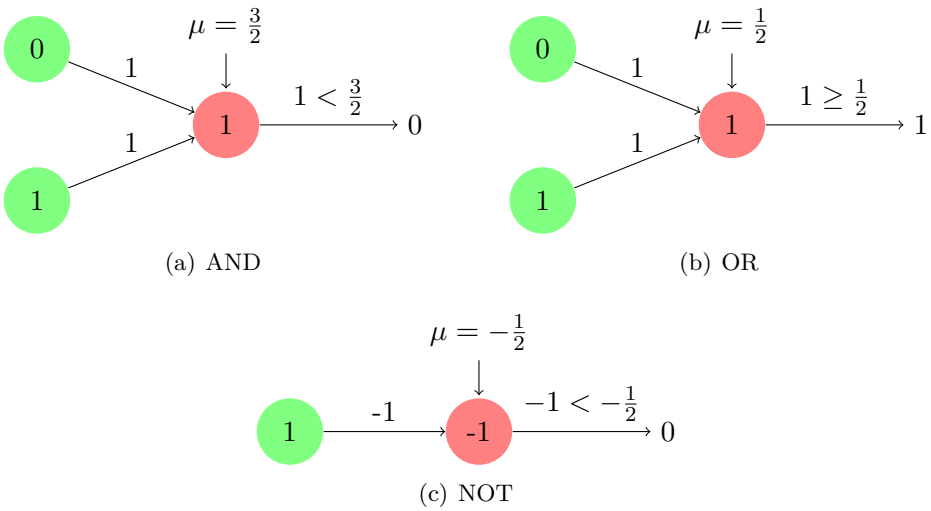


Figura 5: Operazioni logiche elementari nel modello M&P.

1.2 TIPI DI ARCHITETTURE DELLA RETE

In generale le reti neurali vengono classificate sulla base di tre caratteristiche:

- **presenza di cicli**: reti *feedforward* (o acicliche) e reti *ricorrenti* (o cicliche);
- **connettività**: reti fortemente o scarsamente connesse;
- **numero di strati**: reti *single layer* e reti *multilayer*.

La scelta della rete neurale da utilizzare dipende dalla tipologia di problema che si vuole risolvere. In generale si identificano tre classi di reti:

- **reti feedforward ad uno strato**: reti di questo tipo contengono i nodi di input (*input layer*) e uno strato di neuroni (*output layer*). Il segnale nella rete si propaga in avanti in modo aciclico, partendo dallo strato di input ed arrivando in quello di output;

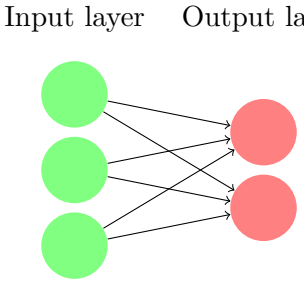


Figura 6: Rete feedforward ad uno strato.

- **reti feedforward a più strati**: in questo caso, tra lo strato di input e quello di output, sono presenti uno o più strati di neuroni nascosti (**hidden layers**). Questo tipo di architettura fornisce alla rete una prospettiva globale in quanto aumentano le interazioni tra neuroni;

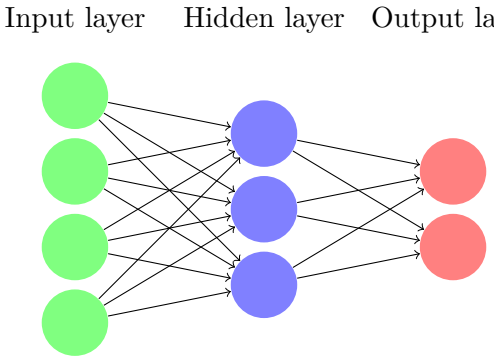


Figura 7: Rete feedforward a più strati.

- **reti ricorrenti**: reti di questo tipo contengono cicli, la cui presenza ha un impatto profondo sulle capacità di apprendimento della rete e sulle sue performance, in quanto la rendono un sistema **dinamico**.

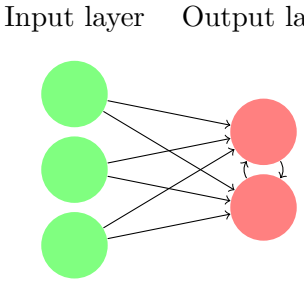


Figura 8: Rete ricorrente.

2

PROBLEMA DI CLASSIFICAZIONE

In un tipico problema di classificazione si hanno a disposizione un insieme di **caratteristiche** $f_1, f_2, \ldots, f_n$ e un insieme di **classi** $c_1, c_2, \ldots, c_m$: l'obiettivo consiste nel classificare degli *oggetti* in base alle loro caratteristiche. Un oggetto può essere rappresentato in uno spazio n-dimensionale e pertanto è possibile trattare problemi di classificazione attraverso metodi geometrici.

Lo spazio multidimensionale è suddiviso in **regioni di decisione**, ovvero aree all'interno delle quali ricadono tutti gli oggetti di una classe. I **confini** di queste regioni sono determinate dalla rete attraverso il processo di addestramento.

Gli oggetti da classificare sono comunemente chiamati **pattern**. Una rete neurale impara a riconosce dei pattern a seguito di una *fase di addestramento*, durante la quale alla rete vengono presentati un insieme di pattern di addestramento con l'indicazione della classe di appartenenza. Quando sarà presentato un nuovo pattern alla rete, appartenente ad una classeche questa ha precedentemente appreso, essa sarà in grado di classificarlo grazie alle informazioni estratte dai dati di addestramento.

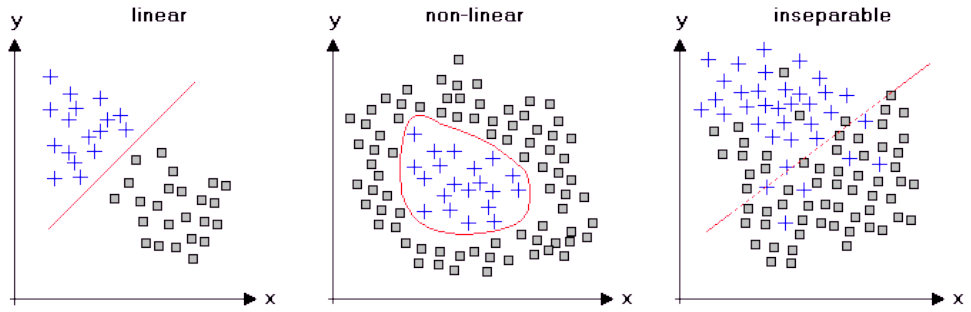


Figura 9: Rappresentazione geometrica di alcuni problemi di classificazione.

Quando si intende risolvere un problema di classificazione con una rete neurale, questa dovrà avere tante unità di input quante sono le caratteristiche degli oggetti e tanti neuroni di output quante sono le possibili classi.

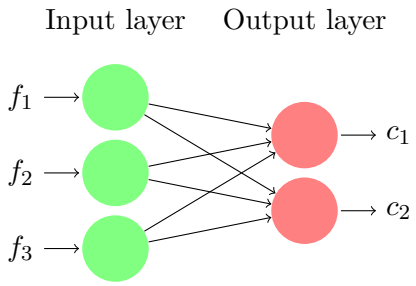


Figura 10: Rete feedforward ad uno strato usata per la classificazione.

2.1 PERCETTRONE A SINGOLO STRATO

La prima rete neurale utilizzata per risolvere problemi di classificazione è il **percettrone** di Rosenblatt (1958), un classificatore **binario** caratterizzato da un singolo strato con connessioni feedforward.

Il percettrone è un idea semplice ed elegante: imita il neurone umano ed è in grado di imparare da esempi che gli vengono presentati. Tuttavia, come dimostrato da M. Minsky e S. Papert (1969), ha una pesante limitazione: è in grado di risolvere solo problemi **linearmente separabili**, ovvero problemi in cui le regioni di decisione si possono separare con un iperpiano.

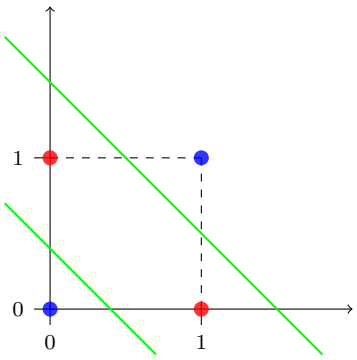


Figura 11: Non separabilità lineare dell'operatore XOR.

La convergenza del percettrone è garantita dal seguente teorema:

Teorema 1 (Teorema di convergenza del percettrone - N. J. Nilsson, 1965)**.** *Se il problema di classificazione è linearmente separabile allora la fase di apprendimento converge ad un'appropriata impostazione dei pesi in un numero finito di passi.*

Nella pratica non è dato sapere se un problema sia linearmente separabile o meno. La convergenza si può comunque ottenere aggiustando i parametri del percettrone come il numero di iterazioni o il fattore di apprendimento: in questo caso, tuttavia, la convergenza è artificiale.

2.2 PERCETTRONE A PIÙ STRATI

È possibile sopperire alle limitazioni dei percettroni a singolo strato aggiungendo strati di neuroni nascosti: il numero di strati, infatti, influenza la forma generale che possono assumere le regioni di decisione. Come si può osservare in figura 12:

- nelle reti a singolo strato si possono avere regioni di decisione separabili mediante un **iperpiano**;
- nelle reti a due strati si possono avere **regioni convesse**, aperte o chiuse;
- nelle reti a tre strati le regioni possono avere forma arbitraria, la cui complessità dipende dal numero di neuroni nascosti.

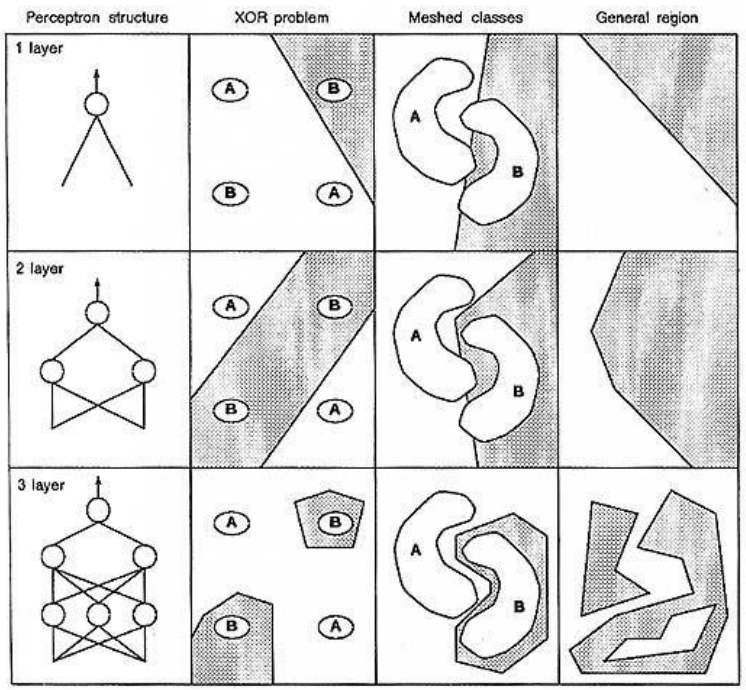


Figura 12: Confronto tra architetture di rete.

Le potenzialità dei percettroni multistrato sono note da tempo, ma gli algoritmi per l'apprendimento sono stati ideati solo recentemente (1986).

2.3 PREVISIONE DI SERIE TEMPORALI

Le reti neurali, oltre alla classificazione e altri problemi *discreti*, sono in grado di affrontare problemi *continui* come la **previsione di una serie temporale**.

Si consideri una sequenza P di vettori che dipendono dal tempo t:

$$P = \{\boldsymbol{x}(t_0), \boldsymbol{x}(t_1), \cdots, \boldsymbol{x}(t_{i-1}), \boldsymbol{x}(t_i), \boldsymbol{x}(t_{i+1}), \cdots\}$$

Per ottenere una serie $\{\boldsymbol{x}[t]\}$ da un segnale continuo $\boldsymbol{x}(t)$ è necessario **campionare** il segnale in punti discreti.

Partendo da un tempo t e procedendo a ritroso, otteniamo la serie temporale $\{\boldsymbol{x}[t], \boldsymbol{x}[t-1], \dots\}$. Lo scopo dell'analisi è di stimare $x(t)$ a un tempo futuro

$$\hat{\boldsymbol{x}}[t+s] = f(\boldsymbol{x}[t], \boldsymbol{x}[t-1], \dots)$$

dove s è noto come **orizzonte della previsione**. Si tratta di un problema di approssimazione di una funzione che può essere risolto utilizzando una rete neurale a due strati.

Teorema 2 (Teorema dell'approssimazione universale - G. Cybenko, 1989 & K. Hornik, 1991)**.** *Una rete neurale multistrato feed-forward con un singolo strato nascosto, un percettrone, che contiene un numero finito di neuroni nascosti è un approssimatore universale di funzioni continue su sottoinsiemi compatti* $\mathbb{R}^n$.

3

ALGORITMO DI BACKPROPAGATION

Negli anni settanta c'è stato un generale disinteresse nei confronti delle reti neurali, a causa della scoperta delle limitazioni dei percettroni a singolo strato e dell'incapacità di addestrare le più potenti reti multistrato.

Il problema incontrato era il seguente: volendo adottare un meccanismo di aggiornamento dei pesi simile a quello nelle reti single layer, in cui l'errore è calcolato come differenza tra l'uscita desiderata e l'uscita effettiva di ciascun neurone, si era in grado di aggiornare solo i pesi relativi ai neuroni di uscita; infatti, mentre per lo strato di uscita si conosce l'output desiderato, che viene dato come secondo elemento delle coppie che costituiscono il training set, non si sa nulla a riguardo dell'uscita desiderata per i neuroni nascosti.

Il problema è stato risolto da D. E. Rumelhart, G. E. Hinton e R. J. Williams, che nel 1986 hanno introdotto l'**algoritmo di backpropagation** per l'addestramento di reti neurali **feedforward multistrato**. L'algoritmo prevede di calcolare l'errore commesso da un neurone dell'ultimo strato nascosto propagando all'indietro l'errore calcolato sui neuroni di uscita collegati ad esso; lo stesso procedimento è poi ripetuto per tutti i neuroni del penultimo strato nascosto e così via.

Requisito fondamentale per l'utilizzo dell'algoritmo è la **derivabilità** della funzione di attivazione dei neuroni, in quanto l'algoritmo è basato sul metodo della **discesa del gradiente** per la minimizzazione dell'**errore** commesso dalla rete.

3.1 METODO DI DISCESA DEL GRADIENTE

Il **gradiente** di una funzione f, indicato con ∇f, è il vettore che ha come componenti le derivate parziali della funzione.

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_n} \right)$$

La discesa del gradiente è una tecnica di ottimizzazione di tipo **locale**: consiste nel valutare, inizialmente in un punto scelto in maniera casuale, sia la funzione stessa sia il suo gradiente.

Dal momento che quest'ultimo indica la direzione di massima crescita della funzione, si considerano i passi proporzionali al **negativo** del gradiente della

funzione nel punto corrente e la si valuta nel nuovo punto. La procedura viene arrestata quando, in seguito ad uno spostamento, il valore della funzione obiettivo aumenta anziché diminuire.[1]

3.2 APPRENDIMENTO SUPERVISIONATO

L'algoritmo di backpropagation è un algoritmo di **apprendimento supervisionato**, ovvero richiede un **training set** L del tipo

$$L = \{(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^P, \boldsymbol{y}^P)\}$$

dove

- $\boldsymbol{x}^\mu$ $(\mu = 1, \ldots, P)$ è il vettore di input;
- $\boldsymbol{y}^\mu$ $(\mu = 1, \ldots, P)$ è il vettore dell'output desiderato.

I pesi vengono tipicamente inizializzati in maniera casuale all'inizio dell'addestramento, poi si presentano, uno alla volta, gli esempi del training set: per ciascuno si calcola l'errore commesso dalla rete, ovvero la differenza tra l'uscita desiderata e l'uscita effettiva della rete, che viene utilizzato per aggiustare i pesi.

Il processo viene ripetuto ripresentando alla rete, in ordine casuale, tutti gli esempi del training set finchè l'errore medio risulta inferiore ad una soglia prestabilita.

Formalmente la fase di apprendimento consiste nel trovare una configurazione di pesi tale da minimizzare la **funzione di errore**

$$\begin{aligned} E(\boldsymbol{w}) &= \frac{1}{2} \sum_{\mu=1}^{P} \|\boldsymbol{y}^\mu - \boldsymbol{out}^\mu(\boldsymbol{w})\|^2 \\ &= \frac{1}{2} \sum_{\mu=1}^{P} \sum_{k=1}^{n} (y_k^\mu - out_k^\mu(\boldsymbol{w}))^2 \end{aligned}$$

dove $\boldsymbol{out}^\mu$ è l'output fornito dalla rete quando $\boldsymbol{x}^\mu$ è dato in input.

Al termine della fase di addestramento, la rete viene testata controllandone il comportamento su un insieme di dati, detto **test set**, costituito da esempi non utilizzati durante la fase di training. La fase di test ha quindi lo scopo di valutare la capacità di **generalizzazione** della rete neurale: si dice che la rete ha imparato se è in grado di fornire risposte anche per ingressi che non le sono mai stati presentati durante la fase di addestramento.

Ovviamente le prestazioni di una rete neurale dipendono fortemente dall'insieme di esempi scelti per l'addestramento: essi devono essere rappresentativi della

1 Quando si è interessati alla ricerca di un punto di massimo si prendono passi proporzionali al positivo del gradiente: in tal caso la procedura è nota come **ascesa del gradiente**.

realtà che la rete deve apprendere e in cui verrà utilizzata. L'apprendimento è infatti un processo ad hoc dipendente dallo specifico problema trattato.

3.3 PROPAGAZIONE DELL'ERRORE

L'algoritmo di backpropagation può essere diviso in due passi:

- **forward pass**: l'input dato alla rete è propagato in avanti, strato dopo strato, fino al livello di uscita, dove viene calcolato l'errore commesso $E(\boldsymbol{w})$;
- **backward pass**: l'errore è propagato all'indietro e i pesi sono aggiornati in maniera appropriata.

Consideriamo la rete neurale in figura 13 per derivare le formule di aggiornamento dei pesi: l'algoritmo può chiaramente essere utilizzato su reti di dimensione arbitraria.

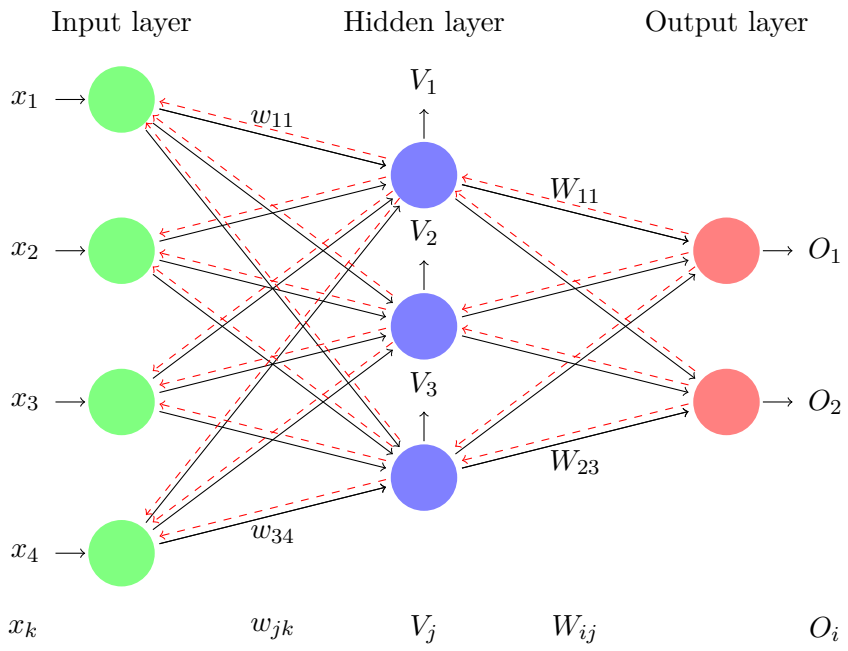


Figura 13: Schema back-propagation: le linee nere indicano il segnale propagato in avanti, mentre quelle rosse indicano l'errore propagato all'indietro.

Dato un pattern μ, l'unità nascosta j riceve un input netto dato da

$$h_j^\mu = \sum_k w_{jk} x_k^\mu$$

e produce come output

$$V_j^\mu = g(h_j^\mu) = g\left(\sum_k w_{jk} x_k^\mu\right)$$

dove g è la funzione di attivazione. In modo del tutto analogo si possono definire input e output dei neuroni dello strato di uscita.

I pesi sono aggiornati nella direzione opposta rispetto al gradiente, in quanto siamo interessati a minimizzare la funzione di errore, in accordo alla seguente formula (dove t è il numero di iterazioni):

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla(E(\boldsymbol{w}^{(t)}))$$

Il parametro η, noto come **fattore di apprendimento**, ha una forte influenza sul comportamento dell'algoritmo (velocità di convergenza, oscillazioni, ecc.).

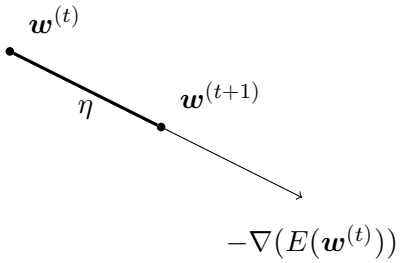


Figura 14: Direzione di aggiornamento dei pesi.

3.3.1 *Aggiornamento pesi strato nascosto-output*

Deriviamo ora la formula per il calcolo della variazione di peso sinaptico tra il neurone j dello strato nascosto e il neurone i dello strato di output.

$$\begin{aligned}\Delta W_{ij} &= -\eta \frac{\partial E}{\partial W_{ij}} \\ &= \eta \sum_\mu \sum_k (y_k^\mu - O_k^\mu) \frac{\partial O_k^\mu}{\partial W_{ij}}\end{aligned}$$

Per $k \neq i$, O_k non dipende da W_{ij}, quindi la derivata è 0.

$$\begin{aligned}&= \eta \sum_\mu (y_i^\mu - O_i^\mu) \frac{\partial O_i^\mu}{\partial W_{ij}} \\ &= \eta \sum_\mu (y_i^\mu - O_i^\mu) g'(h_i^\mu) \frac{\partial h_i^\mu}{\partial W_{ij}}\end{aligned}$$

Calcoliamo ora $\partial h_i^\mu / \partial W_{ij}$:

$$\begin{aligned}\frac{\partial h_i^\mu}{\partial W_{ij}} &= \frac{\partial}{\partial W_{ij}} \sum_l V_l^\mu W_{il} \\ &= \frac{\partial}{\partial W_{ij}} V_j^\mu W_{ij} = V_j^\mu\end{aligned}$$

Quindi si ha

$$\Delta W_{ij} = \eta \sum_\mu (y_i^\mu - O_i^\mu) g'(h_i^\mu) V_j^\mu = \eta \sum_\mu \delta_i^\mu V_j^\mu$$

dove

$$\delta_i^\mu = (y_i^\mu - O_i^\mu) g'(h_i^\mu)$$

è l'errore del neurone i-esimo ed è noto come **gradiente locale**.

3.3.2 *Aggiornamento pesi strato input-nascosto*

Passiamo ora al calcolo della variazione di peso sinaptico tra il neurone nascosto j e l'unità di input k.

$$\begin{aligned}\Delta w_{jk} &= -\eta \frac{\partial E}{\partial w_{jk}} \\ &= \eta \sum_\mu \sum_i (y_i^\mu - O_i^\mu) \frac{\partial O_i^\mu}{\partial w_{jk}} \\ &= \eta \sum_\mu \sum_i (y_i^\mu - O_i^\mu) g'(h_i^\mu) \frac{\partial h_i^\mu}{\partial w_{jk}}\end{aligned}$$

In questo caso la seconda sommatoria rimane, in quanto l'output di ogni neurone nello strato di uscita dipende indirettamente da w_{jk}.

Calcoliamo ora $\partial h_i^\mu / \partial w_{jk}$:

$$\frac{\partial h_i^\mu}{\partial w_{jk}} = \sum_l W_{il} \frac{\partial V_l^\mu}{\partial w_{jk}}$$

Per $l \neq j$, V_l non dipende da w_{jk}.

$$\begin{aligned}&= W_{ij} \frac{\partial V_j^\mu}{\partial w_{jk}} \\ &= W_{ij} g'(h_j^\mu) \frac{\partial h_j^\mu}{\partial w_{jk}}\end{aligned}$$

La derivata $\partial h_j^\mu / \partial w_{jk}$ è:

$$\begin{aligned}\frac{\partial h_j^\mu}{\partial w_{jk}} &= \frac{\partial}{\partial w_{jk}} \sum_m w_{jm} x_m^\mu \\ &= \frac{\partial}{\partial w_{jk}} w_{jk} x_k^\mu \\ &= x_k^\mu\end{aligned}$$

Riassumendo abbiamo

$$\begin{aligned}\Delta w_{jk} &= \eta \sum_\mu \sum_i (y_i^\mu - O_i^\mu) g'(h_i^\mu) W_{ij} g'(h_j^\mu) x_k^\mu \\ &= \eta \sum_\mu \sum_i \delta_i^\mu W_{ij} g'(h_j^\mu) x_k^\mu \\ &= \eta \sum_\mu \delta_j^\mu x_k^\mu\end{aligned}$$

dove

$$\delta_j^\mu = g'(h_j^\mu) \sum_i \delta_i^\mu W_{ij}$$

è il gradiente locale del neurone nascosto j e la sommatoria rappresenta l'errore medio sullo strato di output causato dal neurone j dello strato nascosto.

3.3.3 *In sintesi*

In generale, per l'aggiornamento dei pesi non sono necessarie informazioni globali, ma solo **locali**. La variazione di peso sinaptico tra un neurone p e q, in accordo alle formule derivate nelle sezioni precedenti, è dato da

$$\Delta w_{pq} = \eta \sum_\mu \delta_p^\mu V_q^\mu$$

dove

$$\delta_p^\mu = \begin{cases} (y_p^\mu - O_p^\mu) g'(h_p^\mu) & \text{se } p \text{ è un neurone di output} \\ g'(h_p^\mu) \sum_i \delta_i^\mu W_{ip} & \text{altrimenti} \end{cases}$$

Questo modello prende il nome di apprendimento **offline**, in quanto calcola la variazione di peso usando contemporaneamente tutti gli esempi dell'insieme di addestramento.

Un modello più coerente con quello biologico è detto apprendimento **online**, dove la rete apprende un esempio per volta:

$$\Delta w_{pq} = \eta \delta_p^\mu V_q^\mu$$

3.4 L'ALGORITMO DI BACKPROPAGATION

Consideriamo una rete con M strati ed indichiamo

- V_i^m: output della i-esima unità del livello m;
- w_{ij}^m: peso della connessione tra il j-esimo neurone dello strato $m-1$ e l'i-esimo neurone dello strato m.

con $m = 0, \ldots, M$. L'algoritmo è il seguente:

① Inizializzazione dei pesi a (piccoli) valori casuali;

② Scelta di un pattern $\boldsymbol{x}^\mu$ da inserire allo strato input:

$$V_k^0 = x_k^\mu \qquad \forall k$$

③ Propagazione del segnale in avanti ($m = 1, \ldots, M$):

$$V_i^m = g(h_i^m) = g\left(\sum_j w_{ij}^m V_j^{m-1}\right)$$

④ Calcolo degli errori δ dello strato di output:

$$\delta_i^M = (y_i - V_i^M)\, g'(h_i^M)$$

⑤ Calcolo degli errori δ per gli strati precedenti ($m = M, \ldots, 2$):

$$\delta_i^{m-1} = g'\left(h_i^{m-1}\right) \sum_j \delta_j^m\, w_{ji}^m$$

⑥ Aggiornamento dei pesi ($m = 1, \ldots, M$):

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij} \qquad \text{dove } \Delta w_{ij} = \eta \delta_i^m V_j^{m-1}$$

⑦ Torna al passo 2 fino alla convergenza.

L'algoritmo si può terminare quando $\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\| < \epsilon$ oppure dopo un certo numero di epoche[2] se la condizione precedente non è rispettata.

2 Un'**epoca** è un ciclo di presentazione di tutti gli elementi del training set.

3.5 IL MOMENTO

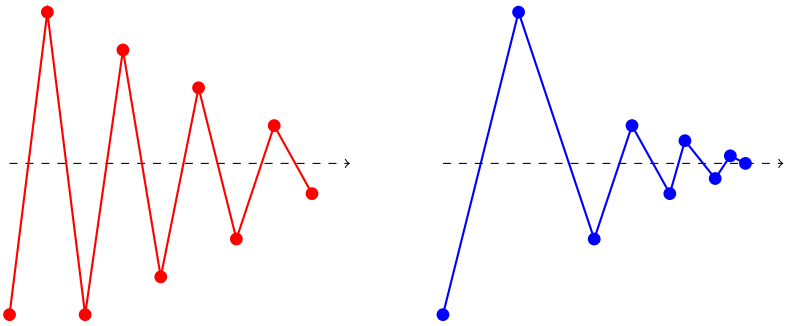
La scelta del fattore di apprendimento η influenza pesantemente il comportamento dell'algoritmo: se scegliamo valori troppo piccoli, la convergenza sarà lenta, se scegliamo valori troppo grandi si rischia di avere un comportamento oscillatorio.

Un metodo semplice per incrementare η senza il rischio di rendere l'algoritmo instabile consiste nel modificare la regola di aggiornamento inserendo un nuovo termine, il **momento**, proporzionale alla precedente variazione dei pesi. La regola di aggiornamento diventa

$$\Delta w_{pq}^{(t+1)} = \alpha \Delta w_{pq}^{(t)} - \eta \frac{\partial E}{\partial w_{pq}}$$

dove α è un numero positivo a scelta preferibilmente compreso in $[0, 1)$. In generale si utilizzano $\alpha = 0.9$ e $\eta = 0.5$.

L'utilizzo del momento ha l'effetto di incrementare o decrementare l'ampiezza di aggiornamento in modo tale da **accellerare** nelle discese o **stabilizzare** l'algoritmo in caso di oscillazioni. Il suo utilizzo, inoltre, ha il vantaggio di **prevenire** (non con certezza) che il processo di apprendimento incappi in minimi locali della funzione d'errore.



(a) Assenza di momento $\alpha = 0$ (b) Con momento $\alpha = 0.5$

Figura 15: Momento e discesa del gradiente su una semplice superficie.

3.6 PROBLEMA DEL MINIMO LOCALE

L'algoritmo di backpropagation non è sempre in grado di trovare il **minimo globale**. Il problema risiede nell'esistenza di *buoni* e *cattivi* punti di minimo.

Per evitare il problema è importante scegliere una configurazione iniziale adeguata dei pesi: se essi sono troppo elevati la derivata della funzione g sarà vicina a zero e di conseguenza la variazione dei pesi sarà piccola, aumentando così il

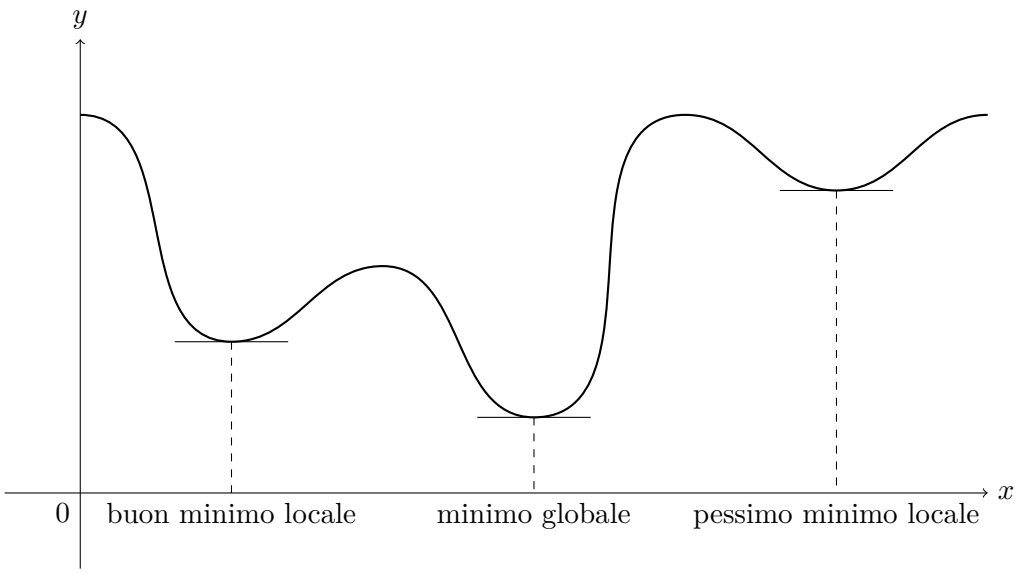


Figura 16: Rappresentazione del problema del minimo locale.

rischio di incappare in un minimo locale. Nella pratica si utilizza la seguente euristica per impostare i pesi iniziali:

$$w_{ij} = \frac{1}{\sqrt{k_i}}$$

dove k_i è il numero di unità entranti nell'unità i.

4

COSTRUZIONE DI UNA RETE E APPRENDIMENTO

Quando si intende utilizzare una rete neurale per risolvere un particolare problema, ci sono diverse questioni da affrontare.

- Da quanti strati dev'essere composta la rete?
- Quante unità sono necessarie per ogni strato?

Per rispondere a queste domande si introduce il concetto di **generalizzazione**.

4.1 GENERALIZZAZIONE

Definizione 1 (Generalizzazione)**.** *Con* **generalizzazione** *si intende la capacità di una rete di fornire risposte corrette ad esempi non incontrati durante la fase di addestramento (esempi di test).*

Si assume che l'insieme di esempi di test (*test set*) sia tratto dalla stessa popolazione usata per generare il training set.

Il problema di apprendimento può essere visto come un problema di *approssimazione di una curva*: la rete è considerata semplicemente come una mappa input/output non lineare, dunque una buona generalizzazione è vista come una buona interpolazione dei dati di input.

Una rete progettata per generalizzare bene produce mappature input/output corrette anche se l'input è lievemente differente dagli esempi usati in fase di training: se però la rete viene addestrata con troppi esempi, si rischia di memorizzare il training set, ovvero la rete apprende il **rumore** presente nel training set. Questo fenomeno è detto **overfitting**: una rete *sovra-addestrata* è troppo rigida e pertanto perde la capacità di generalizzazione.

In generale una rete troppo piccola impara poco, mentre una rete troppo grande impara molto, ma non generalizza abbastanza. In entrambi i casi si rischia di non riuscire a risolvere il problema: è necessario trovare un *giusto compromesso*.

La capacità di generalizzazione è influenzata da **tre fattori** principali: le dimensioni del training set, l'architettura della rete neurale e la complessità del problema.

Dal momento che non si ha alcun controllo sulla complessità del problema, è possibile affrontare il problema della generalizzazione sotto due punti di vista differenti:

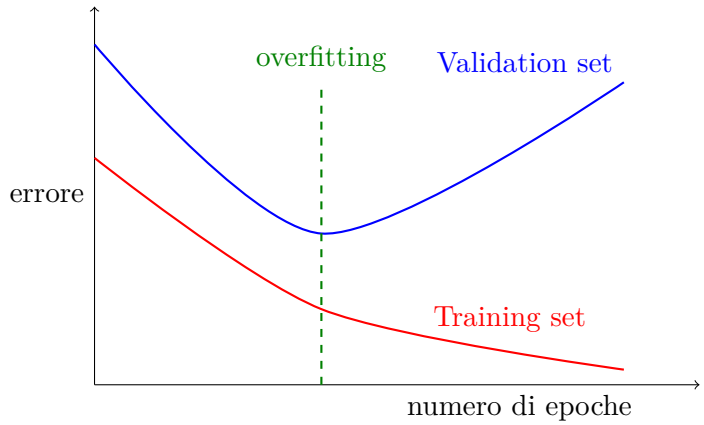


Figura 17: La curva rossa mostra l'andamento dell'errore nel classificare i dati di training, mentre la curva blu mostra l'errore nel classificare i dati di test o validazione. Se l'errore di validazione aumenta mentre l'errore sui dati di training diminuisce, ciò indica che siamo in presenza di un possibile caso di overfitting.

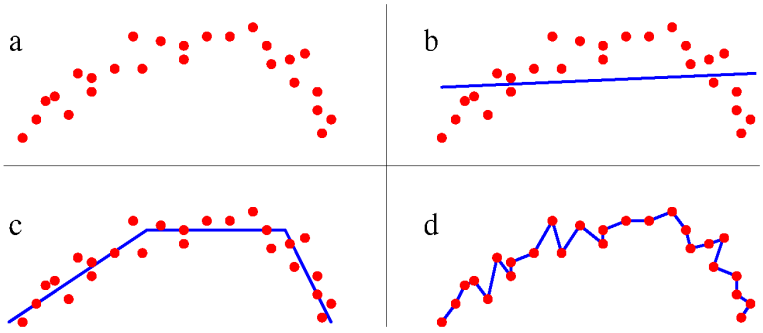


Figura 18: (a) dati del training set, (b) sotto approssimazione, (c) una buona stima sui dati, (d) overfitting: la curva di apprendimento è perfettamente disposta sul training set.

1. l'architettura della rete è prefissata e lo scopo è determinare una dimensione del training set ottimale per una buona generalizzazione;
2. la dimensione del training set è prefissata e lo scopo è di determinare la migliore architettura di rete per una buona generalizzazione.

4.2 CROSS-VALIDATION

La **cross-validation** è una tecnica utilizzata per selezionare, tra un insieme di modelli utilizzabili per risolvere un problema, il modello migliore secondo certi criteri.

Durante un **round** di cross-validation si partiziona l'insieme di addestramento in due sottoinsiemi complementari:

- **estimation subset**, usato per l'addestramento del modello;
- **validation subset**, usato per la validazione del modello.

La procedura viene iterata per diversi partizionamenti, al fine di ridurre la varianza dei risultati, e si calcola la media delle performance ottenute durante i diversi round.

Si distinguono solitamente due tipi di cross-validation:

- **cross-validation esaustiva**, quando si considerano tutti i modi possibili per il partizionamento del training set (**leave-p-out** cross-validation);
- **cross-validation non esaustiva**, quando non si considerano tutti i possibili partizionamenti (**k-fold** cross-validation).

La tipologia di cross-validation più utilizzata è la k-fold cross-validation. In questo tipo di validazione, il training set è suddiviso in k sottoinsiemi e si itera k volte la seguente procedura: addestramento della rete su $k-1$ sottoinsiemi e validazione sul rimanente. Ad ogni round si utilizza un diverso sottoinsieme come validation set.

Una volta selezionato il modello da utilizzare si ricorre al test set per verificare la capacità di generalizzazione della rete.

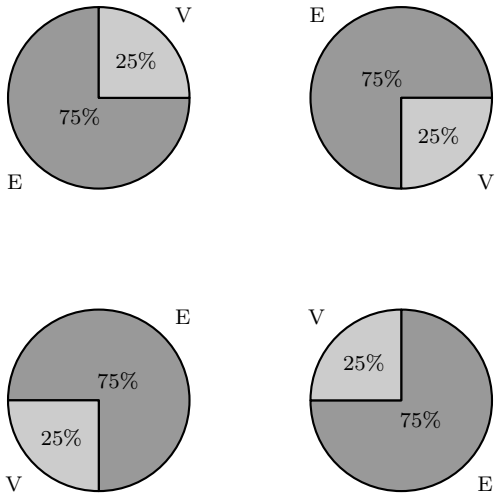


Figura 19: Esempio di 4-fold cross-validation: partizionamenti del training set in estimation subset E e validation subset V.

4.3 METODO DI TRAINING EARLY-STOPPING

Con l'obiettivo di una buona generalizzazione è molto difficile decidere quando è il momento di bloccare il training: c'è il rischio di overfitting dei dati se non si ferma l'addestramento al punto giusto.

Il processo di addestramento **early-stopping** è il seguente:

- dopo un periodo di addestramento sull'estimation set, si calcola l'errore di validazione per ogni esempio del validation set;
- quando la fase di validazione è completa, si riprende la fase di addestramento per un altro periodo.

Se si osserva la sola curva dell'errore dell'estimation set (figura 20), questo si riduce all'aumentare delle epoche ed è spontaneo pensare di proseguire l'addestramento anche oltre il punto di minimo della curva del validation set: in realtà ciò che la rete apprende dopo quel punto è il rumore contenuto nei dati del training set. L'euristica suggerisce quindi di fermare l'addestramento in corrispondenza del minimo della curva relativa al validation set.

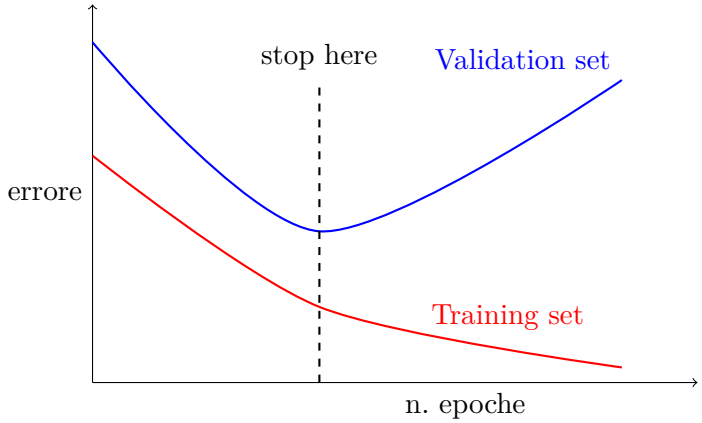


Figura 20: Andamento di errore e metodo di early stopping.

4.4 TECNICHE DI PRUNING

Le capacità funzionali e di generalizzazione di una rete sono fortemente influenzate dalla sua dimensione, ovvero il numero di neuroni nascosti. Con una rete troppo piccola si rischia di non riuscire a risolvere il problema, mentre con una troppo grande si rischia di apprendere il rumore deteriorando la capacità di generalizzare.

Per scegliere la dimensione corretta della rete si può procedere in due modi:

1. **pruning**: si parte da una rete sovradimensionata per poi ridurla eliminando connessioni sinaptiche o neuroni interi;
2. **growing**: si parte da una rete piccola per poi espanderla.

Il primo approccio è tipicamente il più adottato: nonostante richieda maggior tempo computazionale, in quanto ci sono più unità da addestrare, è più veloce in

termini di numero di epoche (convergenza più veloce) ed è del tutto indipendente dall'algoritmo di addestramento utilizzato.

Consideriamo per semplicità una rete con un singolo strato nascosto.

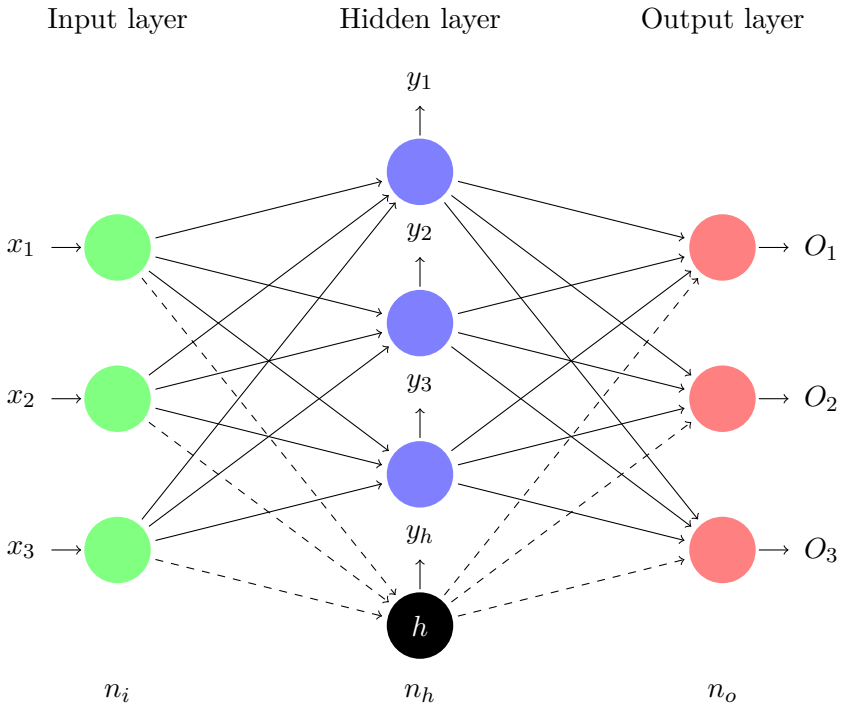


Figura 21: Rimozione del neurone h in una rete neurale.

L'approccio del pruning consiste nel rimuovere un neurone h e successivamente aggiustare i **pesi entranti** nelle unità servite da h in modo tale da preservare il comportamento di input/output dell'intera rete. Questo è equivalente a risolvere il sistema

$$\underbrace{\sum_{j=1}^{n_h} w_{ij} y_j^\mu}_{\text{prima}} = \underbrace{\sum_{\substack{j=1 \\ j\neq h}}^{n_h} \left(w_{ij} + \delta_{ij}\right) y_j^\mu}_{\text{dopo}} \qquad \forall i = 1, \ldots, n_o, \quad \forall \mu = 1, \ldots, P$$

dove δ_{ij} sono i fattori di aggiustamento da scoprire e y_j^μ è l'output dell'unità j quando viene dato in input il pattern μ.

L'equazione precedente può essere riscritta come

$$\sum_{j=1}^{n_h} w_{ij} y_j^\mu = \sum_{\substack{j=1 \\ j\neq h}}^{n_h} w_{ij} y_j^\mu + \sum_{\substack{j=1 \\ j\neq h}}^{n_h} \delta_{ij} y_j^\mu$$

da cui si ottiene

$$\sum_{\substack{j=1 \\ j \neq h}}^{n_h} \delta_{ij} y_j^\mu = w_{ih} y_h^\mu$$

che è un sistema lineare di equazioni con incognite $\{\delta_{ij}\}$.

È possibile scrivere il sistema in forma matriciale. Sia $N = (V, E, w)$ il grafo pesato che rappresenta la rete neurale da ridurre e siano P_i e R_i definiti come segue:

$$P_i = \{j : (i, j) \in E\}$$
$$R_i = \{j : (j, i) \in E\}$$

Per ogni $i \in P_h$ definiamo la matrice $\boldsymbol{Y}_{i,h} = [\boldsymbol{y}_{j_1} \dots \boldsymbol{y}_{j_{r_i-1}}]$, dove ogni colonna è l'output dell'unità $j_k \in R_i \setminus \{h\}$. Dobbiamo risolvere $|P_h|$ sistemi del tipo

$$\boldsymbol{Y}_{i,h} \boldsymbol{\delta}_i = \underbrace{w_{ih} \boldsymbol{y}_h}_{\boldsymbol{z}_{i,h}}$$

che possono essere combinati in un unico sistema

$$\boldsymbol{Y}_h \boldsymbol{\delta} = \boldsymbol{z}_h$$

dove

$$\boldsymbol{Y}_h = \operatorname{diag}(\boldsymbol{Y}_{i_1,h}, \dots, \boldsymbol{Y}_{i_{p_h},h})$$
$$\boldsymbol{\delta} = (\boldsymbol{\delta}_{i_1}^T \dots \boldsymbol{\delta}_{i_{p_h}}^T)^T$$
$$\boldsymbol{z} = (\boldsymbol{z}_{i_1}^T \dots \boldsymbol{z}_{i_{p_h}}^T)^T$$

Dal momento che il sistema è sovradeterminato si può utilizzare il metodo dei minimi quadrati per approssimare una soluzione:

$$\arg\min_{\boldsymbol{\delta}} \|\boldsymbol{Y}_h \boldsymbol{\delta} - \boldsymbol{z}_h\|$$

I fattori di aggiustamento così calcolati permettono di preservare il comportamento della rete avendo rimosso il neurone h.

Rimane ora da determinare come scegliere l'unità da rimuovere: idealmente, la scelta più appropriata sarebbe eliminare l'unità nascosta con il più piccolo **residuo finale** calcolato sul sistema corrispondente. Questo garantisce che la rimozione dell'unità avrà un impatto minimo sul comportamento della rete, tuttavia il calcolo del residuo minimo finale ha un costo computazionale elevato (a

causa del numero di sistemi da risolvere, uno per ogni neurone nascosto).

Nella pratica si cerca pertanto una soluzione subottimale: i metodi di riduzione dei residui per risolvere il problema dei minimi quadrati (come CGPCNE) partono da una soluzione iniziale $\boldsymbol{\delta}_0$ e producono una sequenza di punti $\{\boldsymbol{\delta}(k)\}$ tale per cui i residui

$$r_k = \|\boldsymbol{Y}_h\boldsymbol{\delta}(k) - \boldsymbol{z}_h\|$$

si riducono, ovvero $r_k \geq r_{k+1}$. Un'approssimazione accettabile alla soluzione ottimale è scegliere l'unità h con il più piccolo **residuo iniziale**. Dal momento che $\boldsymbol{\delta}(0)$ è solitamente il vettore nullo si ha

$$\boldsymbol{\delta}(0) = \mathbf{0} \implies r_0 = \|\boldsymbol{z}_h\|$$

e quindi si deve scegliere l'unità con $\|\boldsymbol{z}_h\|$ minimo. Naturalmente questo criterio non garantisce la soluzione ottimale, in quanto non è detto che partendo dal residuo iniziale minimo si giunga al residuo finale minimo, tuttavia nella pratica si è dimostrato efficace e semplice da implementare.

Riassumendo, data una rete sovradimensionata, i passi fondamentali dell'algoritmo di pruning sono i seguenti:

① Trovare l'unità h con $\|\boldsymbol{z}_h\|$ minimo.
② Risolvere il sistema corrispondente (aggiustare i pesi).
③ Rimuovere l'unità h.
④ Ritorna al punto 1 se $Performance(pruned) - Performance(original) < \epsilon$, altrimenti scarta l'ultima rete ridotta.

5

RETI DI HOPFIELD

In questo capitolo considereremo le reti neurali viste come **sistemi dinamici**[1] non lineari, con particolare attenzione al problema della loro stabilità o neurodinamica (vedi appendice B): per introdurre la variabile tempo in una rete è sufficiente aggiungere dei cicli, ovvero operare con **reti ricorrenti**.

Le reti ricorrenti con unità non lineari sono generalmente difficili da analizzare: possono convergere a uno stato stabile, oscillare o seguire **traiettorie caotiche** il cui andamento non è prevedibile.

Tuttavia, il fisico americano J. J. Hopfield si accorse che se le connessioni sono **simmetriche** esiste una funzione di **energia globale**. Le reti di Hopfield sono:

- **reti ricorrenti ad uno strato** in cui ogni neurone è connesso a tutti gli altri (escluso sè stesso);
- **simmetriche**: perché hanno la matrice dei pesi sinaptici simmetrica, ovvero $\boldsymbol{W} = \boldsymbol{W}^T$;
- **non lineari**: nella formulazione continua, ogni neurone ha una funzione di attivazione non lineare invertibile.

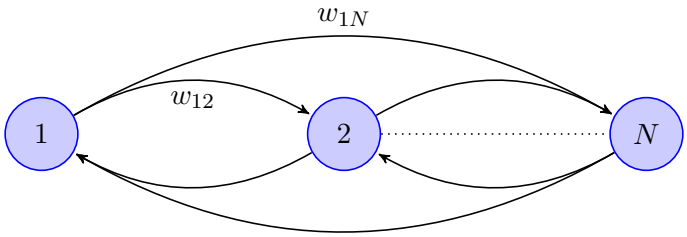


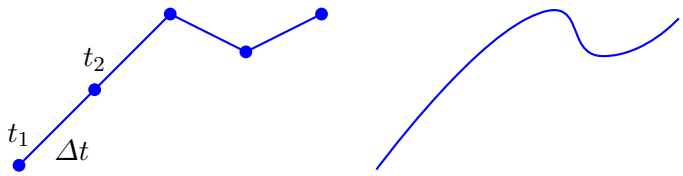
Figura 22: Una rete ricorrente.

Per quanto riguarda l'aggiornamento di un neurone si possono scegliere tre strade diverse:

- **aggiornamento asincrono**, in cui si aggiorna un neurone alla volta;
- **aggiornamento sincrono**, dove tutti i neuroni vengono aggiornati nello stesso istante;
- **aggiornamento continuo**, in cui tutti i neuroni si aggiornano continuamente.

1 Un sistema dinamico è un modello matematico utilizzato per descrivere situazioni il cui stato varia nel tempo.

Esistono infine due formulazioni del modello di Hopfield, **discreto** e **continuo**, che si differenziano per la modalità di scorrimento del tempo.



(a) Caso discreto: Δt costante (b) Caso continuo: $\Delta t \to 0$

Figura 23: Confronto tra caso discreto e continuo.

5.1 MODELLO DI HOPFIELD: CASO DISCRETO

In questa sezione si considerano reti di Hopfield in cui il tempo scorre in maniera discreta e i neuroni si aggiornano in modo asincrono.

Per quanto riguarda l'input al neurone si adotta lo stesso modello di McCulloch e Pitts, con l'aggiunta di un fattore di influenza esterno:

$$H_i = \underbrace{\sum_{j \neq i} w_{ij} V_j}_{\text{modello M\&P}} + \underbrace{I_i}_{\text{input esterno}}$$

La funzione di attivazione (discreta) è la seguente:

$$V_i = \begin{cases} +1 & \text{se } H_i > 0 \\ -1 & \text{se } H_i < 0 \end{cases} \tag{1}$$

L'aggiornamento dei neuroni è un processo casuale e la selezione dell'unità da aggiornare può essere fatta in due modi:

1. ad ogni istante temporale si sceglie a caso l'unità i da aggiornare (utile nelle simulazioni);
2. ogni unità si aggiorna indipendentemente con probabilità costante ad ogni istante.

A differenza delle reti feedforward, una rete di Hopfield è un sistema dinamico: parte da uno stato iniziale

$$\boldsymbol{V}(0) = (V_1(0), \ldots, V_n(0))^T$$

ed evolve lungo una traiettoria fino a raggiungere un punto fisso in cui $\boldsymbol{V}(t+1) = \boldsymbol{V}(t)$ (convergenza).

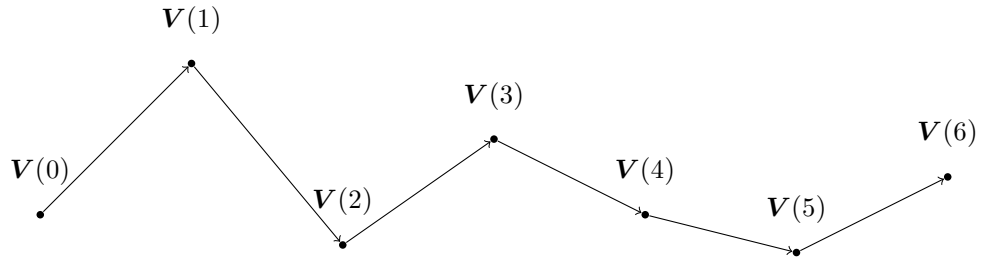


Figura 24: Traiettoria di una rete di Hopfield nel modello discreto.

Per dimostrare la convergenza, introduciamo la **funzione di energia** E che governa il sistema

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_{ij}V_iV_j - \sum_{i=1}^{n} I_iV_i \tag{2}$$

dove il fattore $1/2$ è aggiunto perché i termini identici $w_{ij}x_ix_j$ e $w_{ji}x_jx_i$ sono presenti nella doppia sommatoria.

Il seguente teorema fornisce una condizione sufficiente per la convergenza del sistema.

Teorema 3 (Teorema di Hopfield (caso discreto) - J. J. Hopfield, 1982)**.** *Se la matrice dei pesi di una rete di Hopfield è simmetrica con* $\mathrm{diag}(\boldsymbol{W}) = \boldsymbol{0}$*, allora la funzione di energia* (2) *è una funzione di Lyapunov per il sistema, quindi*

$$\Delta E = E(t+1) - E(t) \leq 0$$

con uguaglianza solo quando il sistema ha raggiunto un punto stazionario.

Dimostrazione. Visto che stiamo utilizzando la modalità di aggiornamento asincrono, supponiamo che il neurone h cambi il proprio stato. La differenza di energia è data da:

$$\begin{aligned}\Delta E = &-\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_{ij}V_i(t+1)V_j(t+1) - \sum_{i=1}^{n} I_iV_i(t+1)\\ &+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_{ij}V_i(t)V_j(t) + \sum_{i=1}^{n} I_iV_i(t)\end{aligned}$$

$$= \underbrace{-\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_{ij}(V_i(t+1)V_j(t+1) - V_i(t)V_j(t))}_{A}$$
$$\underbrace{-\sum_{i=1}^{n} I_i(V_i(t+1) - V_i(t))}_{B}$$

Per quanto riguarda il termine B abbiamo:

$$\begin{aligned} B &= -\sum_{i=1}^{n} I_i \Delta V_i \\ &= -I_h \Delta V_h \end{aligned}$$

Passiamo ora al primo termine A; per prima cosa espandiamo la sommatoria isolando il neurone h:

$$A = \underbrace{-\frac{1}{2}\sum_{i\neq h}\sum_{j\neq i} w_{ij}(V_i(t+1)V_j(t+1) - V_i(t)V_j(t))}_{A_1}$$
$$\underbrace{-\frac{1}{2}\sum_{j\neq h} w_{hj}(V_h(t+1)V_j(t+1) - V_h(t)V_j(t))}_{A_2}$$

Poiché $V_i(t) = V_i(t+1)$ per $i \neq h$ si può riscrivere A_1 come:

$$\begin{aligned} A_1 &= -\frac{1}{2}\sum_{i\neq h}\sum_{j\neq i} w_{ij}(\underbrace{V_i(t+1)}_{=V_i(t)} V_j(t+1) - V_i(t)V_j(t)) \\ &= -\frac{1}{2}\sum_{i\neq h}\sum_{j\neq i} w_{ij} V_i(t) \underbrace{(V_j(t+1) - V_j(t))}_{=\Delta V_j} \\ &= -\frac{1}{2}\sum_{i\neq h} w_{ih} V_i(t) \Delta V_h \end{aligned}$$

Per quanto riguarda A_2 il procedimento è lo stesso:

$$\begin{aligned} A_2 &= -\frac{1}{2}\sum_{j\neq h} w_{hj}(V_h(t+1)V_j(t+1) - V_h(t)V_j(t)) \\ &= -\frac{1}{2}\sum_{i\neq h} w_{hi} V_i(t) \Delta V_h \end{aligned}$$

Siccome la matrice dei pesi è simmetrica si ha

$$A = A_1 + A_2 = -\Delta V_h \sum_{i \neq h} w_{ih} V_i(t)$$

da cui:

$$\begin{aligned}\Delta E &= -\Delta V_h \sum_{i \neq h} w_{ih} V_i(t) - I_h \Delta V_h \\ &= -\Delta V_h \underbrace{\left[\sum_{i \neq h} w_{ih} V_i(t) + I_h \right]}_{\text{input di } h \text{ al tempo } t+1} \\ &= -\Delta V_h H_h(t+1)\end{aligned}$$

A questo punto è necessario dimostrare che il prodotto $\Delta V_h H_h(t+1)$ è sempre non negativo, in modo tale da provare la tesi. Per la regola di apprendimento (1) si ha

1. $\Delta V_h > 0$ se e solo se $H_h(t+1) > 0$;
2. $\Delta V_h < 0$ se e solo se $H_h(t+1) < 0$;

quindi il prodotto è sempre non negativo. □

5.2 REGOLA DI HEBB

Nei computer, se si vuole accedere ad una certa informazione, si utilizza il suo preciso indirizzo di memoria: questo tipo di memoria è definita **byte-addressable memory**. Nel cervello umano, invece, la memoria è indirizzata in base al contenuto, **content-addressable memory**: ad esempio pensare alla parola *volpe* potrebbe automaticamente attivare memorie relative ad altri animali simili, alla caccia, oppure al concetto di furbizia.

La **regola di Hebb** è stata introdotta per descrivere questo meccanismo di accesso alle informazioni e si basa sul principio che, se due neuroni si attivano contemporaneamente, la loro interconnessione deve essere rafforzata. In dettaglio il postulato di Hebb afferma:

> *Quando un assone di un neurone A è abbastanza vicino da eccitare un neurone B e questo, in modo ripetitivo e persistente, gli invia un potenziale di azione, inizia un processo di crescita in uno o entrambi i neuroni tale da incrementare l'efficienza di A.*

La memoria è rappresentata da un insieme di P patterns $\boldsymbol{x}^\mu$, con $\mu = 1, \dots, P$: quando viene presentato un nuovo pattern $\boldsymbol{x}$, la rete tipicamente risponde producendo il pattern salvato in memoria che più somiglia a $\boldsymbol{x}$.

In accordo al postulato di Hebb, sono utilizzati pesi proporzionali alla correlazione nell'attivazione tra un neurone pre e post sinaptico

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} x_i^\mu x_j^\mu$$

dove N è il numero di unità binarie con output $s_1, \dots, s_N$.

Il meccanismo di **recall** è il seguente:

$$s_i = \text{sgn}\left(\sum_j w_{ij} s_j \right)$$

Ci sono tuttavia alcuni problemi nell'utilizzo di reti di Hopfield come memorie indirizzate dal contenuto:

- il numero massimo di pattern[2] è $0.15N$;
- talvolta la rete produce degli **stati spuri**, ovvero stati che non fanno parte dei pattern memorizzati;
- il pattern evocato non è necessariamente il più simile a quello di input;
- i pattern non sono richiamati tutti con la stessa enfasi.

5.3 MODELLO DI HOPFIELD: CASO CONTINUO

In questo modello i neuroni non sono più dispositivi binari con stati $(0, 1)$ o $(-1, 1)$, ma producono un output che identifica la quantità di corrente elettrica: lo scopo di Hopfield, infatti, era quello di fornire un'implementazione fisica del suo modello che imiti il più possibile il funzionamento di un cervello biologico.

L'output di un neurone i è dato da

$$V_i = g_\beta(\mu_i) = g_\beta \left(\sum_j w_{ij} V_j + I_i \right)$$

dove g_β è la funzione di attivazione **crescente**, **continua** e **non lineare**. Le funzioni più utilizzate come g_β sono la tangente iperbolica e la sigmoidea:

$$\tanh_\beta(\mu) = \frac{e^{\beta\mu} - e^{-\beta\mu}}{e^{\beta\mu} + e^{-\beta\mu}} \quad \in]-1, 1[\qquad g_\beta(\mu) = \frac{1}{1 + e^{-\beta\mu}} \quad \in]0, 1[$$

2 I pattern sono memorizzati negli stati di equilibrio della rete.

Il parametro β indica la *stickiness* della funzione: per $\beta \to \infty$ la funzione diventa sempre più ripida fino a diventare una funzione a gradino.

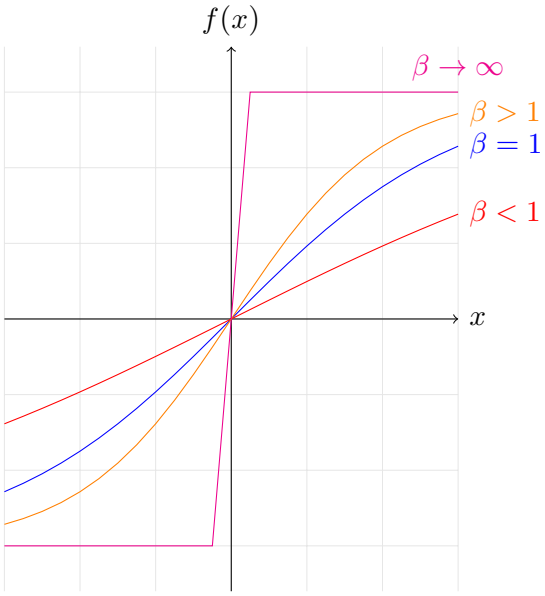


Figura 25: La funzione tangente iperbolica.

L'utilizzo di valori continui permette la modalità di **aggiornamento continuo** dei neuroni. Secondo questa modalità, il sistema evolve in accordo al seguente insieme di equazioni differenziali

$$V_i + \tau_i \frac{\mathrm{d}V_i}{\mathrm{d}t} = g_\beta \left(\sum_j w_{ij} V_j + I_i \right)$$

dove τ_i è una costante positiva che rappresenta la resistenza elettrica. Il sistema raggiunge la stabilità quando $\mathrm{d}V_i / \mathrm{d}t = 0 \, \forall i$.

In maniera del tutto equivalente, è possibile esprimere l'equazione di stato non incentrando l'attenzione sulla variazione dello stato V_i nel tempo, quanto sulla variazione dell'input netto μ_i nel tempo, pervenendo alla seguente equazione:

$$\mu_i + \tau_i \frac{\mathrm{d}\mu_i}{\mathrm{d}t} = \sum_j w_{ij} V_j + I_i = \sum_j w_{ij} g_\beta(\mu_j) + I_i$$

La **funzione energia** nel modello continuo è simile a quella nel caso discreto:

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} V_i V_j + \sum_i \int_0^{V_i} g_\beta^{-1}(V) \, \mathrm{d}V - \sum_i I_i V_i \tag{3}$$

Analogamente al caso discreto, il seguente teorema fornisce una condizione sufficiente per la convergenza del sistema.

Teorema 4 (Teorema di Hopfield (caso continuo) - J. J. Hopfield, 1982)**.** *Se la matrice dei pesi di una rete di Hopfield è simmetrica con* $\operatorname{diag}(W) = \mathbf{0}$*, allora la funzione di energia* (3) *è una funzione di Lyapunov per il sistema, quindi* $\mathrm{d}E/\,\mathrm{d}t \leq 0$ *con uguaglianza quando il sistema ha raggiunto un punto stazionario.*

Dimostrazione. Calcoliamo la derivata della funzione energia:

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\frac{1}{2}\sum_i\sum_j w_{ij}\frac{\mathrm{d}V_i}{\mathrm{d}t}V_j - \frac{1}{2}\sum_i\sum_j w_{ij}V_i\frac{\mathrm{d}V_j}{\mathrm{d}t} + \sum_i \underbrace{g_\beta^{-1}(V_i)}_{=\mu_i}\frac{\mathrm{d}V_i}{\mathrm{d}t} - \sum_i I_i\frac{\mathrm{d}V_i}{\mathrm{d}t}$$

Per simmetria di W si possono sommare i primi due termini:

$$\begin{aligned}
&= -\sum_i\sum_j w_{ij}\frac{\mathrm{d}V_i}{\mathrm{d}t}V_j + \sum_i \mu_i\frac{\mathrm{d}V_i}{\mathrm{d}t} - \sum_i I_i\frac{\mathrm{d}V_i}{\mathrm{d}t} \\
&= -\sum_i \frac{\mathrm{d}V_i}{\mathrm{d}t}\underbrace{\left(\sum_j w_{ij}V_j - \mu_i + I_i\right)}_{=\tau_i\frac{\mathrm{d}\mu_i}{\mathrm{d}t}} \\
&= -\sum_i \tau_i\frac{\mathrm{d}V_i}{\mathrm{d}t}\frac{\mathrm{d}\mu_i}{\mathrm{d}t} \\
&= -\sum_i \tau_i g'_\beta(\mu_i)\left(\frac{\mathrm{d}\mu_i}{\mathrm{d}t}\right)^2 \leq 0
\end{aligned}$$

L'ultima disuguaglianza è verificata poiché g_β è crescente, quindi $g'_\beta > 0$, $\tau_i > 0$ per ipotesi e il quadrato di un numero è sempre non negativo. Vale quindi la doppia implicazione

$$\frac{\mathrm{d}E}{\mathrm{d}t} = 0 \Leftrightarrow \frac{\mathrm{d}\mu_i}{\mathrm{d}t} = 0\,\forall i$$

cioè il valore dell'energia nel tempo rimane fisso se e solo se la rete ha raggiunto un punto di equilibrio. □

5.4 CORRISPONDENZA TRA I DUE MODELLI

Esiste una relazione stretta tra il modello continuo e quello discreto. Si noti che

$$V_i = g_\beta(\mu_i) = g_1(\beta\mu_i)$$

da cui si ricava:

$$\mu_i = \frac{1}{\beta}g_1^{-1}(V_i)$$

Il secondo termine della funzione energia diventa

$$\sum_i \int_0^{V_i} g_\beta^{-1}(V)\ \mathrm{d}V = \frac{1}{\beta} \sum_i \int_0^{V_i} g_1^{-1}(V)\ \mathrm{d}V$$

che, per $\beta \to \infty$, diventa trascurabile e la funzione di energia risulta uguale a quella nel modello discreto.

$$\begin{aligned} E &= -\frac{1}{2} \sum_i \sum_j w_{ij} V_i V_j + \underbrace{\frac{1}{\beta} \sum_i \int_0^{V_i} g_1^{-1}(V)\ \mathrm{d}V}_{=\,0 \text{ per } \beta \to \infty} - \sum_i I_i V_i \\ &= -\frac{1}{2} \sum_i \sum_j w_{ij} V_i V_j - \sum_{i=1}^{n} I_i V_i \end{aligned}$$

6

OTTIMIZZAZIONE CON LE RETI NEURALI

In questo capitolo saranno introdotti alcuni famosi problemi NP-difficili:[1] il problema del commesso viaggiatore (*Traveling Salesman Problem*) e il problema della cricca massima (*Maximum Clique Problem*). Dal momento che questi problemi non sono risolvibili in tempo polinomiale, sarà fornita una formulazione alternativa in modo tale da poter utilizzare tecniche di ottimizzazione continua per approssimare una soluzione.

6.1 PROBLEMA DEL COMMESSO VIAGGIATORE

Il **problema del commesso viaggiatore** è il seguente: data una rete di città, trovare il percorso di lunghezza minima che un commesso viaggiatore deve seguire per visitare tutte le città una e una sola volta e poi tornare alla città di partenza.

6.1.1 *Formalizzazione del problema*

La rete di città può essere rappresentata mediante un grafo completo pesato $G = (V, E, w)$ dove V è l'insieme delle città, E l'insieme delle strade che le collegano e w la funzione peso che assegna ad ogni arco la distanza tra i nodi corrispondenti. Il problema può dunque essere formalizzato come segue:

> *Dato un grafo completo pesato $G = (V, E, w)$, trovare il ciclo di costo minimo (di lunghezza $|V|$) che tocchi tutti i nodi in V una sola volta.*

6.1.2 *Soluzione del TSP con le reti di Hopfield*

Per affrontare il problema utilizzando le reti di Hopfield è necessario trovare una **funzione di energia** adeguata, in modo tale che il minimo corrisponda alla soluzione del problema che si sta affrontando.

Per rappresentare il percorso si può usare una **matrice di permutazione**: dato il grafo in figura 26, il percorso $B \to A \to C \to D$ può essere rappresentato con la matrice in tabella 1.
Per n città sono quindi necessari n^2 neuroni nella rete di Hopfield, in cui ogni neurone identifica una città e una fermata. Introduciamo ora alcune notazioni:

1 Un problema P si dice **NP-difficile** se ogni problema $L \in NP$ è riducibile in tempo polinomiale a P. Un problema P si dice **NP-completo** se $P \in NP$ ed è NP-difficile.

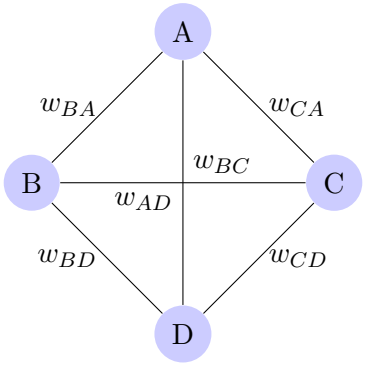


Figura 26: Grafo completo pesato.

Città \ Fermata	1	2	3	4
A	0	**1**	0	0
B	**1**	0	0	0
C	0	0	**1**	0
D	0	0	0	**1**

Tabella 1: Matrice di permutazione del percorso BACD

- X indica una città;
- i una fermata in cui è visitata una città;
- $V_{X,i}$ è l'output dell'unità X, i;
- $T_{Xi,Yj}$ sono i pesi delle connessioni;
- $V_{X,i} = 1$ se la città X è visitata alla fermata i, 0 altrimenti;
- $d_{X,Y}$ distanza tra la città X e Y.

La funzione che rappresenta il costo totale del cammino è

$$E_1 = \frac{D}{2} \sum_X \sum_{Y \neq X} \sum_i d_{X,Y} V_{X,i} (V_{Y,i-1} + V_{Y,i+1})$$

dove D è una costante positiva e gli indici sono intesi modulo n.

Per imporre i vincoli richiesti dal problema del commesso viaggiatore si introducono le seguenti funzioni di energia, che sono minimizzate quando i vincoli corrispondenti sono verificati.

- **Vincolo sulle righe**. Ogni città deve essere visitata al più una sola volta:

$$E_2 = \frac{A}{2} \sum_X \sum_i \sum_{j \neq i} V_{X,i} V_{X,j}$$

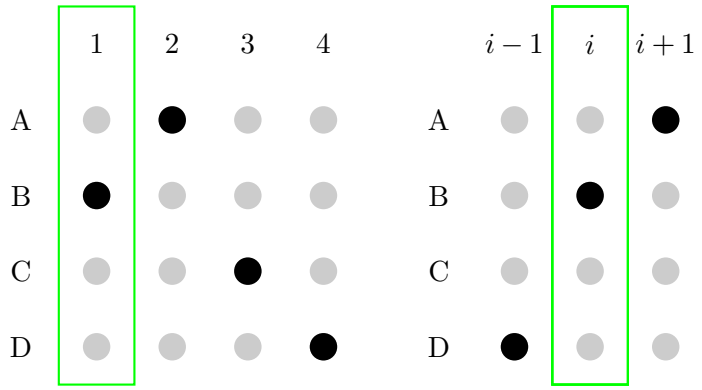


Figura 27: Interpretazione grafica di E_1

- **Vincolo sulle colonne**. In una fermata si visita al più una città:

$$E_3 = \frac{B}{2} \sum_i \sum_X \sum_{Y \neq X} V_{X,i} V_{Y,i}$$

- **Vincolo sulle entrate**. Devono essere attraversate esattamente n città:

$$E_4 = \frac{C}{2} \left(\sum_X \sum_i V_{X,i} - n \right)^2$$

Abbiamo quattro funzioni energia da minimizzare, tuttavia per poter utilizzare una rete di Hopfield è necessaria un'unica funzione: per questo motivo si esprime la funzione di costo totale come combinazione lineare delle funzioni energia E_i:

$$E = E_1 + E_2 + E_3 + E_4$$

dove il peso da attribuire a ciascun termine è determinato dalle costanti positive A, B, C, D. C'è un ultimo problema da risolvere, ovvero la funzione energia dovrà essere della forma tipica di una rete di Hopfield:

$$E = -\frac{1}{2} \sum_{X,Y} \sum_{i,j} T_{Xi,Yj} V_{X,i} V_{Y,j} - \sum_{X,i} I_{X,i} V_{X,i}$$

Per farlo, si introduce il termine

$$\delta_{i,j} = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{altrimenti} \end{cases}$$

e si riscrivono le funzioni E_i come segue:

$$\begin{aligned} E_1 &= \frac{D}{2} \sum_X \sum_{Y \neq X} \sum_i d_{X,Y} V_{X,i} (V_{Y,i+1} + V_{Y,i-1}) \\ &= \frac{D}{2} \sum_{X,Y} \sum_{i,j} d_{X,Y} V_{X,i} V_{Y,j} (\delta_{j,i-1} + \delta_{j,i+1}) \end{aligned}$$

$$\begin{aligned} E_2 &= \frac{A}{2} \sum_X \sum_i \sum_{j \neq i} V_{X,i} V_{X,j} \\ &= \frac{A}{2} \sum_{X,Y} \sum_{i,j} V_{X,i} V_{Y,j} \delta_{X,Y} (1 - \delta_{i,j}) \end{aligned}$$

$$\begin{aligned} E_3 &= \frac{B}{2} \sum_i \sum_X \sum_{Y \neq X} V_{X,i} V_{Y,i} \\ &= \frac{B}{2} \sum_{X,Y} \sum_{i,j} V_{X,i} V_{Y,j} \delta_{i,j} (1 - \delta_{X,Y}) \end{aligned}$$

$$\begin{aligned} E_4 &= \frac{C}{2} \left(\sum_X \sum_i V_{X,i} - n \right)^2 \\ &= \frac{C}{2} \left(\left(\sum_X \sum_i V_{X,i} \right)^2 - 2n \sum_X \sum_i V_{X,i} + n^2 \right) \end{aligned}$$

dove

$$\begin{aligned} \left(\sum_X \sum_i V_{X,i} \right)^2 &= \left(\sum_X \sum_i V_{X,i} \right) \left(\sum_X \sum_i V_{X,i} \right) \\ &= \sum_{X,Y} \sum_{i,j} V_{X,i} V_{Y,j} \end{aligned}$$

In E_4 il termine costante n^2 si può tralasciare, in quanto influenza solamente il valore della funzione E_4 in corrispondenza del minimo, ma non la sua posizione. Mettendo il tutto assieme si ottiene la **funzione di energia totale**

$$\begin{aligned} E &= \frac{A}{2}\sum_{X,Y}\sum_{i,j} V_{X,i}V_{Y,j}\delta_{X,Y}(1-\delta_{i,j}) \\ &+ \frac{B}{2}\sum_{X,Y}\sum_{i,j} V_{X,i}V_{Y,j}\delta_{i,j}(1-\delta_{X,Y}) \\ &+ \frac{C}{2}\left(\sum_{X,Y}\sum_{i,j} V_{X,i}V_{Y,j} - 2n\sum_{X,i} V_{X,i}\right) \\ &+ \frac{D}{2}\sum_{X,Y}\sum_{i,j} d_{X,Y}V_{X,i}V_{Y,j}(\delta_{j,i-1}+\delta_{j,i+1}) \\ &= -\frac{1}{2}\sum_{X,Y}\sum_{i,j} T_{Xi,Yj}V_{X,i}V_{Y,j} - \sum_{X,i} I_{X,i}V_{X,i} \end{aligned}$$

dove $T_{Xi,Yj}$ sono dati da

$$\begin{aligned} -T_{Xi,Yj} &= A\delta_{XY}(1-\delta_{ij}) && \text{(Peso inibitorio in ogni riga)} \\ &+ B\delta_{ij}(1-\delta_{XY}) && \text{(Peso inibitorio in ogni colonna)} \\ &+ C && \text{(Inibizione globale)} \\ &+ Dd_{XY}(\delta_{j,i-1}+\delta_{j,i+1}) && \text{(Costo del cammino minimo)} \end{aligned}$$

e il vettore di corrente esterna $\boldsymbol{I}$ ha come Xi-esima componente

$$I_{X,i} = nC \qquad \text{(Corrente esterna eccitatoria)}$$

6.1.3 *Un'altra formulazione*

Determinare i parametri A, B, C, D è particolarmente difficile. Un altro modo per esprimere i vincoli del TSP, che richiede di determinare un parametro in meno è il seguente:

$$E_2 = \frac{A}{2}\sum_{X}\left(\sum_{i} V_{X,i} - 1\right)^2 \qquad \text{(Vincolo sulle righe)}$$

$$E_3 = \frac{B}{2}\sum_{i}\left(\sum_{X} V_{X,i} - 1\right)^2 \qquad \text{(Vincolo sulle colonne)}$$

La funzione energia diventa:

$$E = \frac{D}{2} \sum_X \sum_{Y \neq X} \sum_i d_{X,Y} V_{X,i} (V_{Y,i+1} + V_{Y,i-1}) + E_2 + E_3$$

6.1.4 *Problema delle N regine*

Una famosa variante del TSP è il **problema delle N regine**: data una scacchiera $N \times N$ e N regine, si vogliono posizionare questi pezzi in modo tale che nessuno di essi possa attaccarne un altro.

Si può costruire una rete di Hopfield $N \times N$ in cui il neurone (i, j) è attivo se e solo se una regina occupa la posizione (i, j), con i seguenti vincoli:

- solo una regina in ciascuna riga;
- solo una regina in ciascuna colonna;
- solo una regina in ciascuna diagonale;
- solo N regine sulla scacchiera.

Il problema è analogo al TSP con l'aggiunta del vincolo sulle diagonali. La matrice dei pesi è la seguente:

$$\begin{aligned}
-T_{ij,kl} = {} & A\delta_{ik}(1 - \delta_{jl}) && \text{(Peso inibitorio in ogni riga)} \\
& + B\delta_{jl}(1 - \delta_{ik}) && \text{(Peso inibitorio in ogni colonna)} \\
& + C && \text{(Inibizione globale)} \\
& + D(\delta_{i+j,k-l} + \delta_{i-j,k-l})(1 - \delta_{ik}) && \text{(Peso inibitorio sulle diagonale)}
\end{aligned}$$

6.1.5 *Conclusioni*

L'ottimizzazione tramite le reti di Hopfield presenta alcuni svantaggi:

1. sono necessari n^2 neuroni;
2. il numero di connessioni è $\mathcal{O}(n^4)$;
3. i parametri A, B, C, D sono difficili da determinare;
4. i risultati ottenuti sono raramente soluzioni ammisibili, ovvero matrici binarie con i vincoli rispettati;
5. è difficile evitare i minimi locali della funzione di energia.

6.2 PROBLEMA DELLA CRICCA MASSIMA

Sia $G = (V, E)$ un grafo non orientato: si definisce **cricca** (o *clique*) un sottoinsieme $C \subseteq V$ di vertici mutuamente adiacenti, ovvero tali per cui $\forall i, j \in C$, se $i \neq j$, allora $(i, j) \in E$.

Definizione 2 (Clique massimale)**.** *Si definisce* **clique massimale** *di G una clique di G che non è contenuta in nessun'altra clique di G.*[2]

Definizione 3 (Clique massima)**.** *Si definisce* **clique massima** *di G una clique massimale di G di cardinalità massima.*

Il problema consiste nel trovare una cricca massima (MCP).

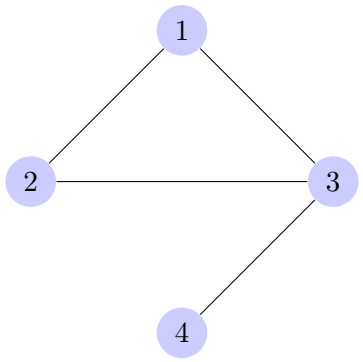


Figura 28: $C_1 = \{1, 2, 3\}$ è massima e massimale, $C_2 = \{3, 4\}$ è massimale, $C_3 = \{1, 2\}$ non è massimale ($C_3 \subseteq C_1$).

Trovare una clique massimale è un problema facile, mentre trovare quella massima è NP-completo, così come trovare la dimensione di tale clique.

Prima di dare una formulazione continua del problema della cricca massima, sono necessarie alcune definizioni:

- Se $C \subseteq V$, $\boldsymbol{x}^C$ indica il **vettore caratteristico** di C:

$$x_i^C = \begin{cases} \dfrac{1}{|C|} & \text{se } i \in C \\ 0 & \text{altrimenti} \end{cases}$$

- Δ è il **simplesso standard** in $\mathbb{R}^n$:

$$\Delta = \left\{ \boldsymbol{x} \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1 \text{ e } x_i \geq 0, \forall i \right\}$$

2 Se C è una clique massimale, allora ogni nodo $j \notin C$ è connesso a meno di $|C|$ nodi della clique. Se è connesso a meno di $|C| - 1$ nodi, la clique è massimale **stretta**.

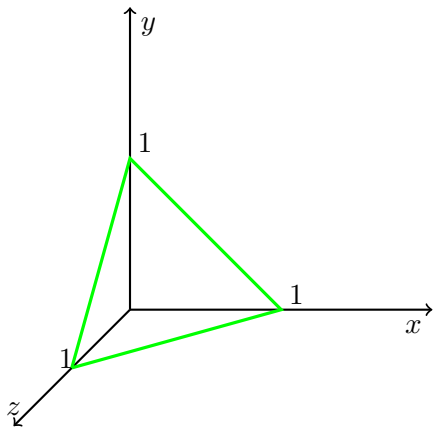


Figura 29: Simplesso standard in $\mathbb{R}^3$.

- $\boldsymbol{A} = (a_{ij})$ è la matrice di adiacenza binaria di G.

Si consideri la seguente funzione quadratica detta **Lagrangiano del grafo**:

$$f_G(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j$$

L'approccio tipico per risolvere il problema della cricca massima è quello di costruire un sistema dinamico che converga ai massimi di f_G in Δ: questi punti corrisponderanno alle soluzioni nello spazio discreto del problema originale. A tale scopo si introduce il teorema di Motzkin-Straus.

Teorema 5 (T. S. Motzkin & E. G. Straus, 1965)**.** *Sia $\boldsymbol{x}^*$ un massimo globale di f_G in Δ, allora la cardinalità della clique massima di G è legata a f_G dalla seguente formula:*

$$\omega(G) = \frac{1}{1 - f_G(\boldsymbol{x}^*)}$$

Inoltre un sottoinsieme di vertici C è una clique massima se e solo se il suo vettore caratteristico $\boldsymbol{x}^C \in \Delta$ è un massimo globale per f_G in Δ.

Purtroppo non tutti i massimizzatori di f_G sono nella forma di vettori caratteristici, pertanto possono essere usati solo per ottenere la cardinalità della cricca massima, ma non i vertici che la compongono: questi massimizzatori sono detti **soluzioni spurie**.

Ad esempio, il grafo in figura 30 contiene due clique massime:

$$C_1 = \{1, 2\} \qquad \boldsymbol{x}^{C_1} = (1/2, 1/2, 0)$$
$$C_2 = \{1, 3\} \qquad \boldsymbol{x}^{C_2} = (1/2, 0, 1/2)$$

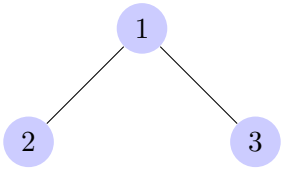


Figura 30: Grafo di esempio.

Tuttavia, come si può vedere dal seguente grafico, sono massimi globali tutti i punti interni al segmento $\boldsymbol{x}^{C_1}\boldsymbol{x}^{C_2}$ ovvero i punti $(1/2, \alpha/2, (1-\alpha)/2)$ con $\alpha \in (0,1)$: tuttavia, poiché non sono nella forma di vettori caratteristici, non possono essere usati per estrarre informazioni sui componenti delle clique massime.

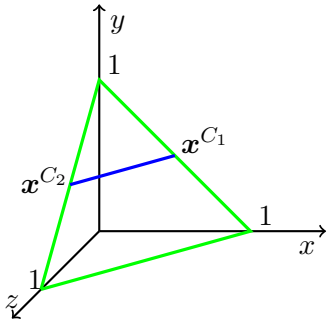


Figura 31: Soluzioni spurie in MCP

Il problema delle soluzioni spurie è stato risolto da Bomze proponendo una versione regolarizzata di $f_G(\boldsymbol{x})$:

$$\hat{f}_G(\boldsymbol{x}) = \boldsymbol{x}^T \left(\boldsymbol{A} + \frac{1}{2}\boldsymbol{I}_{|V|} \right) \boldsymbol{x}$$

Teorema 6 (I. M. Bomze, 1995)**.** *Sia $C \subseteq V$ e sia $\boldsymbol{x}^C$ il suo vettore caratteristico, allora:*

- *C è una clique massima di G se e solo se $\boldsymbol{x}^C$ è un massimo globale di $\hat{f}_G$ in Δ;*
- *C è una clique massimale di G se e solo se $\boldsymbol{x}^C$ è un massimo locale di $\hat{f}_G$ in Δ;*
- *ogni massimo locale è un vettore caratteristico ed è locale stretto.*

Parte II

TEORIA DEI GIOCHI

7

TEORIA DEI GIOCHI

La **teoria dei giochi** analizza situazioni in cui vi sono interazioni tra **agenti** diversi, ove ognuno è interessato a **massimizzare** il proprio **beneficio**, tali per cui le scelte di un agente possono influire sul beneficio che possono ottenere gli altri agenti. Essa è stata sviluppata con il preciso scopo di sopperire alle limitazioni legate all'ottimizzazione di una **singola** funzione obiettivo.

Vediamo brevemente gli avvenimenti più importanti nello studio della teoria dei giochi:

- **1921 - 1928**: prima formulazione di strategia mista e l'idea di trovare soluzioni di giochi in forma normale (E. Borel, J. von Neumann);
- **1944, 1947**: pubblicazione di *Theory of Games and Economic Behavior* (J. von Neumann, O. Morgenstern);
- **1950 - 1953**: contributi alla teoria dei giochi non cooperativi (J. Nash);
- **1972 - 1982**: applicazione della teoria dei giochi a problemi biologici (J. M. Smith).

7.1 GIOCHI FINITI IN FORMA NORMALE

Un gioco finito in forma normale consiste di un insieme di **giocatori**

$$I = \{1, \dots, n\} \qquad (n \geq 2)$$

ciascuno dei quali ha a disposizione un insieme di **azioni** (dette anche **strategie pure**):

$$S_i = \{1, \dots, m_i\} \qquad (m_i \geq 2)$$

L'insieme delle azioni giocate dagli individui in un dato istante prende il nome di **profilo strategico puro**

$$\boldsymbol{s} = (s_1, \dots, s_n)$$

e l'insieme dei profili strategici puri forma lo **spazio delle strategie pure**:

$$S = \prod_{i=1}^{n} S_i$$

In seguito alla giocata di un profilo strategico $\boldsymbol{s} \in S$ ciascun individuo $i \in I$ ottiene un **payoff**, ovvero la quantificazione del beneficio ottenuto dal giocatore in seguito alla giocata. Il payoff del giocatore i è rappresentato dalla funzione $\pi_i : S \to \mathbb{R}$; la **funzione di payoff combinata** $\boldsymbol{\pi} : S \to \mathbb{R}^n$ assegna ad ogni profilo strategico $\boldsymbol{s} \in S$ il vettore

$$\boldsymbol{\pi}(\boldsymbol{s}) = (\pi_1(\boldsymbol{s}), \dots, \pi_n(\boldsymbol{s}))$$

Un gioco in forma normale può dunque essere rappresentato dalla tripletta $(I, S, \boldsymbol{\pi})$.

7.2 GIOCHI A DUE GIOCATORI

Nel caso speciale di giochi a due giocatori, è comodo rappresentare le funzioni di payoff in forma matriciale:

- $\boldsymbol{A} = (a_{ij})$ è la matrice di payoff del primo giocatore, dove $a_{ij} = \pi_1(i, j)$ con $i \in S_1, j \in S_2$;
- $\boldsymbol{B} = (b_{ij})$ è la matrice di payoff del secondo giocatore, dove $b_{ij} = \pi_2(i, j)$ con $i \in S_1, j \in S_2$;

In base alle caratteristiche di $\boldsymbol{A}$ e $\boldsymbol{B}$ si possono distinguere delle categorie particolari di giochi:

- **giochi a somma zero**: $\boldsymbol{A} + \boldsymbol{B} = \boldsymbol{0}$;
- **giochi simmetrici**: $\boldsymbol{A} = \boldsymbol{B}^T$;
- **giochi doppiamente simmetrici**: $\boldsymbol{A} = \boldsymbol{A}^T = \boldsymbol{B}^T$.

7.3 GIOCHI SUCCINTI

Per descrivere un gioco in forma normale, con n giocatori e m strategie pure per ciascuno, occorre memorizzare nm^n numeri. Un **gioco succinto** è un gioco rappresentabile in forma molto più ridotta rispetto alla forma normale; ad esempio:

- **giochi sparsi**, dove la maggior parte dei payoff è 0;
- **giochi grafici**, dove i payoff di un giocatore dipendono dalle scelte di $d < n$ giocatori;
- **giochi simmetrici**.

7.4 POLYMATRIX GAMES

Un **polymatrix game** è un gioco non cooperativo[1] succinto in cui l'influenza relativa della scelta di una strategia da parte di un giocatore sul payoff di un altro è sempre la stessa, indipendentemente da cosa faranno i rimanenti giocatori. Formalmente:

- ci sono n giocatori con m strategie pure ciascuno;
- c'è una matrice di payoff $A^{ij} = (a^{ij}_{kl})$ di dimensione $m \times m$ per ogni coppia di giocatori (i, j);
- il payoff del giocatore i per il profilo strategico puro $\boldsymbol{s}$ è:

$$u_i(\boldsymbol{s}) = \sum_{j \neq i} a^{ij}_{s_i s_j}$$

Il numero di payoff da memorizzare è $\mathcal{O}(m^2 n^2)$. Il problema di trovare un equilibrio di Nash in un gioco di questo tipo è PPAD-completo.[2]

7.5 STRATEGIE MISTE

Supponiamo che il giocatore $i \in I$ decida la strategia pura da usare in base ad una distribuzione di probabilità sull'insieme di strategie pure S_i: tale distribuzione prende il nome di **strategia mista**. Essa può essere rappresentata da un vettore m_i-dimensionale $\boldsymbol{x}_i$ dove x_{ih} è la probabilità che il giocatore i giochi la sua strategia pura h. Per definizione di distribuzione di probabilità, $\boldsymbol{x}_i$ appartiene al simplesso standard m_i-dimensionale Δ_i:

$$\Delta_i = \left\{ \boldsymbol{x}_i \in \mathbb{R}^{m_i} : \sum_{h=1}^{m_i} x_{ih} = 1 \text{ e } x_{ih} \geq 0 \; \forall h \right\}$$

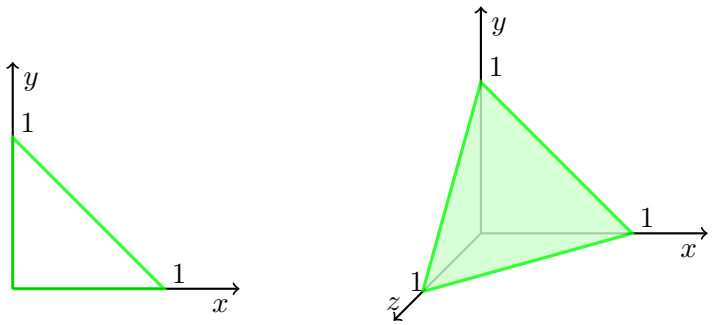
L'insieme delle strategie pure cui è assegnata una probabilità positiva prende il nome di **supporto** di $\boldsymbol{x_i}$:

$$\sigma(\boldsymbol{x_i}) = \{h \in S_i : x_{ih} > 0\}$$

Una strategia pura h può essere vista come una **strategia mista estrema** $\boldsymbol{x}_i$ dove $x_{ih} = 1$ e $x_{ij} = 0$ per $j \neq h$. Ogni strategia mista può essere espressa come combinazione lineare di strategie miste estreme.

1 Un gioco si dice **non cooperativo** se non c'è collaborazione tra i giocatori: tutti i giochi che tratteremo sono di questo tipo.

2 PPAD è un sottoinsieme di TFNP, un'estensione della classe di problemi decisionali NP a relazioni totali. Una relazione binaria $P \in TFNP$ se e solo se esiste un algoritmo che può verificare in tempo polinomiale se, dati x e y, $P(x, y)$ è verificata e $\forall x \exists y$ tale che $P(x, y)$ vale.



(a) (b)

Figura 32: A sinistra il simplesso a due dimensioni, a destra quello a tre dimensioni. Le strategie pure corrispondono ai vertici del simplesso.

Un **profilo strategico misto** è un vettore $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ dove la componente $\boldsymbol{x}_i$ è una strategia mista del giocatore $i \in I$. Lo **spazio delle strategie miste** è il multi-simplesso $\Theta = \prod_{i=1}^{n} \Delta_i$.

Nei giochi non cooperativi si assume che le scelte dei giocatori siano indipendenti tra loro, pertanto la probabilità che un profilo strategico puro $\boldsymbol{s}$ sia usato quando viene giocato un profilo strategico misto $\boldsymbol{x}$ è data da

$$\boldsymbol{x}(\boldsymbol{s}) = \prod_{i=1}^{n} x_{is_i}$$

e il **payoff atteso** del giocatore $i \in I$ è

$$u_i(\boldsymbol{x}) = \sum_{\boldsymbol{s} \in S} \boldsymbol{x}(\boldsymbol{s}) \pi_i(\boldsymbol{s})$$

7.6 BEST REPLIES ED EQUILIBRI DI NASH

Definizione 4 (Best reply)**.** *La* **best reply** *di un giocatore $i \in I$ al profilo strategico $\boldsymbol{x}_{-i}$[3] è una strategia mista $\boldsymbol{x}_i^* \in \Delta_i$ tale per cui:*

$$u_i(\boldsymbol{x}_i^*, \boldsymbol{x}_{-i}) \geq u_i(\boldsymbol{z}_i, \boldsymbol{x}_{-i}) \quad \forall \boldsymbol{z}_i \in \Delta_i$$

L'**insieme delle best replies** a una strategia $\boldsymbol{x}_{-i}$ per $i \in I$ si indica con $\beta_i^*(\boldsymbol{x}_{-i})$. Si noti che, escluso il caso in cui c'è un'unica miglior risposta (che è una strategia pura), il numero di best replies è infinito. Infatti:

- quando il supporto di una best reply include due o più strategie pure, una loro combinazione è una best reply;

3 Con $\boldsymbol{x}_{-i}$ si intende il profilo strategico misto $\boldsymbol{x} \in \Theta$ senza la componente i-esima.

- se due strategie pure sono individualmente best replies, una loro combinazione è una best reply.

Il concetto di equilibrio di Nash è motivato dall'idea che una teoria di decision-making razionale non dovrebbe creare un incentivo a deviare da essa per coloro i quali ci credono.

Definizione 5 (Equilibrio di Nash)**.** *Un profilo strategico $\boldsymbol{x} \in \Theta$ è un* **equilibrio di Nash** *se è una best reply a sè stesso, ovvero:*

$$u_i(\boldsymbol{x}_i, \boldsymbol{x}_{-i}) \geq u_i(\boldsymbol{z}_i, \boldsymbol{x}_{-i}) \quad \forall \boldsymbol{z}_i \in \Delta_i, \forall i \in I$$

Un equilibrio di Nash si dice **stretto** *se la disuguaglianza è stretta per $\boldsymbol{z}_i \neq \boldsymbol{x}_i$.*

Teorema 7. *Un profilo strategico $\boldsymbol{x} \in \Theta$ è un equilibrio di Nash se e solo se, per ogni giocatore $i \in I$, ogni strategia pura nel supporto di $\boldsymbol{x}_i$ è una best reply a $\boldsymbol{x}_{-i}$.*

L'esistenza di un equilibrio di Nash è garantita dal seguente teorema.

Teorema 8 (J. Nash, 1951)**.** *Ogni gioco finito in forma normale ammette un equilibrio di Nash misto.*

7.7 TEORIA DEI GIOCHI EVOLUTIVI

La **teoria dei giochi evolutivi** è una disciplina introdotta da J. M. Smith per modellare l'evoluzione del comportamento animale utilizzando mezzi e principi della teoria dei giochi. Si basa sulle seguenti assunzioni:

- una popolazione grande (idealmente infinita) di individui appartenenti alla stessa specie competono per risorse disponibili in **quantità limitata**;
- il conflitto è modellato come un gioco **simmetrico a due giocatori** selezionati in maniera casuale tra la popolazione;
- gli individui non giocano razionalmente, ma seguono un **pattern preprogrammato**;
- la riproduzione è **asessuata** per cui, a meno di mutazioni, ogni nascituro sarà un clone del genitore, ovvero programmato alla sua stessa strategia;
- il payoff è espresso in termini di **successo riproduttivo**.

In questo framework, una strategia mista può essere interpretata in due modi matematicamente equivalenti:

- ogni individuo gioca una strategia pura, ma parte della popolazione segue una strategia, il resto ne segue altre;
- ogni individuo gioca la stessa strategia mista.

Per trattare aspetti dinamici, la prima formulazione è la più conveniente.

7.8 STRATEGIE EVOLUTIVAMENTE STABILI

Assumiamo che un piccolo gruppo di **invasori**, che segue una strategia $\boldsymbol{y} \in \Delta$, appaia in una popolazione che segue la strategia $\boldsymbol{x} \in \Delta$; sia $\epsilon \in (0,1)$ la percentuale di invasori nella popolazione complessiva.

Il **payoff** in un gioco di questo tipo è lo stesso che si otterrebbe in un gioco con individui che seguono la strategia $\epsilon\boldsymbol{y} + (1-\epsilon)\boldsymbol{x} \in \Delta$.

Definizione 6 (Strategie evolutivamente stabili (ESS))**.** *Una strategia $\boldsymbol{x} \in \Delta$ si dice* **evolutivamente stabile** *se per ogni $\boldsymbol{y} \in \Delta \setminus \{\boldsymbol{x}\}$ esiste $\delta \in (0,1)$ tale per cui, per ogni $\epsilon \in (0,\delta)$, si ha*

$$\underbrace{u(\boldsymbol{x}, \epsilon\boldsymbol{y} + (1-\epsilon)\boldsymbol{x})}_{\textit{strategia } \textbf{\textit{attuale}}} > \underbrace{u(\boldsymbol{y}, \epsilon\boldsymbol{y} + (1-\epsilon)\boldsymbol{x})}_{\textit{strategia } \textbf{\textit{mutante}}}$$

Teorema 9. *Una strategia $\boldsymbol{x} \in \Delta$ è evolutivamente stabile se e solo se:*

- $u(\boldsymbol{y}, \boldsymbol{x}) \leq u(\boldsymbol{x}, \boldsymbol{x})$ *per ogni $\boldsymbol{y} \in \Delta$ (***equilibrio di Nash***);*
- $u(\boldsymbol{y}, \boldsymbol{x}) = u(\boldsymbol{x}, \boldsymbol{x}) \Rightarrow u(\boldsymbol{y}, \boldsymbol{y}) < u(\boldsymbol{x}, \boldsymbol{y})$ *per ogni $\boldsymbol{y} \in \Delta \setminus \{\boldsymbol{x}\}$ (***condizione di stabilità***).*

Dal teorema segue che:

- $\Delta^{ESS} \subseteq \Delta^{NE}$, dove Δ^{ESS} e Δ^{NE} sono l'insieme delle strategie evolutivamente stabili e l'insieme degli equilibri di Nash;
- se $\boldsymbol{x} \in \Delta$ è un equilibro di Nash stretto, allora $\boldsymbol{x} \in \Delta^{ESS}$.

Dal punto di vista computazionale:

- dire se un gioco simmetrico a due giocatori ha un ESS è NP-hard e coNP-hard;
- dire se $\boldsymbol{x} \in \Delta$ è ESS di un gioco simmetrico a due giocatori è coNP-hard.

7.9 DINAMICHE DI REPLICAZIONE

Le **dinamiche di replicazione** sono una classe di sistemi dinamici utilizzati nel contesto della teoria dei giochi evolutivi per modellare la replicazione.

Le equazioni di replicazione si distinguono rispetto ad altri modelli in quanto permettono di incorporare la distribuzione dei tipi di popolazione nella funzione di fitness, catturando così l'essenza della **selezione**. Non incorporano tuttavia le mutazioni, pertanto non sono in grado di creare nuovi tipi (strategie pure).

Sia $\boldsymbol{x}(t) \in \Delta$ il vettore che rappresenta lo **stato** della popolazione al tempo t, dove $x_i(t)$ è la percentuale di popolazione programmata alla strategia

$i \in \{1, \ldots, n\}$, e sia $\boldsymbol{A} = (a_{ij})$ la matrice $n \times n$ di payoff dove a_{ij} è il payoff che si ottiene giocando la strategia i contro un individuo che usa la strategia j. Il **payoff atteso** di un giocatore che segue la strategia i è dato da

$$\pi_i(\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x})_i = \sum_j a_{ij} x_j$$

mentre il **payoff medio** sull'intera popolazione è:

$$\pi(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} = \sum_i x_i \pi_i(\boldsymbol{x})$$

Esistono due formulazioni per le dinamiche di replicazione, una in cui il tempo scorre in maniera **continua**

$$\frac{\mathrm{d}}{\mathrm{d}t} x_i(t) = x_i(t) \left(\pi_i(\boldsymbol{x}(t)) - \sum_j x_j(t) \pi_j(\boldsymbol{x}(t)) \right)$$

e una in cui il tempo scorre in maniera **discreta**

$$x_i(t+1) = \frac{x_i(t)\pi_i(\boldsymbol{x}(t))}{\sum_j x_j(t)\pi_j(\boldsymbol{x}(t))}$$

con il vincolo che $\boldsymbol{A}$ sia non negativa.[4]

Si noti che il simplesso Δ è invariante rispetto entrambe le dinamiche, cioè se $\boldsymbol{x}(0) \in \Delta$ allora $\boldsymbol{x}(t) \in \Delta$ per ogni $t \geq 0$. Tipicamente la dinamica viene avviata dal baricentro di Δ, ovvero $\boldsymbol{x}(0) = \boldsymbol{e}/n$.[5]

Un punto $\boldsymbol{x}$ si dice **stazionario** se $\frac{\mathrm{d}}{\mathrm{d}t} x_i(t) = 0$ per ogni i; si dice **asintoticamente stabile** se qualunque traiettoria avviata in un intorno sufficientemente piccolo di $\boldsymbol{x}$ converge a $\boldsymbol{x}$ per $t \to \infty$.

Il seguente teorema fornisce alcuni risultati riguardanti gli stati stazionari e stabili delle dinamiche di replicazione.

4 Esistono delle equazioni alternative per cui la convergenza è più veloce. Sia $\kappa > 0$, la formulazione continua della dinamica di replicazione diventa

$$\frac{\mathrm{d}}{\mathrm{d}t} x_i(t) = x_i(t) \left(e^{\kappa \pi_i(\boldsymbol{x}(t))} - \sum_j x_j(t) e^{\kappa \pi_j(\boldsymbol{x}(t))} \right)$$

mentre quella discreta è:

$$x_i(t+1) = \frac{x_i(t) e^{\kappa \pi_i(\boldsymbol{x}(t))}}{\sum_j x_j(t) e^{\kappa \pi_j(\boldsymbol{x}(t))}}$$

5 $\boldsymbol{e}$ è un vettore di lunghezza n i cui elementi sono tutti 1.

Teorema 10 (P. D. Taylor, L. B. Jonker, 1978 & J. Nachbar, 1990)**.** *Un punto $\boldsymbol{x} \in \Delta$ è un equilibrio di Nash se e solo se $\boldsymbol{x}$ è il punto limite di una traiettoria di una dinamica di replicazione avviata all'interno di Δ.*

Inoltre, se $\boldsymbol{x}$ è un ESS, allora è un punto di equilibrio asintoticamente stabile per la dinamica di replicazione.

7.10 GIOCHI DOPPIAMENTE SIMMETRICI

Nel caso di giochi doppiamente simmetrici si possono dimostrare proprietà molto interessanti riguardanti le dinamiche di replicazione.

Teorema 11 (Teorema fondamentale della selezione naturale - V. Losert, E. Akin, 1983)**.** *In un gioco doppiamente simmetrico, il payoff medio della popolazione $f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}$ è* **strettamente crescente** *lungo una qualunque traiettoria non costante di una dinamica di replicazione, ovvero per $t \geq 0$ si ha*

- $\frac{\mathrm{d}}{\mathrm{d}t} f(\boldsymbol{x}(t)) \geq 0$ *nel caso continuo;*
- $f(\boldsymbol{x}(t+1)) \geq f(\boldsymbol{x}(t))$ *nel caso discreto;*

con uguaglianza se e solo se $\boldsymbol{x}(t)$ è un punto stazionario.

Teorema 12 (J. Hofbauer, K. Sigmund, 1988)**.** *In un gioco doppiamente simmetrico le seguenti affermazioni sono equivalenti:*

- $\boldsymbol{x} \in \Delta^{ESS}$*;*
- $\boldsymbol{x} \in \Delta$ *è un massimizzatore locale stretto di $f(\boldsymbol{x})$ in Δ;*
- $\boldsymbol{x} \in \Delta$ *è un punto asintoticamente stabile nelle dinamiche di replicazione.*

Inoltre $\boldsymbol{x} \in \Delta$ è un equilibrio di Nash se e solo se è un massimizzatore locale di $f(\boldsymbol{x})$ in Δ.

8

CLUSTERING

Il clustering è con ogni probabilità il più importante problema di **apprendimento non supervisionato**: l'obiettivo che si pone è organizzare dati non classificati in gruppi, detti **cluster**, i cui membri sono simili secondo qualche criterio.

Sebbene non esista una definizione formale universalmente accettata, tutti concordano nell'affermare che un cluster deve soddisfare i seguenti criteri:

- **criterio interno**: tutti gli oggetti all'*interno* di un cluster devono essere il più possibile simili tra loro;
- **criterio esterno**: tutti gli oggetti all'*esterno* di un cluster devono essere il più possibile dissimili rispetto a quelli contenuti al suo interno.

A seconda dell'input che viene fornito all'algoritmo, si distingue tra:

- clustering **feature-based** (o *central* clustering), dove gli oggetti sono rappresentati mediante vettori di caratteristiche;
- clustering **pairwise**, dove viene fornita una **matrice di similarità** tra gli oggetti.

Un approccio feature-based in alcuni casi può non essere applicabile, in quanto non si riesce a fornire una rappresentazione vettoriale degli oggetti su cui fare clustering, mentre è possibile determinarne la similarità: un esempio è dato da oggetti che sono rappresentati mediante grafi.

È utile distinguere il classico approccio al clustering dalla sua formulazione più moderna. Nel caso **classico**, l'obiettivo del clustering è di **partizionare** gli oggetti in insiemi massimalmente omogenei, il cui numero è tipicamente dato come input all'algoritmo; secondo il nuovo approccio:

- il numero di cluster non è necessario, in quanto vengono estratti sequenzialmente;
- è possibile non assegnare degli elementi, utile nel caso di segmentazione di immagini o one-class clustering;
- è possibile estrarre cluster sovrapposti.

8.1 K-MEANS

K-means è probabilmente il più famoso algoritmo di clustering feature-based. Dato un insieme di osservazioni $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, dove $\boldsymbol{x}_i$ è un vettore m-dimensionale,

e il numero desiderato di cluster K, lo scopo dell'algoritmo è di trovare un assegnamento di dati ai cluster e vettori $\boldsymbol{\mu}_k$ $(k = 1, \ldots, K)$, che rappresentano i centroidi dei cluster, tali per cui la distanza dei dati dal centroide del gruppo cui sono stati assegnati sia minima.

Sia $\boldsymbol{R} = (r_{ij})$ una matrice $n \times K$ tale per cui $r_{ij} = 1$ se $\boldsymbol{x}_i$ è assegnato al cluster j e 0 altrimenti. L'algoritmo minimizza la **misura di distorsione**

$$J = \sum_i \sum_k r_{ik} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$$

iterando i seguenti step:

- minimizzare J rispetto a $\boldsymbol{R}$, tenendo i centroidi fissi:

$$r_{ij} = \begin{cases} 1 & \text{se } j = \arg\min_k \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 \\ 0 & \text{altrimenti} \end{cases}$$

- minimizzare J rispetto $\boldsymbol{\mu}_k$, tenendo gli assegnamenti fissi:

$$\boldsymbol{\mu}_k = \frac{\sum_i r_{ik} \boldsymbol{x}_i}{\sum_i r_{ik}}$$

L'algoritmo viene arrestato quando non viene prodotta alcuna variazione negli assegnamenti oppure dopo un numero fisso di iterazioni: trattandosi di un problema NP-difficile, non è dato sapere a priori se l'algoritmo converga o meno.

Per quanto riguarda l'inizializzazione dei centroidi, questi vengono tipicamente scelti in maniera casuale dall'insieme delle osservazioni.

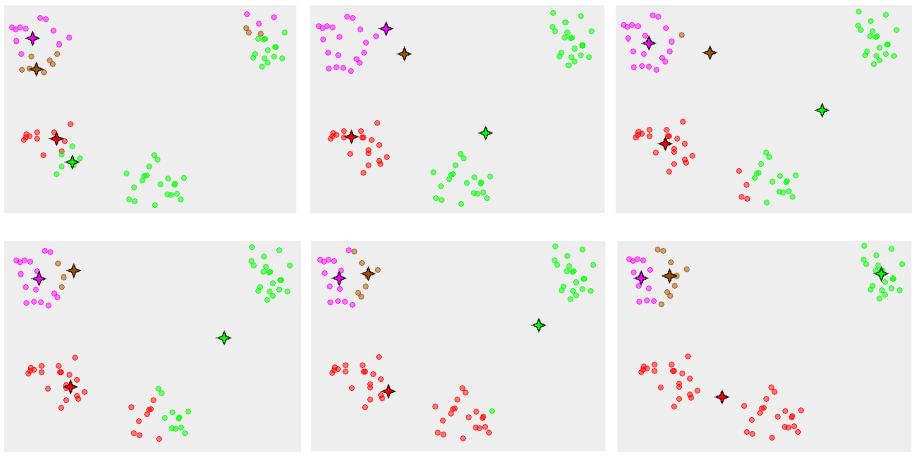


Figura 33: Convergenza (dopo 5 iterazioni) di K-means ad un minimo locale.

8.2 NORMALIZED CUT

Normalized cut è un algoritmo di clustering pairwise che utilizza tecniche della teoria spettrale dei grafi.

Sia $G = (V, E, w)$ un grafo pesato, dove V è l'insieme degli oggetti su cui fare clustering, E è l'insieme degli archi che indicano le relazioni tra gli oggetti e w è la funzione peso che assegna la similarità a coppie di vertici connessi. L'obiettivo è di individuare un partizionamento non banale (J, K) dei nodi in V tale da minimizzare

$$\text{NCUT}(J, K) = \frac{\text{CUT}(J, K)}{\text{ASSOC}(J, V)} + \frac{\text{CUT}(J, K)}{\text{ASSOC}(K, V)}$$

dove:

$$\text{CUT}(J, K) = \sum_{\substack{u \in J \\ t \in K}} w(u, t) \qquad \text{ASSOC}(J, V) = \sum_{\substack{u \in J \\ t \in V}} w(u, t)$$

Sia $\boldsymbol{y}$ un vettore $|V|$-dimensionale, dove

$$y_i = \begin{cases} 1 & \text{se } i \in J \\ -1 & \text{altrimenti} \end{cases}$$

$\boldsymbol{D} = (d_{ij})$ la matrice diagonale dove $d_{ii} = \sum_j w_{ij}$ e $\boldsymbol{A}$ la matrice di adiacenza pesata. Si può dimostrare che

$$\begin{aligned} \boldsymbol{y} &= \arg\min_{\boldsymbol{x}} \text{NCUT}(\boldsymbol{x}) \\ &= \arg\min_{\boldsymbol{x}} \frac{\boldsymbol{x}^T(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{D} \boldsymbol{x}} \text{ tale che } \boldsymbol{x}^T \boldsymbol{D} \boldsymbol{e} = 0 \end{aligned}$$

dove con $\text{NCUT}(\boldsymbol{x})$ si intende il costo del taglio normalizzato descritto da $\boldsymbol{x}$.

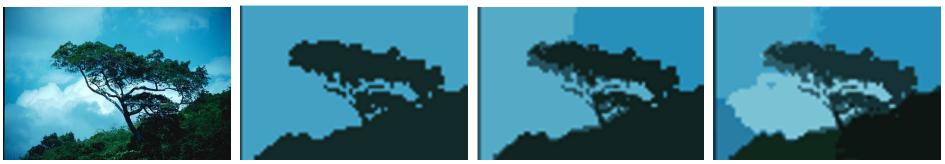
Rilassando il vincolo su $\boldsymbol{y}$ in modo da fargli assumere valori reali, possiamo approssimare la soluzione risolvendo un'equazione del tipo $(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{y} = \lambda \boldsymbol{D} \boldsymbol{y}$, ovvero $\boldsymbol{y}$ è un autovettore di $\boldsymbol{D} - \boldsymbol{A}$.

L'autovettore associato all'autovalore più piccolo è nullo (corrisponde al taglio banale $J = V$, $K = \emptyset$), mentre quello associato al secondo autovalore più piccolo rappresenta il taglio normalizzato ottimale. L'algoritmo è dunque il seguente:

① Crea un grafo pesato a partire dai dati in ingresso.
② Risolvi l'equazione $(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{y} = \lambda \boldsymbol{D} \boldsymbol{y}$.
③ Usa il segno degli elementi dell'autovettore associato al secondo autovalore più piccolo per partizionare il grafo.

④ Partiziona ricorsivamente le parti segmentate, se necessario.

Il problema di calcolare gli autovettori di una matrice $n \times n$ ha complessità $\mathcal{O}(n^3)$, ma nel caso di matrici sparse (come in questo caso) è possibile ridurla a $\mathcal{O}(n\sqrt{n})$.



(a) *original* (b) $k = 2$ (c) $k = 3$ (d) $k = 4$

Figura 34: Normalized cut usato per la segmentazione di immagini.

8.3 INSIEMI DOMINANTI

Il concetto di insieme dominante nasce dallo studio sulla formulazione continua del problema della cricca massima su grafi pesati.

Sia dunque $G = (V, E, w)$ un grafo non orientato dove i vertici sono gli oggetti su cui fare clustering, gli archi rappresentano le relazioni tra essi e i pesi rispecchiano la similarità tra coppie di vertici connessi. Sia inoltre $\boldsymbol{A} = (a_{ij})$ la matrice di adiacenza pesata.

Si consideri un insieme non vuoto di vertici $S \subseteq V$ e $i \in S$. Il **grado pesato medio** di i rispetto a S è definito come segue:

$$\text{AWDEG}_S(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij}$$

Se $j \notin S$, definiamo

$$\phi_S(i, j) = a_{ij} - \text{AWDEG}_S(i)$$

che misura la **similarità** tra j e i rispetto alla similarità media tra i e i suoi vicini in S.

Il **peso** di i rispetto ad S è dato dalla definizione ricorsiva

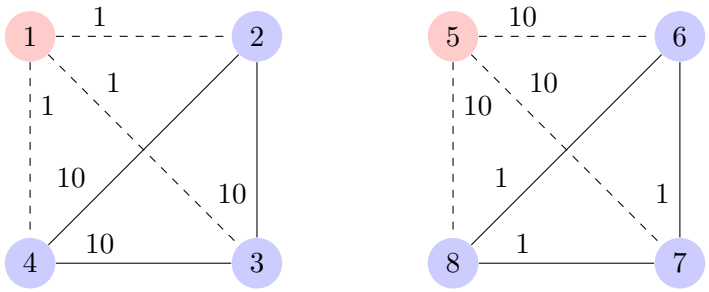
$$w_S(i) = \begin{cases} 1 & \text{se } |S| = 1 \\ \displaystyle\sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}(j, i) w_{S \setminus \{i\}}(j) & \text{altrimenti} \end{cases}$$

ed intuitivamente dà una misura di similarità complessiva tra i e i vertici in $S \setminus \{i\}$ rispetto alla similarità complessiva tra i vertici in $S \setminus \{i\}$.

Il **peso totale** di S è:

$$W(S) = \sum_{i \in S} w_S(i)$$

Vediamo ora come interpretare il significato del peso di un nodo rispetto ad un insieme di vertici. Nel grafo in figura 8.35(a), i vertici $\{2, 3, 4\}$ sono altamente simili tra loro: se proviamo ad aggiungere il vertice 1, che è dissimile, la similarità complessiva diminuisce, cioè $w_{\{1,2,3,4\}}(1) < 0$. Nel grafo in figura 8.35(b), il vertice 5 è altamente similare ai nodi $\{6, 7, 8\}$, dunque aggiungendolo all'insieme la similarità complessiva aumenta, cioè $w_{\{5,6,7,8\}}(5) > 0$.



(a) $w_{\{1,2,3,4\}}(1) < 0$ (b) $w_{\{5,6,7,8\}}(5) > 0$

Figura 35: Esempi sui pesi.

È stata dunque definita una misura che descrive cosa comporta, in termini di similarità, l'aggiunta o la rimozione di un nodo. Questo porta alla definizione di insiemi dominanti.

Definizione 7 (Insieme Dominante)**.** *Un sottoinsieme di vertici non vuoto $S \subseteq V$ tale che $W(T) > 0$ per ogni insieme non vuoto $T \subseteq S$ è detto* **dominante** *se:*

- $w_S(i) > 0 \quad \forall i \in S$ *(omogeneità interna)*
- $w_{S \cup \{i\}}(i) < 0 \quad \forall i \notin S$ *(disomogeneità esterna)*

Un insieme dominante è quindi un insieme di vertici massimalmente coesi tra loro e questa definizione corrisponde con quella di cluster.

Quando la matrice di affinità è binaria, l'insieme dominante coincide con la **clique massimale stretta** in un grafo. Se la disuguaglianza del secondo punto nella definizione 7 non fosse stretta, la clique sarebbe massimale, ma non necessariamente stretta.

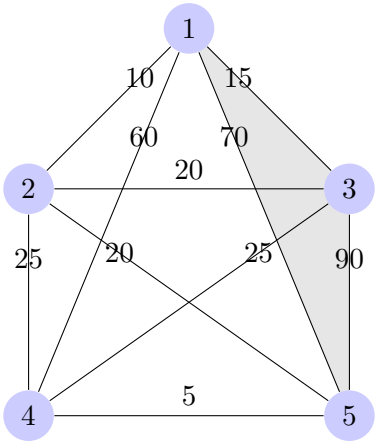


Figura 36: L'insieme $\{1, 3, 5\}$ è dominante.

8.3.1 *Gioco del clustering*

Si supponga di avere un insieme di oggetti O ed una matrice di affinità $\boldsymbol{A}$, possibilmente asimmetrica, degli elementi di O. Due giocatori selezionano contemporaneamente un elemento di O e ricevono un payoff proporzionale all'affinità tra gli oggetti scelti: è dunque nell'interesse di ogni giocatore selezionare un elemento supportato da quelli che l'avversario probabilmente sceglierà.

In questo **clustering game**:

- ci sono due giocatori;
- gli oggetti su cui fare clustering sono le strategie pure;
- la matrice di affinità (con diagonale nulla) corrisponde alla matrice di similarità.

Teorema 13 (A. Torsello, M. Pelillo, S. Rota Bulò, 2006)**.** *Le strategie evolutivamente stabili nel gioco del clustering con matrice di affinità* $\boldsymbol{A}$ *sono in corrispondenza uno a uno con gli insiemi dominanti.*

8.3.2 *Insiemi dominanti e affinità simmetriche*

Consideriamo la seguente funzione, che esprime il grado di coesione di un cluster espresso in forma vettoriale $\boldsymbol{x}$:

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}$$

Si può dimostrare il seguente risultato.

Teorema 14 (M. Pavan, M. Pelillo, 2007)**.** *Se S è un sottoinsieme dominante di vertici, il suo vettore caratteristico $\boldsymbol{x}^S$ definito come*

$$x_i^S = \begin{cases} \frac{w_S(i)}{W(S)} & \text{se } i \in S \\ 0 & \text{altrimenti} \end{cases}$$

è un massimizzatore locale stretto di f in Δ. Al contrario, se $\boldsymbol{x}^*$ è un massimizzatore locale stretto di f in Δ, il suo supporto σ è un insieme dominante, a condizione che $w_{\sigma\cup\{i\}}(i) \neq 0$ per ogni $i \notin \sigma$.

La condizione $w_{\sigma\cup\{i\}}(i) \neq 0$ è un tecnicismo dovuto alla presenza di soluzioni spurie. Il teorema può essere visto come la generalizzazione del teorema di Motzkin-Straus ai grafi con archi pesati.

Si può utilizzare una qualunque tecnica di ottimizzazione quadratica per massimizzare f in Δ, tuttavia le **dinamiche di replicazione** si sono dimostrate lo strumento ideale. Se la matrice di adiacenza $\boldsymbol{A}$ è simmetrica, per il teorema della selezione naturale, i punti asintoticamente stabili di una qualunque traiettoria non costante sono in corrispondenza uno a uno con i massimizzatori stretti di f in Δ e, dunque, con gli insiemi dominanti per la matrice $\boldsymbol{A}$.

8.3.3 *Insiemi dominanti e cluster sovrapposti*

L'idea alla base della tecnica per l'estrazione di cluster possibilmente sovrapposti consiste nel rendere iterativamente **instabili** le strategie evolutivamente stabili associate agli insiemi dominanti precedentemente estratti: per fare ciò, si introducono **nuove strategie** nel gioco del clustering che siano best replies solo ed esclusivamente per le ESS già trovate.

Sia Σ una tupla di ESS di un gioco con matrice di payoff $\boldsymbol{A}$ di dimensione $n \times n$ e sia Σ_i l'i-esima ESS contenuta nella tupla.

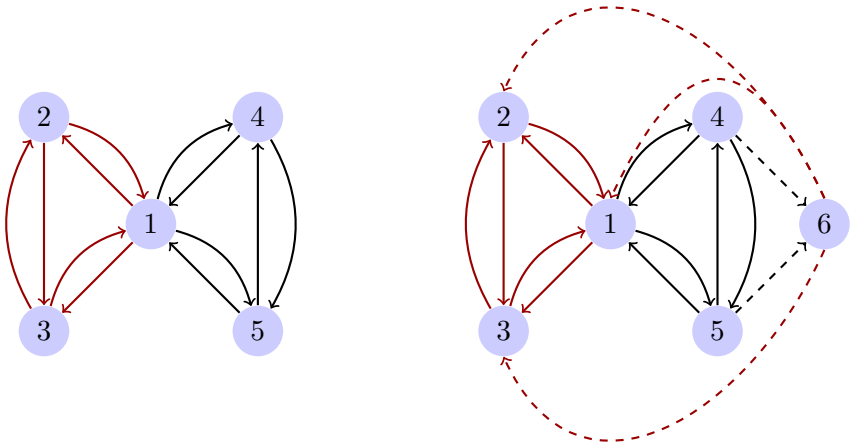
La Σ-extension $\boldsymbol{A}^\Sigma = (a_{ij}^\Sigma)$ della matrice di payoff $\boldsymbol{A}$ è definita come

$$a_{ij}^\Sigma = \begin{cases} a_{ij} & \text{se } i,j \in \{1,\dots,n\} \\ \alpha & \text{se } j > n \text{ e } i \notin \sigma(\Sigma_{j-n}) \\ \beta & \text{se } i,j > n \text{ e } i = j \\ \frac{1}{|\Sigma_{i-n}|}\sum_{k\in\Sigma_{i-n}} a_{kj} & \text{se } i > n \text{ e } j \in \sigma(\Sigma_{i-n}) \\ 0 & \text{altrimenti} \end{cases}$$

dove $\alpha > \beta$ e $\beta = \max_{i,j} a_{ij}$.

Teorema 15 (A. Torsello, M. Pelillo, S. Rota Bulò, 2008)**.** *Sia Φ un gioco doppiamente simmetrico a due giocatori con matrice di payoff $\boldsymbol{A}$ e sia Σ una tupla di ESS di Φ. Sia inoltre Φ^{Σ} un gioco a due giocatori con matrice di payoff $\boldsymbol{A}^{\Sigma}$. Allora $\boldsymbol{x}$ è una ESS di Φ non in Σ se e solo se $\boldsymbol{x}$ è una ESS di Φ^{Σ}.*

Un esempio di Σ-extension segue.



Supponiamo che l'insieme dominante $D = \{1, 2, 3\}$ sia stato estratto e non si voglia più ottenerlo in futuro: per fare ciò si introduce un nuovo nodo (6), si crea un arco $(n, 6)$ per ogni $n \notin D$ e un arco $(6, n)$ per ogni $n \in D$. I pesi sono impostati in accordo alla formula data sopra.

8.3.4 *Insiemi dominanti e partizionamento*

Anche se gli insiemi dominanti sono stati introdotti con il preciso scopo di superare l'approccio classico al clustering, visto come un problema di partizionamento, è comunque possibile creare una partizione dei dati in ingresso, secondo la strategia **peel-off**. L'algoritmo è il seguente:

① Trova un insieme dominante.
② Rimuovilo dal grafo.
③ Torna al punto 1 se il grafo contiene ancora dei vertici.

Il problema di questo approccio è dovuto al fatto che la rimozione di vertici introduce un **implicito cambio di scala** per quanto riguarda i pesi degli archi.

9

APPLICAZIONI DELLA TEORIA DEI GIOCHI

In questo capitolo si mostrano applicazioni della teoria dei giochi in contesti diversi dal clustering: in particolare, vedremo il problema dell'**etichettatura consistente** e della **trasduzione di grafi**.

9.1 PROBLEMA DI ETICHETTATURA

In un problema di **etichettatura consistente** si hanno

- un insieme di oggetti $B = \{b_1, \ldots, b_n\}$;
- un insieme di etichette $\Lambda = \{\lambda_1, \ldots, \lambda_m\}$;

e l'obiettivo è assegnare a ciascun oggetto in B un'etichetta in Λ utilizzando due fonti di informazione:

- **misurazioni locali** che catturano le caratteristiche salienti di ogni oggetto visto in isolamento, espresse sotto forma di n vettori m-dimensionali del tipo

  $$\boldsymbol{p}_i^{(0)} = (p_i^{(0)}(\lambda_1), \ldots, p_i^{(0)}(\lambda_m))$$

  con $p_i^{(0)}(\lambda) \geq 0$ per ogni $\lambda \in \Lambda$ e $\sum_\lambda p_i^{(0)}(\lambda) = 1$. L'elemento $p_i^{(0)}(\lambda)$ rappresenta il grado di confidenza iniziale (non contestuale) dell'ipotesi b_i *ha etichetta* λ;
- **informazioni contestuali** espresse in termini di matrici $n^2 \times m^2$ di coefficienti di compatibilità $\boldsymbol{R} = (r_{ij}(\lambda, \mu))$ dove $r_{ij}(\lambda, \mu)$ è la compatibilità tra l'ipotesi b_i *ha etichetta* λ e b_j *ha etichetta* μ.

Un processo di **relaxation labeling** riceve in input un'etichettatura iniziale $\boldsymbol{p}^{(0)} = (\boldsymbol{p}_1^{(0)} \ldots \boldsymbol{p}_n^{(0)})$ e la aggiorna iterativamente tenendo in considerazione il modello di compatibilità $\boldsymbol{R}$ in accordo alla formula

$$p_i^{(t+1)}(\lambda) = \frac{p_i^{(t)}(\lambda) q_i^{(t)}(\lambda; \boldsymbol{p})}{\sum_\mu p_i^{(t)}(\mu) q_i^{(t)}(\mu; \boldsymbol{p})}$$

dove

$$q_i^{(t)}(\lambda; \boldsymbol{p}) = \sum_j \sum_\mu p_j^{(t)}(\mu) r_{ij}(\lambda, \mu)$$

è nota come **funzione di supporto** e quantifica la compatibilità al tempo t tra l'ipotesi b_i *ha etichetta* λ e il contesto.

Lo **spazio degli assegnamenti pesati** delle etichette è l'insieme convesso

$$\mathcal{K} = \prod_{i=1}^{m} \Delta$$

dove Δ è il simplesso standard in $\mathbb{R}^n$. I vertici di $\mathcal{K}$ rappresentano **assegnamenti non ambigui** dove si assegna con probabilità 1 un'etichetta a ciascun oggetto.

Un assegnamento non ambiguo $\boldsymbol{p}$ si dice **consistente** se l'etichetta assegnata a ciascun oggetto da $\boldsymbol{p}$ riceve il più alto supporto dal contesto. Per analogia, si definisce la consistenza di un assegnamento pesato come segue.

Definizione 8 (Consistenza di un assegnamento pesato)**.** *L'assegnamento pesato* $\boldsymbol{p} \in \mathcal{K}$ *è* **consistente** *se*

$$\sum_{\lambda} p_i(\lambda) q_i(\lambda; \boldsymbol{p}) \geq \sum_{\lambda} v_i(\lambda) q_i(\lambda; \boldsymbol{p})$$

per ogni $i = 1, \ldots, n$ *e* $\boldsymbol{v} \in \mathcal{K}$*. Se la disuguaglianza è stretta per* $\boldsymbol{v} \neq \boldsymbol{p}$*, allora* $\boldsymbol{p}$ *è* **strettamente consisitente**.

Condizione necessaria affinché $\boldsymbol{p}$ sia strettamente consistente è che sia non ambiguo. Il seguente teorema fornisce un criterio operazionale per testare la consistenza di un etichettatura.

Teorema 16 (R. A. Hummel, S. W. Zucker, 1983)**.** *L'etichettatura pesata* $\boldsymbol{p} \in \mathcal{K}$ *è consistente se per ogni* $i = 1, \ldots, n$ *valgono*

- $q_i(\lambda) = c_i$ *se* $p_i(\lambda) > 0$*;*
- $q_i(\lambda) \leq c_i$ *se* $p_i(\lambda) = 0$*;*

per costanti non negative c_i*.*

La **consistenza locale media** di $p \in \mathcal{K}$ è data da

$$A(\boldsymbol{p}) = \sum_{i} \sum_{\lambda} p_i(\lambda) q_i(\lambda; \boldsymbol{p})$$

per la quale si dimostra il seguente risultato.

Teorema 17 (R. A. Hummel, S. W. Zucker, 1983)**.** *Se la matrice di compatibilità* $\boldsymbol{R}$ *è simmetrica, ovvero* $r_{ij}(\lambda, \mu) = r_{ji}(\mu, \lambda)$*, allora ogni massimizzatore locale* $\boldsymbol{p} \in \mathcal{K}$ *di* A *è consistente.*

Inoltre, se $p \in \mathcal{K}$ è strettamente consistente e $\boldsymbol{R}$ è simmetrica, si può dimostrare che $\boldsymbol{p}$ è un massimizzatore locale stretto di A.

Teorema 18 (M. Pelillo, 1997)**.** *Se la matrice di compatibilità* $\boldsymbol{R}$ *è simmetrica, il processo di rilassamento incrementa in maniera stretta la consistenza locale media ad ogni iterazione, ovvero* $A(\boldsymbol{p}^{(t+1)}) > A(\boldsymbol{p}^{(t)})$ *fino al raggiungimento di un punto stazionario.*

Teorema 19 (T. Elfving, J. O. Eklundh, 1982 & M. Pelillo, 1997)**.** *Sia* $\boldsymbol{p} \in \mathcal{K}$ *un etichettamento strettamente consistente, allora* $\boldsymbol{p}$ *è un punto di equilibrio asintoticamente stabile per il processo di rilassamento, indipendentemente dal fatto che la matrice* $\boldsymbol{R}$ *sia simmetrica o meno.*

9.1.1 *Relaxation labelling come gioco*

Il problema dell'etichettamento consistente è equivalente a un polymatrix game dove:

- i giocatori sono gli oggetti;
- le etichette sono le strategie pure;
- gli assegnamenti pesati corrispondono alle strategia miste;
- la matrice di compatibilità è la matrice di payoff.

In questo gioco gli equilibri di Nash (stretti) corrispondono ad etichettamenti (strettamente) consistenti.

9.1.2 *Apprendimento dei coefficienti di compatibilità*

Si supponga di avere a disposizione un insieme di campioni

$$L = \{L_1, \ldots, L_N\}$$

dove L_γ è un insieme di oggetti etichettati del tipo:

$$L_\gamma = \{(b_i^\gamma, \lambda_i^\gamma) : 1 \leq i \leq n_\gamma, b_i^\gamma \in B, \lambda_i^\gamma \in \Lambda\}$$

Indichiamo con $\boldsymbol{p}^{(L_\gamma)}$ l'assegnamento non ambiguo di etichette agli oggetti in L_γ, dove:

$$p_{i\alpha}^{(L_\gamma)} = \begin{cases} 1 & \text{se } \alpha = \lambda_i^\gamma \\ 0 & \text{altrimenti} \end{cases}$$

Indichiamo inoltre con $\boldsymbol{p}^{(I_\gamma)}$ un etichettamento iniziale degli oggetti sulla base delle informazioni in L_γ e $\boldsymbol{p}^{(F_\gamma)}$ l'etichettamento prodotto dall'algoritmo di rilas-

samento con $\boldsymbol{p}^{(I_\gamma)}$ in input: si noti che $\boldsymbol{p}^{(F_\gamma)}$ non deve necessariamente essere un punto fisso del processo di rilassamento.

L'idea per ricavare $\boldsymbol{R}$ è di definire una **funzione di errore** che misuri la *perdita* subita quando si ottiene in output $\boldsymbol{p}^{(F_\gamma)}$ anziché $\boldsymbol{p}^{(L_\gamma)}$, al fine di minimizzarla. Si può ad esempio adottare una funzione **quadratica**

$$E_\gamma^{(Q)}(\boldsymbol{R}) = \sum_{i=1}^{n_\gamma} \sum_{j=\lambda_1}^{\lambda_m} (p_{ij}^{(F_\gamma)}(\boldsymbol{R}) - p_{ij}^{(L_\gamma)})^2$$

dove l'**errore totale** commesso è

$$E^{(Q)}(\boldsymbol{R}) = \sum_{\gamma=1}^{N} E_\gamma^{(Q)}(\boldsymbol{R})$$

o, in alternativa, si può usare una funzione **logaritmica** del tipo

$$E_\gamma^{(I)} = -\sum_{i=1}^{n_\gamma} \log p_{i\lambda_i^\gamma}^{(F_\gamma)}(\boldsymbol{R})$$

con errore totale

$$E^{(I)}(\boldsymbol{R}) = \sum_{\gamma=1}^{N} E_\gamma^{(I)}(\boldsymbol{R})$$

Per risolvere il problema di minimizzazione si può usare il metodo di **proiezione del gradiente** di Rosen, un'estensione del metodo di discesa del gradiente ai problemi con vincoli lineari.

9.2 TRASDUZIONE DI GRAFI

La **trasduzione di grafi** è una classe di tecniche di **apprendimento semi-supervisionato** che mirano a stimare la funzione di classificazione definita su un grafo di nodi classificati e non. L'idea è di propogare in maniera consistente l'informazione appresa nei nodi etichettati a quelli non etichettati.

Supponiamo che sia dato un dataset $D = \{D_l, D_u\}$ dove

- $D_l = \{(x_1, y_1), \ldots, (x_l, y_l)\}$ sono gli oggetti etichettati, con Y l'insieme delle etichette;
- $D_u = \{x_{l+1}, \ldots, x_n\}$ sono gli oggetti non etichettati;

con $l \ll n$.

Le relazioni tra questi oggetti sono date da un grafo pesato non orientato $G = (V, E, w)$ dove i vertici rappresentano gli oggetti e i pesi degli archi rappresentano la similarità tra le coppie di nodi.

Nel caso di matrice di adiacenza **binaria**, il problema della trasduzione può essere formulato come un **problema di soddisfacimento di vincoli** (CSP) dove:

- $V = \{v_1, \ldots, v_n\}$ è l'insieme delle variabili;
- $D = \{D_{v_1}, \ldots, D_{v_n}\}$ è l'insieme dei domini, con

$$D_{v_i} = \begin{cases} \{y_i\} & \text{se } 1 \leq i \leq l \\ Y & \text{se } l+1 \leq i \leq n \end{cases}$$

- $R = \{R_{ij} : R_{ij} \subseteq D_{v_i} \times D_{v_j}\}$ è l'insieme dei vincoli binari, dove R_{ij} contiene coppie di valori compatibili tra v_i e v_j.

Il problema della trasduzione di grafi è una generalizzazione del CSP, dove ad ogni vincolo è assegnato un peso che ne descrive il livello di confidenza.

9.2.1 *Graph transduction game*

Il **graph transduction game** è un polymatrix game in cui i giocatori sono i vertici di un grafo ed ogni arco corrisponde ad un gioco tra i due giocatori connessi.

In questo gioco ogni giocatore $i \in I$ corrisponde ad un elemento del dataset D e può scegliere una strategia dall'insieme $S_i = \{1, \ldots, c\}$, che esprime l'ipotesi di appartenenza a una certa classe, con c il numero totali di classi.

I giocatori possono essere suddivisi in due gruppi:

- i **giocatori etichettati** $I_l = \{I_{l|1}, \ldots, I_{l|c}\}$ dove ogni sottoinsieme $I_{l|k}$ contiene i giocatori consapevoli di appartenere alla classe k e dunque giocheranno sempre la loro k-esima strategia pura;
- i **giocatori non etichettati** I_u.

In questo gioco, i giocatori I_l non mirano a massimizzare il loro payoff, dal momento che hanno già scelto la loro strategia: si può dimostrare che il transduction game può essere ridotto a un gioco tra i soli giocatori I_u dove le strategie dei giocatori I_l agiscono da bias sulle scelte dei giocatori non etichettati.

Il **payoff** di un giocatore $i \in I$, se viene giocato il profilo strategico misto $\boldsymbol{x}$, è dato dalla formula:

$$u_i(\boldsymbol{x}) = \begin{cases} \sum_{j \in I_u} (\boldsymbol{A}^{ij} \boldsymbol{x}_j)_h + \sum_{k=1}^{c} \sum_{j \in I_{l|k}} a_{hk}^{ij} & \text{se } i \in I_{l|h} \\ \sum_{j \in I_u} \boldsymbol{x}_i^T \boldsymbol{A}^{ij} \boldsymbol{x}_j + \sum_{k=1}^{c} \sum_{j \in I_{l|k}} (\boldsymbol{x}_i^T \boldsymbol{A}^{ij})_k & \text{se } i \in I_u \end{cases}$$

Data la matrice pesata $\boldsymbol{W} = (w_{ij})$, la **matrice di payoff parziale** tra i giocatori i, j è $\boldsymbol{A}^{ij} = w_{ij} \boldsymbol{I}_c$.

Si noti infine che, se sono ammesse solo strategie pure, il problema della trasduzione di grafi si riduce al CSP, dove un equilibrio di Nash corrisponde al caso in cui tutti i giocatori vicini giocano la medesima strategia pura al fine di ottenere il massimo supporto.

Sperimentalmente si è verificato che si ottengono prestazioni migliori **normalizzando** la matrice dei pesi

$$\widehat{\boldsymbol{W}} = \boldsymbol{D}^{-1/2} \boldsymbol{W} \boldsymbol{D}^{-1/2}$$

con $\boldsymbol{D} = (d_{ij})$ matrice diagonale dove $d_{ii} = \sum_j w_{ij}$.

Per il calcolo dell'equilibrio di Nash si usano le **dinamiche di replicazione**: raggiunta la convergenza, si assegna al giocatore $i \in I$ l'etichetta che ha la più alta probabilità nel suo equilibrio misto.

Parte III

APPENDICE

A

RICHIAMI DI MATEMATICA

A.1 REGOLE DI DERIVAZIONE

Siano $f(x)$ e $g(x)$ funzioni reali di variabile reale x derivabili e sia D l'operazione di derivazione rispetto a x:

$$\mathrm{D}[f(x)] = f'(x) \qquad \mathrm{D}[g(x)] = g'(x)$$

- **Regola della somma**:

$$\mathrm{D}[\alpha f(x) + \beta g(x)] = \alpha f'(x) + \beta g'(x) \qquad \alpha, \beta \in \mathbb{R}$$

- **Regola del prodotto**:

$$\mathrm{D}[f(x) \cdot g(x)] = f'(x) \cdot g(x) + f(x) \cdot g'(x)$$

- **Regola del quoziente**:

$$\mathrm{D}\left[\frac{f(x)}{g(x)}\right] = \frac{f'(x) \cdot g(x) - f(x) \cdot g'(x)}{g(x)^2}$$

- **Regola della funzione reciproca**:

$$\mathrm{D}\left[\frac{1}{f(x)}\right] = -\frac{f'(x)}{f(x)^2}$$

- **Regola della funzione inversa**:

$$\mathrm{D}[f^{-1}(y)] = \frac{1}{f'(x)}$$

con:

$y = f(x) \qquad x = f^{-1}(y)$

- **Regola della catena**

$$\mathrm{D}\left[f\left(g(x)\right)\right] = f'\left(g(x)\right) \cdot g'(x)$$

A.2 INTEGRALI

Teorema 20 (Teorema Fondamentale del Calcolo Integrale (prima parte))**.** *Sia* $f\colon [a,b] \to \mathbb{R}$ *una funzione integrabile. Si definisce funzione integrale di* f *la funzione* F *tale che:*

$$F(x) = \int_a^x f(t)dt \qquad a \leq x \leq b$$

Se f *è limitata, allora* F *è una funzione continua in* $[a,b]$*.*

Se inoltre f *è una funzione continua in* (a,b)*, allora* F *è funzione differenziabile in tutti i punti in cui* f *è continua e si ha:*

$$F'(x) = f(x)$$

cioè F *risulta essere una primitiva di* f*.*

B

SISTEMI DINAMICI

Un modello matematico per descrivere le dinamiche di un sistema non lineare è quello dello **spazio degli stati**: in questo modello si pensa in termini di variabili di stato i cui valori ad un certo istante temporale sono considerati sufficienti a predire la futura evoluzione del sistema.

Sia $\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t))$ il **vettore di stato** contenente le variabili di stato di un sistema dinamico non lineare, dove la variabile indipendente è il tempo t e N è l'ordine del sistema. È possibile descrivere un largo numero di sistemi dinamici non lineari mediante un sistema di equazioni differenziali di primo ordine chiamate **equazioni dello spazio degli stati** del tipo

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \boldsymbol{F}(\boldsymbol{x}(t))$$

dove $\boldsymbol{F}(\boldsymbol{x})$ è una funzione che ritorna un vettore la cui j-esima componente è $F_j(x_j)$ con F_j una qualche funzione: questo vettore prende il nome di **vettore di velocità**.

Un sistema in cui la funzione vettore $\boldsymbol{F}$ non dipende esplicitamente dal tempo è detto **autonomo**. In questo capitolo saranno trattati solo sistemi dinamici autonomi.

L'equazione dello spazio degli stati può essere vista come il descrittore del movimento di un punto nello spazio N-dimensionale: con lo scorrere del tempo, il punto descrive una curva nello spazio degli stati detta **traiettoria** o orbita del sistema.

La famiglia delle traiettorie, per differenti condizioni iniziali, è chiamato **ritratto degli stati** del sistema e comprende tutti i punti in cui $\boldsymbol{F}(\boldsymbol{x})$ è definita: nel caso di sistemi autonomi, per ogni punto dello spazio esiste una sola traiettoria che vi passa.

B.1 CONDIZIONE DI LIPSCHITZ

Affinché l'equazione dello spazio degli stati abbia soluzione e sia unica è necessario imporre alcune restrizioni su $\boldsymbol{F}(\boldsymbol{x})$: condizione sufficiente perché ci sia soluzione è che $\boldsymbol{F}(\boldsymbol{x})$ sia continua in tutti i suoi argomenti, mentre per quanto riguarda l'unicità serve una restrizione più forte, nota come **condizione di Lipschitz**.

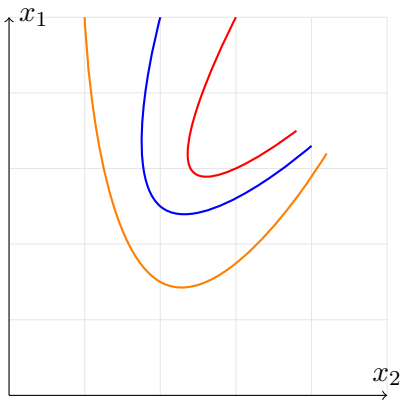


Figura 37: Ritratto degli stati di un sistema dinamico di ordine 2.

Definizione 9 (Condizione di Lipschitz)**.** *Una funzione* $\boldsymbol{F} : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$ lipschitziana *su* Ω *se esiste* $K \geq 0$ *tale per cui*

$$\|\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{F}(\boldsymbol{y})\| \leq K \|\boldsymbol{x} - \boldsymbol{y}\| \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \Omega$$

La funzione in figura 38 è lipschitziana con $K = 4$: se si considera un qualunque punto lungo la funzione e si tracciano le rette di coefficienti angolari 4 e -4 passanti per quel punto, la funzione non sarà mai confinata tra le due rette.

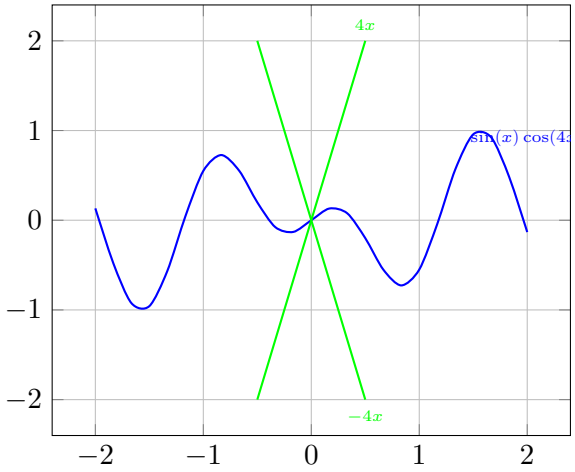


Figura 38: Interpretazione grafica della Condizione di Lipschitz.

B.2 STATI DI EQUILIBRIO

Un punto di equilibrio **stabile** è un punto che non risente delle piccole perturbazioni ovvero, se ci si sposta di poco da un punto stabile, il sistema continuerà a rimanere anche in futuro nelle vicinanze di quel punto. Viceversa un punto di

equilibrio **instabile** è tale per cui basta una perturbazione arbitrariamente piccola dall'equilibrio per far allontanare significativamente il sistema dalla posizione iniziale.

Un vettore costante $\boldsymbol{x}$ è detto stato di equilibrio (o stazionario) se è soddisfatta la seguente condizione:

$$\mathbf{F}(\boldsymbol{x}) = \mathbf{0}$$

Lo stato di equilibrio è anche detto punto singolare e la traiettoria **degenera** nel punto stesso.

Si enunciano ora una serie di definizioni sulla stabilità degli stati di equilibrio.

Definizione 10. *Lo stato di equilibrio $\boldsymbol{x}^*$ è detto* **uniformemente stabile** *se, per ogni ϵ positivo, esiste δ positivo tale per cui*

$$\|\boldsymbol{x}(0) - \boldsymbol{x}^*\| < \delta \Rightarrow \|\boldsymbol{x}(t) - \boldsymbol{x}^*\| < \epsilon \; \forall t > 0$$

Definizione 11. *Lo stato di equilibrio $\boldsymbol{x}^*$ è detto* **convergente** *se esiste δ positivo tale per cui*

$$\|\boldsymbol{x}(0) - \boldsymbol{x}^*\| < \delta \Rightarrow \lim_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}^*$$

Definizione 12. *Lo stato di equilibrio $\boldsymbol{x}^*$ è detto* **asintoticamente stabile** *se è uniformemente stabile e convergente.*

Definizione 13. *Lo stato di equilibrio $\boldsymbol{x}^*$ è detto* **globalmente asintoticamente stabile** *se è stabile e se tutte le traiettorie del sistema convergono a $\boldsymbol{x}^*$ per $t \to \infty$.*

B.3 TEOREMA DI LYAPUNOV

Definizione 14. *La funzione $V(\boldsymbol{x})$ è* **definita positiva** *nello spazio degli stati L se, per ogni $\boldsymbol{x} \in L$, soddisfa le seguenti proprietà:*

- *$V(\boldsymbol{x})$ ha derivate parziali rispetto agli elementi di $\boldsymbol{x}$ continue;*
- *$V(\boldsymbol{x}^*) = 0$;*
- *$V(\boldsymbol{x}) > 0$ se $\boldsymbol{x} \neq \boldsymbol{x}^*$.*

Si enunciano ora i due **teoremi di Lyapunov** che forniscono condizioni sufficienti per l'esistenza di un punto stazionario.

Teorema 21 (Primo teorema di Lyapunov)**.** *Lo stato di equilibrio $\boldsymbol{x}^*$ è stabile se in un piccolo intorno di $\boldsymbol{x}^*$ esiste una funzione definita positiva $V(\boldsymbol{x})$ tale che la sua derivata rispetto al tempo è minore o uguale a 0 in quella regione.*

Teorema 22 (Secondo teorema di Lyapunov)**.** *Lo stato di equilibrio $\boldsymbol{x}^*$ è asintoticamente stabile se in un piccolo intorno di $\boldsymbol{x}^*$ esiste una funzione definita positiva $V(\boldsymbol{x})$ tale che la sua derivata rispetto al tempo sia strettamente minore di 0 in quella regione.*

Una funzione scalare $V(\boldsymbol{x})$ che soddisfa queste proprietà è detta **funzione di Lyapunov** per lo stato di equilibrio $\boldsymbol{x}^*$.

Il criterio è una generalizzazione del fatto, noto nella fisica, che un sistema meccanico, se lasciato libero di evolvere, tende a portarsi in una configurazione dove la sua energia potenziale è minima. La funzione di Lyapunov può quindi essere interpretata come una funzione di energia potenziale generalizzata. Il criterio è tuttavia una condizione sufficiente ma **non necessaria**.

Inoltre non esiste un algoritmo per trovare la funzione di Lyapunov relativa a un sistema, ma si deve cercare per tentativi, basandosi sul tipo di funzione di stato e su considerazioni puramente fisiche.

BIBLIOGRAFIA

[1] Gulli Daniel Andrea Casavola. Identificazione di modelli attraverso reti neurali.

[2] Giovanna Castellano, Anna Maria Fanelli, and Marcello Pelillo. An iterative pruning algorithm for feedforward neural networks. *Neural Networks, IEEE Transactions on*, 8(3):519–531, 1997.

[3] Aykut Erdem and Marcello Pelillo. Graph transduction as a noncooperative game. *Neural Computation*, 24(3):700–723, 2012.

[4] John Hertz. *Introduction to the theory of neural computation*, volume 1. Basic Books, 1991.

[5] Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1):167–172, 2007.

[6] Marcello Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, 11(8):1933–1955, 1999.

[7] Marcello Pelillo. Replicator dynamics in combinatorial optimization. In *Encyclopedia of optimization*, pages 3279–3291. Springer, 2009.

[8] Marcello Pelillo. Slides del corso intelligenza artificiale, 2013. http://www.dsi.unive.it/~pelillo/Didattica/.

[9] Marcello Pelillo, Kaleem Siddiqi, and Steven W Zucker. Matching hierarchical structures using association graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(11):1105–1120, 1999.

[10] Samuel Rota Bulò. Appunti di reti neurali. http://www.dsi.unive.it/~srotabul/files/AppuntiRetiNeurali.pdf.

[11] Samuel Rota Bulò. Clustering con affinità asimmetriche: un approccio basato sulla teoria dei giochi. Master's thesis, Università Ca' Foscari di Venezia, 2005.

[12] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8): 888–905, 2000.

[13] Peter Norvig Stuart Russel. *Artificial Intelligence, A Modern Approach.* Pearson, third edition, 2001.

[14] Mauro Tempesta. Appunti di Mauro Tempesta, 2012.

[15] Andrea Torsello, Samuel Rota Bulò, and Marcello Pelillo. Grouping with asymmetric affinities: A game-theoretic perspective. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 292–299. IEEE, 2006.