I Big Data

A.Carini – Elettronica per l'audio e l'acustica

Big Data

- Quando un ingegnere sviluppa un sistema basato sulla voce, l'efficacia di quel sistema e le prestazioni che si otterranno dipenderanno principalmente sulla conoscenza del problema che l'ingegnere ha.
- Questa argomentazione è valida solo fino a che i dettagli rilevanti del contenuto della voce non divengono troppo complessi per essere compresi dall'uomo o finché il numero di dati da trattare non diviene più esteso dei limiti della comprensione umana.
- Raccogliendo sempre più dati di sempre maggiore complessità, alla fine le loro caratteristiche satureranno i limiti della comprensione umana.
- Qualcosa di simile è successo proprio nel riconoscimento vocale.

Big Data

- L'enorme disponibilità di dati è stata spinta in parte dalla riduzione del costo dei sistemi di data storage nelle ultime decadi:
  - 100,000€ / GB nel 1980
  - 10,000€ / GB nel 1990
  - 10€ / GB nel 2000
  - 0.1€ / GB nel 2010
- in parte dalla rapida proliferazione di sensori (dispositivi e reti).
- Applicazioni dei big data:
  - Monitoraggio ambiente (cambiamenti climatici, inquinamento),
  - Medicina, Finanza,
  - Studio comportamento sociale,
  - Studio singoli individui (pubblicità mirata, credito basato su storia individuo, head hunters risorse umane).

Big Data

- Per molte aziende oggi i dati hanno un valore superiore alle infrastrutture hardware e ai servizi usati per raccoglierli.
- Questo ha spinto alla fornitura di tanti servizi «gratuiti» (accessi liberi a internet, servizi di ricerca, posta elettronica, giochi, video, chiamate, network di amici, pagine web, …).
- Questi servizi sono gratuiti solo perché pagati con i nostri dati.
- *«se qualcosa è gratis il prodotto sei tu»*

Big Data e voce

- Alla fine dello scorso millennio, c'era nella comunità di ricerca un senso crescente che le prestazioni delle tecnologie voce e in particolare del riconoscimento vocale avessero raggiunto un asintoto.
- Nei primi anni della ricerca i progressi erano stati rapidi, ma in quegli anni bisognava lavorare sempre più duramente per ottenere prestazioni sempre più risicate.
- Sono però anche gli anni in cui si rendono disponibili enormi quantitativi di dati voce e audio registrati. La naturale risposta è stata quella di fare uso di questi dati registrati.
- Sono state pertanto utilizzate le tecniche di *machine learning* per analizzare le registrazioni voce e audio e per inferire regole e modelli legati alle loro caratteristiche.

Big Data e voce

- E’ cambiato il modo di fare ricerca: invece che basarsi sulla comprensione del ricercatore sulla voce o sull’udito, si basa ora principalmente sulla qualità e quantità dei dati raccolti e sull’abilità delle tecniche computazionali di imparare dai dati (machine learning).
- Divennero sempre più importanti la raccolta di dati e le tecniche di elaborazione dei grandi dati, ma il risultato fu un nuovo significativo passo in avanti nel miglioramento delle prestazioni.

Trend attuale nella ricerca voce

- Viene inventata una nuova idea/tecnica in ambito voce o audio.
- La comprensione del problema e dei relativi dati porta alla definizione di regole che consentono lo sviluppo di applicazioni.
- I miglioramenti sono inizialmente spinti da una sempre maggiore comprensione del problema e dei dati. Le prestazioni dapprima aumentano rapidamente, poi rallentano, per arrivare alla fine a un asintoto.
- Per risolvere il problema si sfruttano i grandi dati combinati con tecniche di machine learning e il progresso ricomincia.
- Migliori tecniche di machine learning, maggiori e migliori dati sono usati per migliorare le prestazioni.
- Alla fine anche le tecniche di machine learning raggiungono l'asintoto.

The rationale behind big data

- L’idea è che maggiori sono i dati disponibili (buoni e rappresentativi) e meglio possiamo sfruttarli.
- Esempio: iFlytek «voice cloud».
- Nel 2010, iFlytek ha lanciato un servizio voce in cloud che fornisce funzionalità di speech recognition per molti servizi (voice assistant, sistemi di risposta automatica telefonici, etc.), localizzati nel mondo ma in particolare in Cina.
- Il sistema voce in cloud riceve una registrazione voce, riconosce le parole dette e restituisce il relativo testo in breve tempo (100ms).
- Il sistema è basato sui big data con i modelli di riconoscimento che sono aggiornati periodicamente sulla base della correttezza del sistema.
- Circa 500 trascrittori sono impegnati a tempo pieno per ascoltare le registrazioni e decidere cosa era stato detto. In caso di errori di riconoscimento della macchina, la trascrizione viene usata per correggere/adattare i modelli.

The rationale behind big data

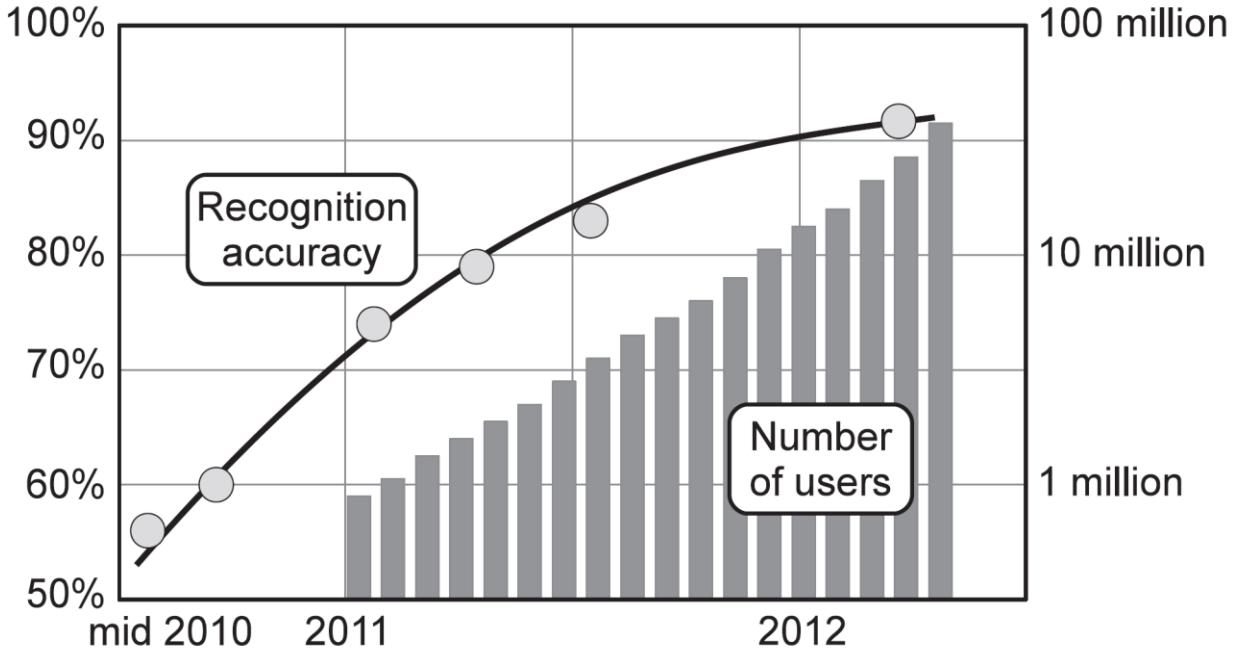


Figure 8.1 Combined plot showing approximate speech recognition accuracy and number of users of the iFlytek voice cloud, plotted using publicly released data.

The rationale behind big data

- I big data sono estratti da più di 1,000,000 di registrazioni voce ricevute ogni giorno.
- L'accuratezza del sistema sta raggiungendo un asintoto.
- L'aspetto più interessante è che il sistema ha già superato in molti scenari il livello di accuratezza dell'uomo.
- Può operare su un maggior numero di dialetti, su voce colloquiale e in ambienti più rumorosi di quelli trattabili dall'uomo medio, con un accuratezza superiore al 90%.

Componenti che accompagnano i big data

- Il data storage (potrebbe richiedere la quantizzazione o la semplificazione dei dati se i requisiti sono troppo grandi).
- L'accesso ai dati (in quale forma sono salvati e organizzati).
- La trasformazione da dati raw a feature rappresentative.
- Una o più tecniche di machine learning.
- Una metodologia di valutazione che assegni degli *score* ai risultati.

Data storage e accesso ai dati

- Ovviamente i big data richiedono grandi risorse di storage.
- Molti database pubblici hanno dimensioni dell'ordine dei gigabyte, quelli commerciali hanno dimensioni dell'ordine di terabyte o superiori.
- I dati raw vengono ridotti di dimensione prima della memorizzazione, mediante trasformazione in feature rappresentative o mediante trasformazione in un formato intermedio ridotto.
- La riduzione delle informazioni comprende:
  - L'eliminazione delle parti inutili (e.g. del silenzio).
  - L'eliminazione di registrazioni rovinate o di bassa qualità,
  - Il cambiamento di formato attraverso quantizzazione, aggiustamento del sample rate, o la compressione (pericolosa: potrebbe influenzare l'efficacia dell'algoritmo di machine learning).
- Sono trasformazioni irreversibili, per questo vengono salvati anche i dati raw.

Data storage e accesso ai dati

- I dati devo essere *accessibili*. Questo comporta problematiche di indicizzazione e di archivio dei dati, di sicurezza e di connettività fisica.
- Si assume che i dati vengano salvati su qualche supporto, ma i vari supporti hanno velocità e abilità di accesso molto diverse.
- Specie quando i dati sono trattati da un sistema multi-processore o multi-macchina, potrebbe essere necessario garantire che i dati siano salvati il più possibile vicino alla macchina che li leggerà, per minimizzare i costi e i colli di bottiglia dovuti al trasferimento dei dati.
- Questo potrebbe richiedere di spezzare un grande database in blocchi distribuiti tra un certo numero di macchine.
- La larghezza di banda è finita e si vuole evitare che un sistema sia rallentato da processi in attesa di ricevere i dati.

Features

- I dati raw, come l'audio registrato, sono in genere di grandi dimensioni. Devono essere trasformati in un formato di minore dimensione, mediante feature più concentrate, prima di essere trattati da un algoritmo di machine learning.
- Queste feature sono una rappresentazione a minore dimensione degli aspetti più importanti dei dati.
- Non tutti i dati raw sono utili (e.g., silenzi all'inizio e alla fine, ripetizioni nella frequenza). Dovremo rimuovere il più possibile le parti inutili.
- Gli algoritmi di machine learning sono generalmente addestrati su dati rappresentativi prima di poter essere usati per una classificazione.
- L'addestramento richiede l'aggiustamento di diversi parametri interni (pesi, bias) aggiornati in modo incrementale.

Features

- I dati concentrati delle feature hanno anche il vantaggio di contenere minori informazioni inutili che potrebbero confondere gli algoritmi di machine learning.
- Se è possibile trasformare i dati raw in una forma più concentrata e di minore dimensionalità, gli algoritmi di machine learning convergono meglio, più velocemente e con maggiore accuratezza.
- Come conseguenza, minori dati saranno richiesti per l'addestramento e il risultato finale sarà meglio addestrato.

Features

Table 8.1 Example features and where they might be used.

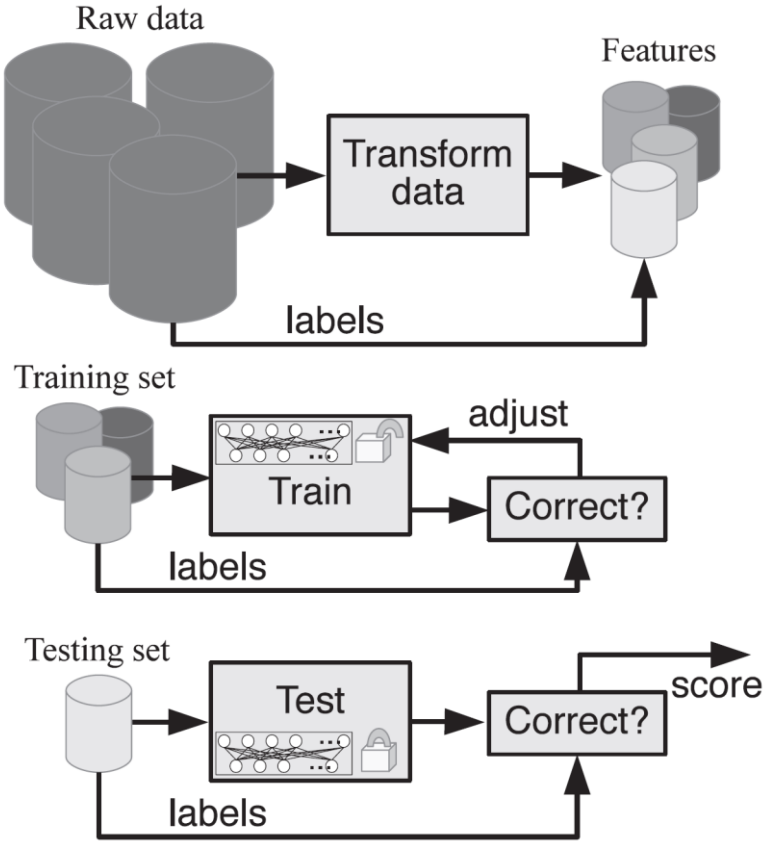
Feature	Dimensionality	Used where
MFCCs	Usually 13	ASR, language/speaker ID, sound detection, diarisation
ΔMFCCs	With MFCCs	– as above –
ΔΔMFCCs	With the above two	– as above –
Pitch	Usually 1	To accompany MFCCs
Δpitch	& ΔΔpitch	– as above –
Energy/power	Usually 1	Commonly used
LPCs	Usually 10 or 12	ASR and similar
LSPs	– as above –	ASR and similar
Filterbank	Up to 40	Sound detection, diarization
Spectrogram	Two-dimensional	Sound detection, VAD
Auditory image	Two-dimensional	Sound detection
i-vector	200∼1024	Diarization, language and speaker ID

Features

- Molti dei dati della precedente tabella sono usati in combinazione, per esempio il pitch e l'energia sono usati in combinazione ai MFCC.
  - I MFCC sono spesso calcolati su frame normalizzate.
  - L'energia rappresenta l'ampiezza assoluta nel frame.
  - Il pitch rappresenta la ripetizione del suono sul lungo periodo.
- Ciascuna feature ha anche una naturale durata su cui viene calcolata.
- Tutte le feature, a parte le ultime tre, sono calcolate su di un frame.
- Lo spettrogramma ha tre informazioni temporali: la durata dei frame di analisi, il passo di avanzamento o l'overlap tra i frame, il numero di frame.

- [L'i-vector è una feature che rappresenta elementi di dimensione/durata diversa con una feature di dimensione costante.]

Machine learning



Machine learning

- Dapprima un grande quantitativo di dati sono trasformati in feature.
- Le feature sono divise almeno in due blocchi.
- Un blocco di dati (in genere il più numeroso) viene usato per il training del sistema di machine learning.
- Durante il training le feature sono passate in ingresso al sistema che cerca di classificarle o identificarle.
- Per ciascun ingresso, il sistema dà in uscita la sua stima che viene confrontata con la label corretta.
- La differenza tra valore corretto e quello predetto viene usata per aggiustare i parametri interni del sistema di machine learning.
- Quando tutti i «training data» sono stati usati (o quando il tasso di errore è diventato sufficientemente piccolo), i parametri del modello vengono fissati e il blocco dati di test viene usato per valutare le sue prestazioni.

Machine learning

- Ci sono molte varianti di sistemi di machine learning:
- *Supervised learning*: classificatori che cercano di classificare i nuovi dati in ingresso in classi con label. Vengono addestrati con dataset di dati con label.
- *Unsupervised learning:* lavorano su dati senza label e generano le proprie classi entro cui classificare i dati.
- Ci sono però molte altre tecniche come *semi-supervised learning, transfer learning, reinforcement learning.*

Machine learning

- Ci sono tanti motivi per cui un sistema di machine learning può fallire:
  - Dati di training insufficienti.
  - Dati di training eccessivi.
  - Feature troppo semplicistiche.
  - Feature troppo complesse.
  - Dati mal etichettati.
  - Dati di training non rappresentativi.
  - Medie o distribuzioni delle feature alterate.
  - I dati non sono distribuiti in modo normale (Gaussiano).
  - Modello incorretto.
  - Un problema mal posto.

Machine learning

- I sistemi di machine learning tendono ad avere una moltitudine di parametri di configurazione.
  - learning rate, batch size, pesi iniziali, bias, scelta della funzione nonlineare, misure della distanza, ….
- Spesso questi parametri vengono scelti sulla base di una procedura *trial and error* o sulla base dell'esperienza passata del ricercatore.
- La procedura *trial and error* dovrebbe usare il *tranining set* o il *test set* ?
- Con il primo, ci adattiamo troppo ai dati di training e il test potrebbe dare risultati inaspettati; con il secondo, ci adattiamo troppo al test set e i risultati potrebbero non essere rappresentativi.
- Soluzione: spesso viene previsto un terzo set di dati, il *development data set.*

Evaluation methodology

- Altro aspetto problematico è rappresentato dal metodo di assegnazione dello score per decidere qual è la migliore tecnica.
- Anche un semplice task come determinare la presenza o meno della voce può essere un problema.
- Per esempio: 10 minuti di registrazione di cui 1 di parlato e 9 di silenzio.
- Due rivelatori:
  1. Rileva correttamente l'80% della voce e confonde il 25% del silenzio.
     - Accuratezza nel tempo: (0.8*1+0.75*9)/10 = 0.755.
  2. Dice sempre silenzio in ogni condizione.
     - Accuratezza nel tempo: 0.9 !!!

Il Percettrone

- Un percettrone è un classificatore binario: consente di dividere due diverse classi di informazione purché siano separabili da una retta o da un piano, ovvero purché siano linearmente separabili.
- L'algoritmo è stato ideato nel 1950. E' alla base della multi-layer neural network e di molte tecniche di machine learning moderne (convolutional neural network).
- La struttura del percettrone è ispirata a quella di una cellula nervosa umana:
  - Prende un certo numero di sorgenti (le uscite di altre cellule, o organi di senso – in machine learning le chiameremo feature vector).
  - Pesa ciascun ingresso prima di sommare tutti i termini e aggiungere un bias.
  - Usa una funzione di sogliatura per decidere se attivare l'uscita.
  - Durante l'addestramento, i pesi sono iterativamente aggiornati con l'effetto di muovere il confine della classificazione.
  - Fissati i pesi, il sistema può essere usato per classificare nuovi dati.

Il Percettrone

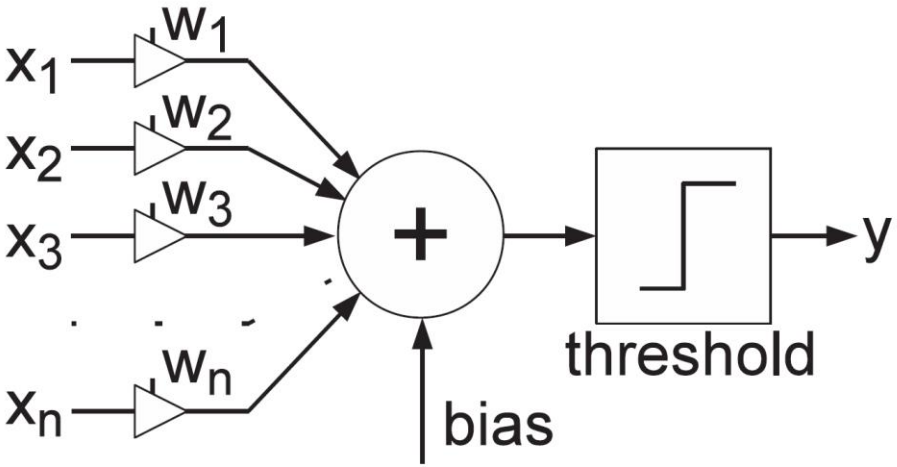


Figure 8.3 Illustration of a single perceptron gathering information from n inputs, using a set of weights and bias to produce a single output, y.

Il Percettrone

- Da un punto di vista matematico:
  - Dato un vettore di ingresso $\mathbf{x}_n = \{x_0, x_1, \ldots, x_N\}$ e un singolo segnale binario d'uscita *y,* la struttura usa *N* pesi $\mathbf{w}_n = \{w_0, w_1, \ldots, w_N\}$ e un bias *b* per calcolare

$$y = \varphi\left\{\sum_{l=1}^{N} w_l x_l + b\right\} = \varphi\left\{\mathbf{w}^{\mathrm{T}}\mathbf{x} + b\right\},$$

  - Dove φ è una funzione di sogliatura, in genere derivabile, e.g., la funzione logistica,

$$\varphi a = 1/(1 + e^{-2a}).$$

Il Percettrone

- Vogliamo minimizzare su tutto il *traning set* la speranza matematica del quadrato dell'errore tra l'uscita *y* ed il relativo valore desiderato *d:*

$$J = E[e^2] = E[(d-y)^2]$$

- Adattiamo il coefficienti applicando la regola del gradiente: feature vector per feature vector ci muoveremo in direzione opposta al gradiente:

$$w_{i+1} = w_i - \frac{\mu}{2}\frac{\partial J}{\partial w} = w_i - \frac{\mu}{2}\frac{\partial E[(d-y)^2]}{\partial w}$$

- Approssimando la derivata della speranza matematica con la derivata del valore istantaneo

$$w_{i+1} = w_i - \frac{\mu}{2}\frac{\partial (d-y)^2}{\partial w} = w_i + \mu\frac{\partial y}{\partial w}e = w_i + \mu\frac{\partial \varphi(a)}{\partial a}\frac{\partial a}{\partial w}e = w_i + \mu\frac{\partial \varphi(a)}{\partial a}x\,e$$

$(a = w^T x + b)$

$$\boxed{w_{i+1} = w_i + \mu\frac{\partial \varphi(a)}{\partial a}x\,e}$$

Il Percettrone

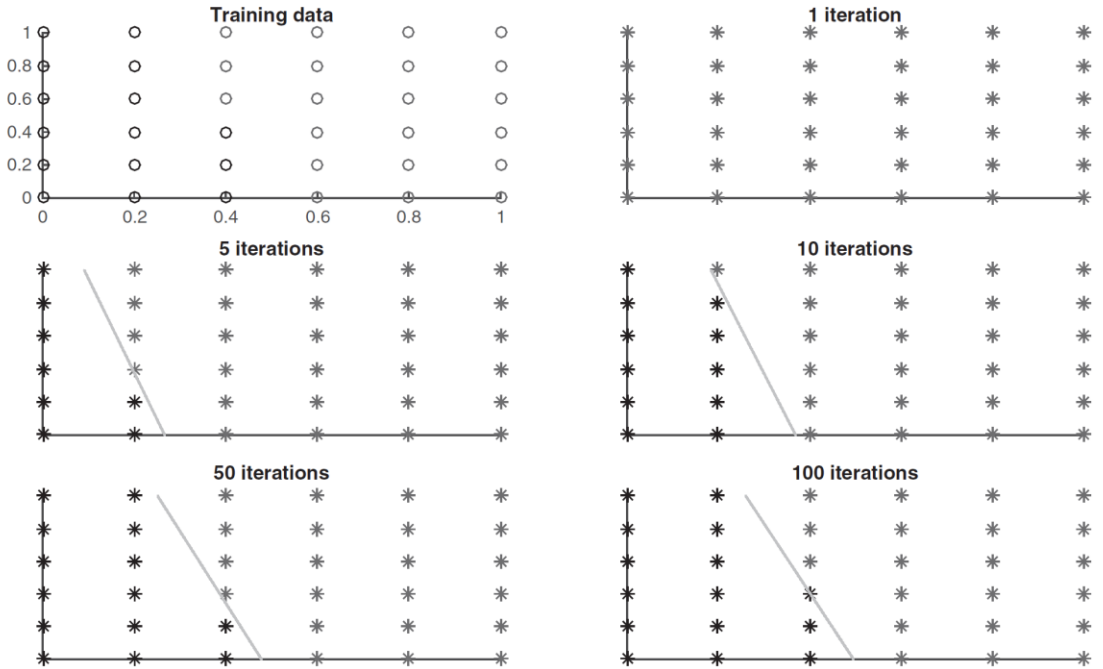


Figure 8.4 Illustration of a perceptron learning to classify two-dimensional data. The training data is plotted top left, and the classifier output data shown after 1, 5, 10, 50 and 100 iterations respectively, with class 1 symbols plotted in black and class 0 symbols plotted in grey.

Il Percettrone

```
%Make a 6x6 square matrix of 2D training data
xin=[[reshape(repmat([0:0.2:1],6,1),[1,6*6])];[repmat
    ([0:0.2:1],1,6)]]';
N = length(xin);
%most are class 0, except a few at lower left
yout=zeros(length(xin));
yout(1:12)=1;
yout(13:15)=1;
```

Il Percettrone

```
%Define a 2D perceptron
b = -0.5;
nu = 0.5;
w = 0.5*rand(3,1); %-1*2.*rand(3,1);
iterations = 100;
%Loop around the training loop
for i = 1:iterations
     for j = 1:N
y = b*w(1,1)+xin(j,1)*w(2,1)+xin(j,2)*w(3,1);
          z(j) = 1/(1+exp(-2*y));
          delta = yout(j)-z(j);
          w(1,1) = w(1,1)+nu*b*delta;
          w(2,1) = w(2,1)+nu*xin(j,1)*delta;
          w(3,1) = w(3,1)+nu*xin(j,2)*delta;
     end
end
```

Multi-layer perceptron (MLP)

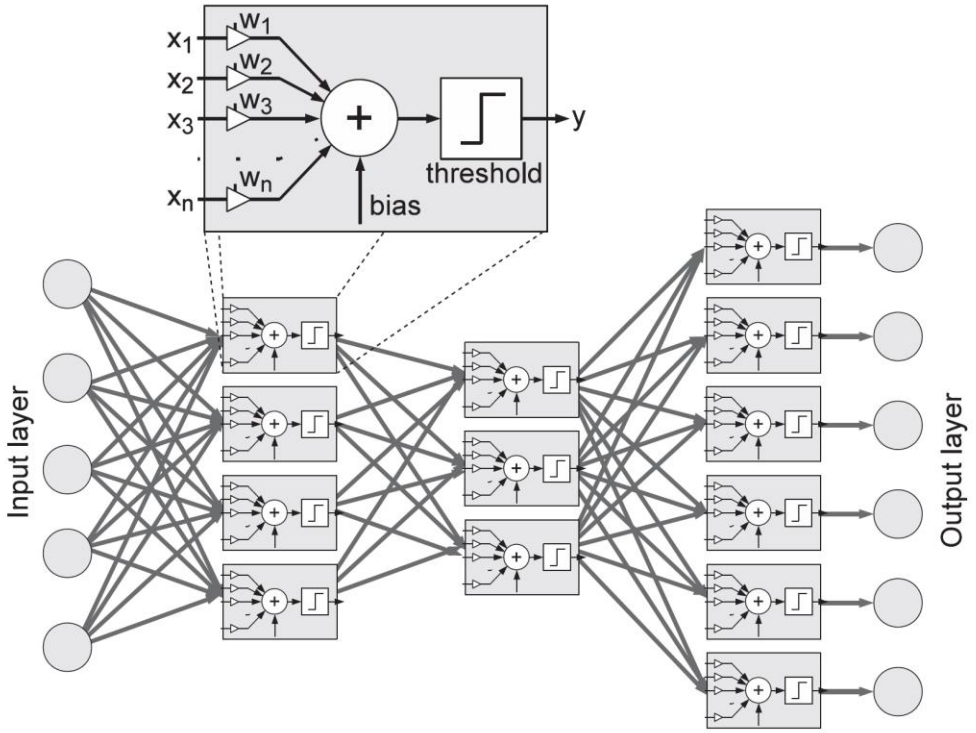


Figure 8.5 Illustration of a multi-layer perceptron network with five-dimensional input data, three internal layers (comprising four, three and six perceptron nodes respectively) and six outputs.

Multi-layer perceptron (MLP)

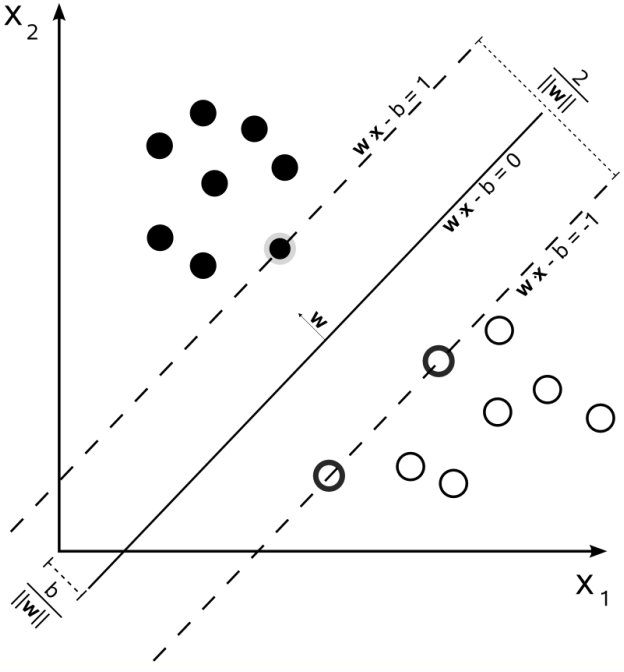
- E’ una struttura che consiste in diversi layer di percettroni.
- Con questa struttura è possibile imparare delle classificazioni più complesse della una semplice classificazione lineare.
- Viene addestrata ancora con un algoritmo a gradiente, l’algoritmo di *back-propagation,* che consente di adattare i pesi e i bias lavorando layer per layer dall’uscita all’ingresso.
- I pesi sono aggiornati ancora sulla base dell’errore tra quello che dovrebbe produrre la rete e quello che produce realmente.

Support vector machine

- Nascono ancora come dei classificatori che cercano di separare i dati in due classi (con i dati etichettati con +1 o -1).
- Si cerca di massimizzare la distanza tra le classi e il confine.
- I vettori più vicini sono detti *support vector*.
- Dato un vettore di D elementi

$$\mathbf{x} = [x_1, x_2, ..., x_D]^T$$

- a questo sarà assegnata una delle due classi.

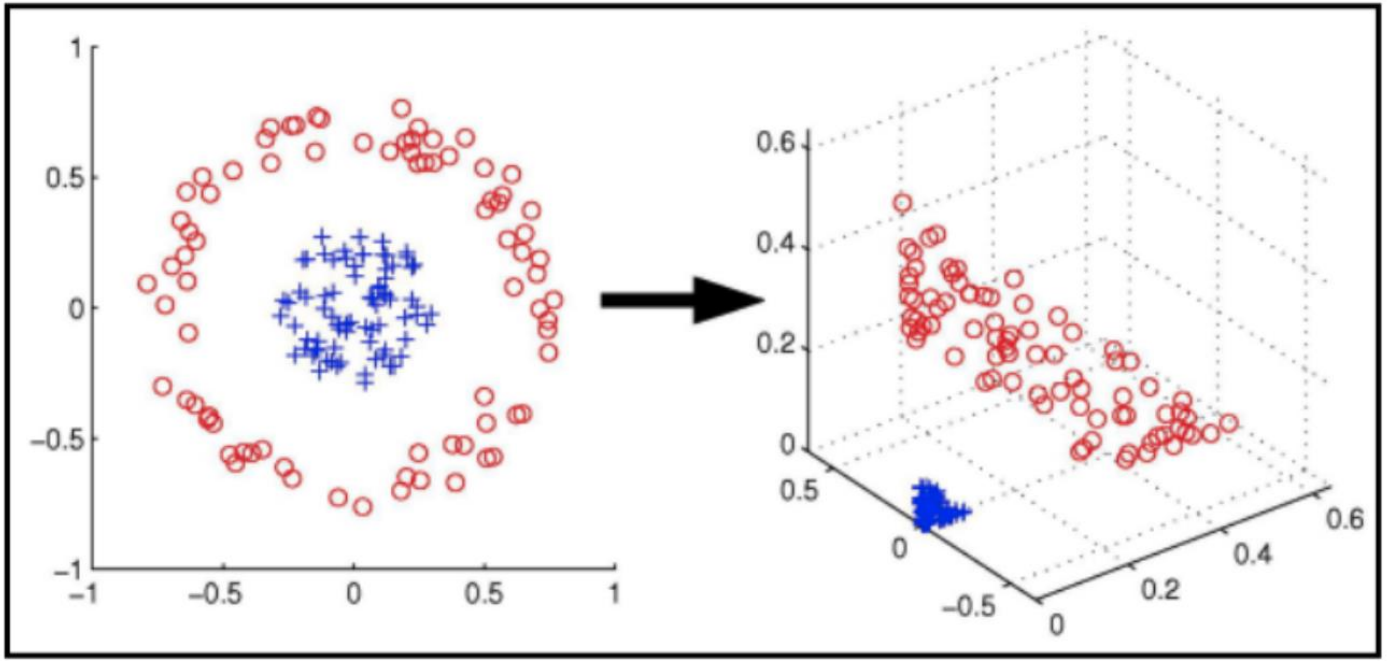


Support vector machine

- Le support vector machine applicano però un trucco per separare cluster sovrapposti o non separabili da un piano, il *kernel trick*:
- Mappano il vettore di ingresso in uno spazio di maggiore dimensione con una trasformazione nonlineare.
- Maggiore la dimensione dello spazio e più facilmente le due classi risulteranno essere separabili.

- E’ anche possibile applicare il metodo a *K* classi, ma avremo bisogno di *K(K-1)/2* classificatori per rappresentare tutte le divisioni in classi.

Support vector machine



Principal component analysis (PCA)

- La PCA è un metodo per determinare l'informazione portata da diverse variabili e per trasformarla.
- Nel dominio speech e audio, la PCA è in genere applicata a *feature vector* di grande dimensione per ridurre la dimensionalità delle feature eliminando quell'informazione che è dipendente o ripetuta tra le feature.
- Si noti che non si vogliono eliminare delle feature ovvero degli elementi dei vettori (*feature selection*), ma si vuole compattare la maggior parte dell'informazione in pochi elementi.
    - [Ridurre la dimensionalità dei feature vector prima di una classificazione agevola l'addestramento]
- La tecnica è *data-driven*, ovvero non ha bisogno di conoscere nulla delle feature o del loro significato.

Principal component analysis (PCA)

- Il metodo PCA applica una trasformazione lineare alle feature che proietta le feature in un nuovo sistema cartesiano dove la feature di maggiore varianza è proiettata sul primo asse, quella di varianza immediatamente più piccola sul secondo, e così via proseguendo.
- Le feature di questo nuovo spazio che hanno minore varianza portano minore informazione.
- La riduzione delle feature viene ottenuta prendendo solo le feature principali.

- Matematicamente, il procedimento parte dai vettori di feature.
- Si suppone che ciascuna feature abbia media nulla, altrimenti possiamo sottrarre la media (valori costanti non portano informazione).

Principal component analysis (PCA)

- Dati i vettori di feature x_i (di dimensione $1 \times L$), formiamo la matrice X che ha come righe gli x_i, (con X di dimensione $\mathrm{N} \times L$).
- Calcoliamo la matrice di covarianza

$$\Sigma = \frac{1}{N} X^T X$$

- (dimensione $L \times L$), una matrice simmetrica definita positiva.
- Troviamone gli autovalori e autovettori: $\Sigma\, \bar{V} = \bar{V}\, \bar{D}$ (ovvero $\Sigma = \bar{V} \bar{D} \bar{V}^T$)
- Possiamo assumere che gli autovalori in $\bar{D}$ siano ordinati in modo decrescente.
- Prenderemo V formato dalle prime L' colonne di $\bar{V}$.
- La trasformazione che applicheremo ai dati sarà: $X' = X \cdot V$
  - [$\bar{V}$ si può considerare una matrice di rotazione]
- Determinata V su un certo numero di feature vector, la potremo applicare a nuovi vettori di feature vector.

Independent component analysis (ICA)

- L'ICA è una tecnica applicata a dati che si assumono contenere una combinazione di componenti, generate da processi non Gaussiani.
- La tecnica identifica e quindi separa questi processi indipendenti.
- Una tipica applicazione è la separazione della voce dal rumore di sottofondo, o anche la separazione di due parlatori da una registrazione che contiene una combinazione dei due.

Independent component analysis (ICA)

- Considereremo il problema del cocktail party, dove si vuole ricostruire la voce originale di diversi parlatori che siano stati registrati con diversi microfoni.
- L'ingresso dell'algoritmo è un insieme di *M* segnali prelevati dai microfoni dai quali si vogliono recuperare le *N* voci, con $\mathrm{M} \geq N$.
- Se i microfoni sono posti in posizioni diverse riceveranno diverse combinazioni delle voci.
- Assumeremo di trascurare l'effetto del riverbero e i diversi ritardi dei segnali.
- Assumeremo anche che i segnali da recuperare siano incorrelati.
- Dobbiamo anche assumere che non siano Gaussiani, altrimenti non è possibile recuperarli (ipotesi che può essere considerata vera per la voce).
- Vedremo un metodo iterativo senza dimostrarlo.

Independent component analysis (ICA)

- L'ICA lavora su *M* segnali ricevuti ciascuno di lunghezza *T*, posti in una matrice di *mixture* X di dimensioni $M \times T$.
- L'ipotesi è che questi derivino da *N* sorgenti separate e siano stati ottenuti con una combinazione lineare.
- Se A rappresenta la matrice di *mixing* ($M \times N$) e S la matrice di sorgenti ($N \times T$), $X = A \cdot S$
- Dobbiamo recuperare la matrice S nell'ipotesi che $\mathrm{M} \geq N$.
- Si noti che potremo solo approssimare S e che non potremo recuperare l'ampiezza dei segnali originali, né l'ordine dei segnali.
- Ci sono vari metodi per risolvere il problema, tutti abbastanza complessi.
- Alcuni minimizzano la mutua informazione, altri massimizzano la non-Gaussianità dei segnali ricostruiti.
- Sia W la matrice di *unmixing* ($N \times M$) tale che $S' = W \cdot X$.

Independent component analysis (ICA)

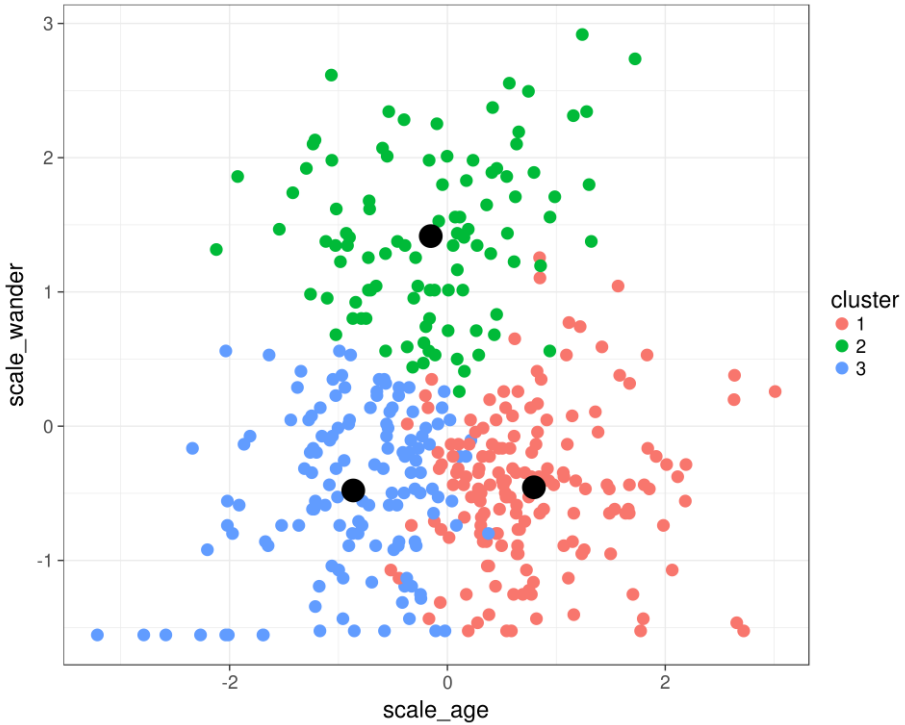
- McLoughlin fa uso di un algoritmo di tipo gradient descent che cerca iterativamente la soluzione a partire da una matrice W inizializzata aleatoriamente, e considera $g(x)$ una funzione distanza nonlineare.

1. Compute $\mathbf{S}' = \mathbf{WX}$, an estimate of the unknown source signals.
2. Find the overall distance, $Z_{n,t} = g(y_{n,t})$ for $n = 1, \ldots, N$ and $t = 1, \ldots, T$.
3. Define $\Delta\mathbf{W} = \eta(\mathbf{I} + (1 - 2\mathbf{Z})\mathbf{S}'^{\mathrm{T}})\mathbf{W}$.
4. Update $\mathbf{W} = \mathbf{W} + \Delta\mathbf{W}$.
5. Repeat from step 1 until either the estimate has converged (which can be found from the magnitude of $\Delta\mathbf{W}$) or a maximum number of iterations has been reached.

K-means

- Supponiamo che i dati raccolti siano generati da diversi processi a distribuzione Gaussiana e che sia possibile dedurre o indovinare il numero di processi coinvolti.
- Possiamo usare il *K-means clustering algorithm* per trovare *K* centri (anche detti *centroidi*) del cluster e quindi assegnare ciascun elemento al cluster opportuno.
- In particolare, si minimizza la somma delle distanze dei dati dal rispettivo centro del cluster.

K-means



K-means

- L'algoritmo di clustering K-means applica un approccio iterativo.
- Inizia da delle posizioni aleatorie per i centroidi, e quindi esegue in loop un processo che dapprima assegna ciascun punto al più vicino centroide del cluster, per poi aggiornare i centroidi per la prossima iterazione sulla base della media di tutti i punti assegnati a ciascun cluster.
- L'algoritmo in genere converge molto rapidamente.
- Dato un dataset, che può avere qualunque dimensione, vengono ricavati *K* centroidi m_i dove ciascun centroide ha la stessa dimensione dei dati:

$$||\mathbf{x} - \mathbf{m}_i|| = \min_j ||\mathbf{x} - \mathbf{m}_j||.$$

- Il metodo definisce una funzione obbiettivo globale da minimizzare in ciascuna iterazione:

$$R = \begin{cases} \sum_t \sum_i ||\mathbf{x} - \mathbf{m}_i||^2 & \text{if } i = \text{argmin}_j ||\mathbf{x} - \mathbf{m}_j||, \\ 0 & \text{otherwise.} \end{cases}$$

K-means

```
function centroids = kmeans(X,k)
%set error threshold
min_thresh=1e-7;
%set maximum iterations
max_iter=10000000;
%centroids
centroids = zeros(k, 2);
len = size(X,2);
nearest_c = zeros(len);
%initialise to random points
rand_i = ceil(len*rand(k, 1));
for i = 1:k
    centroids(i,:) = X(:,rand_i(i));
end
%Iteration loop
for i=1:max_iter
```

K-means

```
17  %updated means
18  new_c = zeros(size(centroids));
19  %no, of points assigned to each mean
20  assigned2c = zeros(k, 1);
21  %Go through all data points
22  for n=1:len
23     % Calculate nearest mean
24     x = X(1, n);
25     y = X(2, n);
26     diff = ones(k,1)*X(:,n)' - centroids;
27     dist = sum(diff.^2, 2);
28
29     [~,indx] = min(dist);
30     nearest_c(n) = indx;
31     new_c(indx, 1) = new_c(indx, 1) + x;
32     new_c(indx, 2) = new_c(indx, 2) + y;
33     assigned2c(indx) = assigned2c(indx) + 1;
34  end
```

K-means

```
36    %Compute new centroids
37    for i = 1:k
38       %Only if a centroid has data assigned
39       if (assigned2c(i) > 0)
40           new_c(i,:) = new_c(i,:) ./ assigned2c(i);
41       end
42    end
43
44    %Early exit if error is small
45    d = sum(sqrt(sum((new_c - centroids).^2, 2)));
46    if d < min_thresh
47        break;
48    end
49    centroids = new_c;
50 end
```

K-means

- L’algoritmo di clustering K-means è molto utile in un certo numero di applicazioni di machine learning per la voce o altri domini.
- E’ semplice da implementare e relativamente efficiente in quanto converge rapidamente.
- Non è garantita la convergenza al minimo globale, potrebbe convergere a un minimo locale.
- Usa una varianza fissa per ciascun cluster, anche se questa in realtà potrebbe essere diversa.
- Possiamo superare questo limite con le GMMs.

Gaussian Mixture Models (GMMs)

- Nelle GMMs si assume che i dati siano generati da diverse funzioni continue con diversa varianza e diversa media.
- Questi dati saranno modellati da combinazioni (*mixture*) di Gaussiane, con ciascun punto che viene generato da una componente del mixture con una certa probabilità.
- Come il K-means, il GMMs inizia con una stima iniziale e quindi iterativamente assegna i dati alle diverse componenti del *mixture* e stima di nuovo le componenti del mixture sulla base dell'allocazione dei dati, lavorando alla ricerca di una soluzione ottima.
- Abbiamo *K* Gaussiane $\mathcal{N}_k$, ciascuna con una sua media μ_k e una matrice di covarianza Σ_k:

$$\mathcal{N}_k(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_k|}} e^{-\frac{1}{2}(x-\mu_k)^{\mathrm{T}} \Sigma_k^{-1} (x-\mu_l)},$$

Gaussian Mixture Models (GMMs)

- Possiamo calcolare la probabilità che il dato x_j appartenga al cluster *k*, w_k^j, e possiamo usare questa probabilità per una assegnazione *soft* (probabilistica) dei punti al cluster:

$$w_k^j = \frac{\mathcal{N}_k(x_j)\phi_k}{\sum_{l=1}^{K}\mathcal{N}_l(x_j)\phi_l}$$

- dove ϕ_k è la probabilità a priori del componente *k*
- (w_k^j è la verosimiglianza che x_j sia stato generato dal componente *k*).
- [La formula deriva da Bayes – è un Expectation maximization algorithm:

w_k^j = p(k | x_j)

$\mathcal{N}_k(x_j)$ = p(x_j | k)

$\phi_k = p(\ k\ )$

$\sum_{l=1}^{K}\mathcal{N}_l(x_j)\phi_l = p(x_j)$]

Gaussian Mixture Models (GMMs)

- Dati i valori di assegnazione soft w_k^j, il passo successivo esegue una nuova stima dei componenti Gaussiani $\phi'_k, \mu'_k, \Sigma'_k$

$$\phi'_k = \frac{1}{m}\sum_{i=1}^{m} w_k^i,$$

$$\mu'_k = \frac{\sum_{i=1}^{m} w_k^i x^i}{\sum_{i=1}^{m} w_k^i},$$

$$\Sigma'_k = \frac{\sum_{i=1}^{m} w_k^i (x^i - \mu_k)(x^i - \mu_k)^{\mathrm{T}}}{\sum_{i=1}^{m} w_k^i}.$$

- Vengono quindi fatte iterativamente delle nuove assegnazioni soft seguite da stime sino a raggiungere la convergenza o il massimo numero di iterazioni.

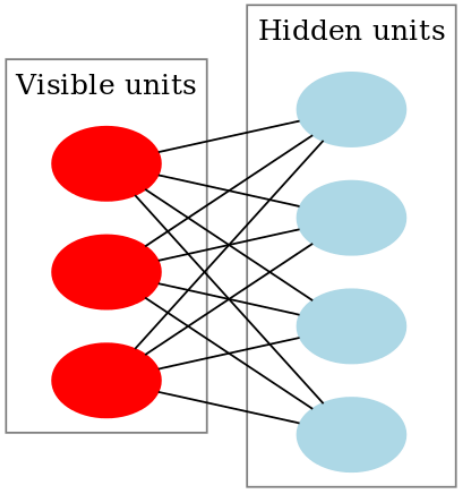
Gaussian Mixture Models (GMMs)

- La stima con GMMs è più complessa e lenta del K-means ma ottiene migliori risultati.
- Le prestazioni dipendono dai dati iniziali e pertanto spesso si usa il K-means per inizializzare la tecnica GMM, in modo da avere valori iniziali buoni.

Deep Neural Networks (DNNs)

- Una DNN è molto simile a una multi-layer perceptron network, ma con un numero molto elevato di layer (> 4).
- Nelle prime DNN proposte veniva fatto un addestramento per layer dall'ingresso all'uscita e quindi un fine-tuning in direzione inversa con l'algoritmo di back-prop.
- Il training per layer nella direzione in avanti, con il tuning all'indietro consente di realizzare strutture con tanti strati (deep).
- Questa DNN tendono a lavorare bene nell'ottenere i dettagli fini dei vettori di feature rappresentativi, ovvero sono in grado di lavorare bene con feature di ingresso a grande dimensionalità.
- Un classificatore DNN a *L* layer viene costruito in modo da produrre *K* classi d'uscita e inizia con un input layer a cui viene applicato il vettore di feature di ingresso.
- La DNN viene costruita per strati dall'ingresso all'uscita facendo uso di coppie pre-addestrate di *Restricted Boltzman Machine* (RBM).

Deep Neural Networks (DNNs)



- Le RBM sono delle reti con *V* nodi visibili e *H* nascosti.
- Vengono in genere usati due tipi di RBM:
  - Bernoulli-Bernoulli (tutti i nodi sono binari)
  - Gaussian-Bernoulli (nodi visibili reali e nascosti binari)
- Infatti, l'ingresso della DNN comprende un vettore di dati reali, mentre il layer d'uscita è una classificazione binaria.
- I layer intermedi e finali sono Bernoulli-Bernoulli mentre quelli di ingresso sono Gaussian-Bernoulli.
- Come nelle MLP, i nodi hanno pesi e bias e c'è piena connettività tra i nodi visibili e quelli nascosti.
- Le RBM possono imparare la distribuzione probabilistica dei vettori di ingresso.
- Vengono addestrate per massimizzare il prodotto delle probabilità assegnate ad alcuni training vector.

Deep Neural Networks (DNNs)

- Viene dapprima addestrato lo strato di ingresso. Dopo il suo training, lo stato inferito della hidden unit 1 diviene il dato visibile per il training della successiva RBM.
- L'uscita è costituita da una serie di moltiplicatori (softmax output labeling layer) che scalano l'uscita della rete per ottenere la probabilità di ciascuna classe *K.*

CD contrastive divergence

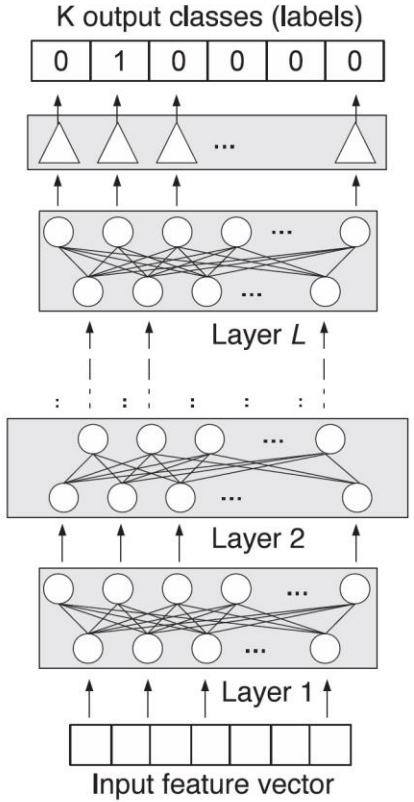


Figure 8.9 A DNN is constructed from a deep stack of Gaussian–Bernoulli and Bernoulli–Bernoulli RBMs, and is trained from the bottom up using the CD algorithm.

Deep Neural Networks (DNNs)

- L'algoritmo di back-propagation è applicato a ritroso su tutti i layer (incluso il softmax layer), per minimizzare una opportuna funzione d'errore.
- Durante il training vengono applicati diversi parametri (dimensioni dei layer, dropout-proporzione di pesi fissati durante ogni iterazione per evitare overfitting, …)
- Dato che ci sono tantissimi dati di training questi sono elaborati in lotti.
- Ci possono essere diverse passate per i dati con un rate di addestramento gradatamente decrescente.
- C'è una miriade di versioni di queste reti e dei loro addestramenti.

Convolutional Neural Networks (CNNs)

- Le CNNs sono delle reti neurali multistrato formate da una pila di layer convoluzionali, layer di subsampling e layer pienamente connessi.
- Anche qui c'è una moltitudine di implementazioni e i dettagli possono variare molto.
- Una CNN in genere inizia con un layer convoluzionale e uno di subsampling che possono essere ripetuti più volte.
- L'ultimo layer di subsampling sarà passato in ingresso a degli strati di uscita pienamente connessi che formano una rete MLP.
- Quindi la novità delle CNN sta in un front-end per le classiche reti MLP.

Convolutional Neural Networks (CNNs)

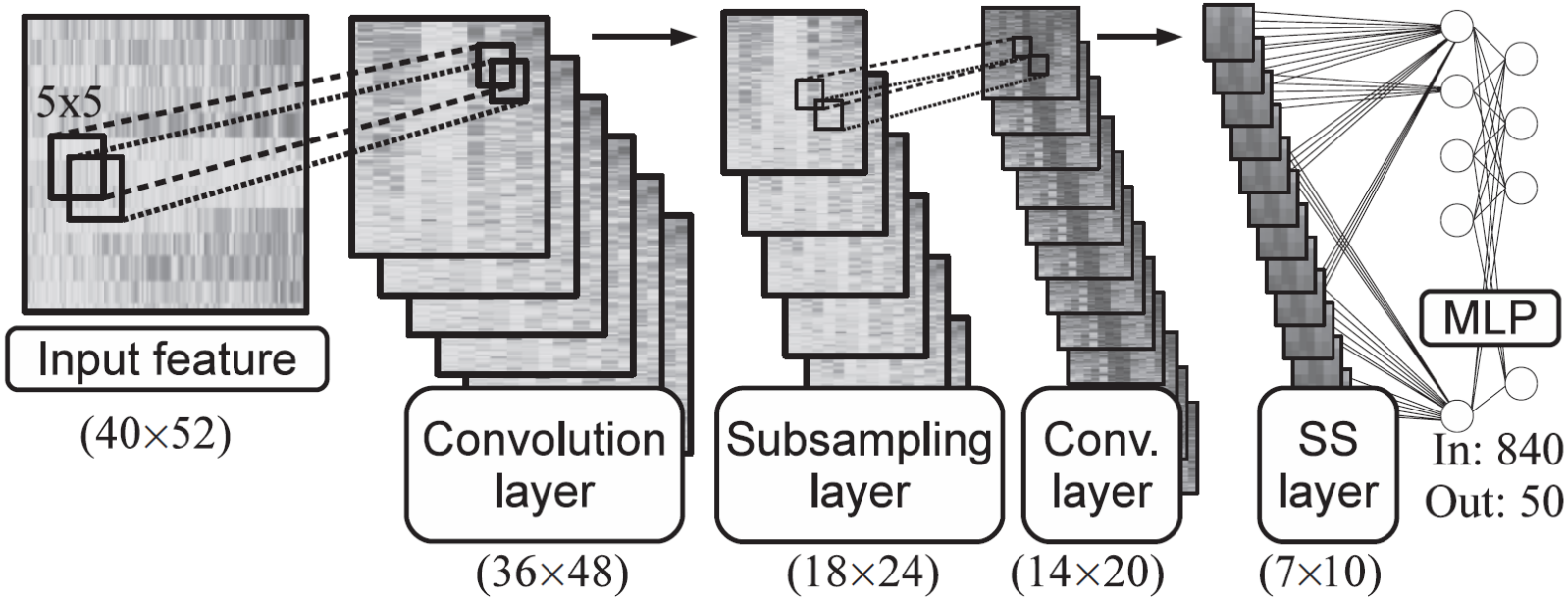


Figure 8.10 A six-layer CNN designed for a machine hearing classifier, comprising two sets of convolution and subsampling layers plus a single fully interconnected output layer. Dimensions are shown for each layer for the specific application example [113].

Convolutional Neural Networks (CNNs)

- Data una immagine di ingresso, i layer convoluzionali lavorano per estrarre le feature locali da una finestra rettangolare e le trasformano in un singolo punto d'uscita.
- Durante l'operazione la finestra rettangolare viene fatta passare su tutta l'immagine di ingresso con le feature d'uscita che formano ancora una immagine, più piccola a seconda della dimensione della maschera convoluzionale.
- Ci sono in genere alcune immagini di uscita generate per ciascuna immagine di ingresso, ciascuna creata con le stesse operazioni ma pesi e bias diversi.
- Il processo di subsampling semplicemente riduce la risoluzione spaziale dell'immagine usando un processo di *max polling* (che rappresenta una finestra rettangolare con il massimo valore nella finestra), di *media,* o processi simili.

Convolutional Neural Networks (CNNs)

- Mentre la complessità della rete è elevata a causa del gran livello di connettività, le CNNs usano pesi condivisi nei layer convoluzionali e questo aiuta a ridurre il numero di parametri di cui fare il training.
- Come le DNNs, le CNNs richiedono grandi quantitativi di dati per l'addestramento.
- I layer convoluzionali formano l'uscita convolvendo l'uscita del layer precedente per un kernel k_{ij} e aggiungendo un bias:

$$\mathbf{x_j^l} = f\left(\sum_{i \in M_j} \mathbf{x_i^{l-1}} * \mathbf{k_{ij}^l} + b_j^l\right),$$

Convolutional Neural Networks (CNNs)

- I layer di sottocampionamento sono più semplici e definiti da:

$$\mathbf{x_j^l} = f(\beta_j^l \downarrow (\mathbf{x_i^{l-1}}) + b_j^l),$$

- Con $\downarrow (.)$ che rappresenta un qualunque subsampling.
- Lo strato finale è costituito da un MLP con la dimensione di ingresso costituita dal numero totale di punti dell'ultimo layer di subsampling e dimensione d'uscita definita dal numero delle classi.
- Viene addestrata con l'algoritmo di back-propagation.
- Dato che le unità dello stesso layer convoluzionale condividono gli stessi parametri, il gradiente di questi coefficienti viene calcolato semplicemente con la somma dei singoli gradienti alla fine di un mini-batch.

Convolutional Neural Networks (CNNs)

- Le CNNs sono molto usate in image-processing ma sono state applicate anche in automatic speech recognition e per lo speech processing.
- Hanno dimostrato di superare le reti tradizionali per molti task, specie quelli che hanno feature di ingresso tipo immagini (e.g., uno spettrogramma di ingresso).

Vedere:

- Ian Vince McLoughlin, “Speech and Audio Processing”- Cambridge Univesity Press (2016)
  - Cap. 8