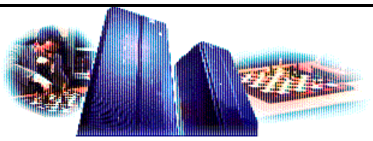
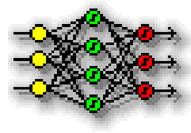




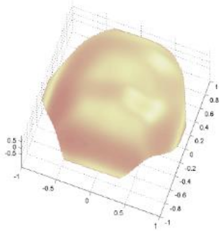
Sistemi Intelligenti
On-line learning

Alberto Borghese
Università degli Studi di Milano
Laboratorio di Sistemi Intelligenti Applicati (AIS-Lab)
Dipartimento di Informatica
Alberto.borghese@unimi.it

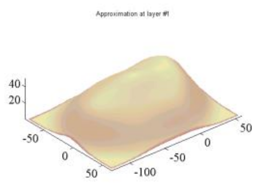
Riassunto

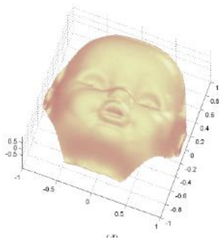
- Valutazione di un modello
- Modelli multi-scala on-line
- Introduzione alle reti neurali

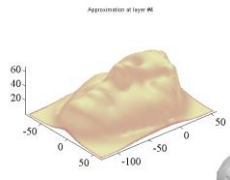
Underfitting e overfitting



Quanti parametri?







Quante unità?

Come valutiamo?

How to classify the error introduced by a model?

Does it cover the input domain (in all dimensions – **dimensionality discovery**)?

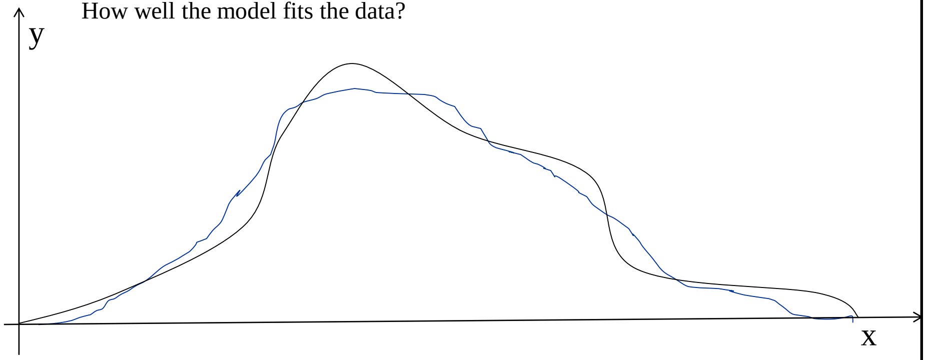
This is not enough to obtain a good model!! Is the model good enough?

The model should be properly tuned to the data. Errors can be classified in:
- Bias
- Variability

Relationship between number of parameters and bias/variability

Bias

How well the model fits the data?



Blue represents the curve of the real data $\{x_{true}, y_{true}\}$
Black represents the curve produce by the model: $y = f(x)$

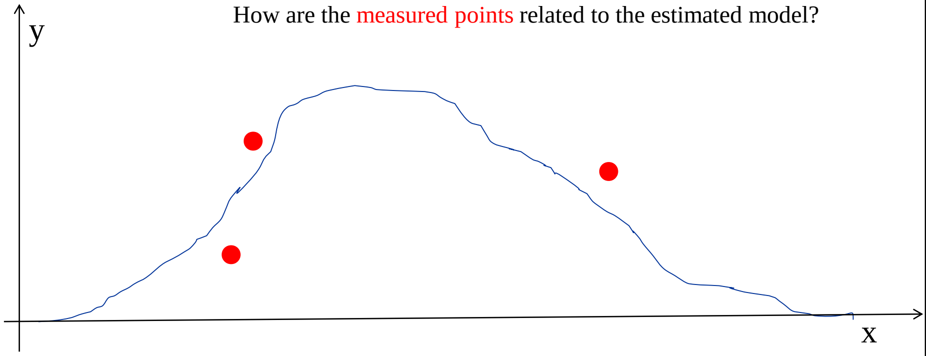
How good is the model?

Model output

We can compute the error: $dist(y_{true}, y=f(x_{true}))$, for instance Euclidean distance.
As such it is the **bias of the model**.

Variability

How are the measured points related to the estimated model?

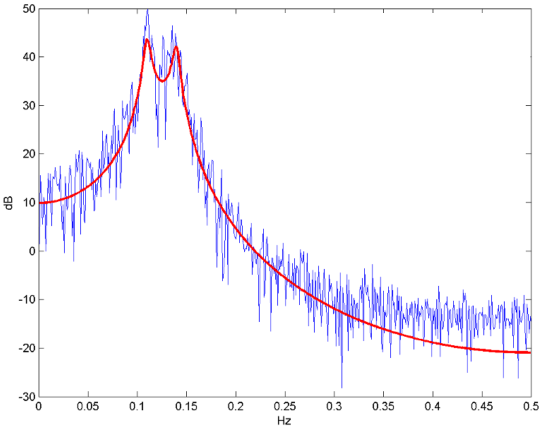


Given $P_m(x_m, y_m)$ and $y = f^*(x)$, the true data behavior, the error is:
$dist(y_m, y=f^*(x_m))$, for instance Euclidean distance.
As such **variability** is the **measurement error**.

If variability goes to zero, bias increases and overfitting arises (model fits data and the noise too).
In a good model, variability tends to the statistics of the measurement noise (cf. regularization parameter setting).

Variability

How are the measured points related to the estimated model?



Given $P_m(x_m,y_{mes})$ and $y = f(x)$, the true data behavior, the error is measured as: $dist(y_m,f(x_m))$, for instance Euclidean distance. It is associated to measurement error.

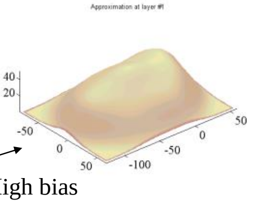
If variability goes to zero, bias increases and overfitting arises.
In a good model, variability tends to the statistics of the data noise.

Bias and variability

Bias and variability trade-off

Bias is the distance of the model curve from the true curve, **that is unknown**. It is the model error.

Variability is the distance of the true curve, **that is unknown**, from the **measured data.** It is the measurement error.



High bias

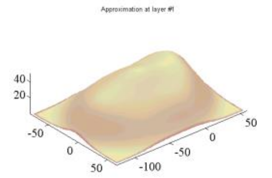
High variability

Error vs number of parameters

Bias and variability trade-off

Bias is the distance of the model curve from the true curve, **that is unknown**. It is the model error.

If we have **few parameters**, we can reproduce only the outline of the data (under-fitting -> bias).



High bias
Low number of parameters

Variability is the distance of the true curve, **that is unknown**, from the **measured data.** It is the measurement error.

If we have many parameters we can reproduce the fastest variations, that are due to noise (over-fitting ->variability).

High variability
High number of parameters

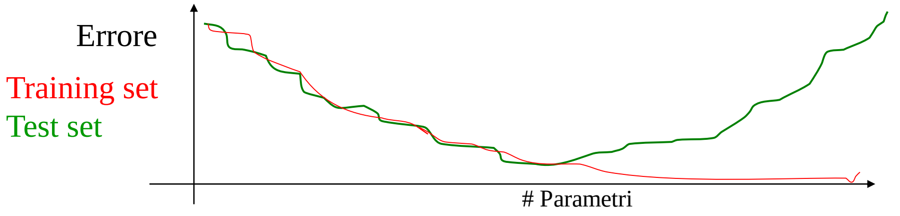
Scelta empirica del numero di parametri – cross-validation

Cross-Validation.

- I dati vengono suddivisi in due sotto-insiemi: training e test.
- Errore sull'insieme di training = Errore sull'insieme di test.
- La procedura viene ripetuta k volte su sottoinsiemi diversi estratti a caso: k-fold cross-validation.

Si vuole evitare che il modello si specializzi troppo sui pattern di training e non sia in grado di interpolare correttamente su altri dati (e.g. dati di test).

*Il numero di parametri viene aumentato fino a quando **entrambi** gli errori diminuiscono.*

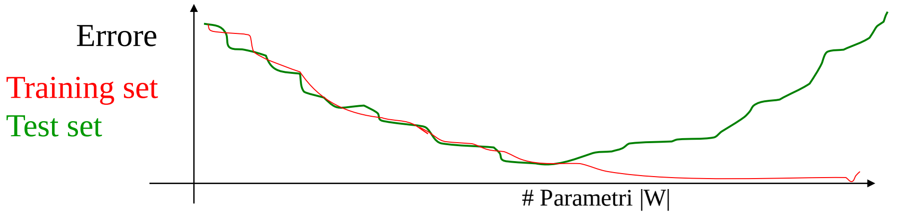


Scelta teorica

Quale funzione costo minimizzo? Come posso inserire l'informazione di complessità del modello nella funzione costo?

Penalizzo i modelli con tanti parametri. Aggiungo nel calcolo della distanza tra dati e modello (funzione costo) un termine che cresce con il numero dei parametri -> Regularization with Reproducible Hilbert Kernels as regularizers.

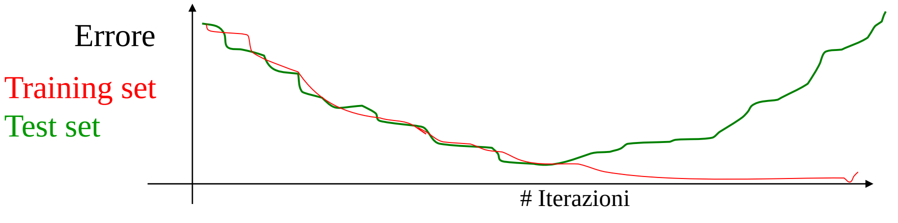
$$w = \underset{w}{\operatorname{argmin}} \left(\sum_i \|f(x_m) - y_m\|^2 + \lambda g(|W|) \right)$$



Altri approcci

Semi-convergenza: non porto l'algoritmo fino alla convergenza nel punto di ottimo ma arresto le iterazioni prima.

Il modello non sarà perfettamente aderente ai dati, ma il residuo sarà tendenzialmente l'errore di misura.



I vari tipi di apprendimento

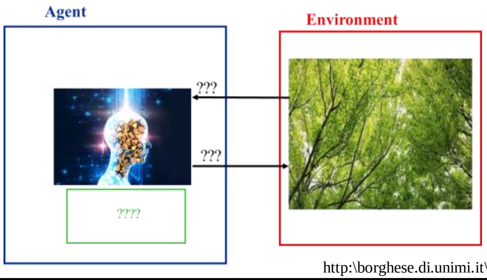
$x(t+1) = f[x(t), a(t)]$ Ambiente
$a(t) = \textcolor{red}{g}[x(t)]$ Agente

Supervisionato (learning with a teacher). Viene specificato per ogni pattern di input, il pattern desiderato in output.

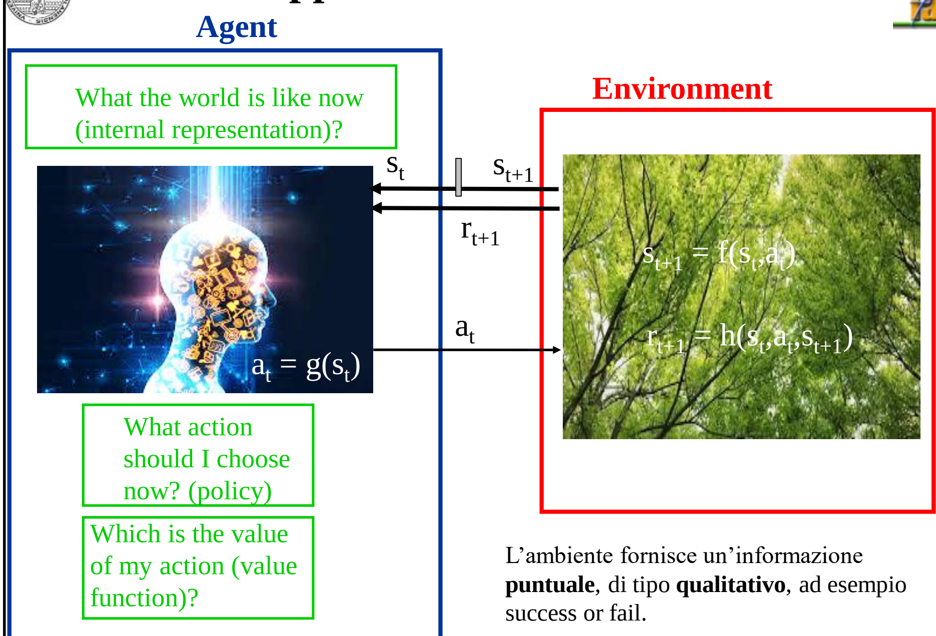
Semi-Supervisionato. Il pattern desiderato in output viene specificato solamente per **alcuni** pattern di input. Gli altri pattern contribuiscono a definire la forma del manifold dei dati.

Non-supervisionato (learning without a teacher). Estrazione di similitudine statistiche tra pattern di input. Clustering. Mappe neurali.

Apprendimento con rinforzo (reinforcement learning, learning with a critic). L'ambiente fornisce un'informazione puntuale, di tipo qualitativo, ad esempio success or fail.



Apprendimento con rinforzo



L'ambiente fornisce un'informazione **puntuale**, di tipo **qualitativo**, ad esempio success or fail.

Apprendimento non supervisionato

Agent

What is «s» similar to?

Environment

s

X No output

No feed-back/information from the environment

Apprendimento supervisionato

Agent

What the world is like now (internal representation)?

What action should I choose now? (policy)

Which is the suggested action?

s_t

$a^D{}_t$

$a_t = \pi(s_t)$

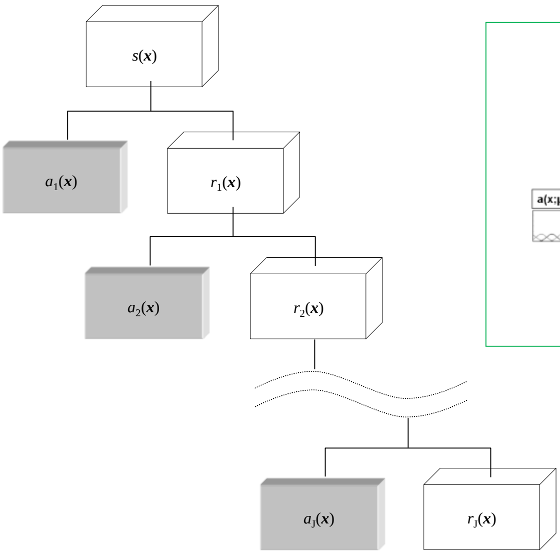
Environment

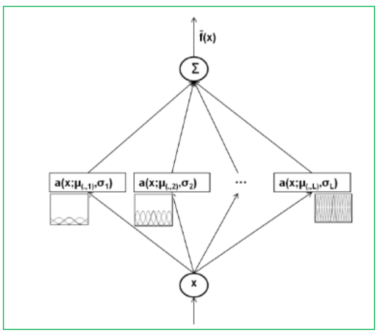
L'ambiente fornisce un'informazione **puntuale**, di tipo **quantitativo**: the optimal (desired action). Learning **with a teacher**.

Riassunto

- Valutazione di un modello
- Modelli multi-scala on-line
- Introduzione alle reti neurali

HRBF networks





It generates a set of hierarchical layers at decreasing scale

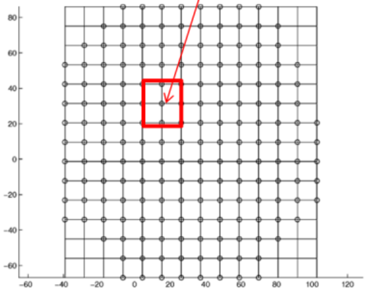
HRBF layer

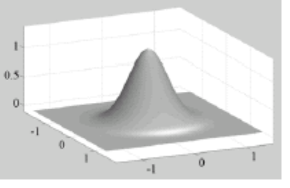
- Convolutional layer, l, with a Gaussian at a given scale σ_l.
- Units are equally spaced with large overlap.
- Weights are computed gridding the data.

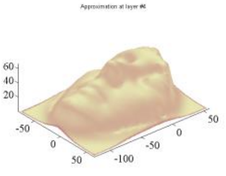
$$f(x_c) = \frac{\sum_{i=1}^{N_c}\left(f(x_i)e^{\frac{-(x_i-x_c)^2}{\sigma^2}}\right)}{\sum_{i=1}^{N_c}\left(e^{\frac{-(x_i-x_c)^2}{\sigma^2}}\right)} \qquad \hat{f}(x) = \sum_{k=1}^{N} f\left(x_{c_k}\right)G\left(x; x_{c_k}, \sigma\right)\Delta x$$

w_{c_k}

x_c







Batch regression

- All data are present at start.

$$\hat{f}(x) = \sum_{k=1}^{N} f\left(x_{c_k}\right)G\left(x; x_{c_k}, \sigma\right)\Delta x$$

- The $w_{c_k} = f(x_{c_k})$ can be computed from all the local data (those inside the receptive field of each unit).
- The local residual can be computed from all the local data.
- Decision for splitting a quad into four quads at half resolution can be taken from all the data and it is taken in parallel for all quads of that layer
- Efficient data support through in-place ordering of the data.

Here we have 1 datum at a time….

On-line single layer

- Each new point, $\{x_k, y_k\}$, contributes to the estimate of the function height inside the receptive fields of the associated Gaussians.
- The estimate of $f(x_c)$ has to be recomputed:

$$f(x_c) = \frac{\sum_{i=1}^{N_c}\left(f(x_i)e^{\frac{-(x_i-x_c)^2}{\sigma^2}}\right)}{\sum_{i=1}^{N_c}\left(e^{\frac{-(x_i-x_c)^2}{\sigma^2}}\right)}$$

- On-line strategy: numerator and denominator are updated separately.

On-line estimate of $f(x_c)$

$$Num(x_c) = \sum_{i=1}^{N_c}\left(f(x_i)e^{\frac{-(x_i-x_c)^2}{\sigma^2}}\right) + \left(f(x_{new})e^{\frac{-(x_{new}-x_c)^2}{\sigma^2}}\right) =$$

$$Num'(x_c) = \sum_{i=1}^{N_c+1}\left(f(x_i)e^{\frac{-(x_i-x_c)^2}{\sigma^2}}\right)$$

$$Den(x_c) = \sum_{i=1}^{N_c}\left(e^{\frac{-(x_i-x_c)^2}{\sigma^2}}\right) + \left(e^{\frac{-(x_{new}-x_c)^2}{\sigma^2}}\right) =$$

$$Den'(x_c) = \sum_{i=1}^{N_c+1}\left(e^{\frac{-(x_i-x_c)^2}{\sigma^2}}\right)$$

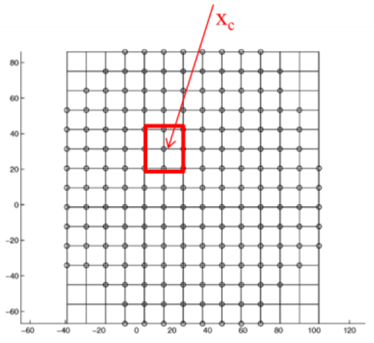
$$f(x_c) = \frac{Num'(x_c)}{Den'(x_c)}$$

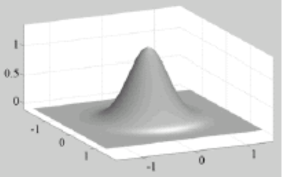
Few computations are required for any number of point: recursive estimate of $f(x_c)$

For each new point a new term is added and the ratio is recomputed only for the Gaussians whose receptive field contains that point.

For Nc -> ∞

$$\lim_{Nc \to \infty} \frac{Num'(x_c)}{Den'(x_c)} = E(f(x_c))$$

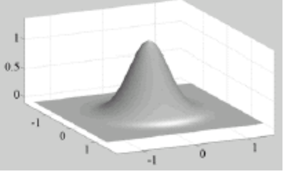


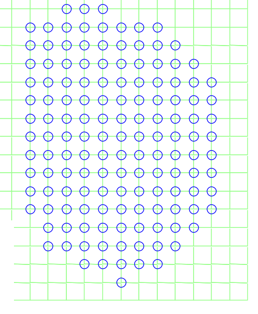


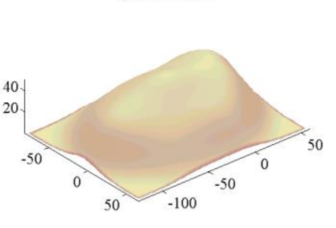
How many points are required to get a good estimate? Experimental answer.

Is it sufficient to obtain a good reconstruction?

First layer



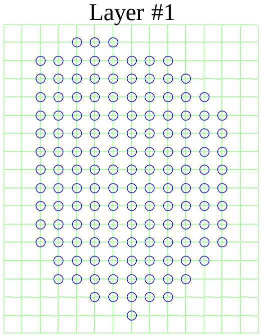




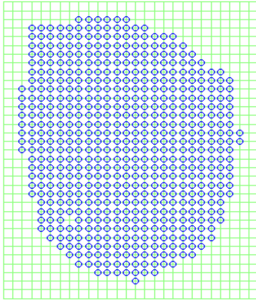
Asymptotically, we cannot obtain anything better than this.
Few Gaussians, large scale.

How to move to next layer?

Layer #1



Layer #2



A reliable estimate of f_c on Layer 1 $\nRightarrow$ a reliable estimate of f_c on Layer 2.

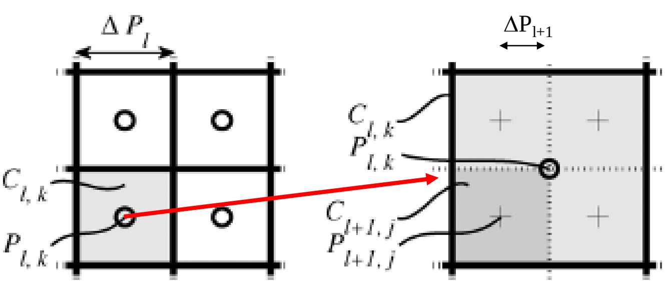
The points that belong to the receptive field of fc on Layer 1 are split over the different quads in Layer 2.

When shall we start to estimate f_c in Layer 2?

Local operations

- Local split of each quad is achieved when:
  - Residual is higher than threshold
  - **Enough points have been sampled**
- **4 new Gaussians are generated at the higher level**

1 quad



4 quads

On-line estimate of $f(x_c)$ on new layer

$$Num'(x_c) = \sum_{i=1}^{N_c+1} \left(r(x_i) e^{\frac{-(x_i - x_c)^2}{\sigma^2}} \right)$$

$$Den'(x_c) = \sum_{i=1}^{N_c+1} \left(e^{\frac{-(x_i - x_c)^2}{\sigma^2}} \right)$$

This requires that the points $\{x_i\}$ inside the receptive field of the 4 Gaussians created are extracted from the $\{x_i\}$ of the Guassian of the current layer that we have split.

How?

In-place ordering of the $\{x_i\}$ associated to the quad of the Gaussian of the current layer such that they are distributed in the quads of the four Gaussians created.

The approximation of the residual, $a(x_c)$ is initialized with few points. Computation of Num'(x_c) and Den'(x_c) for the four new Gaussians requires little effort.

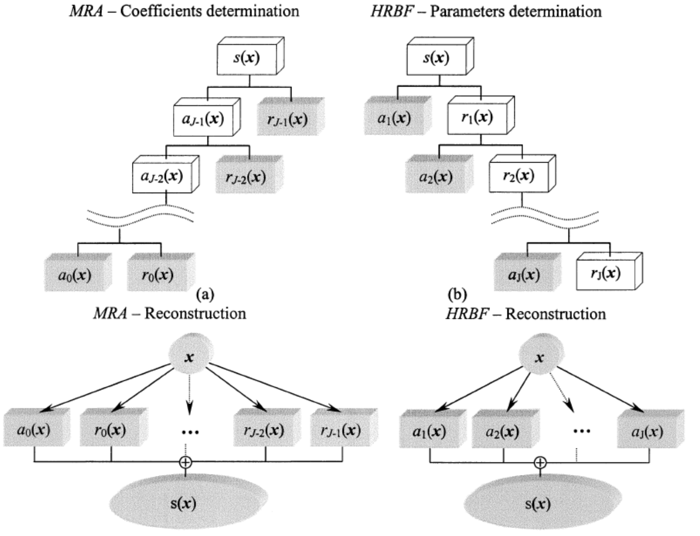
$$r(x_c) = \frac{Num'(x_c)}{Den'(x_c)}$$

On-line version

- Data do not arrive all together (batch)
- One data at a time.
- **Growing while scanning**

2 min video

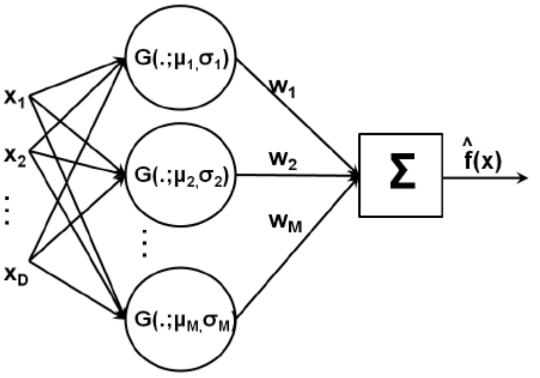
Comparison with Multi-Resolution Analysis (wavelet)



Riassunto

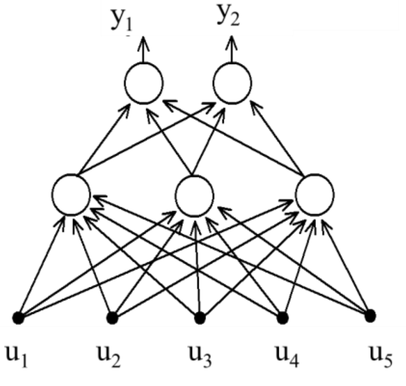
- Valutazione di un modello
- Modelli multi-scala on-line
- Introduzione alle reti neurali

Verso le reti neurali artificiali



- G(.) può essere sostituita da un'altra funzione di base
- Le unità **non devono essere necessariamente equispaziate**
- I parametri sono **personalizzati su ciascuna unità**

Neuroni artificiali «moderni»



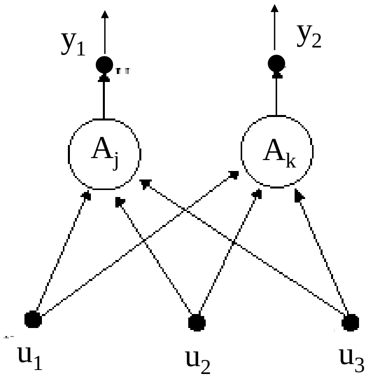
Connessionismo classico. Uscita compresa tra min – Max. Tra 0 e 1 (o tra -1 e 1): y_i = [0 1].

L'uscita può essere binarizzata definendo una soglia.

L'input può essere compreso tra –oo e +oo o limitato a seconda delle unità utilizzate in input.

La rete neurale a un livello

La rete opera una trasformazione dallo spazio di input allo spazio di output.



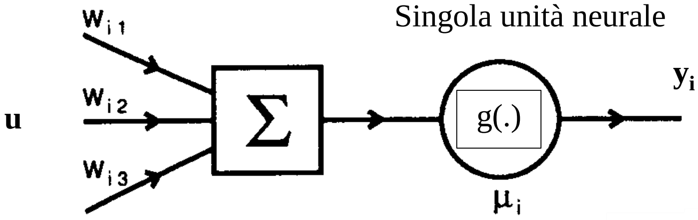
$y_i = \mathbf{g}(\sum_j(w_{ij}u_j) - \mu_i)$

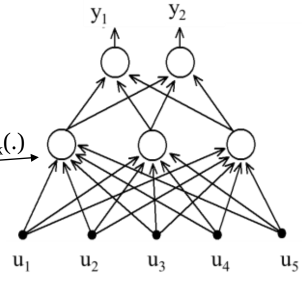
La trasformazione o mappatura dipende dai parametri $\{w_{ij}\}$ e $\{\mu_i\}$ in modo tale che la rete neurale approssimi la trasformazione tra i pattern di input e di output.

Se g(.) = 1, la rete diventa un **modello lineare**: $y_i = \sum_j(w_{ij}u_j) - \mu_i$

Una rete neurale a più livelli

Singola unità neurale



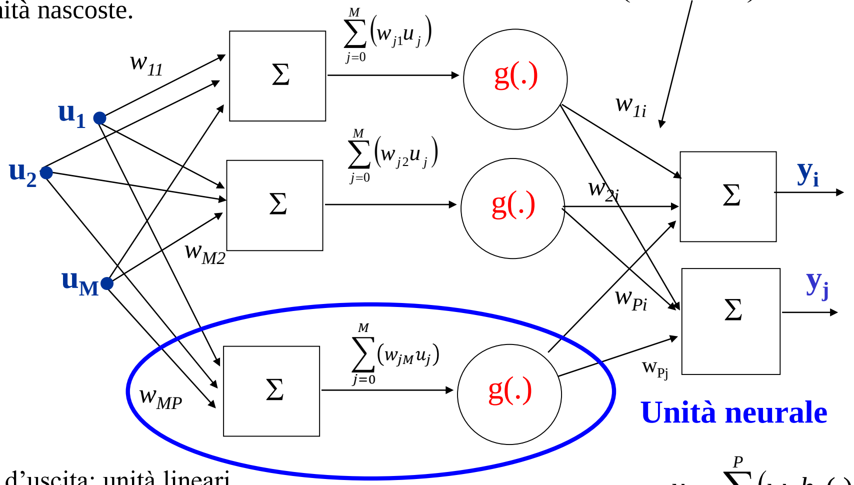


Unità nascoste – Hidden layer

MLP : Multi-layer Perceptron

•M ingressi
•N uscite
•P unità nascoste.

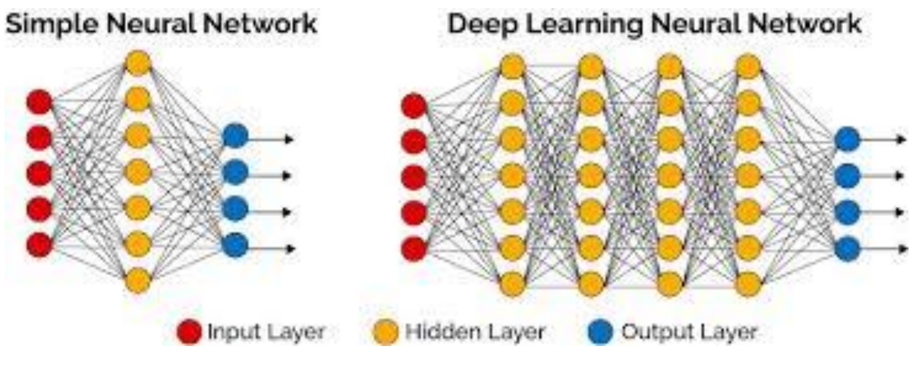
Hidden output $h_i = g\left(\sum_{j=0}^{M}\left(w_{ji}u_j\right)\right)$



Livello d'uscita: unità lineari.
Livello intermedio (hidden): unità non-lineari

$$y_i = \sum_{k=0}^{P}\left(w_{ki}h_k(.)\right)$$

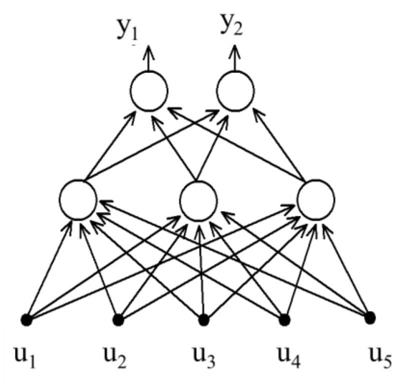
Deep Neural Networks



Convolutional layers (e.g. RBF networks) - filters
Pooling
Subsampling
Gather-scatter…)

Operazioni semplici e parallele -> match perfetto per le GPU

Caratteristiche



Livelli di unità di attivazione

Collegamento in cascata

Input convergenti, output divergenti.

Capacità di approssimazione universale

Perceptrone: layered networks, flusso unidirezionale dell'elaborazione.

L'output viene interpretato come frequenza di scarica del neurone d'uscita della rete.

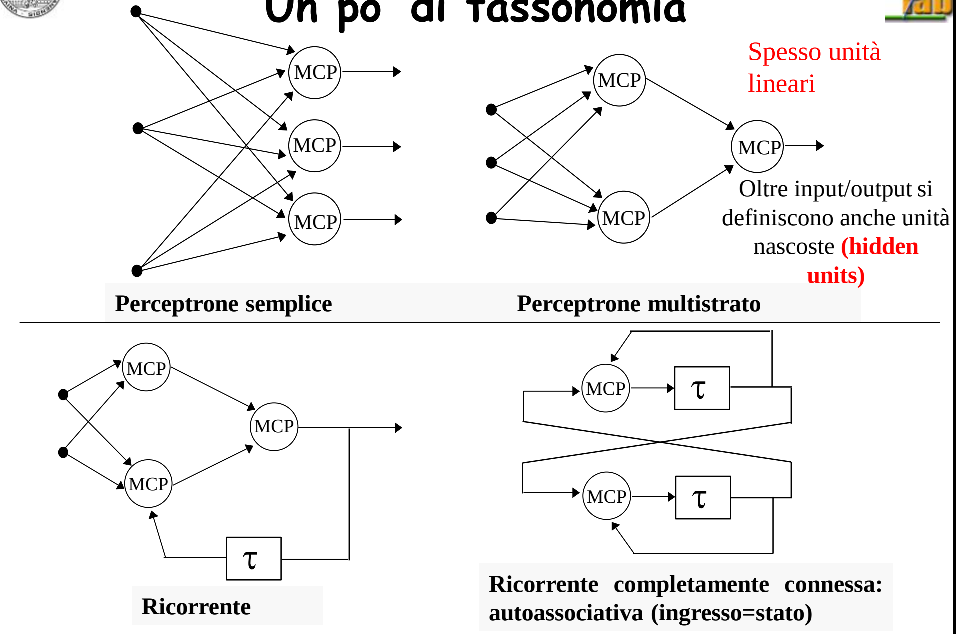
Complessità della funzione realizzabile

Quanti più neuroni artificiali vengono connessi tanto più la funzione complessiva approssimabile diviene più complessa

$$Y=|y_1, y_2, y_3, \dots y_n)|^T$$
$$X=|x_1, x_2, x_3, \dots x_m|^T$$
$$y=F(X)$$

Reti neurali = approssimatori universali.

Un po' di tassonomia



Riassunto

- Valutazione di un modello
- Modelli multi-scala on-line
- Introduzione alle reti neurali