Self Organizing Maps (Kohonen Maps)

Modello neurale ispirato dal cervello umano.

Il cervello umano è organizzato a seconda degli input sensoriali che riceve, in diverse cortecce. All’interno di ogni corteccia, gli stimoli sono **organizzati in modo topologico**: neuroni vicini si “eccitano” in corrispondenza di stimoli vicini.

Esempio: stimolo visuale

* due immagini di due gatti simili tra di loro, attiveranno due zone della corteccia cerebrale che sono spazialmente vicini fra loro.
* un albero e un gatto, si attivano due neuroni spazialmente distanti tra loro

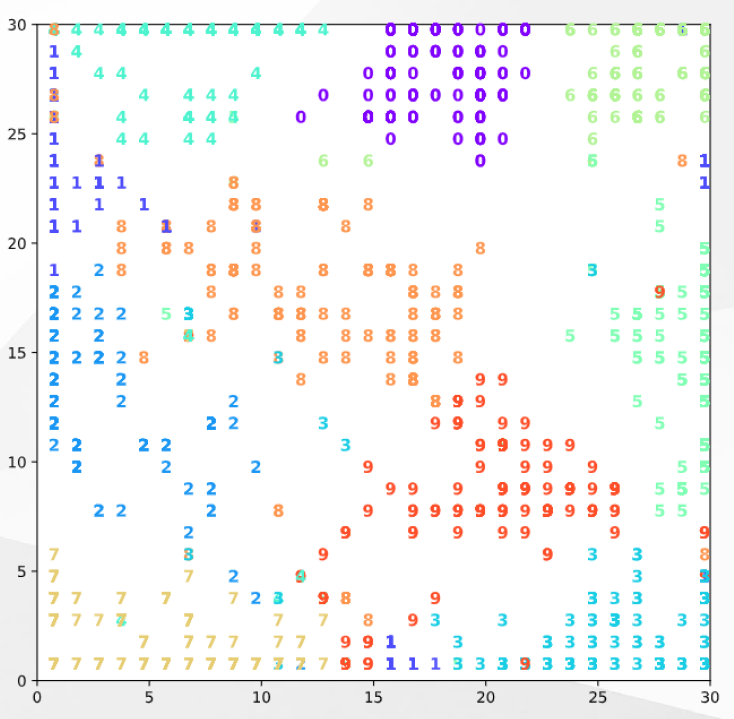
Le SOM non hanno un’unica declinazione, esistono diverse varianti. Si affrontano di seguito le **Kohonen Maps** che non implementano fedelmente questa idea di riproduzione del funzionamento del cervello umano. Ma sono un buon strumento computazionale in quanto ha dato frutti in molti campi. Esistono altre mappe oltre alle Kohonen Maps, ad esempio le Wilshaw-von der Malsburg.

# Obiettivi

Sono **modelli non supervisionati**: si tratta di un **task descrittivo**. Le mappe sono in genere oggetti monodimensionali o bidimensionali. Una volta addestrata la SOM, gli stimoli provenienti dal dataset sono organizzati in modo topologico: stimoli vicini vengono mappati su neuroni vicini nella mappa.

In figura: mappa di 30x30 neuroni. Ogni cella rappresenta un neurone e sulla mappa si sta plottando il dataset MNIST.

MNIST è un dataset usato molto nelle reti neurali (task di classificazione delle immagini): immagini 28x28 pixels, ogni pixel riporta un valore da 0 a 255 che rappresenta l’intensità del pixel in scale di grigio. Le immagini contengono i numeri scritti a mano da utenti. Il task è il mapping dall’immagine al valore numerico corrispondente. In questa mappa, si osserva il mapping del dataset sulla mappa: per ognuno degli esempi del dataset, si sottopone alla mappa l’esempio, l’esempio eccita in particolare un neurone e il neurone eccitato viene etichettato con il numero corrispondente al suo eccitamento.



Ciò che succede durante l’addestramento della mappa, la mappa è riuscita a riconoscere le differenze e le similarità degli esempi e ha individuato delle aree omogenee. I 9 sono vicini agli 8 perché molto simili. Idem per il 3. E così via.

Ha organizzato gli stimoli in maniera topologica: a stimoli vicini corrispondono neuroni vicini.

# Principi delle SOM

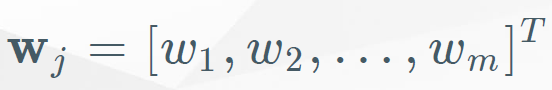
Esistono 3 principi che guidano l’apprendimento delle SOM:

1. **Competizione**: ogni volta che arriva un nuovo stimolo, tutti i neuroni competono per diventare la **Best Matching Unit (BMU)**, il neurone della mappa che meglio si adatta allo stimolo. E’ confrontata la rappresentazione\* presa dal neurone fino a quel momento con lo stimolo e se questa è la più vicina diventa la BMU.
   * ad esempio: esistono più rappresentazioni del numero 3, in quanto può essere scritto in tanti modi diversi.
2. **Cooperazione**: tutti i neuroni vicini al neurone vincitore (BMU) devono diventare un po’ simili allo stimolo. La BMU attiverà un certo suo intorno di neuroni.
3. **Adattamento sinaptico**: sono aggiornati i pesi associati ai neuroni attivati in modo tale da renderli più vicini all’esempio osservato. Ad ogni iterazione vengono apportate modifiche ai pesi in modo da migliorare la risposta del neurone vincente a modelli di input simili.

# Competizione

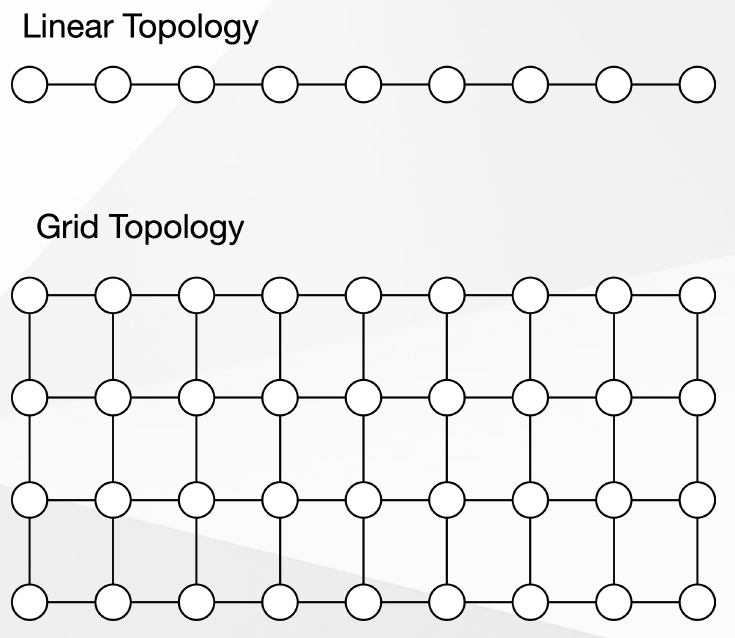
Sia  un vettore di input, è un vettore colonna.

Si consideri una SOM con l neuroni. Ogni unità j nella SOM ha un vettore peso con stessa dimensione del vettore di input.



I neuroni sono disposti all'interno della mappa secondo un **ordinamento topologico**.

Le topologie più usate sono la **topologia lineare (monodimensionale)** e la **topologia a griglia (bidimensionale)**.

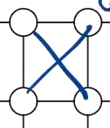


Si può salire di dimensionalità ma non è comune farlo.

**Topologia lineare**: i neuroni sono distribuiti su un vettore. I rapporti di vicinanza → il neurone 1 è vicino al 2. Il neurone 2 è vicino a 1 e a 3.

**Topologia grigia**: ogni neurone ha almeno 4 vicini. Esistono tante topologie di questo tipo ma si possono aggiungere altri rapporti di vicinanza.

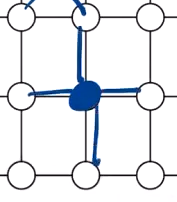
Altri esempi di topolgie a griglia

rapporto di vicinanza pentagonale.

Adattare la griglia in base al problema che si sta affrontando.

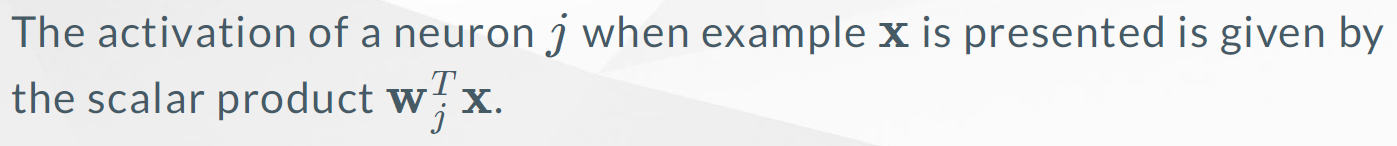
I **rapporti di vicinanza** sono fondamentali per l’attivazione del neurone BMU e dei neuroni vicini. Per questo la topologia è importante.



**Definizione di topologia**. La topologia è una branca della matematica che si occupa di studiare le relazioni di vicinanza degli oggetti. Ciò che è importante non è tanto la forma dell’oggetto (studio geometrico) ma i rapporti di vicinanza. Come sia disposto nello spazio è superfluo. La topologia è alla base di tutti gli spazi vettoriali R, N, etc.

In questo caso quindi è importante conoscere il **rapporto di vicinanza** di un neurone (quali sono i vicini di un neurone).

L’input è confrontato con i vettori pesi di tutti i neuroni e i neuroni si attivano con una forza pari al loro prodotto scalare. Più il prodotto scalare è alto, più il neurone viene attivato.



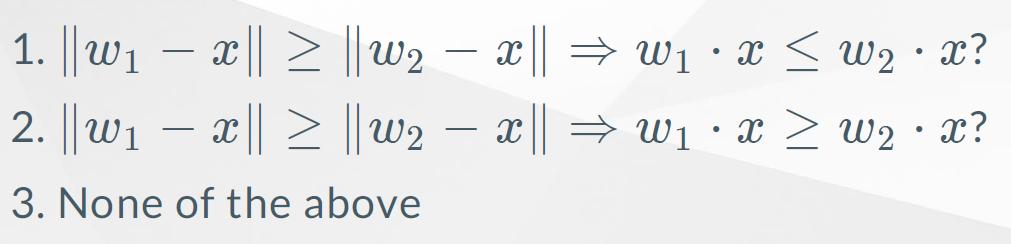
Dopodiché, si osservano tutti questi valori e si sceglie la Best Matching Unit.

Nella maggior parte delle implementazioni non si usa il prodotto scalare per trovare la BMU, ma si usa la distanza euclidea.

**Il neurone che ha la distanza euclidea minore diventa la BMU**.

## Osservazione

Si consideri un vettore x e due vettori w1 e w2. Quale tra i seguenti è vero?



La risposta è la 3.

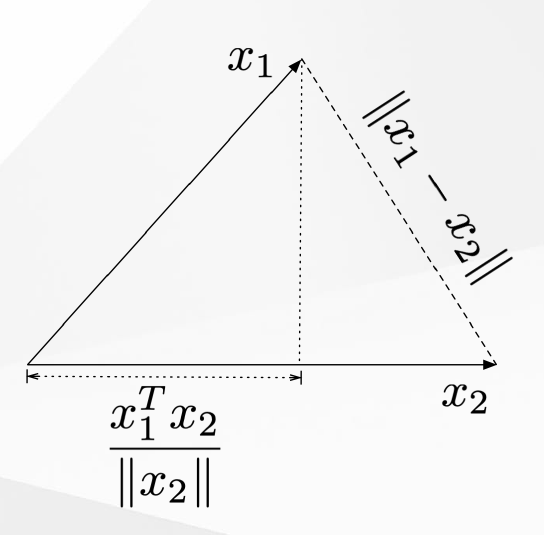
## Relazione tra prodotto scalare e norma euclidea

Qual è la relazione tra il prodotto scalare e la norma euclidea?

1. Non hanno alcuna relazione
2. Se i vettori sono normalizzati: forniscono la stessa informazione
3. La distanza euclidea cresce quando il prodotto scalare diminuisce e viceversa
4. La distanza euclidea diminuisce quando il prodotto scalare diminuisce e viceversa

Risposta: 2. Se i vettori sono normalizzati: forniscono la stessa informazione. Sono intimamente correlati tra di loro ma per avere un’equivalenza vera è necessaria qualche proprietà in più.

La relazione tra prodotto scalare e norma euclidea è descritta dalla seguente figura:



Siano dati due vettori x1 e x2.

La distanza euclidea tra x1 e x2 è la lunghezza del segmento ||x1-x2||.

Il prodotto scalare invece è la proiezione di x1 su x2.

Se l’angolo compreso tra i due vettori x1 e x2 diminuisce, la norma euclidea diminuisce, mentre il prodotto scalare cresce.

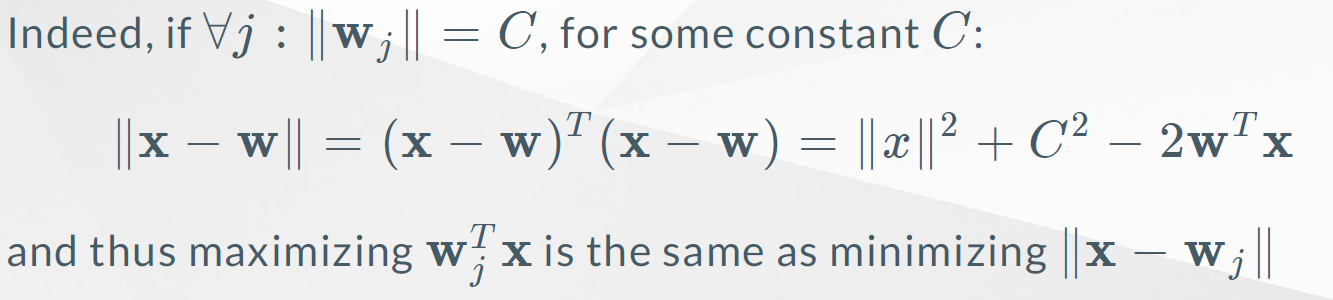
Esiste una relazione inversa tra uno e l’altro.

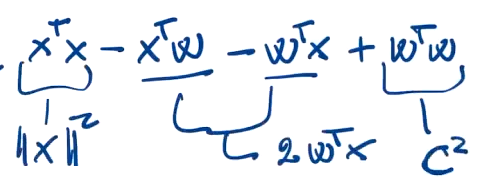
Assumendo che i vettori peso w hanno la stessa lunghezza, selezionando w con il più grande prodotto scalare allora questo è identico a selezionare il w che minimizza la distanza euclidea.



**i(x) è l’indice del neurone BMU (Best Matching Unit)**.

Se i vettori peso w hanno la stessa lunghezza, la norma euclidea di tutti i vettori è uguale a una certa costante C.





Minimizzando rispetto a w, allora ||x||^2 e C^2 sono costanti.

**Minimizzare la distanza euclidea ||x-w||** significa minimizzare il termine -2wx, che **corrisponde a massimizzare il prodotto scalare wx**.

|  |
| --- |
| Definizione di norma 2 di un vettore x    corrisponde alla lunghezza di un vettore o di una matrice.  La norma 2 è la distanza euclidea. |

Riassumendo, **per trovare la best matching unit**

**si massimizza il prodotto scalare**



oppure

**si minimizza la norma euclidea**

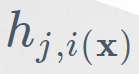


Se i w hanno tutti la stessa lunghezza sono identici.

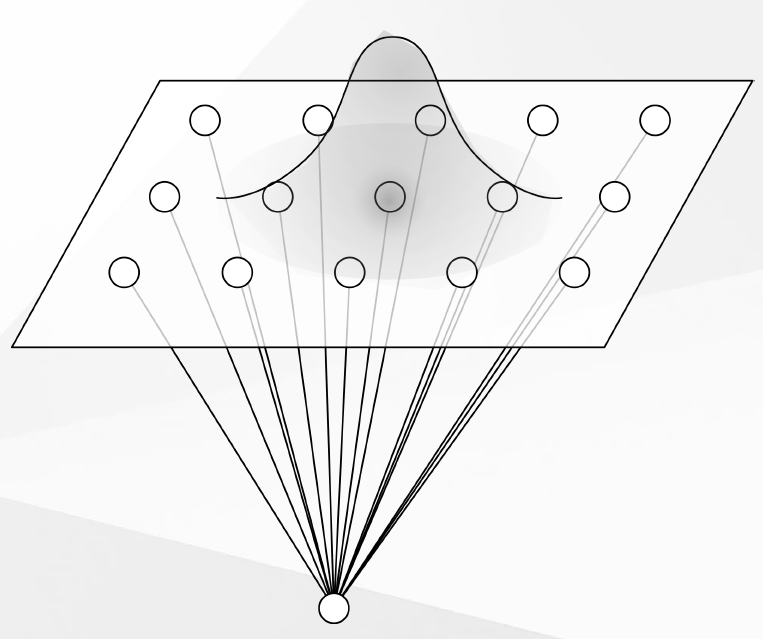
Se non hanno la stessa lunghezza possono portare a risultati leggermente diversi.

# Cooperazione

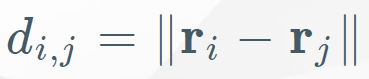
Dopo aver trovato la BMU, non solo si attiva il neurone BMU ma si attivano anche tutti i neuroni nel suo intorno.

Bisogna trovare la **funzione di vicinanza topologica**  dove i(x) rappresenta l’indice del neurone BMU corrente e j rappresenta l’indice di un neurone.

La funzione h è una **funzione che ha il suo massimo in corrispondenza della BMU e decrescente rapidamente man mano che ci si allontana dalla BMU**.



Per definire formalmente h, bisogna introdurre la **distanza laterale** tra i nodi i e j.

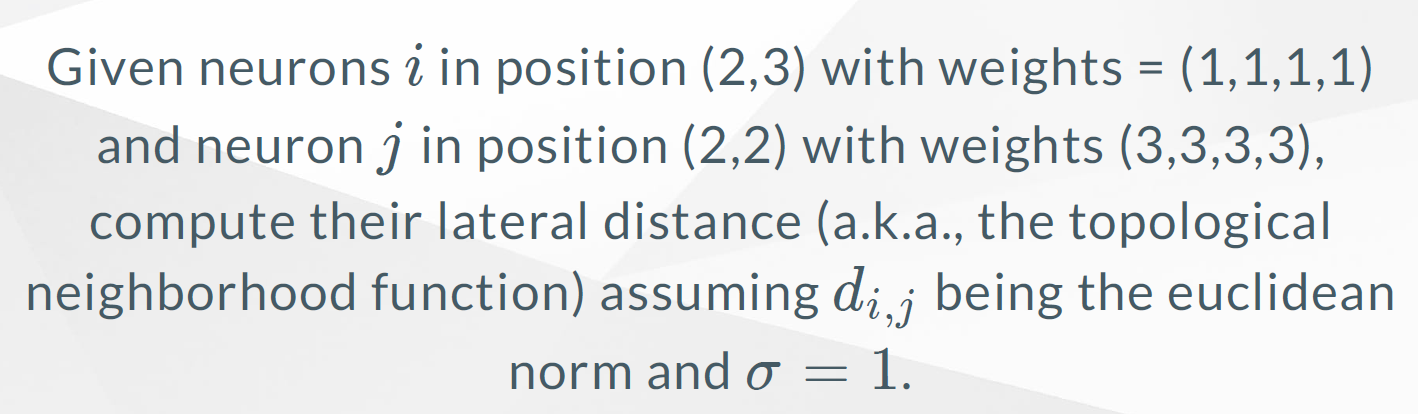


dove e denotano la posizione (coordinate) dei neuroni i e j nella mappa.

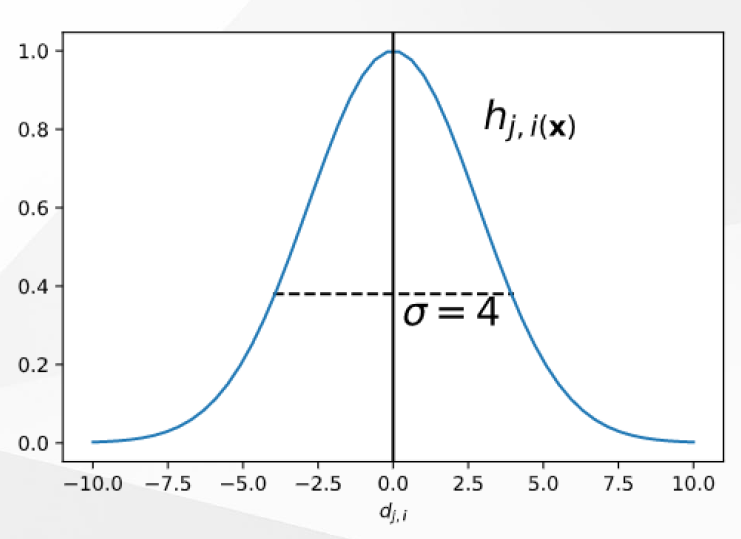
Allora la funzione h è così definita



## Esempio: calcolo della funzione h



Per riassumere, la funzione di vicinanza topologica h ha una forma a campana molto prossima a una gaussiana. Il parametro indica con che velocità l’eccitazione dei neuroni verrà attenuata. Più è stretta la campana (più piccolo è sigma), i vicini saranno considerati meno man mano che ci si allontana dalla BMU.



Il parametro indica con che velocità l’influenza della BMU diminuisce.

## Il parametro

Dovendo far apprendimento sul parametro , ha senso rendere il parametro variabile dipendente dall’apprendimento oppure tenerlo fisso?

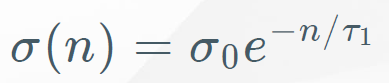
Nel caso in cui convenga cambiarlo è meglio che decresce o cresce nel tempo?

E’ ragionevole diminuire la scala di influenza della BMU man mano che l’apprendimento procede.

All’inizio i pesi della SOM sono inizializzati random quindi i neuroni sono disposti a caso, non c’è una topologia ben definita. Quando si individua una BMU, si modificano anche i pesi di tutti i vicini della BMU per renderli simili alla BMU e per ripristinare le relazioni topologiche che vogliamo cogliere.

Quando siamo più avanti nell’apprendimento, si vuole ridurre questa sfera di influenza per permettere ad altre BMU di creare la propria, altrimenti continuerei a cambiare i pesi di tanti neuroni attorno a me andando ad interferire sul lavoro degli altri neuroni.

Per ridurre questa sfera di influenza, si usa la seguente formula:



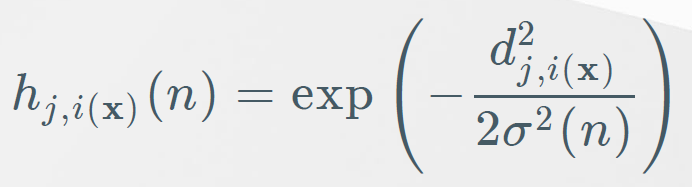
dove n e sono due parametri definiti dall’utente.

Inizialmente il contributo sarà alto, con l’aumentare delle iterazioni il contributo sarà sempre più basso.



spesso è impostato ad un numero prossimo al numero massimo di iterazioni che si vogliono fare nell’apprendimento.

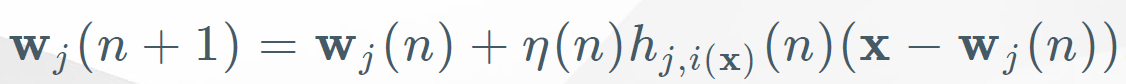
Quindi la funzione di vicinanza topologica è riscritta nel seguente modo:



con il parametro variabile, che decresce ad ogni iterazione.

# Adattamento sinaptico

La regola di aggiornamento del peso w del neurone j è definita:



n è il numero di iterazioni

è il learning rate al tempo n

è la funzione di vicinanza topologica

è una quantità che serve per rendere w più vicino al neurone x.

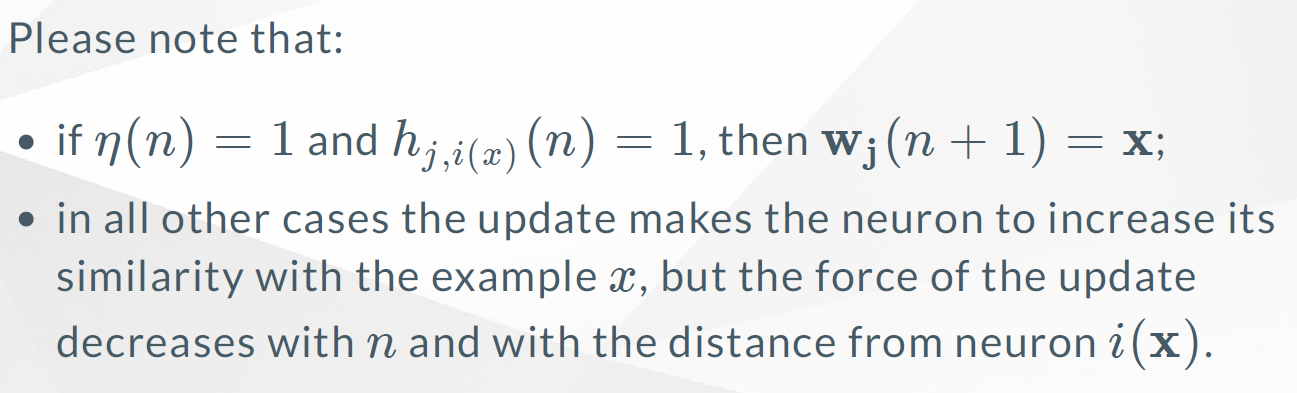
## Valori della funzione h e del learning rate

Quali devono essere i valori della funzione h e il learning rate in queste due condizioni

1. dopo l’aggiornamento del peso, il peso sia stato settato esattamente uguale allo stimolo iniziale, quindi come far sì che
   * h = 1
   * learning\_rate = 1

Arrivare fino ad x è estremo. Meglio arrivare a qualcosa di meno a x. Quindi che h e learning rate siano un po’ meno di 1 in modo tale da avvicinare il peso alla descrizione dell’esempio x.

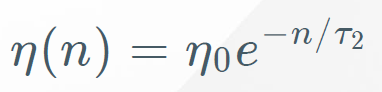
1. Come far sì che , ovvero il peso non è stato modificato
   * h = 0 oppure learning\_rate = 0 oppure entrambi 0



Il processo di aggiornamento dei pesi dipende da due componenti: topological neighbourhood function e learning rate.

Come è definito il learning rate?

E’ definito in modo molto simile al parametro .



dipende dal learning rate iniziale e dal parametro che determina con che velocità il learning rate viene decrementato, diventa un valore vicino a 0.

Questi due parametri sono definiti dall’utente e spesso sono definiti con valori molto prossimi al numero massimo di iterazioni che bisogna compiere.

# Due fasi del processo adattivo

Le SOM sono inizializzate a random usando pesi molto vicini a 0, spesso si usa una funzione gaussiana con varianza non troppo grande centrata in 0. Dopodiché si estrae un esempio, si aggiornano i pesi come descritto dalla regola di update.

Ogni volta che si sottopone un nuovo esempio, n viene incrementato.

L’apprendimento della SOM solitamente si articola in due fasi:

1. **Fase di auto-organizzazione oppure di ordinamento** in cui la struttura topologica viene inizializzata. Inizializzando a caso i pesi, inizialmente la struttura topologica è completamente sbagliata: neuroni che dovrebbero stare lontani sono molto vicini e viceversa. La struttura topologica viene rapidamente ripristinata e si ottiene una struttura in cui neuroni vicini hanno pesi vicini.
2. **Fase di convergenza** in cui la struttura viene raffinata e vengono sbrogliati alcuni difetti topologici. Spesso può succedere che la struttura topologica sia aggrovigliata. Nella fase di convergenza questi nodi si riescono a sciogliere. (vedi video per capire di più).

# Fase di auto-organizzazione

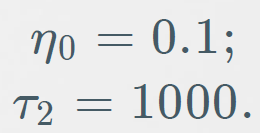
Questa prima fase richiede almeno 1000 iterazioni (valori indicativi da adattare in base al problema ma sono buoni punti di partenza che funzionano in molti problemi pratici).

**Il learning rate e l’ampiezza della funzione di vicinanza devono essere scelti con cura perché l’apprendimento è molto sensibile ai cambiamenti di questi parametri**. Spesso capita che si passa una prima fase del tempo ad apprendere questi parametri per migliorare le prestazioni.

Nelle reti neurali non esiste una teoria solida che spiega come settare i parametri, è un gioco guidato da un minimo di intuizione da parte dell’utente che setta i parametri.

Il learning rate dovrebbe partire da un valore intorno a 0.1 e deve diminuire gradualmente e rimanere intorno a 0.01.

Queste proprietà possono essere raggiunte scegliendo:



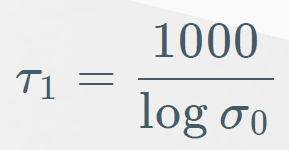
In questa prima fase vogliamo che la funzione h raggiunga almeno tutti i neuroni.

I neuroni sono inizializzati a caso e avere una funzione h molto ampia permette di farli convergere ad una struttura topologica più sensata.

Man mano che l’apprendimento procede, l’ampiezza della funzione h viene ridotta in modo tale da permettere la formazione di altre aree e avere la propria influenza.

Se si parla di un reticolo bidimensionale (a griglia), si consiglia di settare il parametro iniziale al raggio del reticolo (numero di neuroni dal centro).

Invece il parametro è definito di seguito:

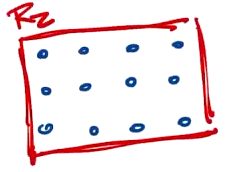
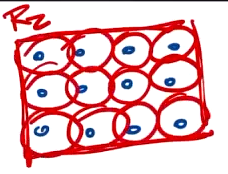


# Fase di convergenza

Le strutture vengono raffinate. In particolare in questa fase ciò che migliora di molto è la quantizzazione dello spazio di input.

Uno dei motivi per cui si usano le SOM è che lo spazio di input sia quantizzato.

Se si immagina la seguente SOM in un piano R2. La dimensione della mappa corrisponde allo spazio che si vuole rappresentare.

Ad un certo punto, i neuroni rappresenteranno una certa zona dello spazio. **Lo spazio viene quantizzato, ovvero un neurone rappresenterà tutto lo spazio di sua competenza**. Ha dei vantaggi: uno dei motivi per cui si utilizzano le SOM è **ridurre la codifica di uno spazio**.

Se voglio trasmettere un'informazione relativa ad uno spazio attraverso la rete, potrei decidere di avere una approssimazione nella trasmissione, **riducendo di molto la quantità di dati da trasmettere**. Posso trasferire i valori dei neuroni e poi quando trovo un punto da trasferire, anziché trasferire il valore completo del punto, trasferisco soltanto l’identificatore del neurone che mi interessa. In questo modo si riduce il peso necessario per trasmettere informazioni relative a questo spazio.

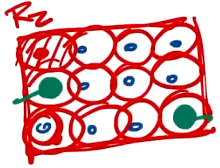
In questa fase di convergenza, l’accuratezza di quantizzazione dello spazio viene migliorata, raffinata.

Affinché questa operazione sia svolta correttamente, il parametro learning rate dovrebbe rimanere ad un valore abbastanza piccolo ma non deve essere troppo vicino a 0 altrimenti la SOM potrebbe rimanere bloccata in uno **stato metastabile in cui sono presenti difetti topologici**.

|  |
| --- |
| Un sistema è in uno **stato metastabile** vuol dire che è in uno stato che non tende più a cambiare da quella posizione ma non è nella sua posizione ottimale. Basterebbe che si spostasse un pochino di più per raggiungere la posizione ottimale.  Se una pallina che deve scendere lunga una pendice, se la pallina si trova in questo punto significa che si trova in un punto metastabile perché piccole perturbazioni a destra e a sinistra non migliorano la sua posizione rispetto alla discesa. Mentre uno spostamento un po’ più grande permetterebbe alla pallina di cambiare la sua corsa. |

Il problema di trovarsi in uno stato metastabile è che se il learning rate diventa troppo piccolo, le modifiche che facciamo ai pesi non permettono alla SOM di migliorare ulteriormente la sua configurazione. Se non migliora la sua configurazione vuol dire che esistono dei difetti topologici.

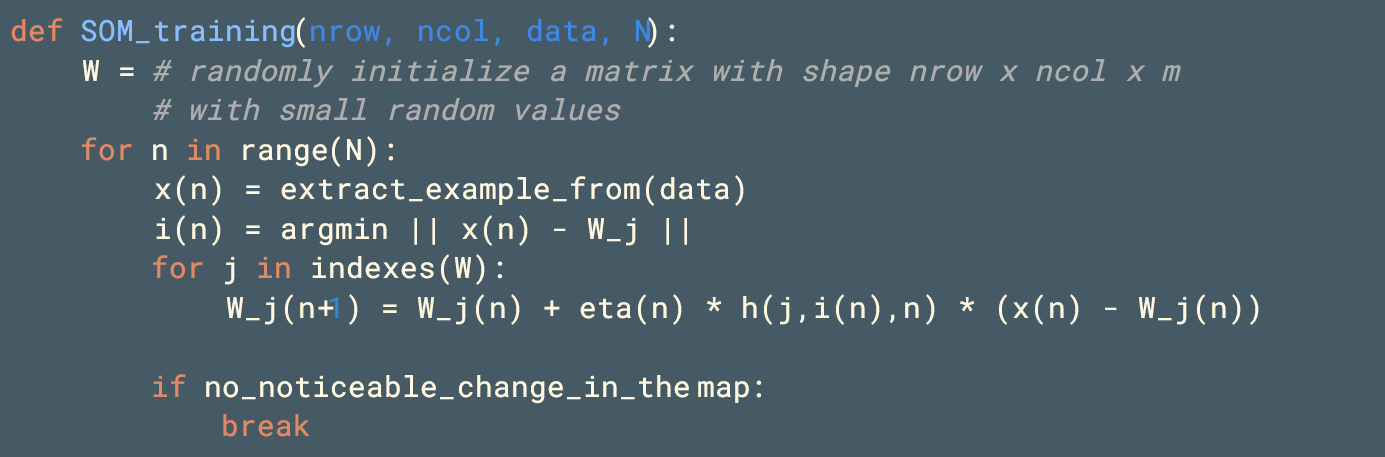
Il **difetto topologico** è una situazione in cui, per esempio, si hanno due oggetti verdi (con rappresentazione simile) nella mappa posizionati distanti. Se la mappa ha appreso la topologia correttamente, questo non dovrebbe succedere.



La **funzione di decadimento del learning rate** è stata introdotta per limitare questo problema, evitare pesi troppo piccoli che ci fanno arrivare a questo stato.

Nella fase di convergenza, la **funzione di vicinanza h** non dovrebbe abbracciare più una grande quantità di pesi. In questa fase si assume che la struttura topologica sia abbastanza buona. Si vuole **raffinare la descrizione compresa nei neuroni**. **Se una BMU si attiva, si vuole attivare quella BMU e non tanto il suo intorno.** In questa seconda fase che automaticamente si attiva nelle SOM man mano che si procede nel numero di iterazioni dell’update della rete, vogliamo che la funzione h si comporti un questo modo.

# Pseudocodice dell’algoritmo SOM



Input

* nrow, ncol: numero di righe e di colonne della SOM
  + nel caso bidimensionale si inseriscono tutte e due
  + nel caso monodimensionale si inizializza nrow=1 e ncol=numero di neuroni da usare
* data: dataset
* N: numero massimo di iterazioni

Si inizializza la matrice dei pesi W: si crea una matrice che contiene un numero reale per ogni neurone (nrow x ncol). I pesi sono inizializzati random con valori piccoli. Solitamente si usa una gaussiana centrata in 0 oppure si tira a sorte con una distribuzione uniforme.

Si fa il ciclo sul numero di iterazioni.

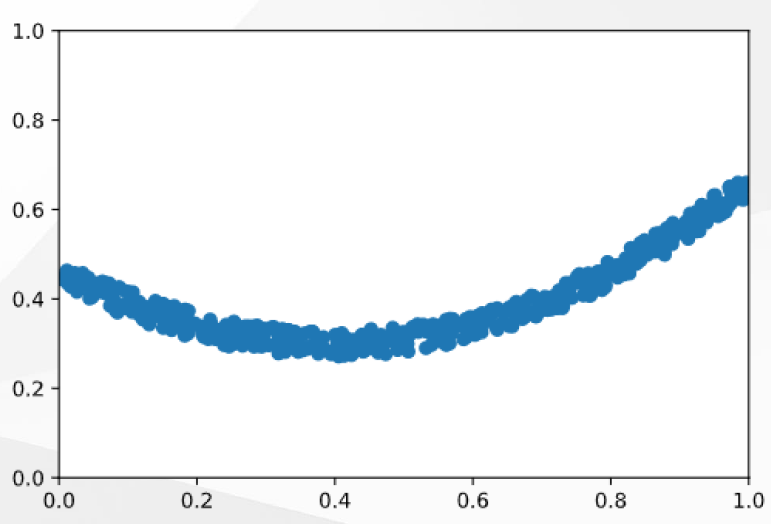
Ad ogni passo di apprendimento, si estrae un esempio a caso dal dataset, si trova l’indice della BMU minimizzando la distanza euclidea oppure massimizzando il prodotto scalare tra x e w.

Per tutti gli indici j della mappa, si setta il peso relativo a quel neurone usando la regola di aggiornamento dei pesi. Si calcola il learning rate e la funzione di vicinanza h.

L’ultima istruzione è solo una miglioria, è facoltativa. Se non ci sono cambiamenti nella mappa, si può uscire dal ciclo prima che siano terminate tutte le N iterazioni. E’ solo una miglioria che serve per stoppare in modo precoce l'apprendimento.

# SOM monodimensionale su un manifold monodimensionale

Si osserva in figura una SOM monodimensionale che cerca di apprendere un manifold monodimensionale. Si tratta di embedding, incluso in uno spazio bidimensionale.



|  |
| --- |
| Il **manifold** è un termine che si usa in topologia per **indicare la dimensionalità intrinseca dei dati**.  Noi viviamo in uno spazio 3D.  Un foglio di carta, pur essendo 3D, possiamo pensarlo come un oggetto bidimensionale immerso in uno spazio 3D. Il manifold del foglio di carta è 2D immerso in uno spazio 3D.  Un granello di sabbia è un manifold monodimensionale immerso nello spazio 3D.  La cartella è un manifold 3D in uno spazio 3D.  E’ un concetto importante perché spesso i dati sono dati con molte dimensioni, vivono in spazi molto grandi, che possono essere approssimati con manifold a bassa dimensione.  Il manifold è alla base della PCA per individuare le dimensioni principali attorno cui i dati si disperdono. Quindi ridurre le dimensioni è molto importante perché semplifica l’apprendimento. |

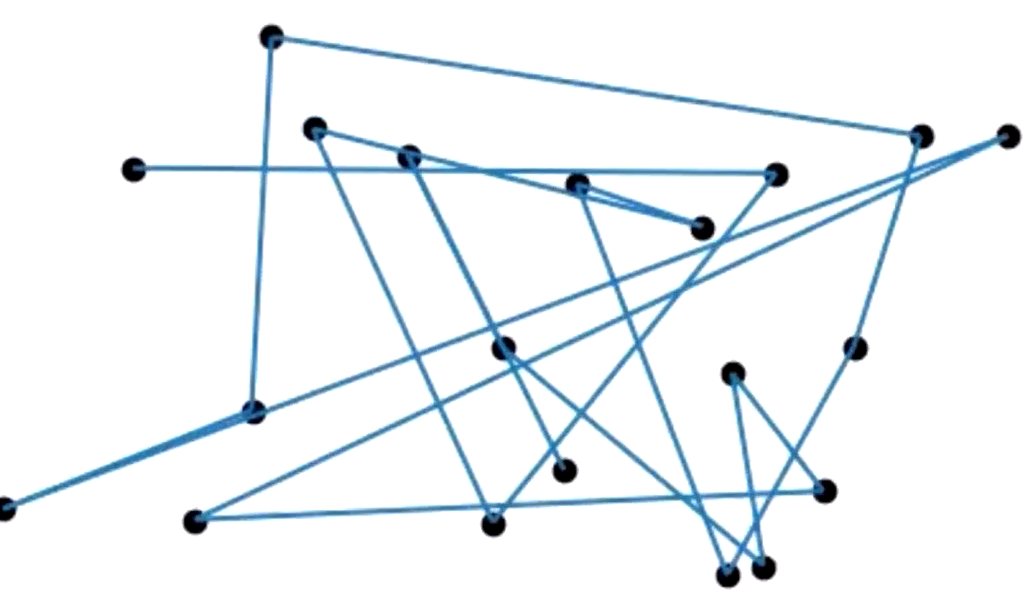
I punti sono inclusi in uno spazio bidimensionale però nella sostanza l’oggetto che descrivono è un oggetto monodimensionale perché è una curva nello spazio.

Questo è uno dei concetti importanti che ci fanno capire perché le SOM funzionano e sono importanti.

Le SOM funzionano molto bene quando gli esempi che stiamo usando vivono in uno spazio a molte dimensioni ma descrivono un manifold (varietà) che ha un numero di dimensioni molto più basso. In questo modo una SOM bidimensionale riesce a catturare una topologia bidimensionale (struttura dei dati che vivono in uno spazio con molte più dimensioni).

Lezione del 23-10, min. 36:00 → video SOM 1D, manifold 1D.

Situazione iniziale



Si osserva un difetto topologico: la parte destra si ripiega su se stessa.



Man mano che si apprende, vengono risolti questi difetti topologici. Si apprende fino a quando non sono sciolti tutti questi nodi e si ottiene una descrizione topologica corretta dello spazio



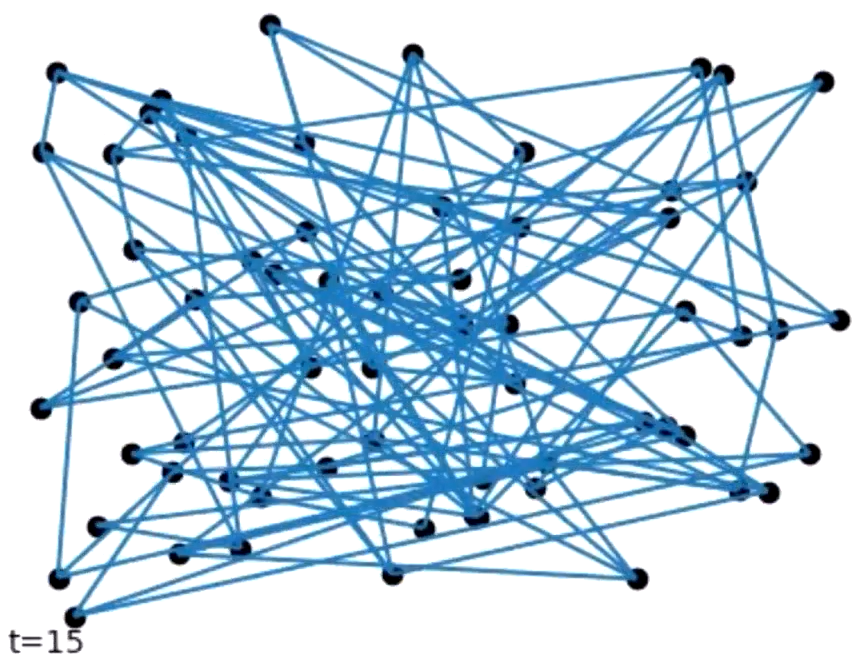
# SOM bidimensionale su un manifold bidimensionale



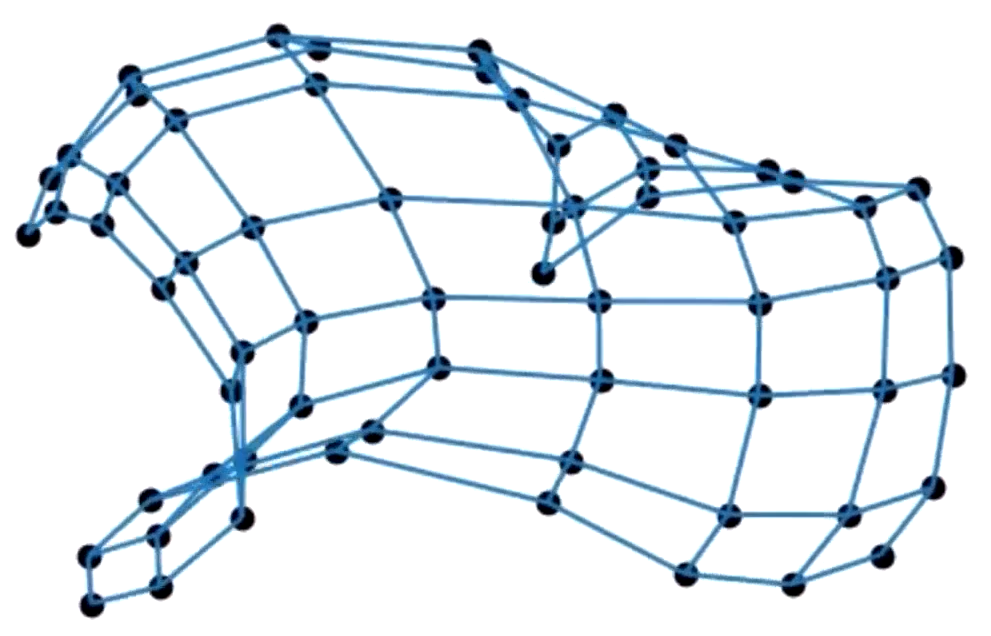
Lezione del 23-10, min. 41:00 → video SOM 2D, manifold 2D.

Si cerca di apprendere una griglia bidimensionale per rappresentare un manifold bidimensionale. Si tratta di un manifold di scarso interesse: sono punti a caso che riempiono l’intero spazio.

Situazione iniziale

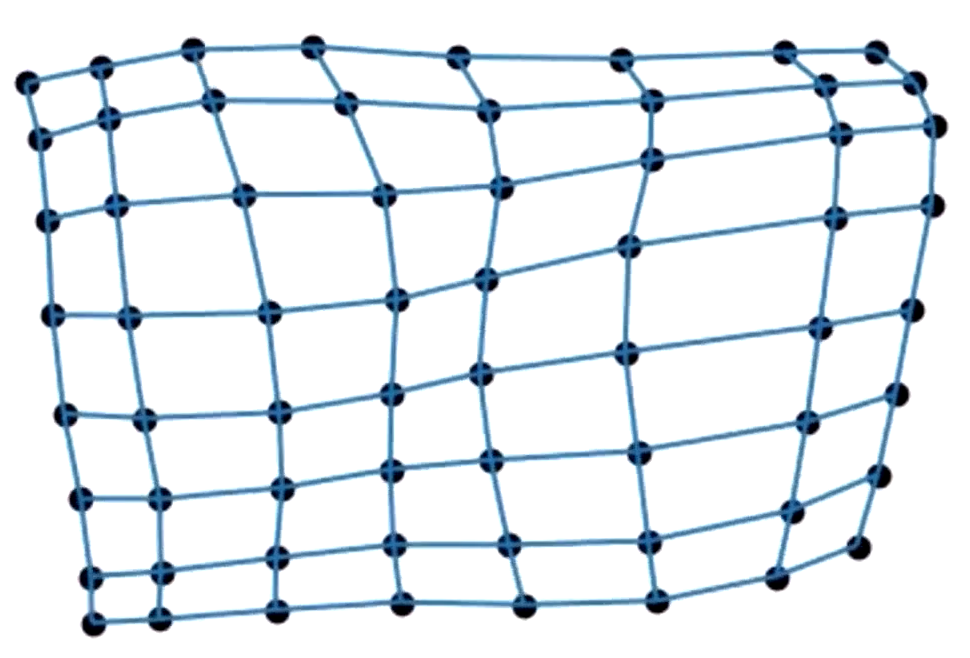


La SOM inizia con posizioni casuali e man mano inizia a districare questa matassa.



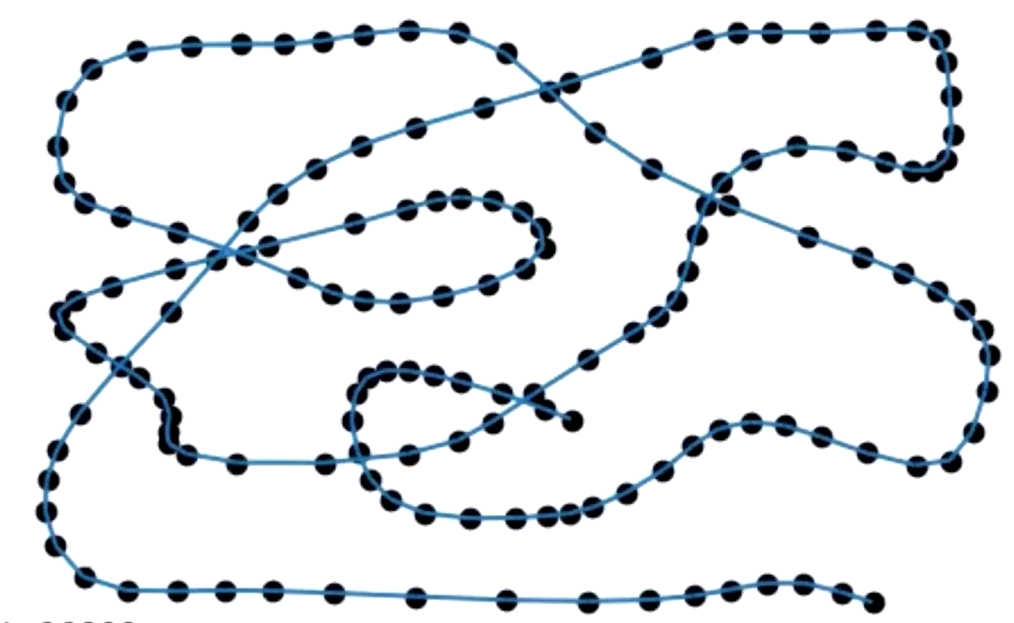
Esistono dei punti difficili da risolvere perché i neuroni si danno fastidio l’uno con l’altro.

Ma in questo caso, la SOM riesce a risolverli. Situazione finale.



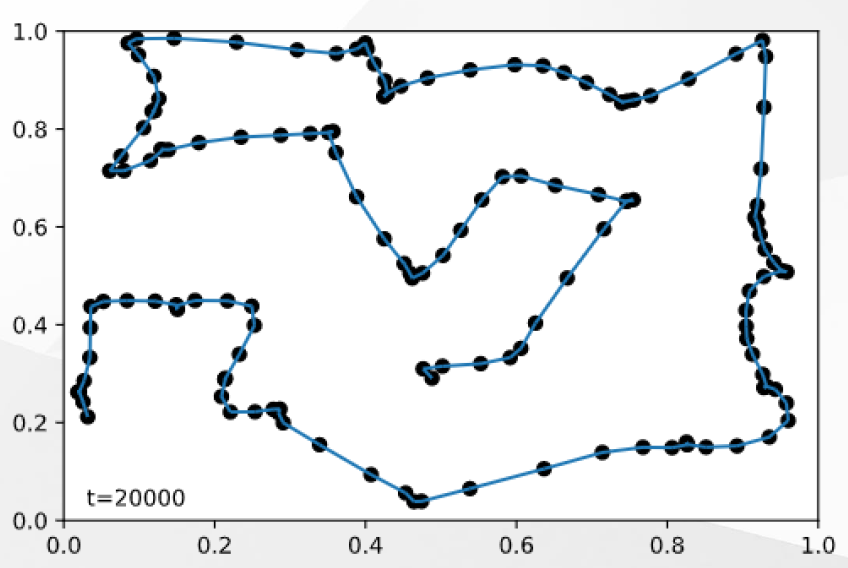
# SOM monodimensionale su un manifold bidimensionale

Nel seguente caso, la SOM non potrà mai risolvere il task in modo soddisfacente perché i dati sono descritti in modo bidimensionale. La SOM cercherà di ottenere un'approssimazione dei dati usando una rappresentazione dei dati che è troppo povera per cogliere questi dati.



La SOM cerca di mantenere la topologia dello spazio in cui siamo. E’ vero che la maggior parte dei punti sono vicini topologicamente a loro. Ciò che non va bene è che abbiamo una sola dimensione → topologia della SOM molto distante da altri punti.

La rete SOM finisce in una situazione abbastanza ingarbugliata. L’apprendimento di questi oggetti dipendono tantissimo dal learning rate iniziale e dal parametro sigma iniziale. Giocando su questi parametri si possono ottenere risultati leggermente migliori, come il seguente.



Ha gli stessi difetti dell’altro ma ha risolto tutti i nodi con 20mila iterazioni.

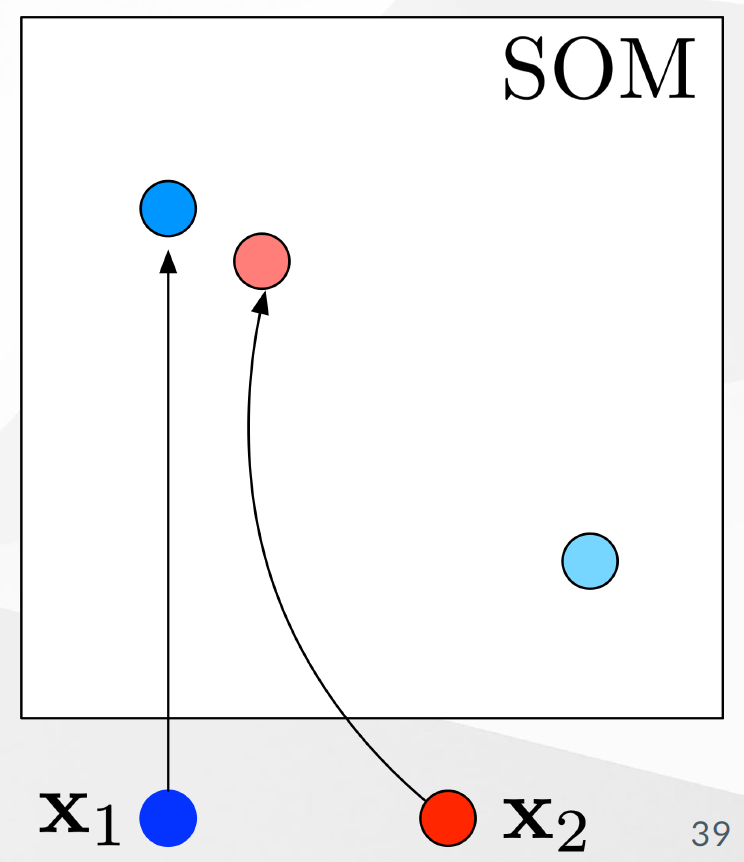
# Qualità della SOM

Come capire se le SOM si stanno comportando bene o male?

Bisogna tenere conto che ci interessano due aspetti importanti:

* misurare la **qualità dell’apprendimento** (qualità della **quantizzazione dei vettori**): corrisponde a quanto i neuroni riescono ad approssimare bene i dati di input. Se i neuroni sono vicini ai dati di input, allora la qualità è alta. Se sono tutti molto lontani dal dato in input, la qualità è bassa.
* misurare la **qualità della proiezione** effettuata dalla SOM: è la qualità dell’**ordinamento topologico**. Se oggetti simili sono in posizioni vicine allora è alta; se oggetti simili sono in posizioni lontane allora è bassa.

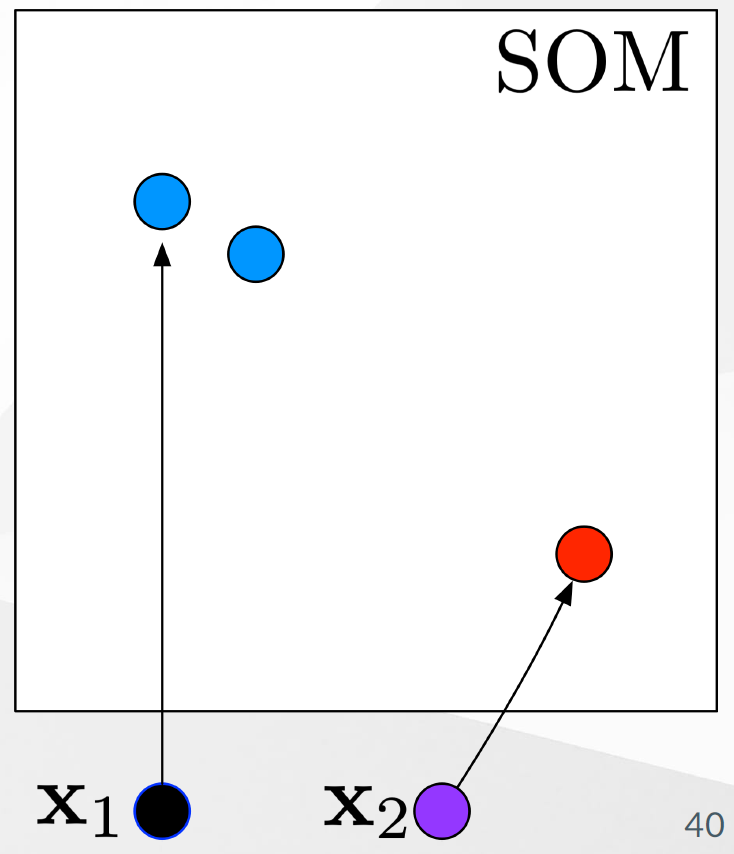
Caso 1



Errore di apprendimento: basso → colori simili sono mappati su colori simili.

Errore di ordinamento topologico: alto → i due blu sono distanti.

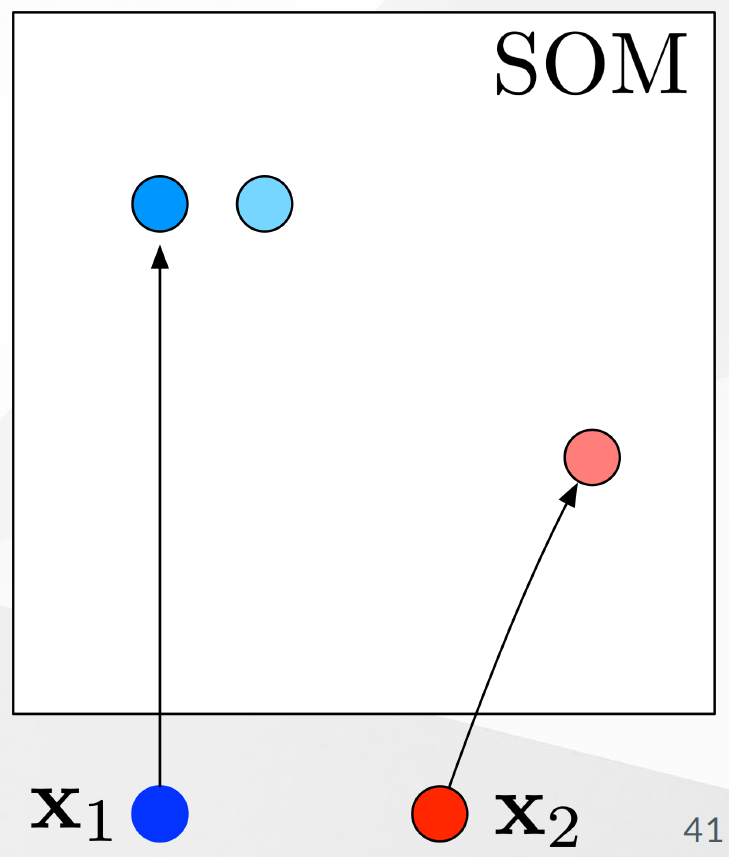
Caso 2



Errore di apprendimento: alto → nero mappato con blu

Errore di ordinamento topologico: basso → blu vicini, rosso distante

Caso 3



Errore di apprendimento: basso

Errore di ordinamento topologico: basso

Per misurare la qualità della SOM esistono tre misure:

* Quantization Error → errore di quantizzazione
* Topographic Error → errore della topologia
* Distortion Measure → mette insieme i due aspetti

# Errore di quantizzazione

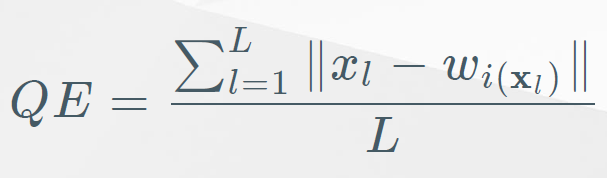
**L’errore di quantizzazione misura quanto bene i dati sono mappati sulla SOM**.

Dato in input uno stimolo x alla SOM che viene descritto con una certa descrizione. Lo stimolo x viene mappato su una Best Matching Unit, ciò che interessa è la distanza tra lo stimolo in input e la rappresentazione della BMU.

Quantizzazione: l’input x è mappato nella SOM su una BMU. La BMU è descritta dal vettore w.



L’errore di quantizzazione è alto se la descrizione di x è molto diversa dalla descrizione di w.



Si calcola con il valore atteso (media aritmetica) della distanza euclidea tra il vettore input e il valore della BMU

peso w e i(x) indice della BMU

Il valore atteso è approssimato dalla media aritmetica.

L’errore di quantizzazione misura la qualità della quantizzazione dello spazio di input.

Non c’è alcun tentativo di misurare la qualità della topologia indotta dalla SOM.

# Topographic Error

La topologia è ben preservata se è vero che punti vicini nello spazio di input, allora è vero che sono vicini anche nello spazio di output.

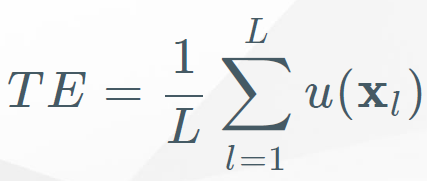
Input: x1 e x2 sono vicini

Output: i(x1) è vicino a i(x2) → x1 e x2 sono mappati in punti vicini nella mappa.

**L’errore topografico misura la percentuale dei data points in cui la prima e la seconda BMU non sono adiacenti nello spazio di output**.

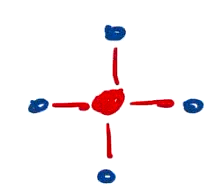
Dato x1, calcolo l’indice i(x1) della BMU e l’indice j(x1) della seconda BMU, cioè l’unità che si attiva in modo più forte subito dopo i(x1).

Se i e j non sono vicini tra loro nella mappa, allora esiste un errore topografico.

 media aritmetica della funzione u(x)

**u(x) = 1** se la prima e la seconda BMU di x **non** sono adiacenti (collegate) nello spazio di output.

**u(x) = 0** altrimenti



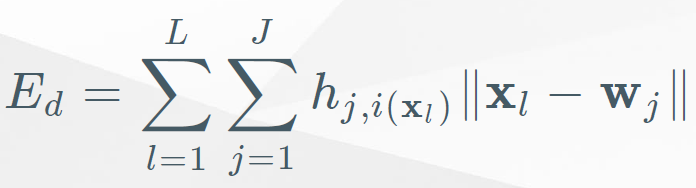
La seconda BMU deve essere uno dei vicini della BMU (rosso), altrimenti esiste un errore topologico.

# Misura di distorsione

La misura di distorsione cerca di catturare tutte e due le proprietà.

Una delle difficoltà nel misurare quanto bene sta andando la SOM è dovuta al fatto che si può dimostrare che non esiste una funzione di loss fissa che la SOM ottimizza nell’apprendimento.

**Ma modificando la regola di apprendimento e assumendo di lavorare con un training set finito, si assume che la funzione h abbia un’ampiezza fissa (non diminuisce), allora si può mostrare che la quantità usata nella regola di update nella SOM è un modo di approssimare efficientemente il gradiente della misura di distorsione**.



Sotto l’ipotesi che la funzione h ha ampiezza fissa, l’algoritmo di apprendimento della SOM applica la discesa del gradiente usando la misura di distorsione come funzione di loss.

La misura di distorsione cattura in parte l’errore di quantizzazione e in parte l’errore topologico.

Somma su tutti gli esempi, somma su tutte le unità della SOM, si calcola il prodotto tra la funzione h e la distanza euclidea.

Si tratta quindi di un **errore di quantizzazione pesato dalla distanza topologica**.

Si può dimostrare che la **misura di distorsione si compone di tre parti**:

1. una componente misura la **qualità di quantizzazione** dello spazio di input
2. una componente misura **la qualità della costruzione della topologia**
3. una componente che **connette le due componenti** precedenti

# Applicazioni

Le SOM hanno avuto molteplici applicazioni:

* analisi dei processi
* geoscienze, controllo remoto
* analisi delle immagini, pattern recognition
* IT
* design dei sistemi intelligenti
* sistemi esperti e data mining

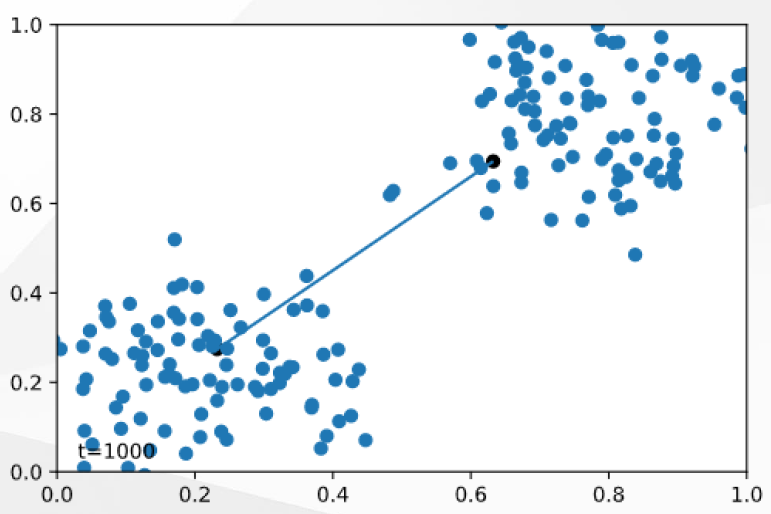
Oggi giorno, si vede un uso meno frequente delle SOM perché in tanti di questi ambiti ha preso piede il deep learning.

Le SOM hanno avuto ampio spazio prima dell’evoluzione del deep learning.

Vantaggi nel costruire una SOM:

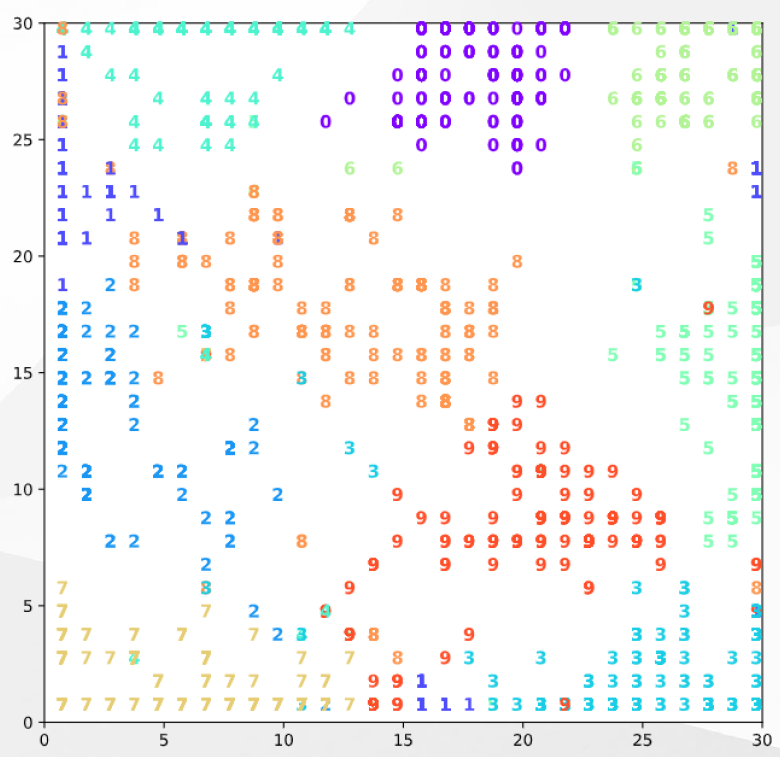
* la SOM può essere usata per fare **clustering** dei dati
* per **visualizzare dataset con dimensionalità alta**
* per discretizzare il dataset in un numero piccolo di vettori, utile se stiamo facendo **compressione dei dati** e per migliorare la comunicazione
* per semplificare un task di apprendimento usando le SOM come uno step di **pre-processamento** associato con un classificatore semplice.

## Clustering dei dati



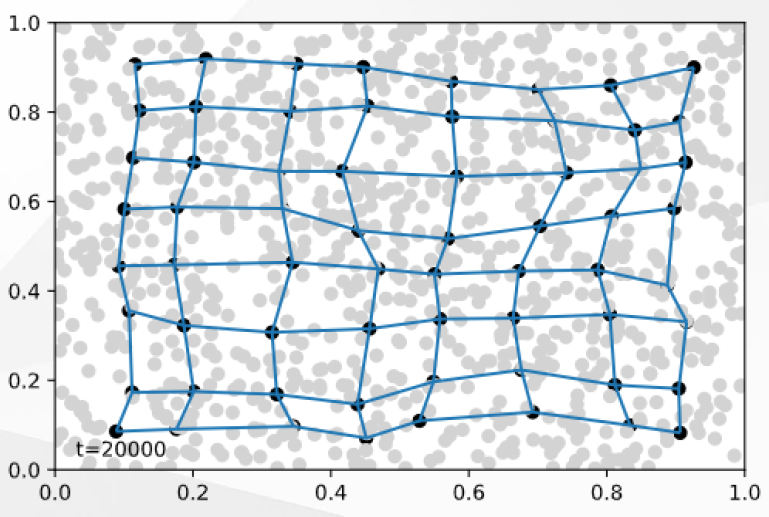
I neuroni neri sono i centroidi dei due cluster.

## Un modo per visualizzare dataset ad alta dimensionalità



Ogni punto di questa immagine rappresenta un immagine di 28x28 pixels → 784 dimensioni. Ogni immagine ha 784 attributi. Si sta visualizzando i punti in uno spazio bidimensionale.

## Data compression



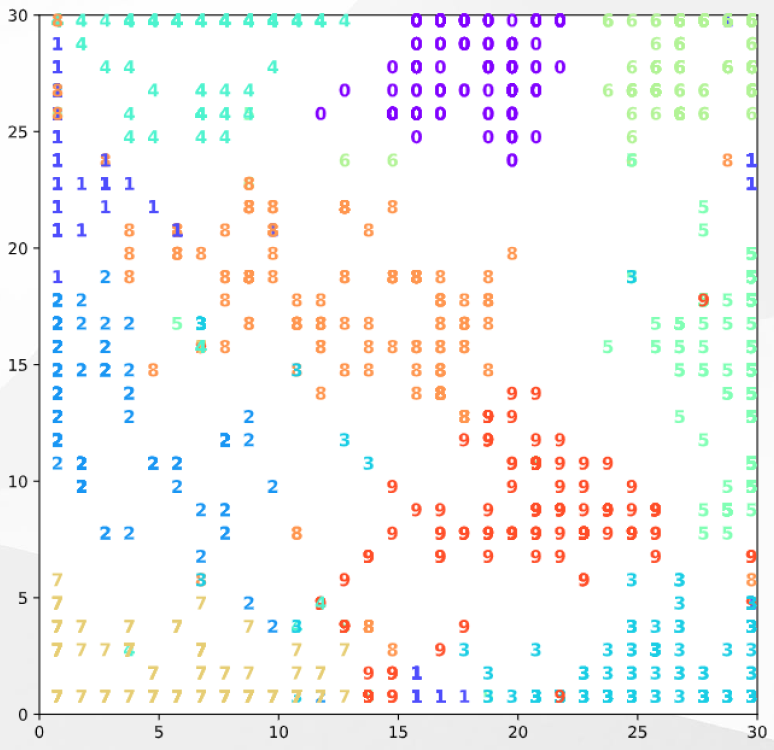
Idea: immaginate di voler inviare sulla rete i punti in grigio. Per inviarli, bisogna inviare le coordinate. Si può decidere di ammettere un po’ di errore dovuto alla compressione dei dati e inviare i punti neri a cui sono più vicini, usando la SOM per fare quantizzazione dello spazio.

64 punti neri → 6 bit per inviare l’informazione.

Pago in termini di precisione, non sto più inviando il punto preciso ma la quantizzazione dello spazio.

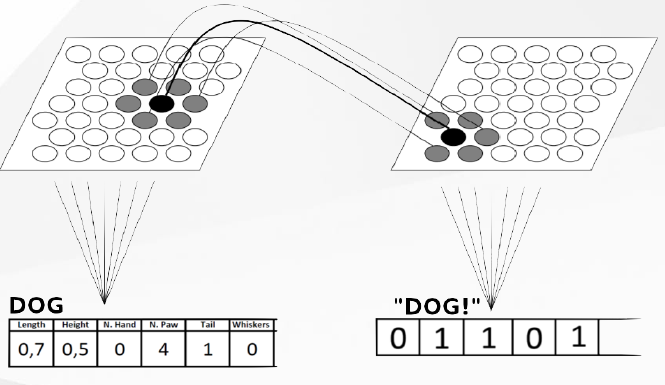
Nella realtà i dati vivono spesso in uno spazio con tante dimensioni ma si possono rappresentare con poche dimensioni. Non perdo nella loro descrizione anche se uso poche dimensioni e riesco ad ottenere un buon risultato.

## Semplificare il task di apprendimento



Si immagini di costruire un albero di decisione su questa mappa. Gli alberi di decisione molto facilmente riescono a catturare le zone identificate dalla SOM, semplifica enormemente il task anziché analizzare l’immagine.

## Mayor and Plunket word learning model



E’ un modello presentato da Mayor e Plunket, due cognitivisti. Avevano intenzione di modellare come i bambini apprendono l’associazione immagine-parola. In un bambino, bastano poche ripetizioni: guarda l’immagine di un cane per associare l’immagine con la parola “cane”.

Ciò si scontra con la complessità dei segnali che riceve il bambino. L’immagine e il suono sono oggetti complicati da analizzare.

M. e P. hanno ipotizzato che l’immagine viene mappata nella corteccia cerebrale visiva in modo topologico come avviene nella SOM su un certo numero di neuroni che si eccitano.

Il suono analogamente è mappato nella corteccia uditiva.

Nel momento in cui entrambe si eccitano, tramite un meccanismo di apprendimento in piano che rinforza le connessioni, si riesce ad apprendere la connessione tra immagine e suono in entrambe le direzioni.