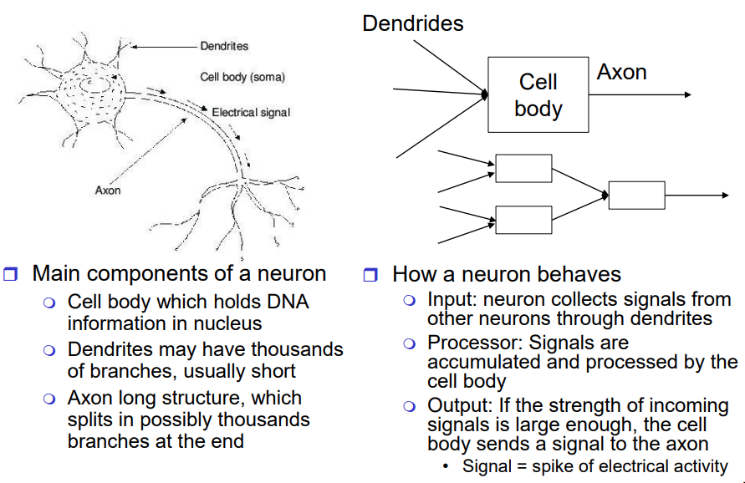
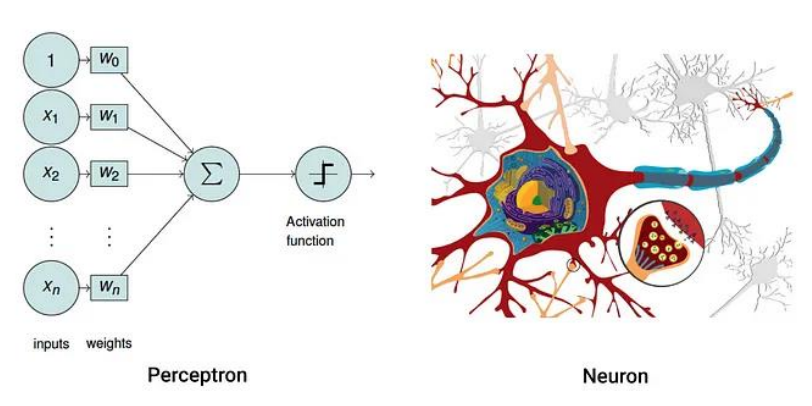
**RETI NEURALI** – NEURAL NETWORKS

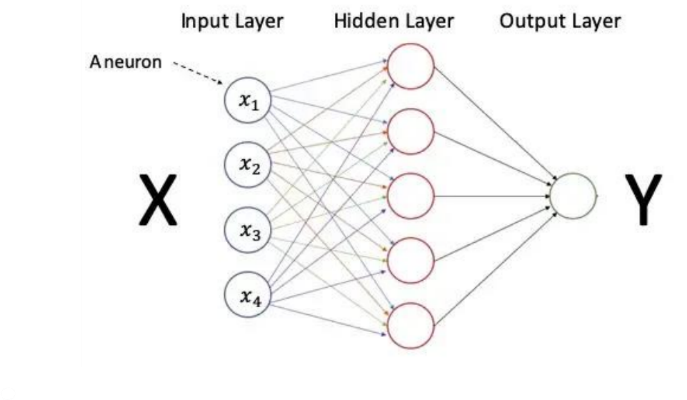
*Cosa è una rete neurale?*

Oltre al nucleo (neurone) abbiamo la parte di input del neurone e la parte di output.

L’informazione in input viene mandata alla cellula (Core della cellula = neurone). Più neuroni si compongono in serie. Le reti artificiali si rifanno al nostro sistema nervoso

* Le reti con feedback fanno in modo che lo stato attuale si somma con l’input in modo da influenzare l’output.
* Le reti neurali fanno una cosa diversa: creano una relazione tra input e output tenendo conto di ciò che è successo in passato (addestramento).

*ANN - Artificial Neural Network*

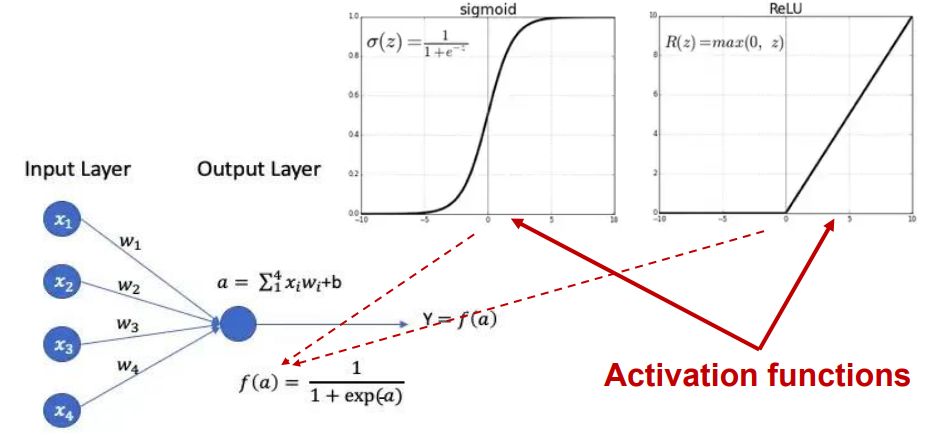
Le reti neurali artificiali sono algoritmi matematici in cui ogni neurone avrà uno o più assoni (decidono se inviare il segnale su 1 o più assoni (OUTPUT))

Le features che andiamo a caricare sono mentre y rappresenta l’assone.

Le features vengono pesate con il peso () indicato sugli archi (vedi figura sotto).

Le features passano poi in funzioni strettamente non lineari, tipicamente si usano la Sigmoide e la ReLU.

**NOTA** - Nel caso di una ANN con un singolo neurone la rete neurale collassa al caso di Logistic Regression

**Sigmoide**

La sigmoide approssima la funzione gradino in quanto quest’ultima non è derivabile e quindi non sarebbe utilizzabile per calcolare i pesi tramite il metodo del gradiente.

Le funzioni non lineari permettono di emulare il comportamento di un neurone reale: ovvero, a seconda del peso faccio passare o meno l’output.

La configurazione della rete neurale consiste nella scelta della funzione di attivazione e della scelta dei pesi (METODO BACK PROPAGATION: vedi ML)

A noi interessa una tipica configurazione delle reti neurali:

* L’altra volta abbiamo visto che la classificazione binaria ATTACCO/NON ATTACCO andava benissimo, però una volta identificato l’evento, l’algoritmo kNN richiedeva troppo tempo per discriminare il tipo di attacco.
* Ho 4 tipi di attacco, questo implica ce devo avere 4 output e di conseguenza si può addestrare una rete neurale per vedere se il suo funzionamento è migliore di kNN. La rete neurale può essere lenta nell’addestramento (per il calcolo dei pesi). Poi è molto veloce (Non lo vedremo).
* Altro approccio: abbiamo 2 approcci (USATI IN PARALLELO):
  1. Conosco i profili associati alle anomalie, quindi posso allenare il modello nel dataset con anomalie note (Supervised Learning): classifica anche il tipo di attacco
  2. Conosco solo il profilo del traffico benigno e vado a vedere se nei dati di test c’è qualcosa che non ha (si discosta dal traffico benigno) => Unsupervised (Distinguo solo BENIGNO/MALIGNO con k-Means: 2 CLUSTER) => uso reti neurali per fare Unsupervised Learning. Uso reti neurali profonde pensate per rilevare le anomalie.

**Differenza rete neurale e rete neurale profonda**

* Rete neurale profonda

Ha almeno due livelli di neuroni nascosti. Vengono utilizzate perché hanno un capacità migliore di apprendimento. Ogni livello aggiunge un grado di astrazione

Hanno il problema dei Decision Tree: soffrono del problema di incorrere in over fitting.

Soluzioni Decision Tree:

* 1. Fisso la profondità
  2. Utilizzo una foresta di alberi: non a tutti gli alberi faccio vedere l’intero dataset => l’addestramento avviene su porzioni simili, ma non uguali del dataset. Quando utilizzo le foreste, all’interno dell’albero, quando decido di splittare vado a considerare un sottoinsieme delle features, non tutte => con gli alberi vado a nascondere parte delle informazioni appartenenti al dataset => per generalizzare non gli faccio vedere tutti gli input.

Nelle reti neurali faccio la stessa cosa: gli nascondo delle informazioni. I due meccanismo fondamentali sono:

1. Funzione di attivazione.
2. Meccanismo che permette di eliminare alcuni neuroni.

**ReLU -** Vantaggi di una funzione di attivazione a soglia

1. Nelle reti neurali abbiamo il problema che il gradiente tende a scomparire (Sigmoide/Tangente Iperbolica). Il gradiente deve convergere. Se va a zero => perdiamo convergenza. Se è molto piccolo => abbiamo bisogno di più incrementi.

Ciò è dovuto dal fatto che sia la tangente iperbolica che la sigmoide tendono ad andare verso un valore costante => il gradiente è basso (…) anche con valori di x grandi (i valori di x grandi li posso avere solo in input, perché la funzione di attivazione li trasforma in valori compresi tra 0 e 1).

Nella ReLU invece, se un valore è alto, rimane alto, mentre tutti i neuroni che danno 0 o un valore negativo, vengono spenti.

1. Altro comportamento che tende a limitare l’over fitting.

DROP OUT: avviene nella fase di training e permette di eliminare alcuni neuroni sui vari livelli. Il processo viene fatto in maniera controllata tramite una funzione di probabilità. Ho 9 neuroni, se considero una p = 0.33 implica che spengo 3 neuroni.

Le reti che ottengo sono molto artificiali, quindi devo fare dei TRIAL and ERROR per vedere se va bene.

**AUTOENCODER** – Rete Neurale Profonda

Sono usate per:

* Riduzione delle dimensione;
* Compressione delle immagini;
* …
* Rilevazione delle anomalie.

*Come funziona?*

È una rete neurale profonda in cui voglio che l’output sia uguale all’input.

Ho due fasi:

* Codifica: utilizzano input per ottenere l’informazione.
* Decodifica: utilizzano questa informazione per estrapolare l’input.

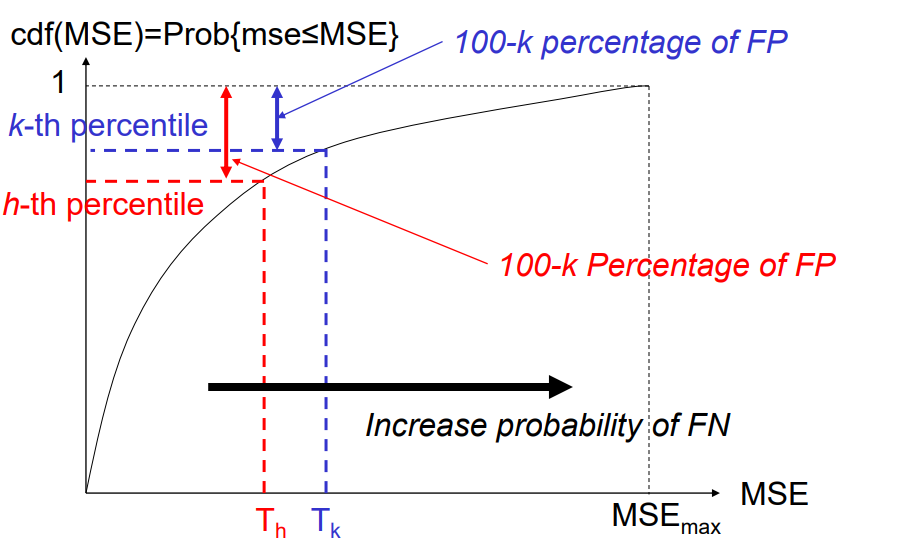
**NOTA** -Informazione: ciò che conta e che permette di ricostruire l’input reale.

Tali reti neurali non hanno bisogno di un target, dato che il target è l’input che inserisco. Dobbiamo codificare il traffico estrapolandole caratteristiche più importanti. Facciamo training solo con il traffico benigno (senza anomalie). Se lo addestro bene nel traffico benigno => nel momento in cui arriva un attacco => avrò un alto errore di decodifica => vedo che è maligno.

La rete di pagina 47 ha una strettoia: deve esserci, altrimenti alcuni neuroni farebbero un lavoro sporco. Avere la strettoia significa costringere la rete ad eliminare le informazioni sovrabbondanti. Nella strettoia abbiamo il max livello di comprensione delle informazioni.

Il numero di livelli di input (features) deve essere uguale al numero di livelli di output (features ricostruite). Per vedere di quanto discostano => uso una qualsiasi metrica (errore quadratico).

1. Il dataset su cui addestro non deve contenere anomalie (possono esserci OUTLIER).
2. Devo scegliere il numero di livelli
   * Livello INPUT
   * Livello OUTPUT
   * Livello CORE
   * Livelli INTERMEDI: quanti ne mettiamo?
3. Ho una anomalia quando l’errore di ricostruzione è alto. Quando è alto? Non esiste un valore assoluto. Solitamente si sceglie una soglia. Oppure utilizziamo il dataset di training, si valuta una statistica sull’errore di ricostruzione dei vari campioni.



[MATRICE DI CONFUSIONE]

Se l’errore > allora ho un falso positivo. Perché verrà rilevata una anomalia, ma l’anomalia nel dataset non c’è.

Se fisso il valore a 0.99 => tollero meglio l’errore di ricostruzione => crescono i falsi negativi (non va bene)

Nel dataset faccio una regolarizzazione con MinMaxScaler. Se come funzione di attivazione utilizzo la ReLU, nell’output non le posso usare. Nel ramo di output utilizzo una sigmoide, dato che la sigmoide permette di avere valori € [0,1] (mentre la ReLU polarizzerebbe il valore).

Auto encoder = encoder + decoder

Metrica con cui valuto auto encoder: MSE – Mean Square Error (tra predizione e vero)

Encoder: livello da 20 – 10 – 5 (strettoia)

Decoder: livello da 5 – 10 – 20

Per trovare la soglia utilizzo una statistica sull’errore di ricostruzione dei campioni del dataset di training. Sulla statistica calcolo la soglia, ad esempio dico che la soglia è pari al 95-esimo percentile della metrica.

Falsi negativi -> eventi anomali in cui MSE è sotto la soglia.

Falsi positivi -> eventi benigni in cui MSE è sopra la soglia.

Calcolo MSE del dataset di training: benigno.

Considero come soglia il 95-esimo percentile del MSE

Se MSE di un campione > MSE generale >0, questo implica che è un anomalia, altrimenti è benigno.

Se fisso la soglia al 99-esimo percentile => tollero meglio i falsi negativi