Metodi Numerici per L'intelligenza Artificiale.

Andrea Cecchini

4 marzo 2023

Indice

1 Introduzione all'Analisi Numerica.

1.1 Analisi Numerica.

Introduciamo nel definire il compito dell'analisi numerica.

Analisi Numerica.

L'Analisi Numerica è la parte di matematica che si occupa di dare una **risposta numerica** ad un problema matematico che modellizza un problema reale.

1.1.1 Fasi della risoluzione di un problema numerico.

Al fine di raggiungere tale problema, ci avvaliamo delle seguenti fasi:

- **Tradurre** il problema reale in un insieme di equazioni matematiche in grado di descriverlo
- **Trasformare** il problema matematica nel continuo in un problema numerico discreto che sia risolubile.
- **Trasportare** il problema discreto in un calcolatore mediante l'applicazione di algoritmi numerici capaci di determinare la soluzione in un tempo ottimale.
- **Interpretare** la soluzione numerica nei termini della situazione reale e **verificare** così sia l'adeguatezza del modello matematico sia l'efficienza dell'algoritmo risolutivo.

1.1.2 Errori nel risolvere un problema numerico.

Nel percorso appena descritto vi possono essere numerevoli errori, le quali sorgenti sono:

- **Errori nel modello matematico** Nascono da una cattiva traduzione del problema reale a quello matematico, per esempio si considerano alcune cose come trascurabili quando non lo sono.
- **Errori nel modello numerico-computazionale** Vengono descritti come errori di *discretizzazione* o *troncamento*.
- **Errori presenti nei dati** Nati da uno strumento di misurazione fallace o da misurazioni che possono essere influenzate da errori sistematici.
- **Errori di arrotondamento nei dati e nei calcoli** Sono gli errori introdotti nella rappresentazone dei numeri sul calcolatore.

1.2 Classificazione dei problemi numerici

1.2.1 Problema numerico

Problema Numerico.

Per **problema numerico** intendiamo una descrizione chiara di una **relazione funzionale** tra i dati (**input**) e i risultati (**output**).

In particolare, in un problema numerico abbiamo i seguenti elementi:

- **F** rappresenta la relazione funzionale tra input ed output.
- **x** rappresenta il dato di input della relazione funzionale.
- **y** rappresenta l'output dell¡a funzione di un determinato input

1.2.2 Classificazione dei problemi numerici.

Descritti questi 3 elementi, è possibile classificare il problema numerico in base a cosa stiamo cercando:

- **Problema diretto F** e **x** sono dati, bisogna **trovare y**.
- **Problema inverso F** e **y** sono dati, bisogna **trovare x**.
- **Problema di identificazione x** e **y** sono noti, bisogna trovare **F**.

Quest'ultimo problema è quello che interesserà di più durante il corso, perchè è proprio il problema numerico che l'intelligenza artificiale cerca di risolvere.

Summary

Abbiamo introdotto la materia dell'**analisi numerica** e quello che si prefissa di risolvere. Successivamente abbiamo definito il concetto di **problema numerico** ed abbiamo elencato i diversi tipi, quali **problema diretto**, **problema inverso** e **problema di identificazione**.

2 Introduzione all'Intelligenza artificiale

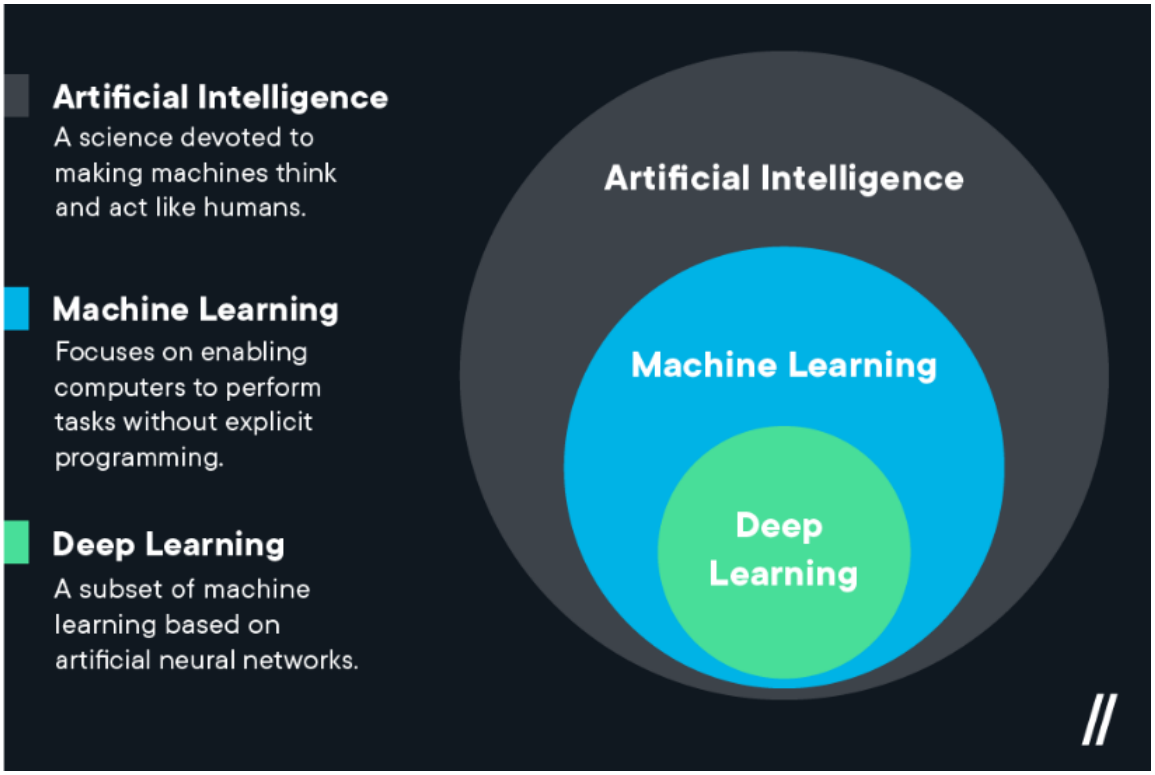
Intelligenza Artificiale.

Per **intelligenza artificiale** si intende una parziale riproduzione dell'attività intellettuale propria dell'uomo.

Esistono due tipi di intelligenze artificiali, basate sul loro dominio applicativo:

- **Intelligenza artificiale "debole"**: Sono dei sistemi basati per risolvere **problemi specifici.**
- **Intelligenza artificiale "forte"**: Sono dei sistemi in grado di replicare **tutte le funzioni cognitive** dell'essere umano. Spesso per riferirci a questa categoria useremo il termine **"Intelligenza Generalista"**.

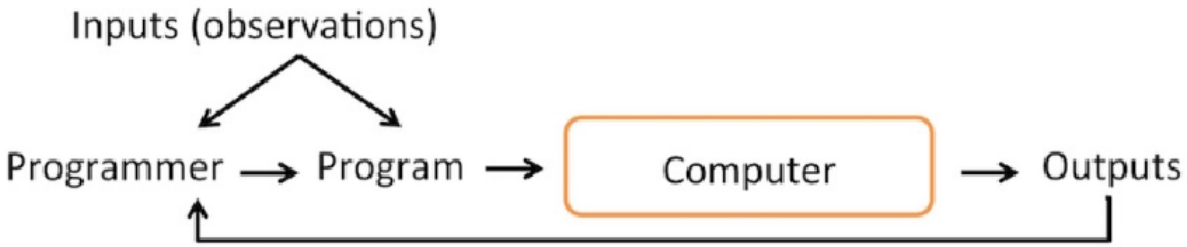
Molti tendono a confondere e a non capire il legame tra **intelligenza artificiale**, ,**machine learning** e **deep learning**.
Rappresenteremo il loro legame attraverso il seguente diagramma di Venn:



Detto ciò, capiamo cosa cambia grazie all'utilizzo di questa tecnologia.

2.1 Cambio di paradigma di programmazione

2.1.1 Paradigma classico

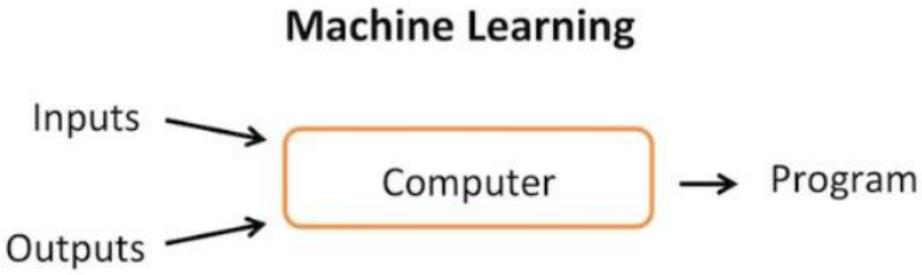


Il paradigma sul quale noi siamo abituare a creare programmi è il seguente:

- Il **programmatore** elabora e crea un **algoritmo** (programma).
- All'algoritmo vengono forniti dei dati come **input**.
- L'elaboratore computa l'input sulla base dell'algoritmo e fornisce un output.

Questo modo di agire prevede una forte presenza dell'essere umano, il quale in veste di programmatore, crea l'algoritmo voluto.

2.1.2 Paradigma del Machine Learning



In questo paradigma il ruolo dell'uomo viene "sostituito" dalla tecnica del **Machine Learning**.

Machine Learning.

Sistema in grado di apprendere automaticamente da esempi specifici (**training data**) e di generalizzare la conoscenza su nuovi campioni (**test data**) dello stesso dominio

Difatti, il machine learning dato input ed output, procede nella risoluzione del problema di identificazione.
Andiamo ad analizzare le diverse fasi di questo paradigma:

- **Acquisizione dati** I dati sono l'elemento base di tutte le applicazioni di M.L.. E' molto importante quindi sapere come acquisire questi dati.
- **Data processing** I dati raccolti nella fase precedente vengono processati al fine da attarli al meglio al al modello M.L. che intendiamo sviluppare.
- **Modello** Insieme di techiche matematiche e statistiche in grado di apprendere da una certa distribuzione di dati.
- **Predizione** Una volta ottenuto il modello è possibile "predire" l'output correlato ad un certo input non presentato nel training data.

2.2 I dati

2.2.1 Acquisizione dei dati

E' possibile ottenere i dati in due modi:

- Usare set di dati pubblici: sono presenti molte piattaforme, come kaggle.
- Acquisento un nuovo set di dati.

E' molto comune nel mondo della ricerca di fornire questi set di dati pubblici, in modo altruista. Visto ciò, non usarli sarebbe un peccato.

2.2.2 Annotazione dei dati

Etichetta.

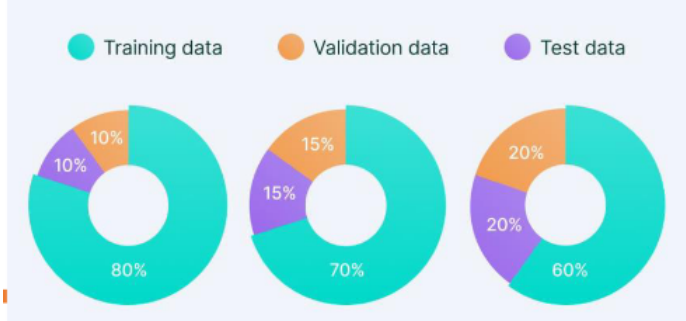
Annotare i dati vuol dire assegnare un **etichetta** (output) ad una determinata istanza di input. L'etichetta rappresenta il contenuto semantico dei dati.

Diremo quindi che un dato è **annotato** se associato ad una **etichetta**.
I dati non annotati sono spesso **inutili**. Tuttavia, grazie alla tecnica di apprendimento **non supervisionata** (vedremo dopo) è possibile comunque estrarre conoscenza da essi.

2.2.3 Organizzazione dei dati

Bisogna **organizzare** i dati come segue:

- **Training set** sono i dati sui quali il modello apprende automaticamente durante la fase di apprendimento.
- **Validation set** sottoinsieme del training set, sono i dati con il quali si informa il sistema della validazione del suo apprendimento.
- **Testing set** dati con il quali si testa il modello. Questa fase verifica l'efficacia del modello, anche attraverso misure numberiche qualitative e quantitative.



Nell'immagine qui sopra si elenca differenti proporzioni in cui si dovrebbe suddividere il set di dati che abbiamo nei subset descritti precedentemente.

2.3 Complessità dei dati

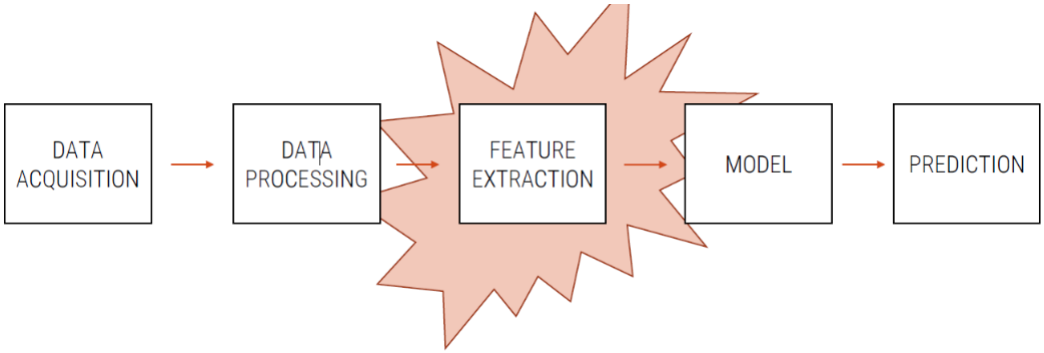
Dimensionalità.

La **dimensionalità** di un dato rappresenta la **densità** di quest'ultimo, ovvero la quantità.

Complessità.

Diremo che un dato è complesso se presenta una **alta dimensionalità**.

Dare al sistema di M.L. una mole spoporzionata di dati, come tutti i pixel di un'immagine, non è una buona cosa.
Se stessimo lavorando su un classificatore di immagini, sarebbe un errore grave dargli dati ad alta dimensionalità, in quanto non riuscirebbe ad apprendere da così tanti dati **inutili**.
La soluzione a questo problema si chiama **feature extraction**.



2.3.1 Feature extraction

Feature.

La **feature** srappresenta la parte più utile del dato grezzo.

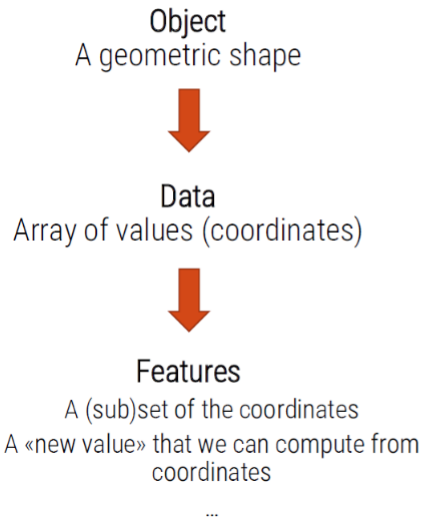
Feature Expansion.

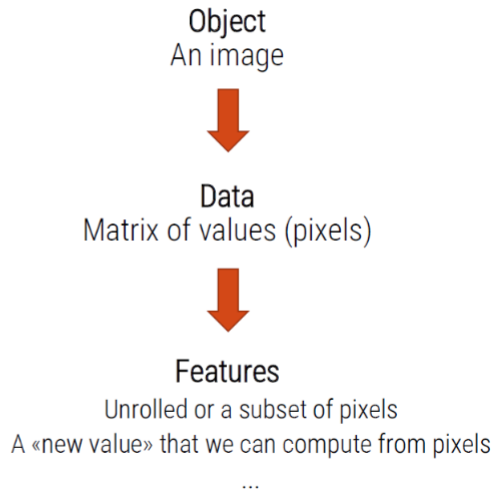
Rappresenta l'operazione di estrazione di features dal dato grezzo.
E' un modo per creare un nuovo e più piccolo insieme di dati che cattura la maggiore parte dell'informazione dei dati grezzi.

Feature Descriptor.

Un **featuer descriptor** rappresenta un vettore n dimensionale di feature numeriche che rappersentano qualche oggetto.

-


Object
A geometric shape
Data
Array of values (coordinates)
Features
A (sub)set of the coordinates
A «new value» that we can compute from coordinates
...


Object
An image
Data
Matrix of values (pixels)
Features
Unrolled or a subset of pixels
A «new value» that we can compute from pixels
...

2.4 Machine Learning tasks

Il machine learning offre diversi **task** a seconda dell'output che vogliamo

- **Classificazione**
- **Regressione**
- **Clustering**

2.4.1 Classificazione

Classe.

Una classe è un **set di dati con proprietà comuni**.
Il concetto di classe è correlato al concetto di "etichetta".

Classificazione.

Dato un input specifico, il modello (**classificatore**) emette una **classe**.

- Se ci sono 2 classi, chiamiamo il problema come **problema di classificazione binaria**
- Se ci sono n classi con $n > 2$, chiamiamo il problema come **problema di classificazione multiclasse**.

2.4.2 Regressione

Regressione.

La **relazione** viene utilizzata per **modellare la relazione** tra le variabili indipendenti e le variabili dipendenti.

Quindi la regressione si occupa di risolvere un **problema di identificazione**.

Regressione Multi-Variata.

La **regressione multivariata** prevede l'impiego di più variabili in gioco rispetto alla classica regressione lineare.

2.4.3 Clustering

Clustering.

Il **clustering** permette di identificare dei gruppi di dati in classi, senza saper a priori le classi.

Il clustering è spesso applicato, infatti, in un ambiente di apprendimento non supervisionato, in cui le classi del problema non sono noti a priori.

2.5 Tipi di apprendimento

E‘ possibile ragruppare i vari tipi di apprendimento di un algoritmo di machine learning in base all'effetivo convolgimento dell'attività umana.

2.5.1 Apprendimento supervisionato

Apprendimento Supervisionato.

Nell'**apprendimento supervisionato** i **training data** dati come input dell'algoritmo di machine learning sono **annotati**.

Esempi di tasks di machine learning dove si utilizza un sistema supervisionato sono la **classificazione** e la **regressione**.
Per quanto riguarda la classificazione, un esempi molto chiaro e‘ lo **spam filter**: l'algoritmo di machine learning e‘ allenato con un training set di esempi di email, indicando quali sono e non sono spam.

2.5.2 Apprendimento non supervisionato

Apprendimento non Supervisionato.

Nell'**apprendimento non supervisionato** i **training data** dati come input dell'algoritmo di machine learning non sono **annotati**.

Un esempio di task di machine learning dove si utiilizza un sistema non supervisionato e‘ proprio il **clustering**.
Immaginiamo di avere un bel blog alla Ferragni.
Abbiamo quindi una quantita‘ di dati riguardanti i visitatori del blog sufficientemente alta per allenare il nostro algoritmo di machine learning. Quello che vogliamo e‘ ragruppare i nostri visitatori, i quali inizialmente non sono annotati, in diverse classi. Quindi, partendo da dei dati **non annotati** ed utilizzando un sistema di **apprendimento non supervisionato**, siamo riusciti ad annotare i nostri dati ed a ragrupparli in diverse classi.

2.5.3 Apprendimento semi-supervisionato

Apprendimento semi-supervisionato.

Nell'**apprendimento semi-supervisionato** ci si affaccia con un **training set parzialmente annotato**, usualmenta tanti dati non annotati e pochi annotati.

Un esempio di apprendimento semi-supervisionato e' il servizio offerto da **Google Photos**: una volta caricate delle foto di famiglia sul cloud di google, e' possibile sapere in quali foto e' comparso un determinato parente. L'unica cosa che serve al sistema e' un'annotazione sulle foto iniziali, non su tutte, del nome del parente. Dalle foto successive il sistema sara' in grado di eseguire del clustering all' interno di foto, quindi assegnare classi a dati non annotati.

2.5.4 Apprendimento con rinforzo

Apprendimento con rinforzo.

Il sistema di apprendimento, chiamato **agent** in questo contesto, seleziona ed esegue delle azioni, ottenendo un **feedback positivo** o **negativo** sull'azione svolta.
Il sistema deve apprendere da solo quale e' la migliore strategia, chiamata **policy**, al fine di massimizzare il numero di feedback positivi , di ridurre invece i negativi, in **funzione del tempo**.

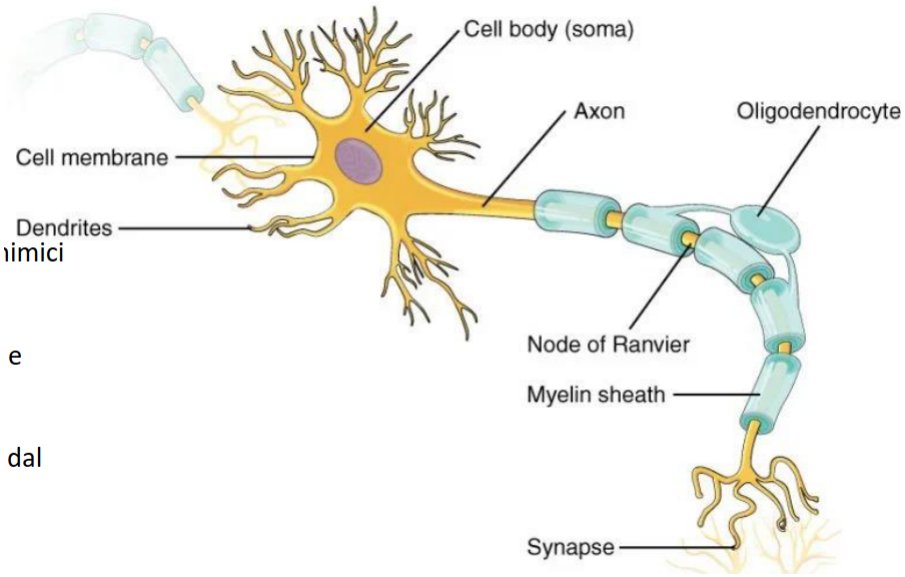
Fun fuct, l'apprendimento con rinforzo e' il tipo di apprendimento che un bambino effettua: le cose che impara le impara effettuando delle azioni dalle quali ricevera' un feedback dai propri genitori.

3 Artificial Neural Networks

Le **Artificial Neural Networks (A.N.N.)** sono il vero punto centrale **Deep Learning**.

3.1 Ispirazione dal modello umano

Il paradigma delle rete neurale artificiali e‘ ispirato al modo in cui il sistema nervoso biologio elabora le informazioni.
L’unita‘ elementare del nostro sistema nervoso si chiama **Neurone**.



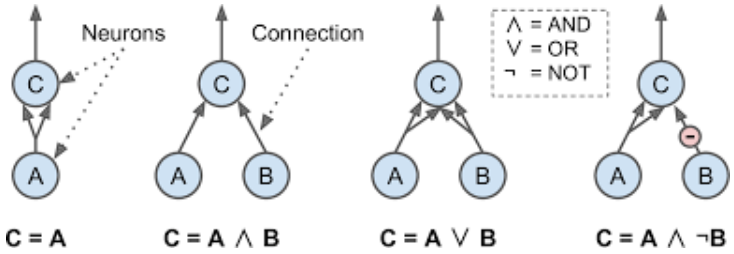
Le parti piu‘ importanti di un neurone sono:

- **Dentriti** sono i canali di ricezione dell’input proveniente da altri neuroni.
- **Soma** E la parte ”core” del neurone.
  Essa elabora l’input nel tempo e converte il valore elaborato in output.
- **Assone** Rappresenta il canale di trasferimento dell’ouput.
  L’assone trasferisce l’informazione elaborato dal soma ad altri neuroni.
- **Sinapsi** sono le estremita‘ dell’assone, le quali funzionano come dei ganci verso i dentriti di altri neuroni.

Quando il some riceve segnali in ingresso, e se quest’ultimi superano una certa **soglia**, viene prodotto un segnale di uscita sull’assone che diventera‘ un segnale di input per altri neuroni. **Il valore di questa soglia e l’efficienza chimica di trasmissione delle sinapsi sono strettamente legate ai processi di apprendimento.** Nella testa l’indicatore **wtf-per-minute** sara‘ andato alle stelle con questa lezione di bioologia, ma non vi preoccupate, rispetto a dopo e‘ ancora basso.

3.2 Neurone artificiale

Le reti neurali sono un tentativo dell'uomo di creare una rappresentazione informatica del nostro sistema nervoso. L'elemento base di esso, ovvero neuroni, vengono usati come caso di studio e base comportamentale per la creazione dell'elemento base delle reti neurali: il **Neurone artificiale**.
Il primo modello di neurone artificiale, create da McCulloch e Pitts, e' in grado di eseguire computazioni logiche dato dei segnali di input. Tale modello riceve e rilascia negnali di tipo **binario**, banalmente **Vero** o **Falso**. Immaginiamo ora di avere delle reti neurali, composte da un **layer di input e uno di output**, le quali eseguono delle computazioni logiche, assumendo che il neurone di output viene attivato (raggiunge la **soglia di eccitazione**) quando almeno 2 dei suoi input sono attivati.
Per **layer di neuroni** intendiamo, al momento, un'insieme di neuroni posti sullo stesso livello. Vedremo dopo in dettaglio.



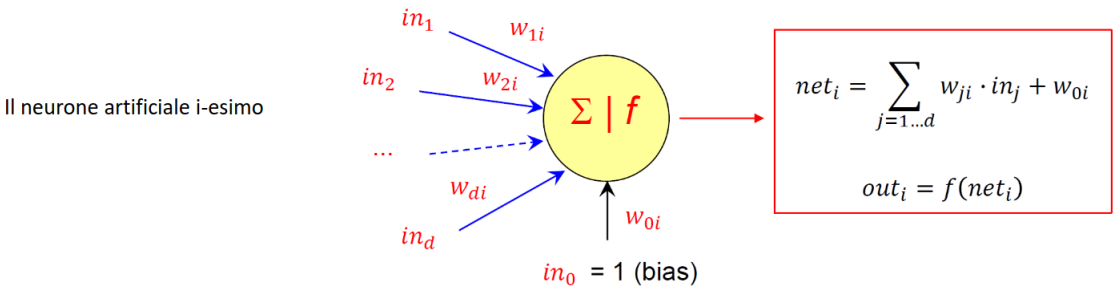
Le computazioni logiche sono le seguenti:

- **C = A**: se il neurone di input **A si attiva**, il neurone **C** riceve due input e **si attiva**
- **C = A AND B**: Sia il neurone **A** che il neurone **B** devono essere attivati al fine di attaviare il nuerone **C**.
- **C = A OR B**: Basta che uno dei due neuroni di input, ovvero **A** o **B**, sia attivato per attivare **C**.
- **C = A OR NOT B**: Questo caso simula la realta', ovvero quando un nuerone (**C**) viene inibito da un altro (**B**).
  In questo caso **C e' attivo** se **A e' attivo** e **B NON e' attivo**.

Ora immagina il risultato che puoi ottenere combinando tantissime reti neurali di questo tipo tra loro.

3.3 Il Perceptron

Il **Perceptron** e' uno delle architetture di reti neurali piu' facili.
Quest'archittettura e' basata su un altro tipo di neurone artificiale, chiamato **TLU**(**Threshold Logic Unit**) o anche chiamato **LTU**(**Linear Threshold unit**).
L'architettura Perceptron prevede **un solo layer di neuroni**, a volte anche uno solo.
In questo tipo di neuroni, **input** ed **output** non sono piu' dei semplici vero o falso, ma **sono dei numeri**.
Ogni **input** e' associato ad un **neurone i-esimo** attraverso un arco pesato di peso w_{ji} (vedremo i dettagli successivamente).
Il TLU **calcola una somma pesata degli input** (combinazione lineare), e successivamente applica una funzione chiamata **funzione di attivazione** per ottenere l'output.
Nel caso dei neuroni TLU, la funzione di attivazione prende il nome di
step function.



L'indicatore wtf-per-minute si e' alzato, almeno nella mia testa, quindi cerchiamo di svelare il mistero nel geroflifico.
Per prima cosa andiamo ad analizzare i vari attore presenti nel disegno superiore:

- **i** rappresenta **l'identificativo del neurone corrente**.
- $\mathbf{in_j}$ con $j <= d$ rappresenta il **segnale di ingresso j-esimo**.
- $\mathbf{w_{ji}}$ rapprensenta il **peso** (approssimiamolo con il termine "**importanza**" per ora) di un certo segnali di input.
  Vale a dire che w_{1i} sara' il peso assegnato al primo segnale di ingresso del neurone i.
- in_0 *e* w_{0i} vengono denominato come **bias**, ovvero del **peso** collegato ad un **input fittizio** $in_0 = 1$.
  La sua utilita' sta nel **calibrare** il punto ottimale di lavoro di un neurone.
- $\mathbf{net_i}$ e' il **livello di eccitazione** del neurone i-esimo.
  Avremo che $\mathbf{net_i} = \sum_{\mathbf{j=0}}^{\mathbf{d}} \mathbf{w_{ji}} * \mathbf{in_j} + \mathbf{w_{0i}}$ dove $\mathbf{w_{ji}} * \mathbf{in_j}$ rappesenta il segnale di ingresso nella sua totalita' (considerando cosi' il suo peso).
  Nella sommatoria troviamo anche la somma w_{0i}, ovvero il bias utile per tarare.
- $\mathbf{f}()$ e' la **funzione di attivazione** che determina il comportamento di un neurone.
  Esamineremo dopo i dettagli di questa funzione. Avremo che $\mathbf{out_i} = \mathbf{f}(\mathbf{net_i})$.

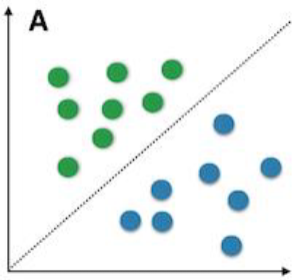
Visto che risulta essere ancora abbastanza completato capiamo il flusso

Vengono acquisiti i vari segnali.

I vari segnali in ingresso vengono acquisiti e trasportati nella "soma" artificiale.

I segnali vengono "pesati".	Ad ogni segnali in ingresso corrisponde un certo peso.
Si calcola il livello di eccitazione.	Si calcola l'eccitazione del neurone sommando tutti i segnali di ingresso moltiplicati per il loro rispettivo peso, sommati a loro volta al bias.
La funzine di attivazione calcola l'output.	In base al livello di eccitazione del neurone la funzione di attivazione calcola l'output. Se $net_i > soglia$ allora viene prodotto un output, altrimenti ne viene prodotto un altro.

Un singolo TLU puo' essere usato per una semplice **classificazione binaria**, **separando in modo lineare le due classi in un grafico**.



Esempi di funzioni di attivazione che permettono di separare linearmente due classi sono le seguenti:

$$headviside(x) = \begin{cases} 0 \; if \; x < 0 \\ 1 \; if \; x > 1 \end{cases}$$

$$sgn(x) = \begin{cases} -1 \; if \; x < 0 \\ 0 \; if \; x = 0 \\ 1 \; if \; x > 0 \end{cases}$$

Come detto, all'inizio, non perforza il modello Perceptron e' costituito da un solo neurone, ma bensi da **solo layer di neuroni**. Un **layer di neuroni** e' uno insieme di neuroni appartenteni allo stesso livello.
Quanto tutti i neuroni di un layer, sono collegati a tutti i neuroni del layer precedente, il layer viene denominato come **fully connected layer**.
La domanda da farsi ora e' questa, come faccio a calcolare l'output di un intero layer di neuroni senza diventare scemo?
Non farti questa domanda, quando a lezione sara' trattato questo caso lo annotero' negli appunti, altrimenti no.