Sapere utile

IFOA
Istituto Formazione Operatori Aziendali

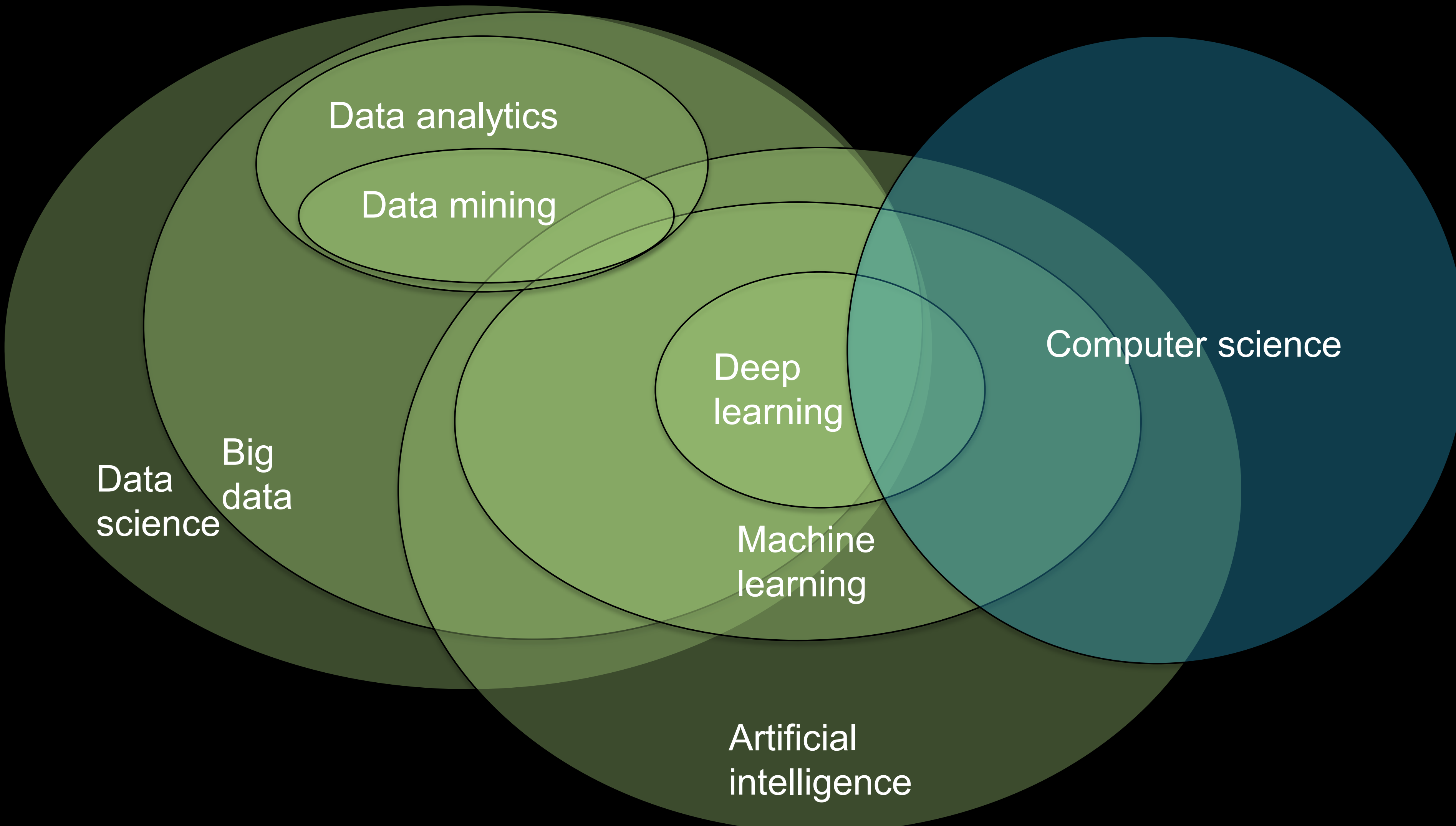
BIG DATA e Analisi dei Dati

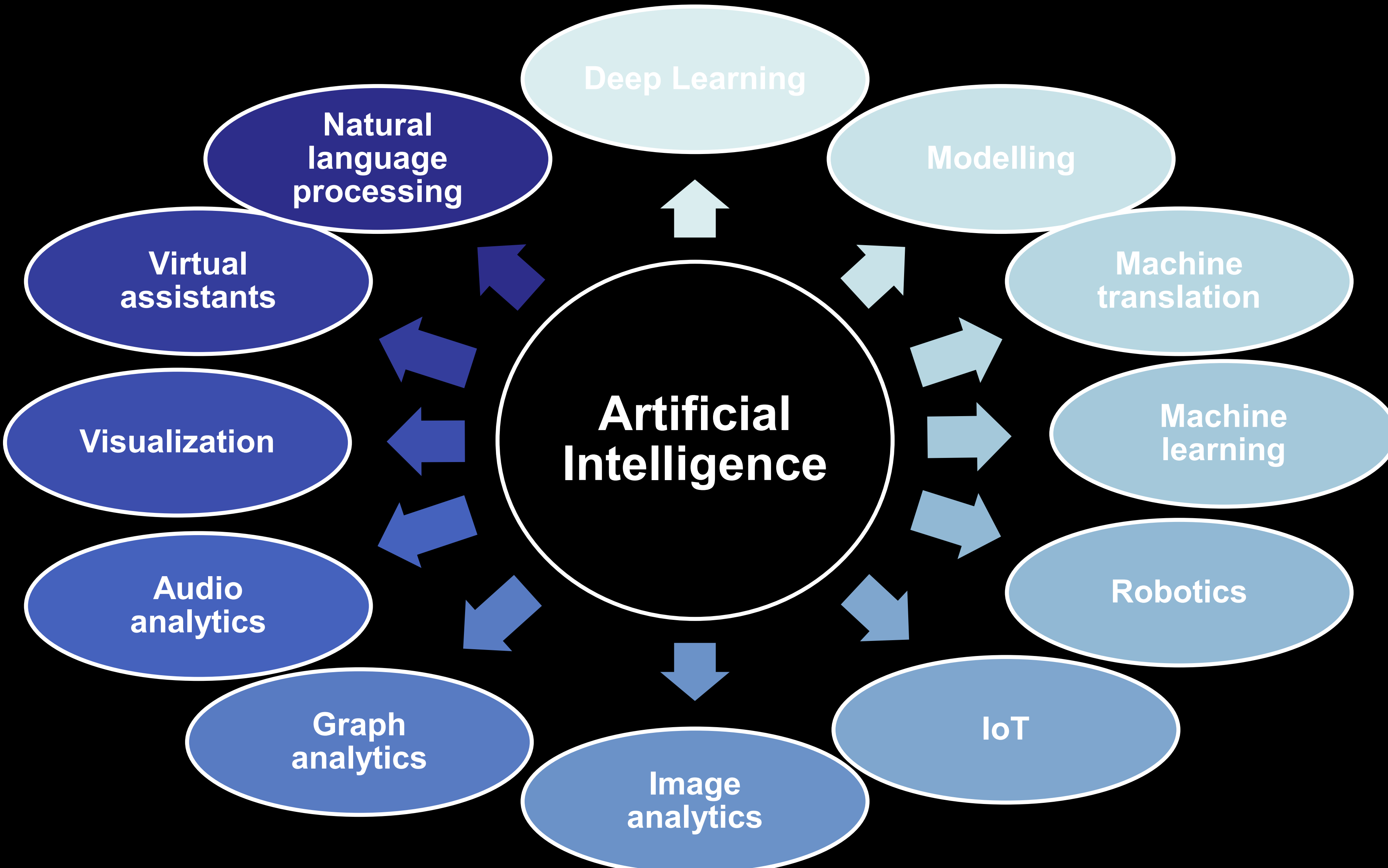
Mauro Bellone,
Robotics and AI researcher

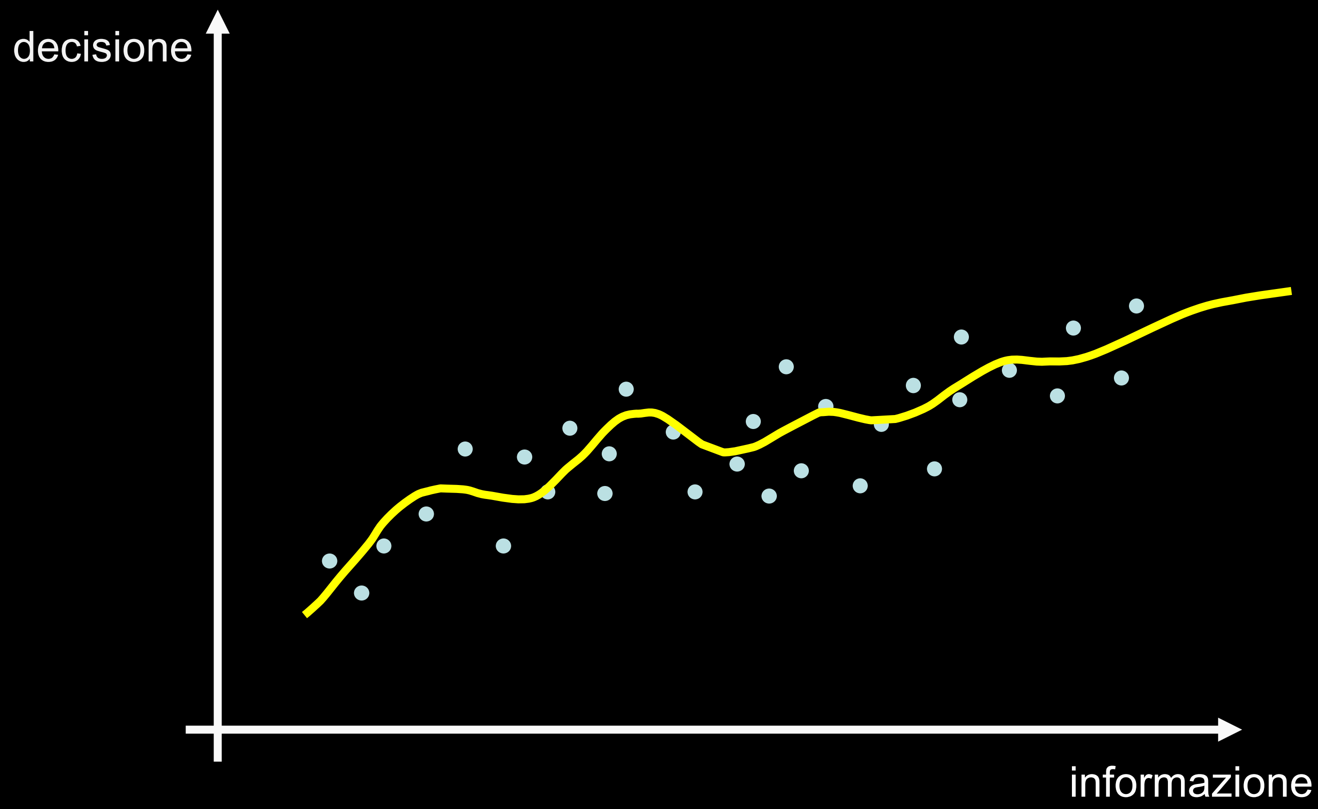
bellonemauro@gmail.com
www.maurobellone.com

Obiettivo

- ✓ introduzione all’intelligenza artificiale e reti neurali
- ✓ Il percettrone - funzioni di attivazione dei neuroni
- ✓ Reti neurali lineari
- ✓ Algoritmi di apprendimento


Data analytics
Data mining
Deep learning
Computer science
Big data
Data science
Machine learning
Artificial intelligence

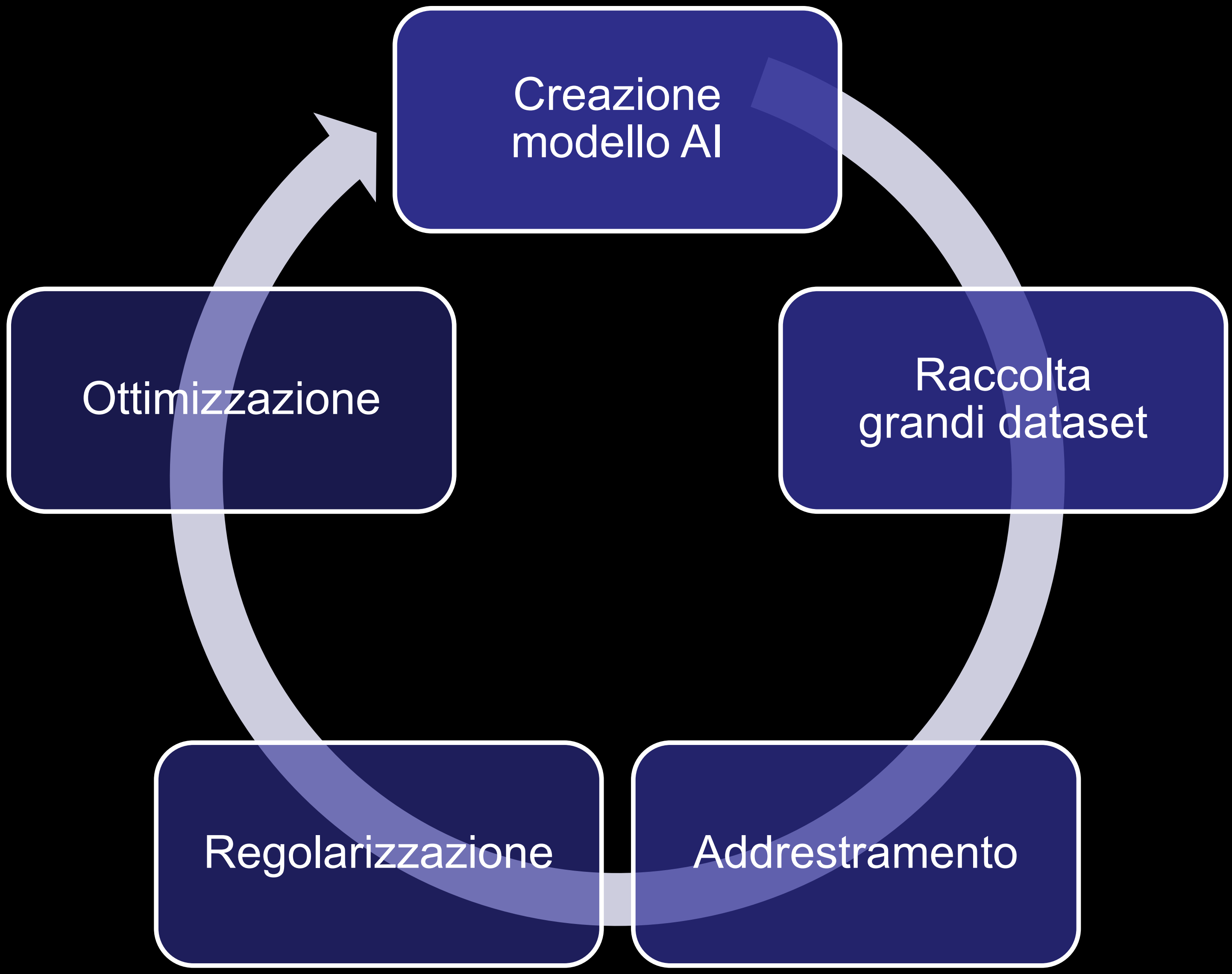

Artificial Intelligence
Deep Learning
Modelling
Machine translation
Machine learning
Robotics
IoT
Image analytics
Graph analytics
Audio analytics
Visualization
Virtual assistants
Natural language processing



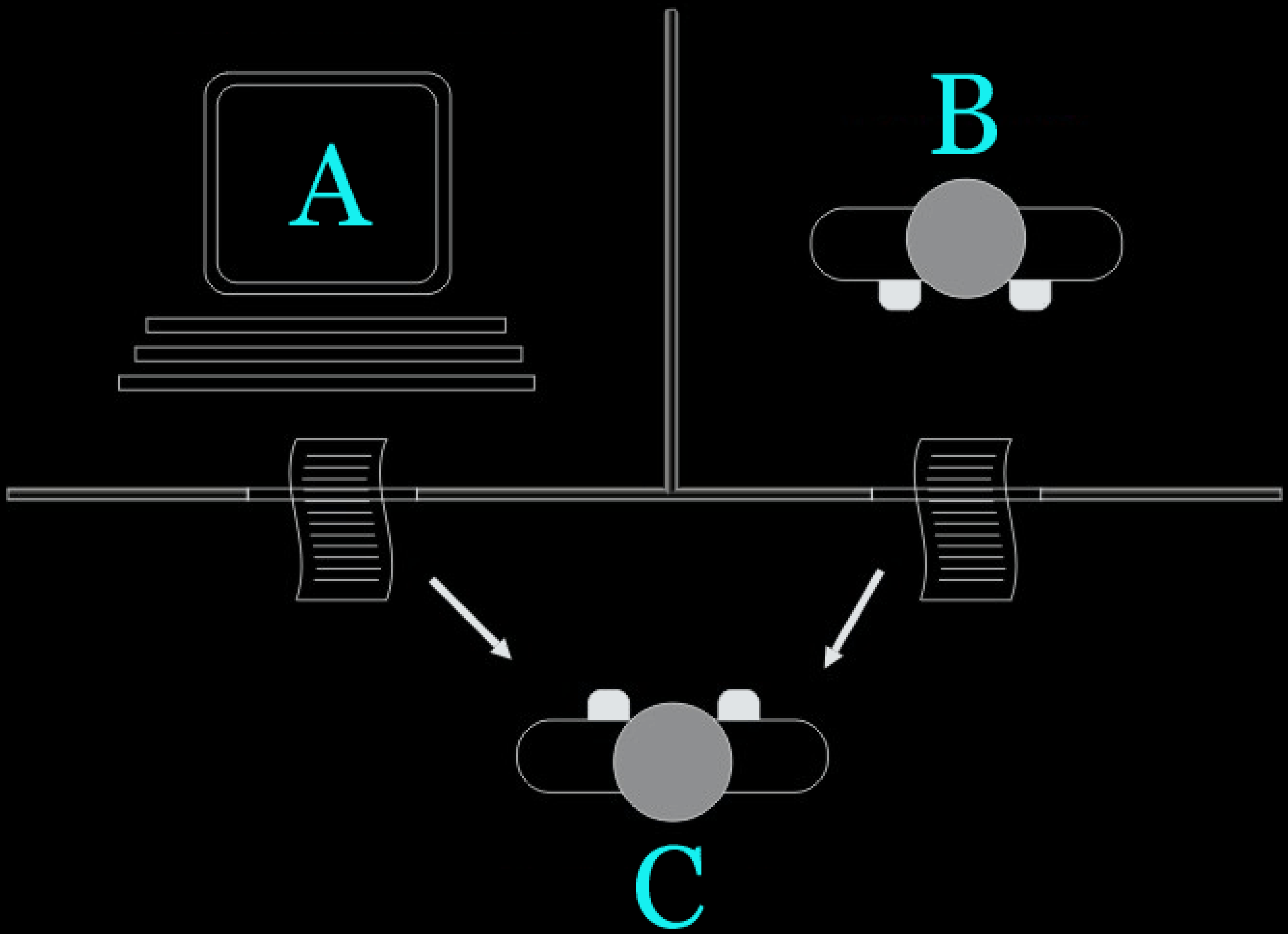
decisione = f (informazione)

Ogni sistema di ingelligenza artificiale è un approssimatore di funzioni

Intelligenza artificiale centralizzata



Test di Turing



Intelligenza artificiale distribuita

Con intelligenza artificiale centralizzata si cerca di spostare dati presenti in un sistema (magari distribuito) per essere processati in un’unica unità computazionale

“spostamento dei dati verso gli algoritmi”

con intelligenza artificiale distribuita si cerca invece di processare dati lungo i diversi sistemi su domini multipli e su devices multipli

“spostamento degli algoritmi verso i dati“

Deep blue Vs Gary Kasparov

Deep blue Vs Gary Kasparov

In realtà deep blue **NON** usa reti neurali, usa brute-force computation e analisi degli alberi decisionali, era in grado di valutare e scegliere oltre 200milioni di posizioni al secondo nel 1999

10 anni di sviluppo di un team di ingegneri di IBM

source: https://en.wikipedia.org/wiki/Deep_Blue_versus_Garry_Kasparov

ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver[1*], Aja Huang[1*], Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver[1*], Aja Huang[1*], Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver[1*], Aja Huang[1*], Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

Esperienza

Pensiero razionale

Autonomo Vs. Intelligente

Autonomo Vs. Intelligente

The term ”**autonomous**” is defined as

”existing or acting separately from other things”

or better

“having the power or right to govern itself”

1: http://www.merriam-webster.com/dictionary/autonomous
2: http://www.merriam-webster.com/dictionary/intelligent

Autonomo Vs. Intelligente

The term ”**autonomous**” is defined as

”existing or acting separately from other things”

or better

“having the power or right to govern itself”

The term ‘‘**intelligent**’’ is defined as

‘‘having or indicating a high or satisfactory degree of intelligence and mental capacity’’

and *‘‘intelligence’’* is *‘‘the ability to learn or understand or to deal with new or trying situations: reason’’*

1: http://www.merriam-webster.com/dictionary/autonomous
2: http://www.merriam-webster.com/dictionary/intelligent

Autonomous Vs. Intelligent

The term "**autonomous**" is defined as

"existing or acting separately from other things"

or better

"having the power or right to govern itself"

The term "**intelligent**" is defined as

"having or indicating a high or satisfactory degree of intelligence and mental capacity"

and *"intelligence"* is *"the ability to learn or understand or to deal with new or trying situations: reason"*

Comprensione Vs. Percezione

Understanding

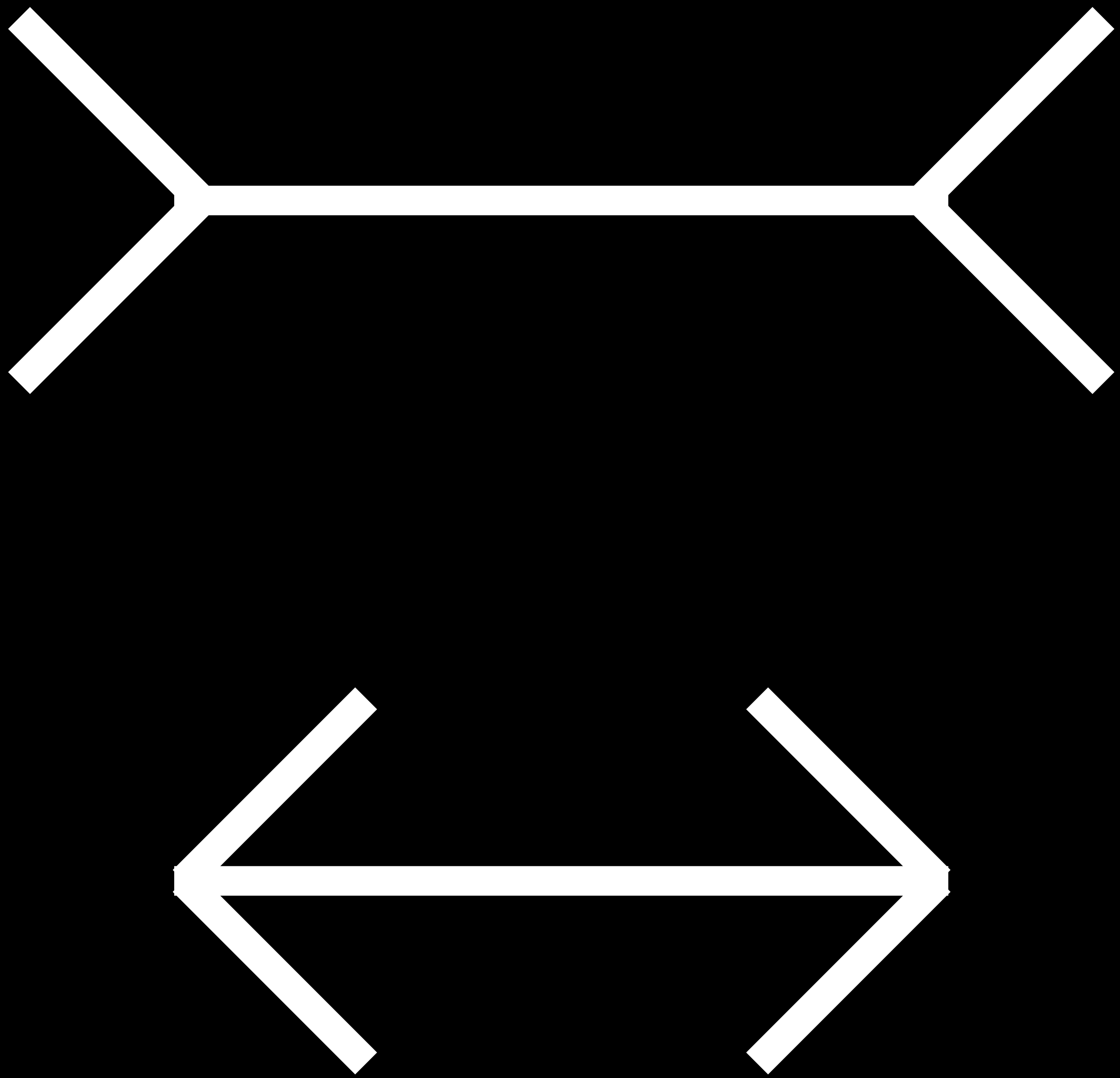
"the knowledge and ability to judge a particular situation or subject" [1]

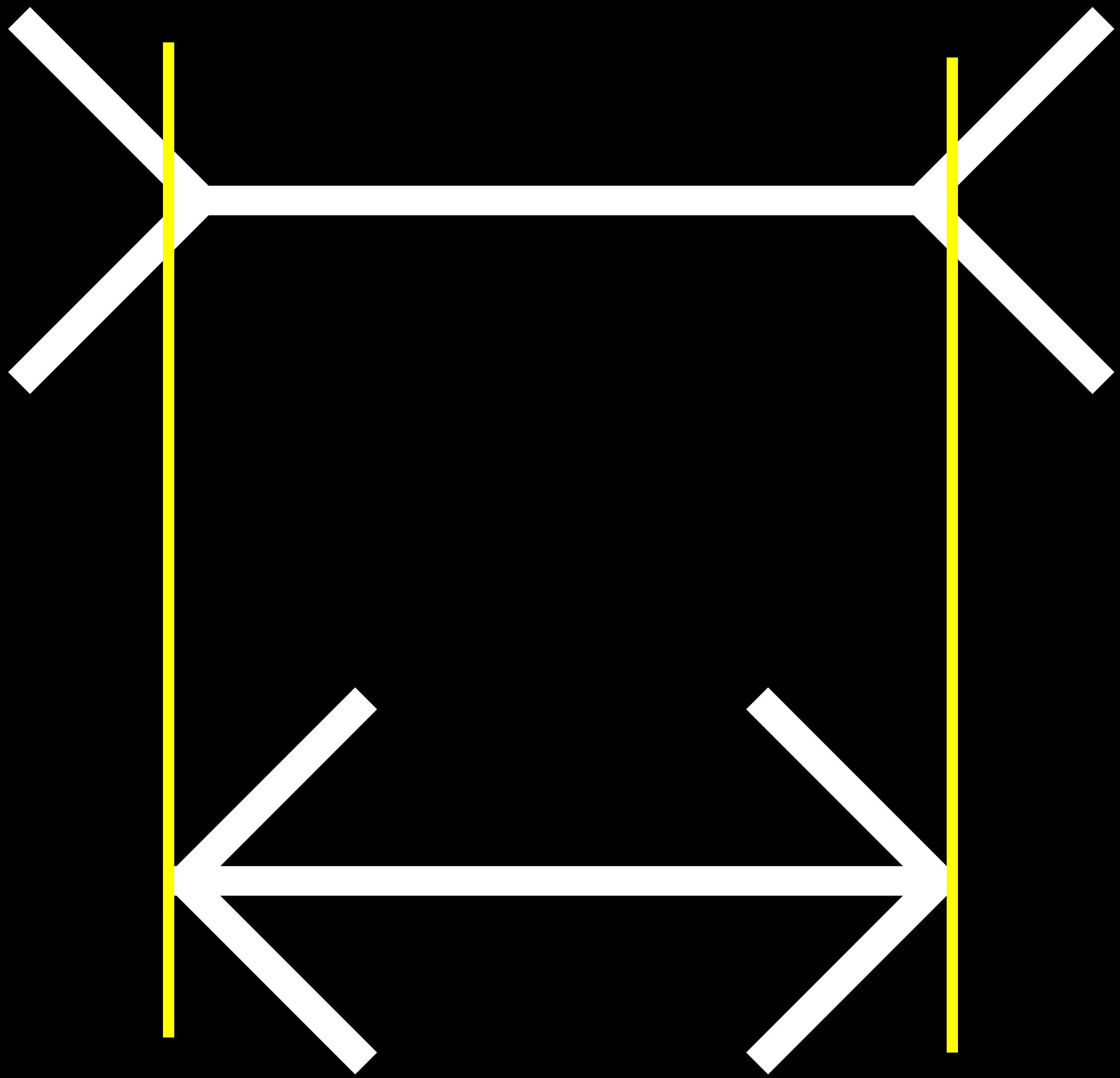
Perception

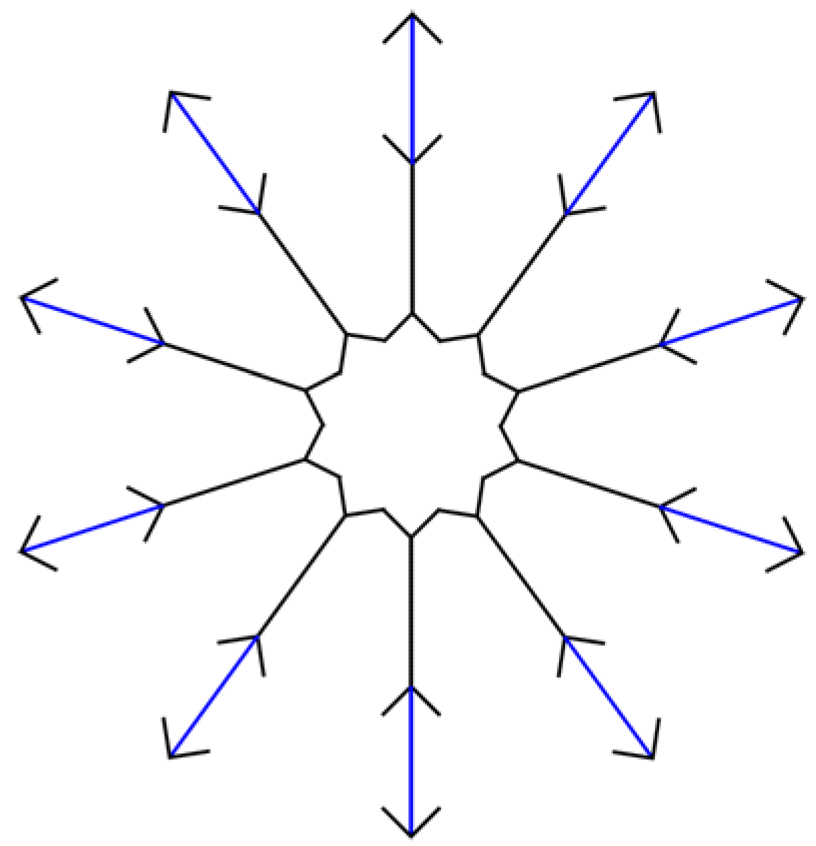
"the way that you notice or understand something using one of your senses" [2]

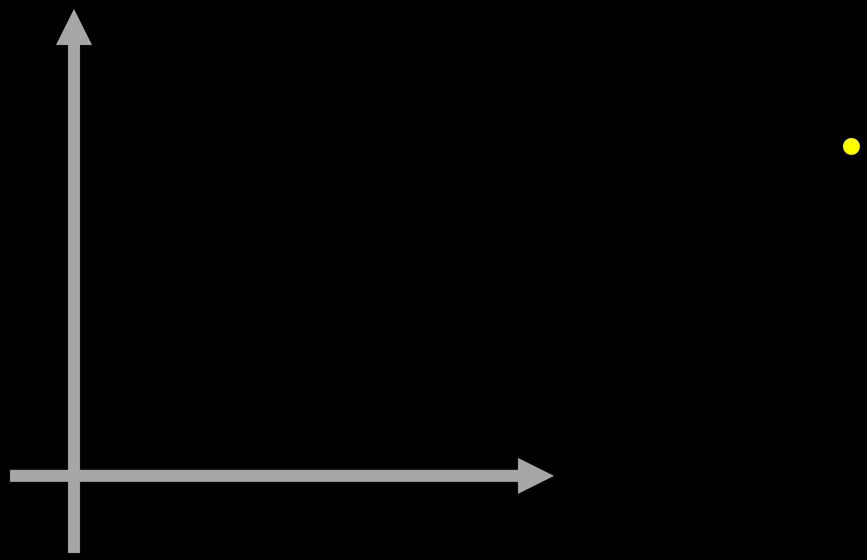
1: http://www.merriam-webster.com/dictionary/understanding
2: http://www.merriam-webster.com/dictionary/perception


6
9
PERCEPTION

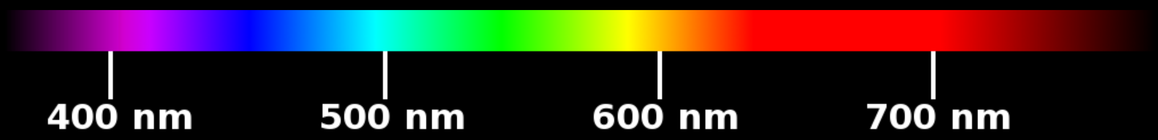


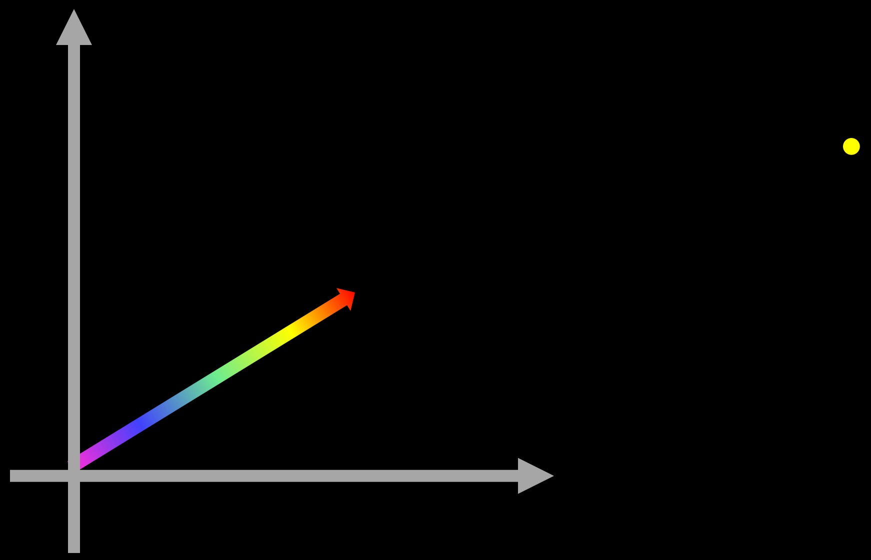






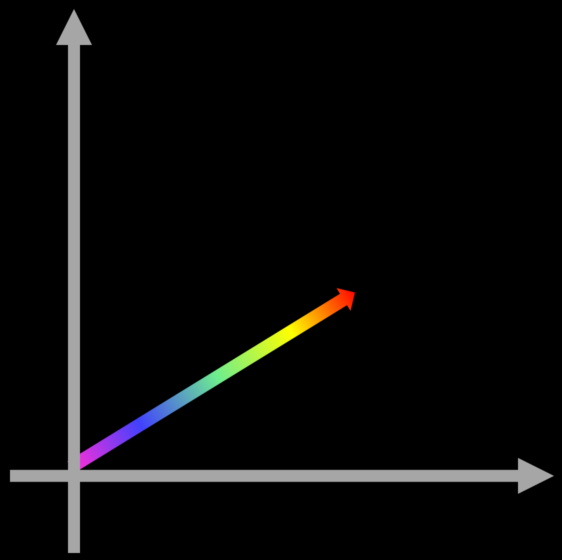
$p = (x,y)$

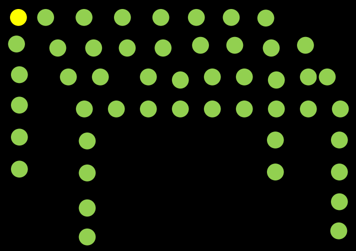




p = (x,y, colore)

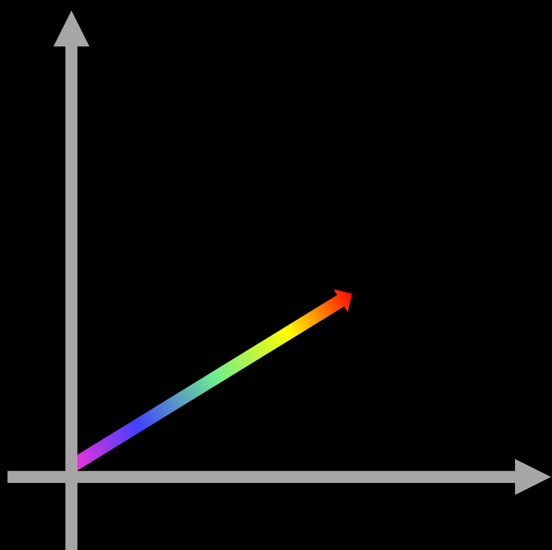
“il problema della descrizione semantica è mal posto se consideriamo solo un insieme limitato di caratteristiche.“

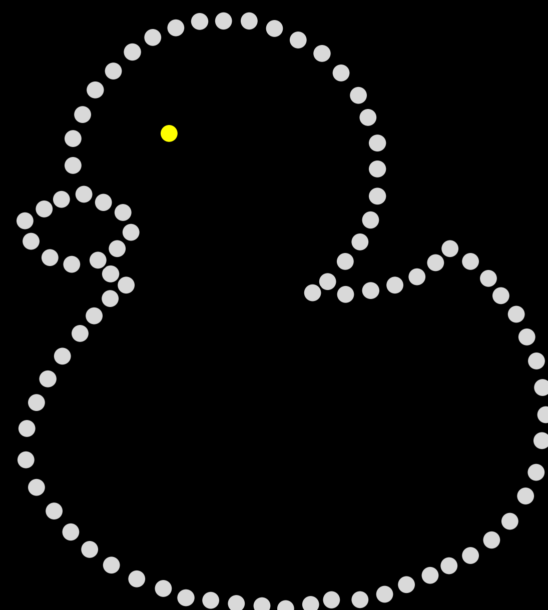




p = (x,y, colore, angolo)

“il problema della descrizione semantica è mal posto se consideriamo solo un insieme limitato di caratteristiche.“





$p = (x,y, color, eye\ )$

Comprensione Vs. Percezione

Understanding

"the knowledge and ability to judge a particular situation or subject" [1]

Perception

"the way that you notice or understand something using one of your senses" [2]

Sensing

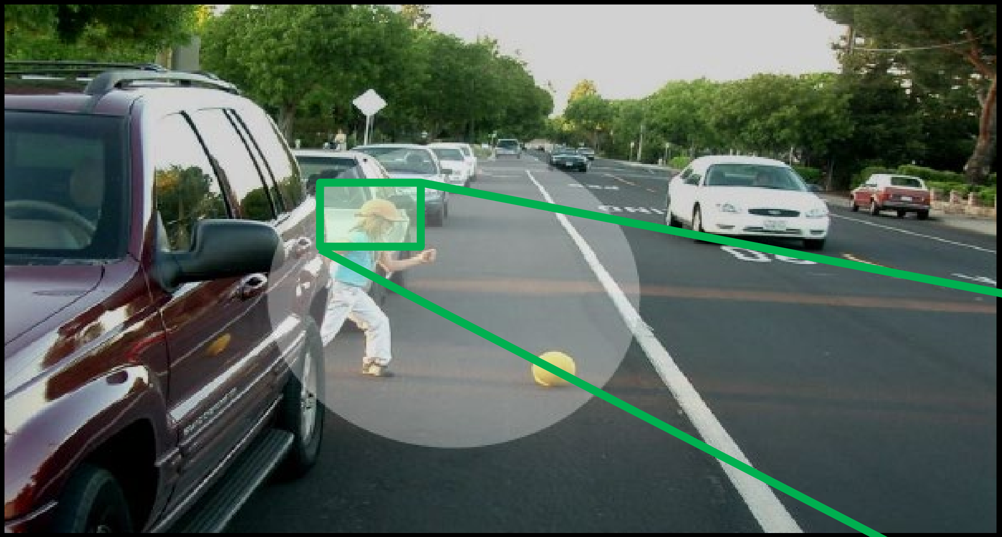
"one of the five natural powers (touch, taste, smell, sight, and hearing) through which you receive information about the world around you" or in the meaning of physical feeling *"something that your body experiences"*

1: http://www.merriam-webster.com/dictionary/understanding
2: http://www.merriam-webster.com/dictionary/perception

Perception: *“the ability to see, hear, or become aware of something through the senses”*

- Child following a ball -

Quale quantità fisica misiruamo? Come facciamo a comprendere la situazione?

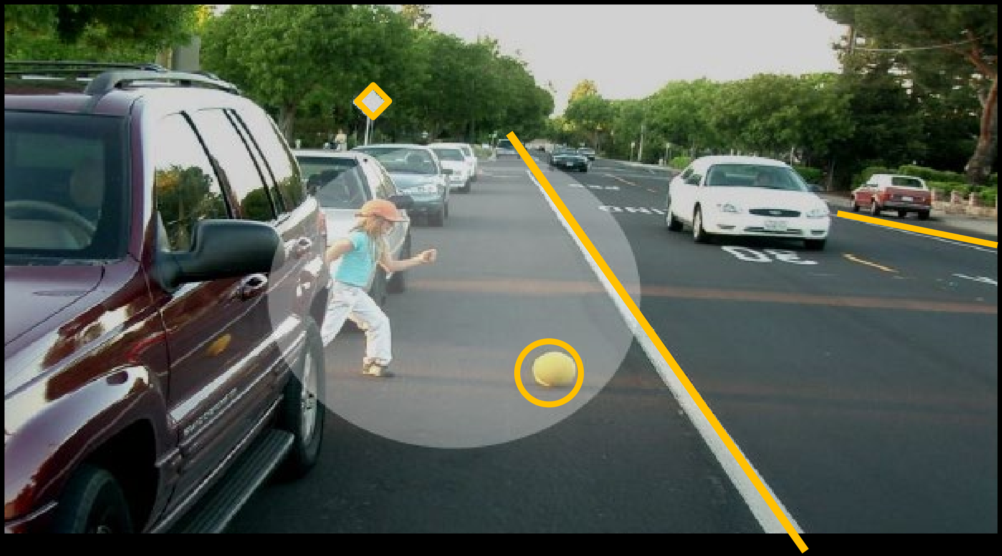


$$\begin{bmatrix} 13 & 2 & 4 & 2 & 32 & 23 & 9 & 1 & 3 \\ 6 & 82 & 45 & 72 & 3 & 24 & 3 & 11 & 4 \\ 21 & 3 & 44 & 6 & 64 & 34 & 2 & 28 & 32 \\ 93 & 53 & 32 & 0 & 77 & 33 & 21 & 7 & 56 \\ 43 & 23 & 87 & 1 & 22 & 23 & 57 & 37 & 22 \\ 37 & 2 & 39 & 11 & 23 & 3 & 45 & 2 & 90 \\ 34 & 3 & 3 & 3 & 3 & 33 & 1 & 25 & 78 \\ 3 & 75 & 63 & 3 & 32 & 4 & 55 & 3 & 72 \\ 56 & 3 & 3 & 21 & 2 & 6 & 69 & 16 & 45 \end{bmatrix}$$

Perception: *"the ability to see, hear, or become aware of something through the senses"*

\- Child following a ball -

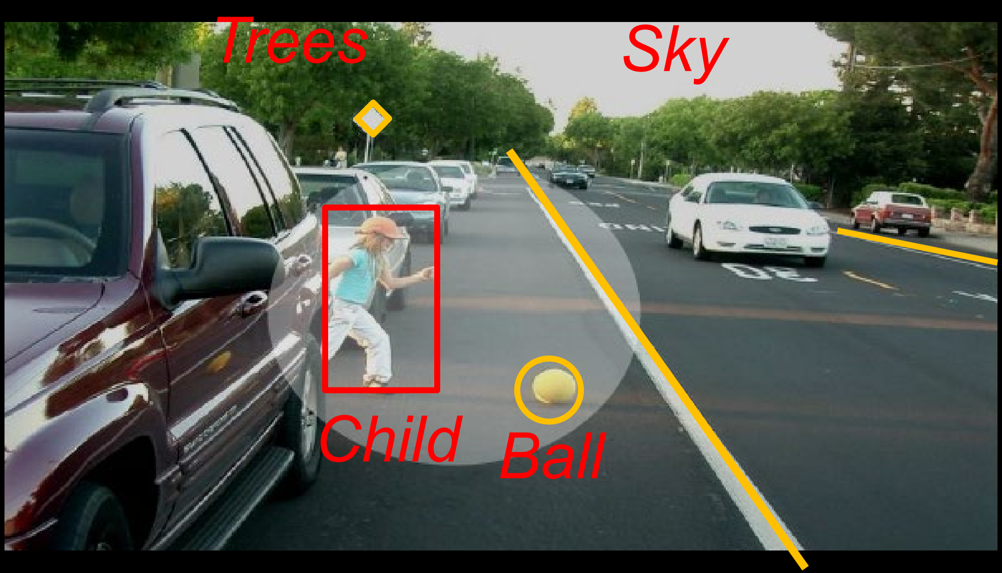
Informazioni geometriche – distanze, linee, circonferenze, ect.



Perception: *“the ability to see, hear, or become aware of something through the senses”*

- Child following a ball -

Informazioni semantiche – bambino, palla, cielo, alberi ect.



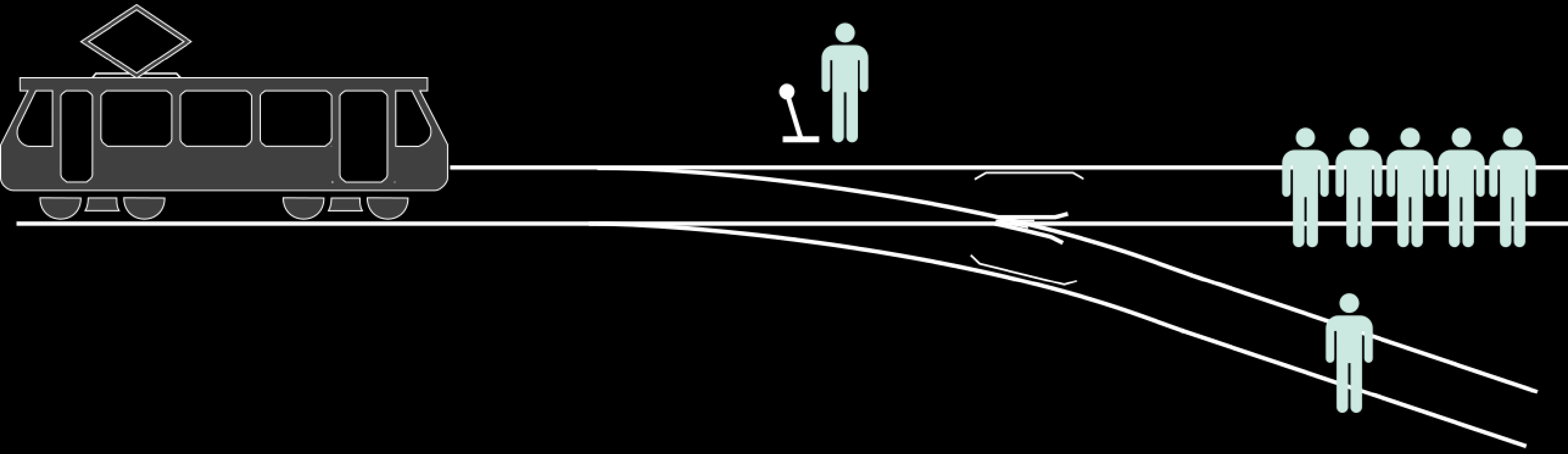
comprensione

stop = $f($  $)$

azione

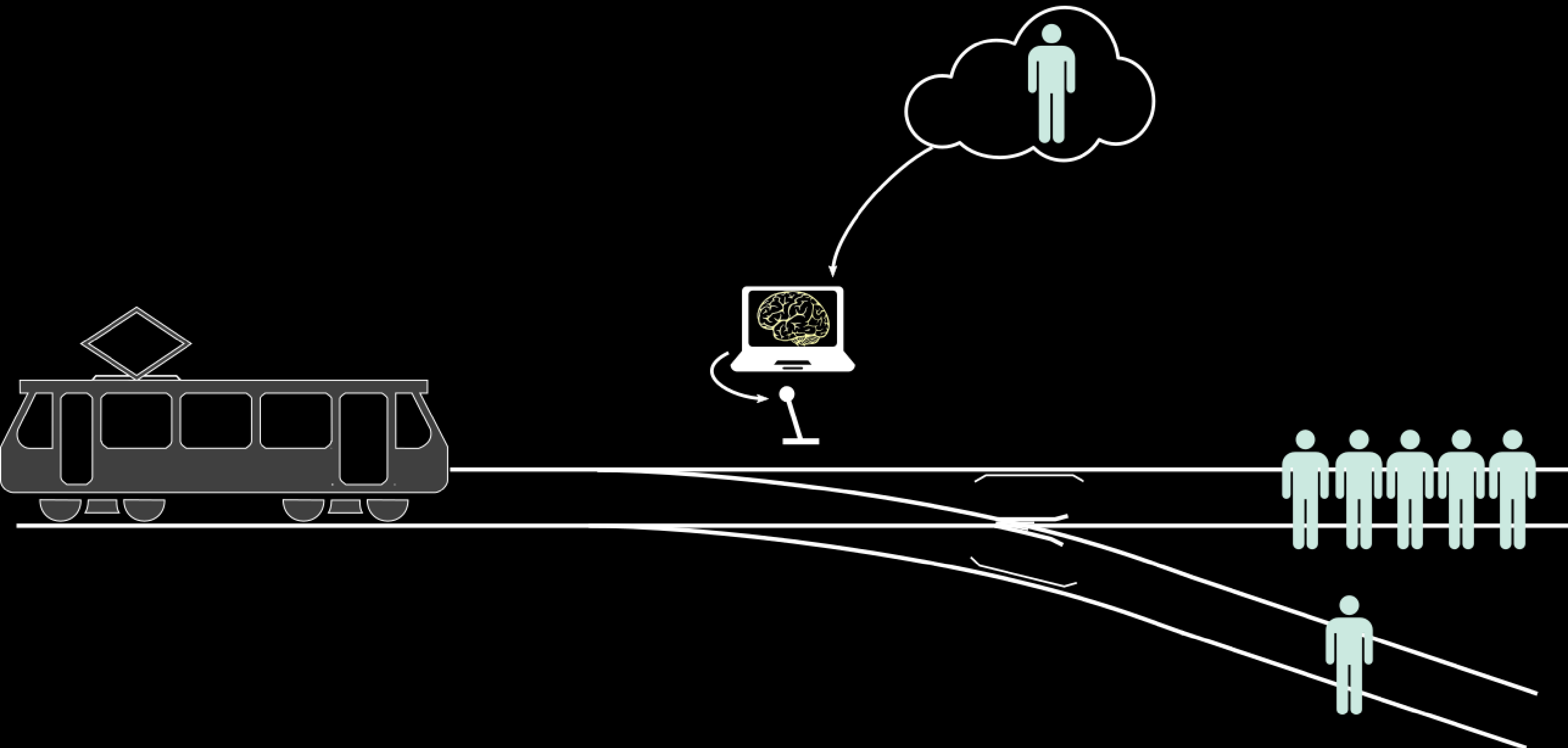
informazione

Problema etico



Problema etico

Problema etico



Simbolismo Vs. connettivismo

Una grande divisione nel mondo dell’intelligenza artificiale è quella di:

Intelligenza connettiva:

Rappresenta l’informazione in maniera distribuita, in forma meno esplicita attraverso una rete. Biologicamente il processo di apprendimento, prestazioni nella risoluzione di un compito e le abilità di problem solving sono limitate.

Simbolismo Vs. connettivismo

Una grande divisione nel mondo dell’intelligenza artificiale è quella di:

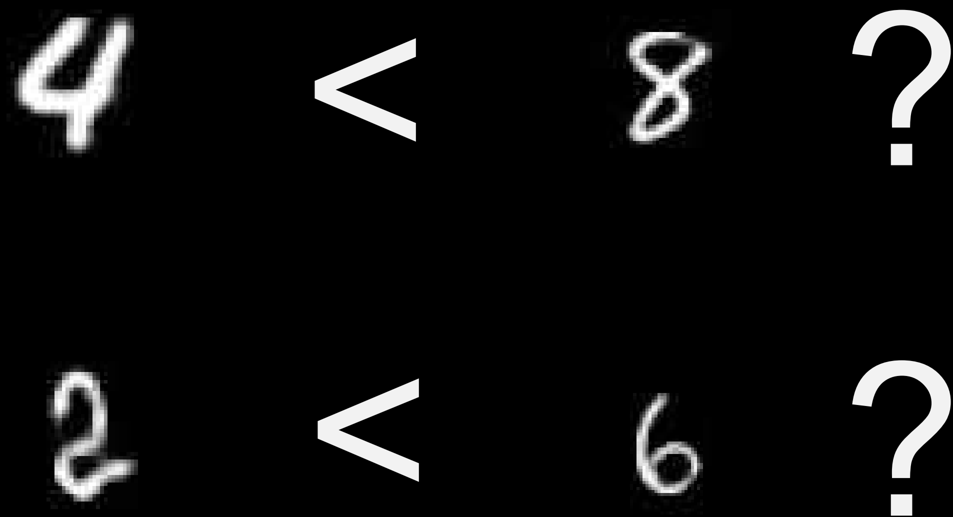
Intelligenza connettiva:

Rappresenta l’informazione in maniera distribuita, in forma meno esplicita attraverso una rete. Biologicamente il processo di apprendimento, prestazioni nella risoluzione di un compito e le abilità di problem solving sono limitate.

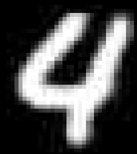
Intelligenza simbolica:

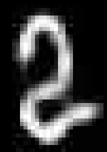
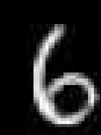
Rappresenta l’informazione attraverso simboli e le loro relazioni. E possibile quindi usare algoritmi per processare questi simboli e risolvere problemi o dedurre nuova conoscenza attraverso regole di inferenza.

Intelligenza simbolica



Intelligenza simbolica

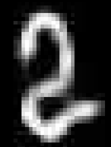
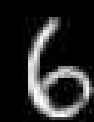
 <  ?

 <  ?

0 < 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9

Intelligenza simbolica

 <  ?

 <  ?

$0 < 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9$

$0 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9$

$0 < 3 < 4 < 5 < 6 < 7 < 8 < 9$

Black box Vs explainable model

Black box Vs explainable model

Storia breve:

Nei primi anni del 1900 un medico dello stato di New York scoprì che mangiare fegato bovino crudo faceva guarire i pazienti da una particolare malattia chiamata “anemia perniciosa”.

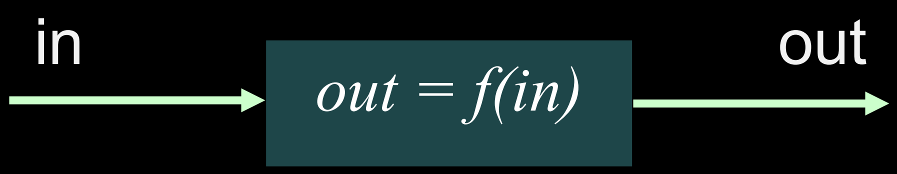
Pur di guarire questa complessa malattia il medico faceva mangiare ai suoi pazienti 200grammi di fegato bovino crudo al giorno

Black box Vs explainable model

Storia breve:

Nei primi anni del 1900 un medico dello stato di New York scoprì che mangiare fegato bovino crudo faceva guarire i pazienti da una particolare malattia chiamata “anemia perniciosa”.

Pur di guarire questa complessa malattia il medico faceva mangiare ai suoi pazienti 200grammi di fegato bovino crudo al giorno

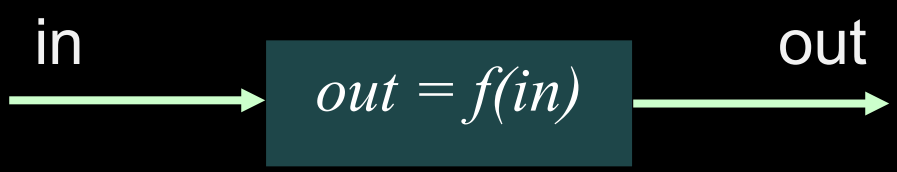


1. Modello: Malattia
2. Prove IN: Diversi cibi
3. Effetto OUT: Miglioramento si/no

Black box Vs explainable model

Storia breve:

Sono stati necessari oltre 40anni a scoprire che questa complessa patologia era risolvibile con un supplemento vitaminico (il fegato è ricco di vitamine e minerali, in particolare per questa patologia la B12)

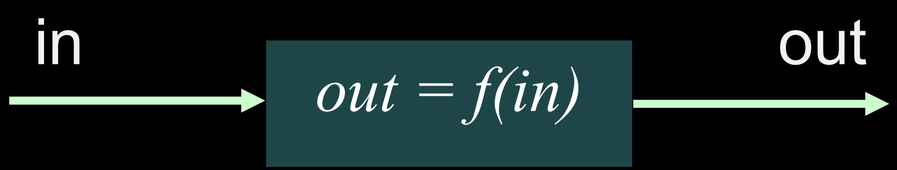


1. Modello: Malattia
2. Prove IN: Diversi cibi
3. Effetto OUT: Miglioramento si/no

Black box Vs explainable model

Storia breve:

Sono stati necessari oltre 40anni a scoprire che questa complessa patologia era risolvibile con un supplemento vitaminico (il fegato è ricco di vitamine e minerali, in particolare per questa patologia la B12)



1. Modello: Malattia
2. Prove IN: Diversi cibi
3. Effetto OUT: Miglioramento si/no

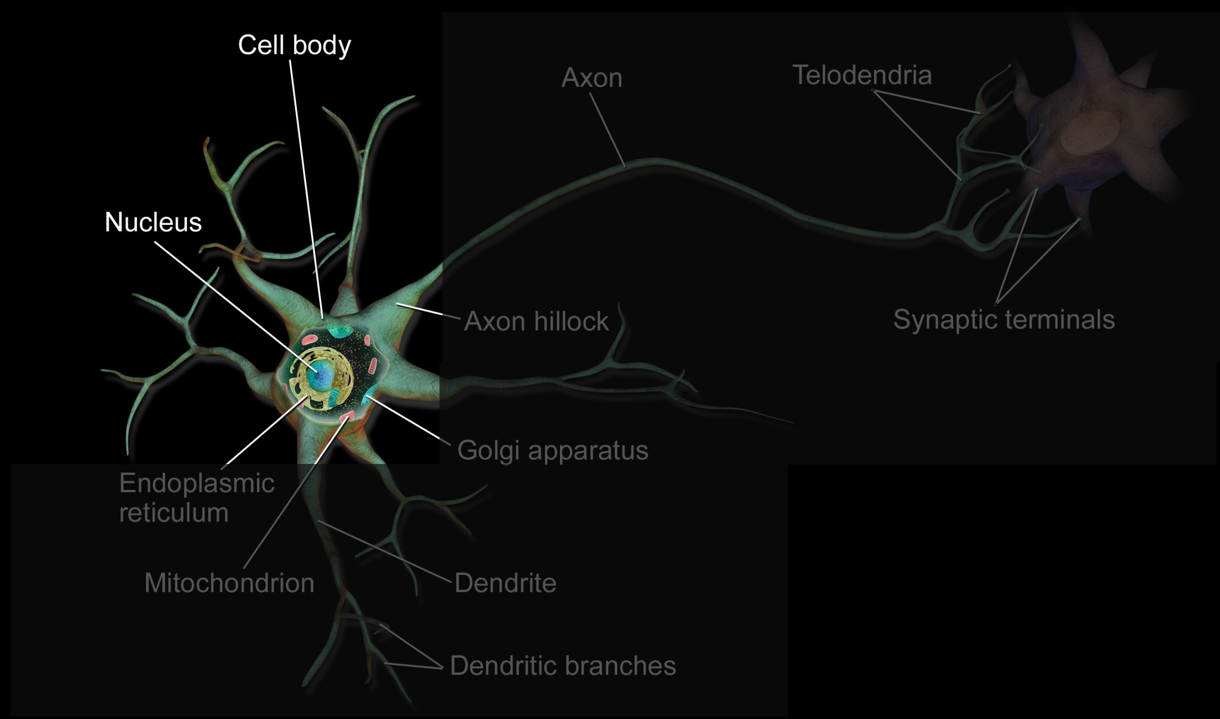
Approccio black box

1. Modello: Malattia
2. Analisi e comprensone del modello
3. Risoluzione del problema

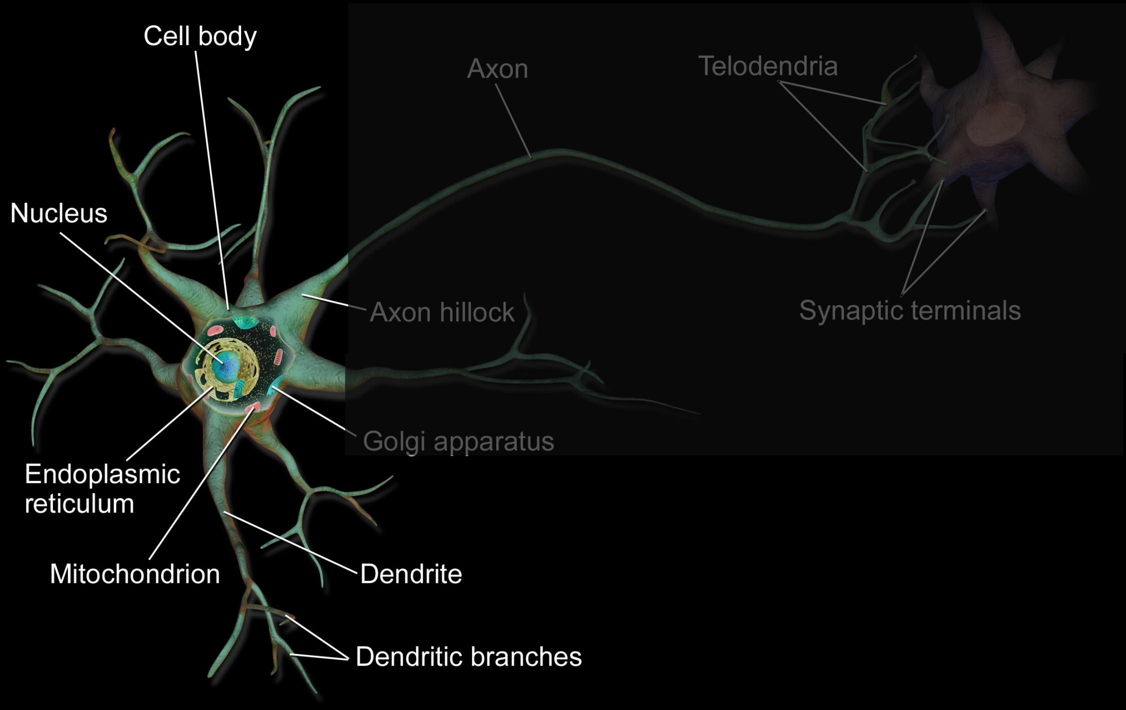
Approccio spiegato

Modello neurale

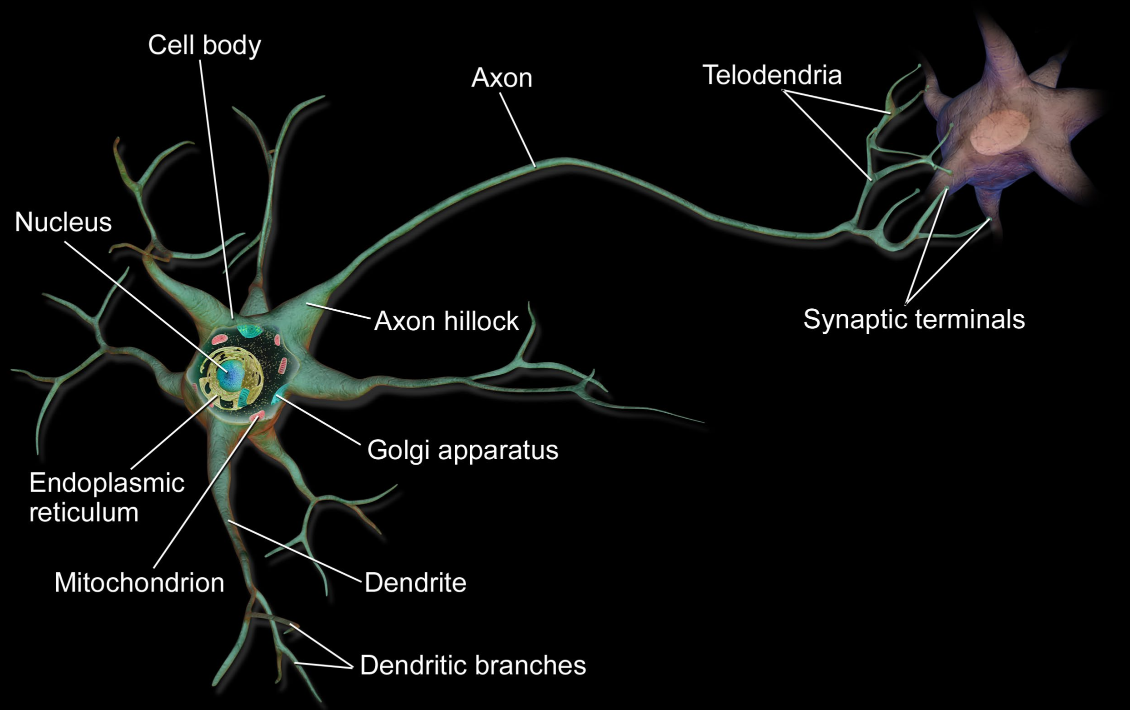
Modello neurale



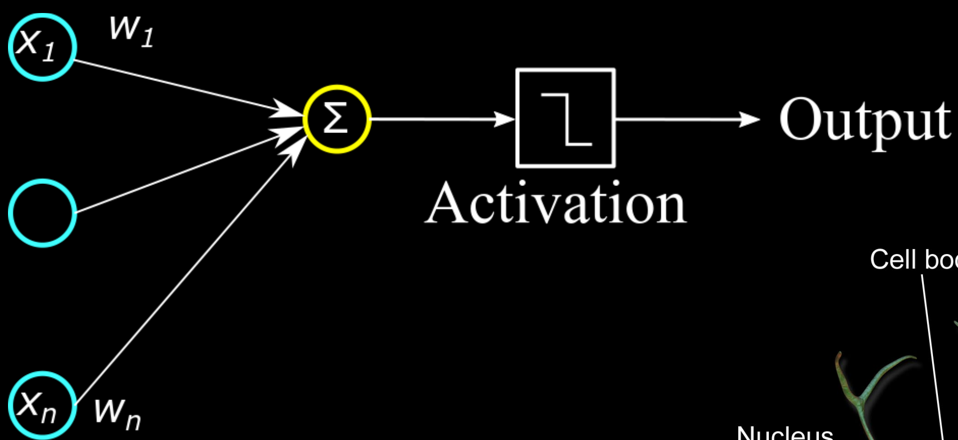
Modello neurale

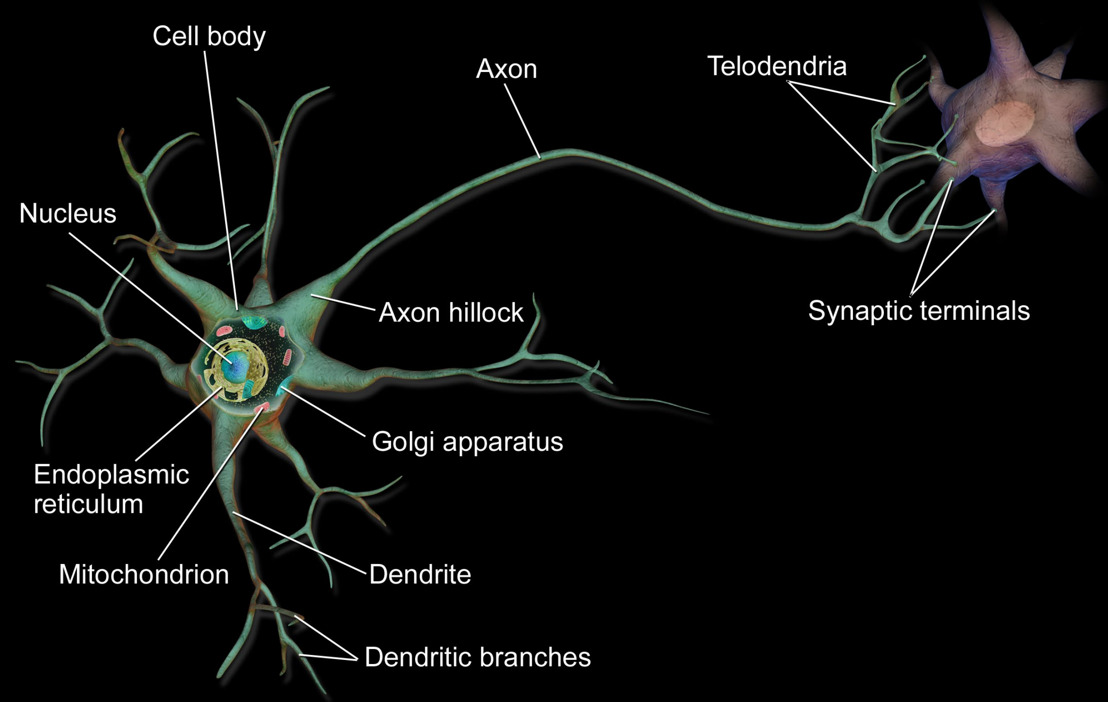


Modello neurale

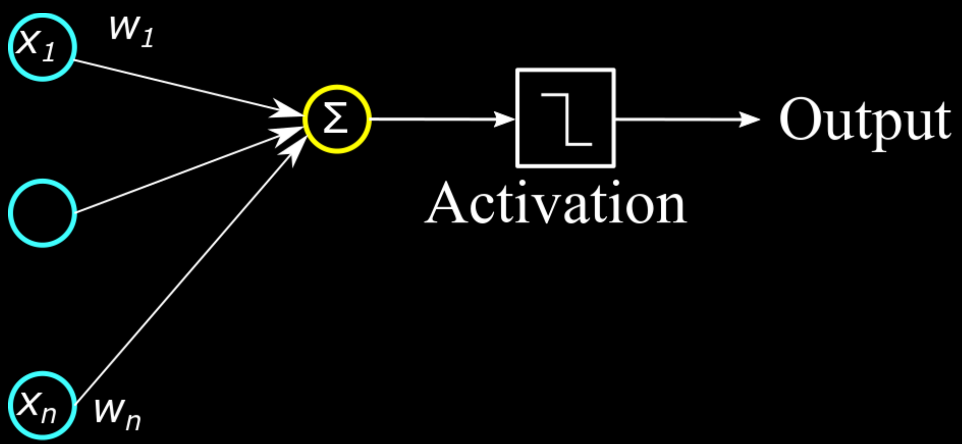


Il percettrone

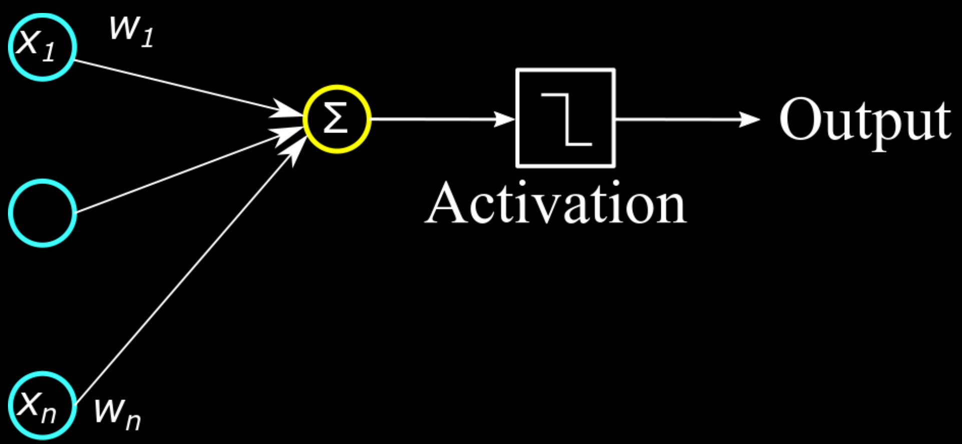




Il percettrone

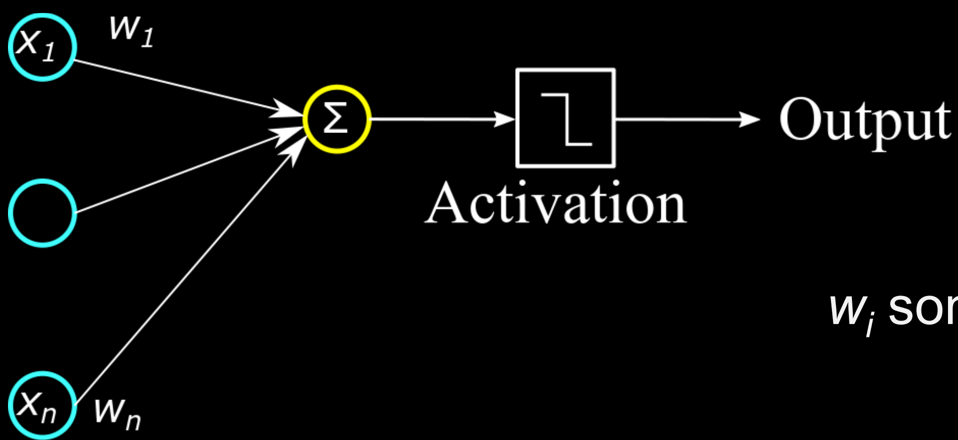


Il percettrone



$$out = \sum x_i w_i * \mathrm{A}$$

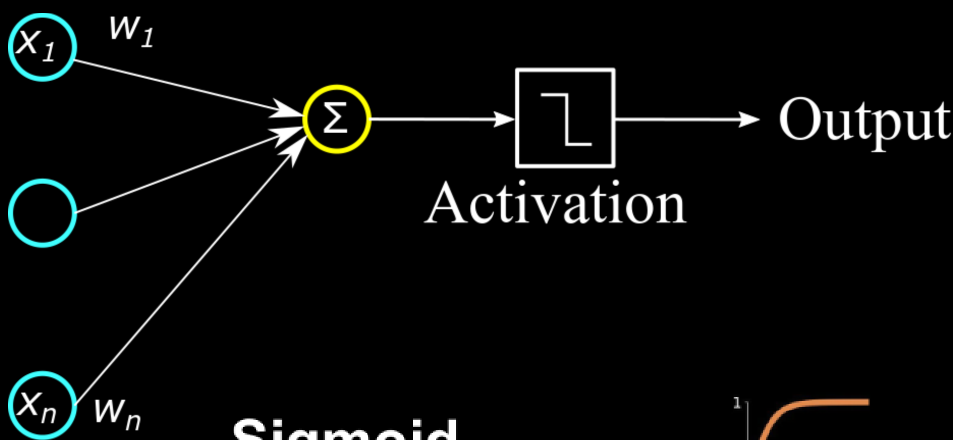
Il percettrone



$$out = \sum x_i w_i * \mathrm{A}$$

w_i sono chiamati “trainable parameters”

Funzioni di attivazione

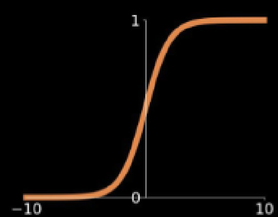


$$out = \sum x_i w_i * \mathrm{A}$$

A è detta funzione di attivazione

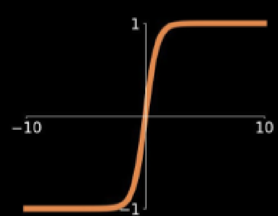
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



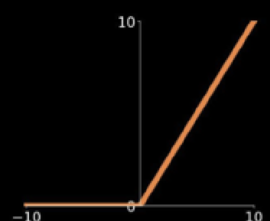
tanh

$$\tanh(x)$$



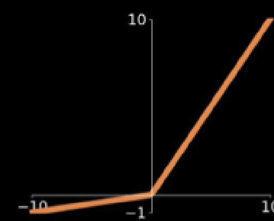
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

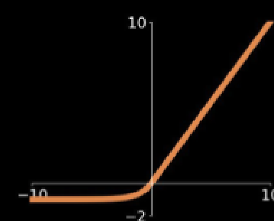


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

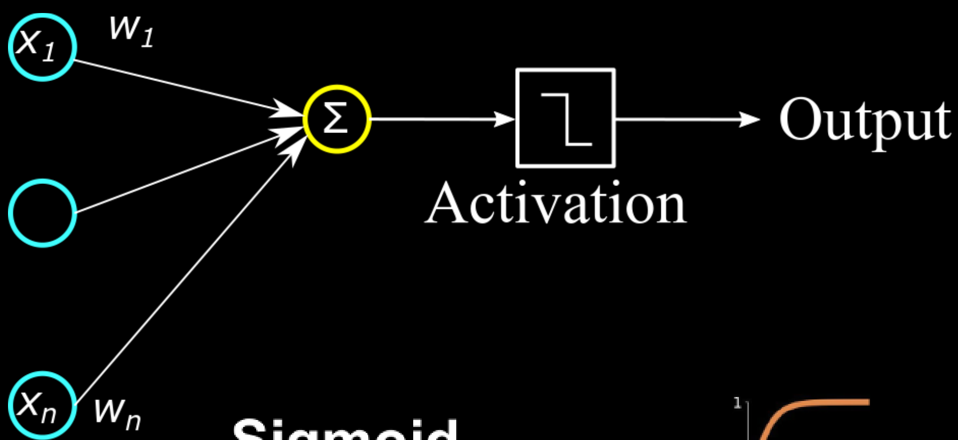
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Funzioni di attivazione

Modella il sistema di attivazione neuronale in relazione a determinati stimoli

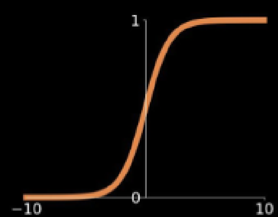


$$out = \sum x_i w_i * \mathrm{A}$$

A è detta funzione di attivazione

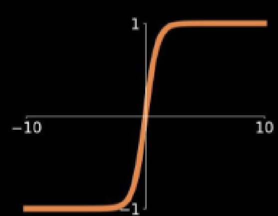
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



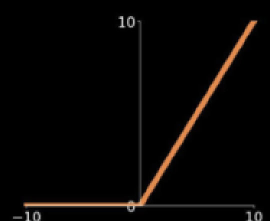
tanh

$$\tanh(x)$$



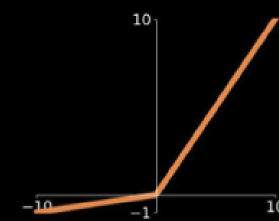
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

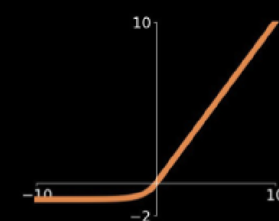


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



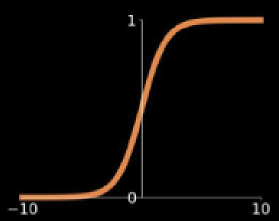
Funzioni di attivazione

Caratteristiche importanti:

- Non linearità
- Differenziabilità
- Devono avere un range di sensibilità
- Avere dei lowerbound o upperbound

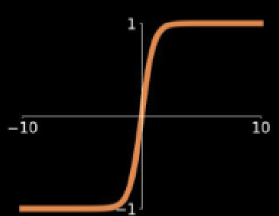
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



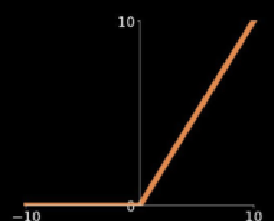
tanh

$\tanh(x)$



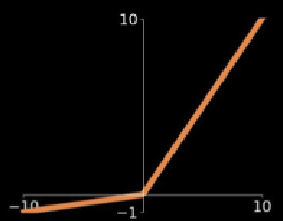
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

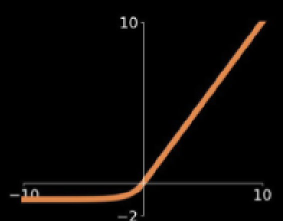


Maxout

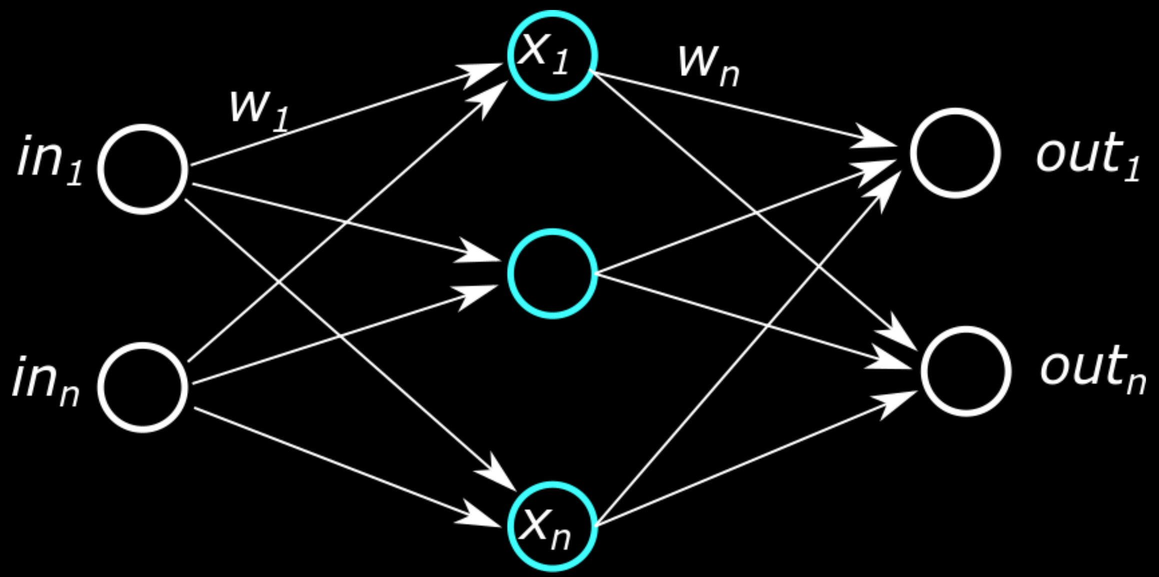
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

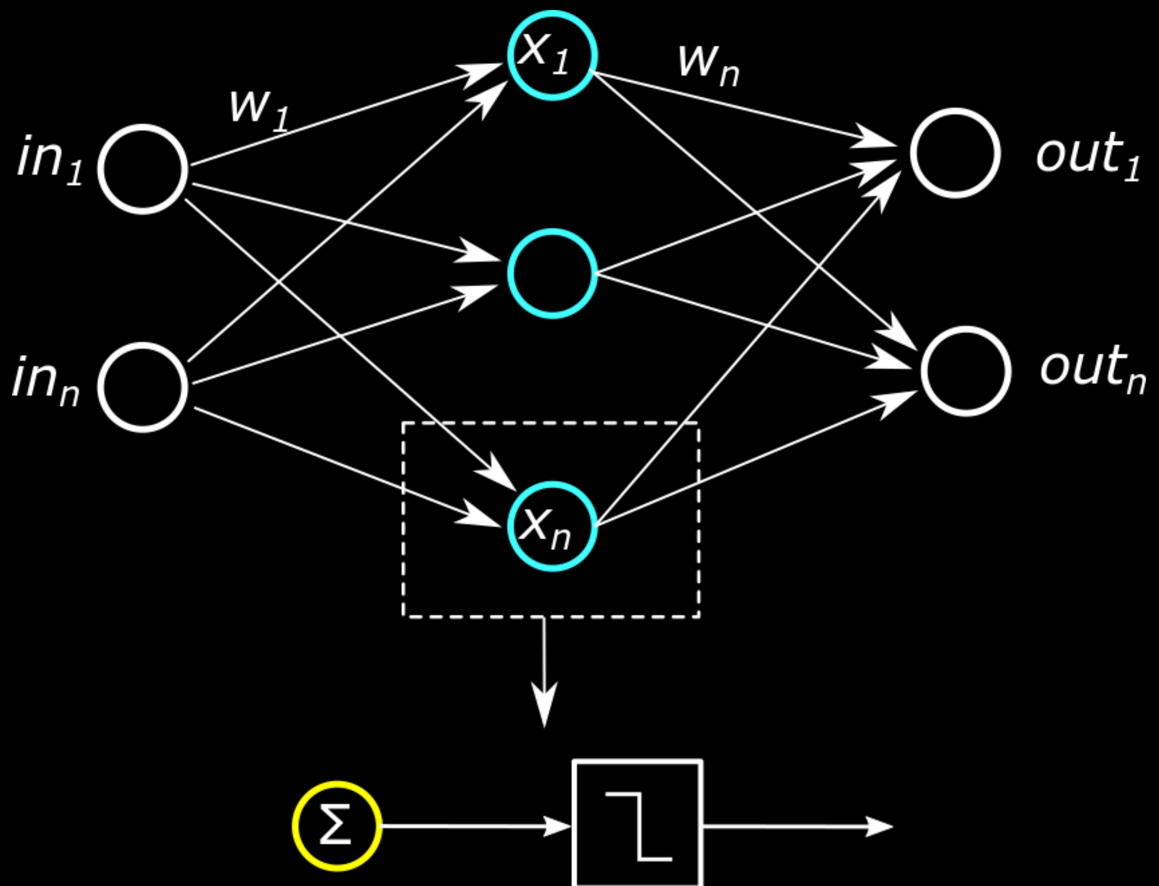


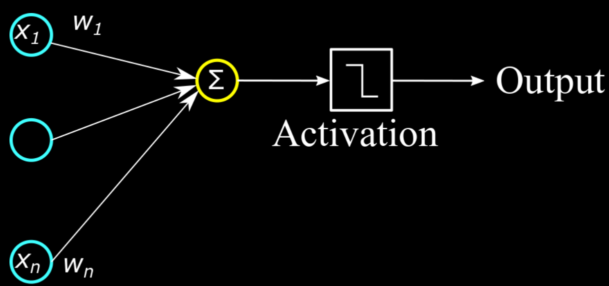
Reti neurali lineari



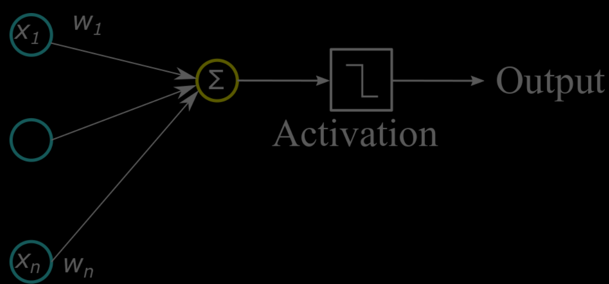
Livello di input *livelli nascosti* *livello di uscita*

Reti neurali lineari

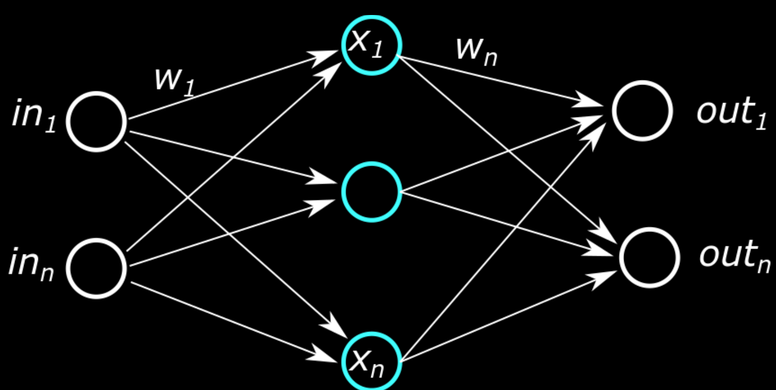




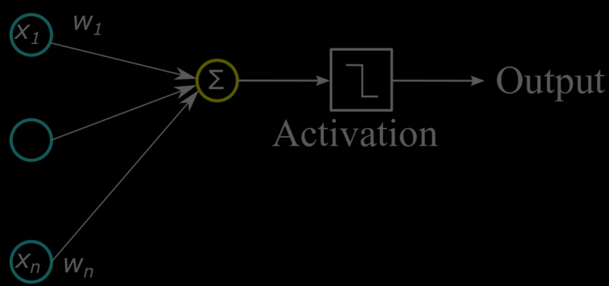
$$out = \sum x_i w_i * \mathrm{A}$$



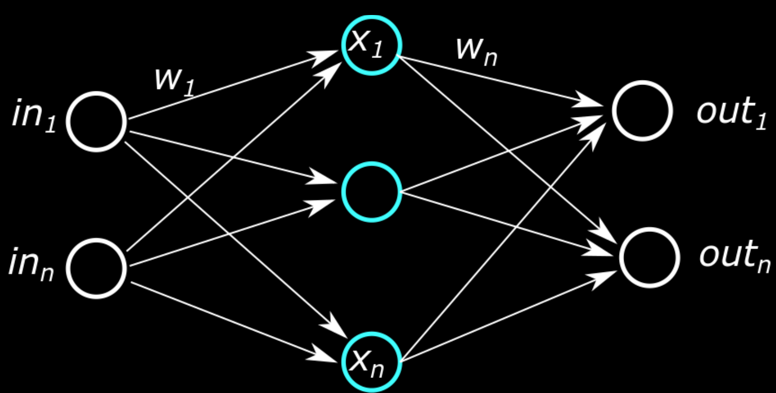
$$out = \sum x_i w_i * \mathrm{A}$$



$$\begin{cases} out_1 = \sum x_i w_i * \mathrm{A} \\ out_n = \sum x_j w_j * \mathrm{A} \end{cases}$$



$$out = \sum x_i w_i * \mathrm{A}$$



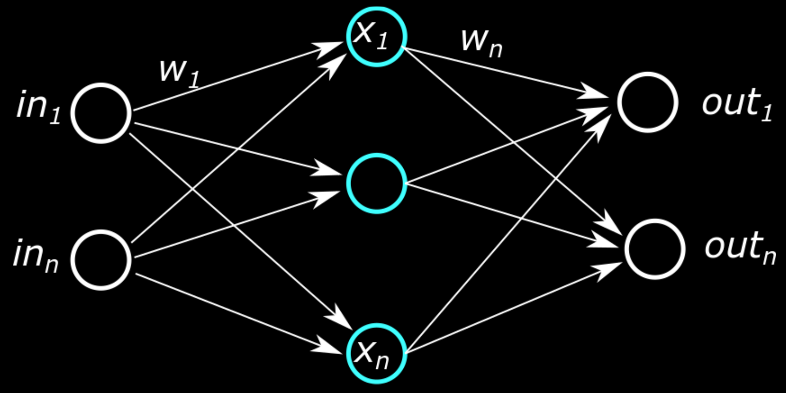
$$x_i = \sum in_i w_i * \mathrm{A}$$

$$\begin{cases} out_1 = \sum x_i w_i * \mathrm{A} \\ out_n = \sum x_j w_j * \mathrm{A} \end{cases}$$

Reti neurali lineari

Connessione:

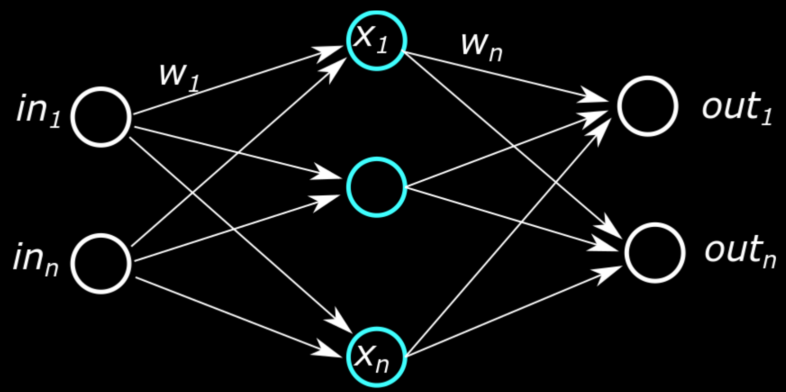
- Feed forward
- Ricorsive



Reti neurali lineari

Connessione:
- Feed forward
- Ricorsive

Task:
- Classificazione
- Regressione



Reti neurali lineari

Connessione:

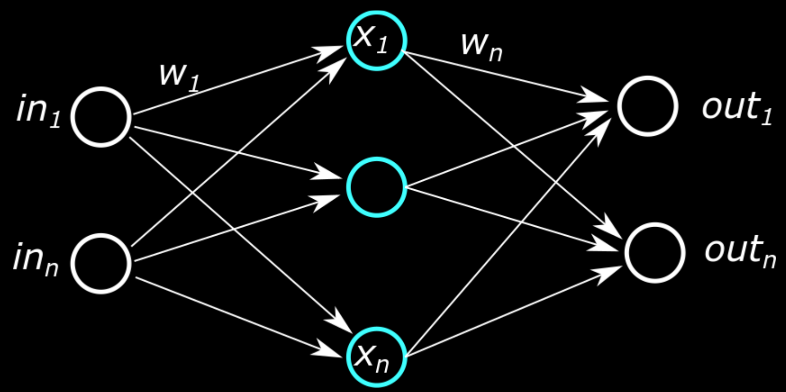
- Feed forward
- Ricorsive

Task:

- Classificazione
- Regressione

Tipologia:

- Singolo livello
- Multi-livello

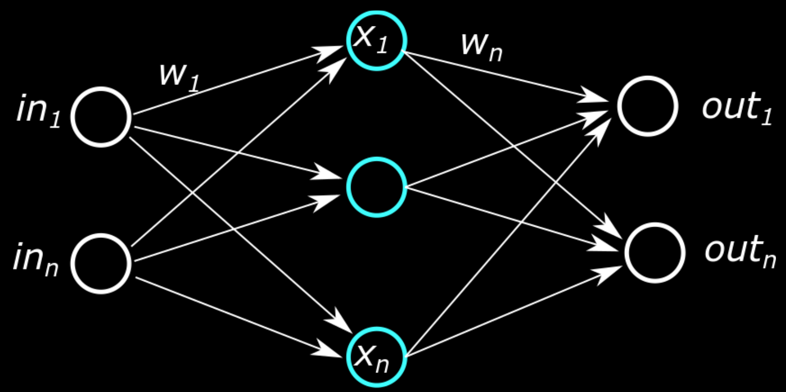


Reti neurali lineari

Connessione:
- Feed forward
- Ricorsive

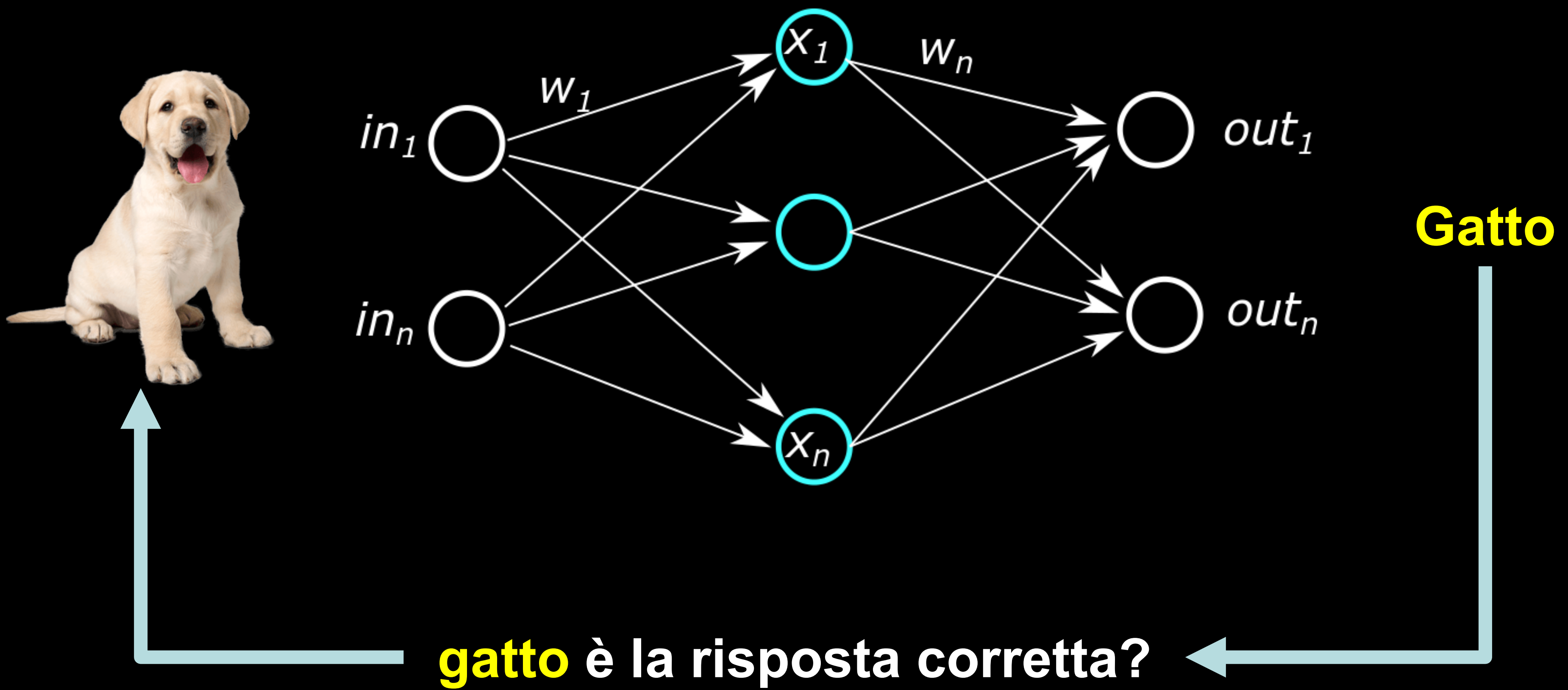
Task:
- Classificazione
- Regressione

Tipologia:
- Singolo livello
- Multi-livello

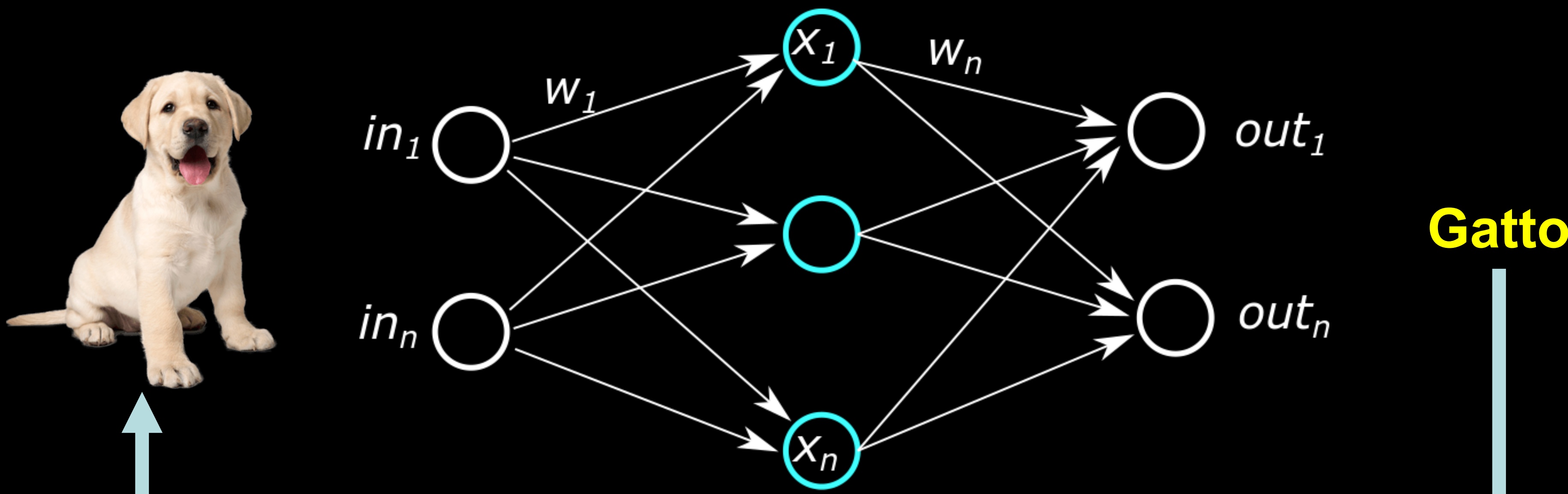


Metodo di apprendimento:
- Supervisionato
- Semi-supervisionato
- Non-supervisionato
- Rinforzato

Procedure di apprendimento



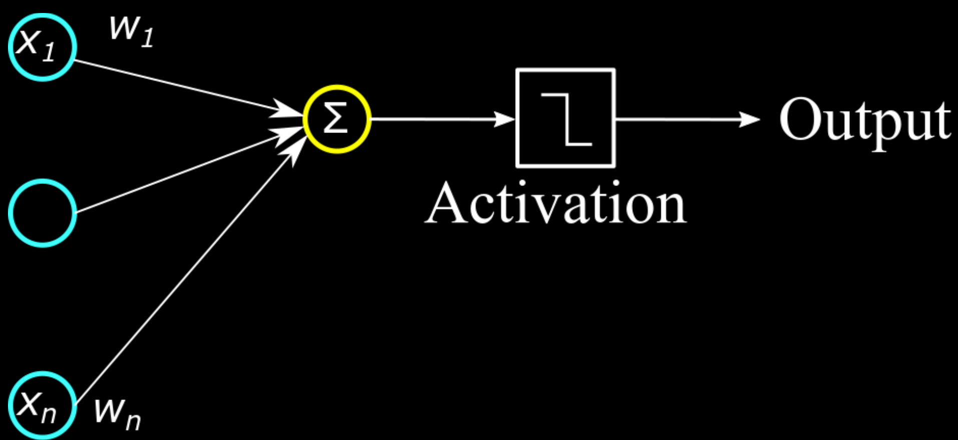
Procedure di apprendimento



gatto è la risposta corretta?

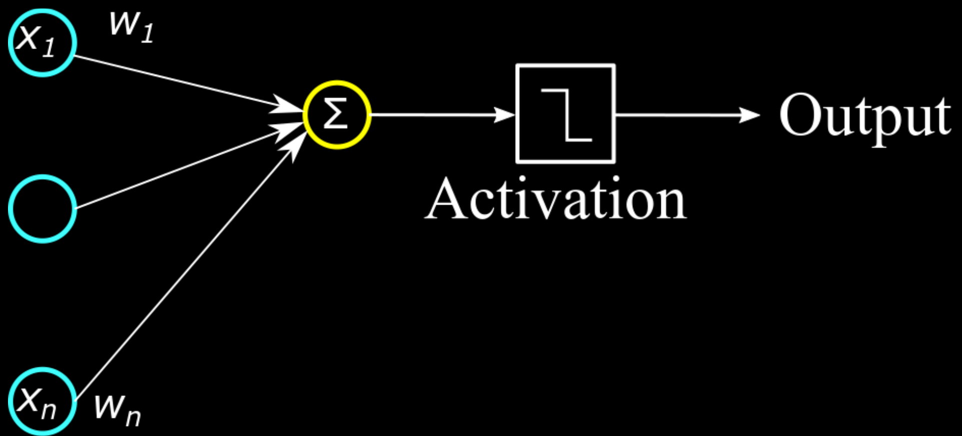
Se SI → ok, la rete sta funzionando bene
se NO → male, I pesi hanno bisogno di essere aggiustati

Procedure di apprendimento (singolo neurone)



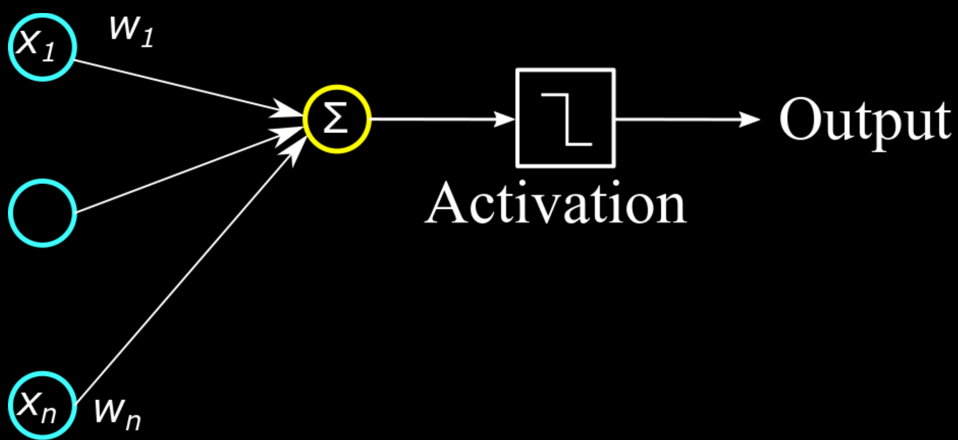
$$y = \varphi\left(\sum \omega_i x + b_i\right)$$

Procedure di apprendimento (singolo neurone)



$$y = \varphi\left(\sum \omega_i x + b_i\right) \qquad \omega_{new} = \omega_{old} - l_r\left(\frac{\delta\ error}{\delta\ \omega}\right)$$

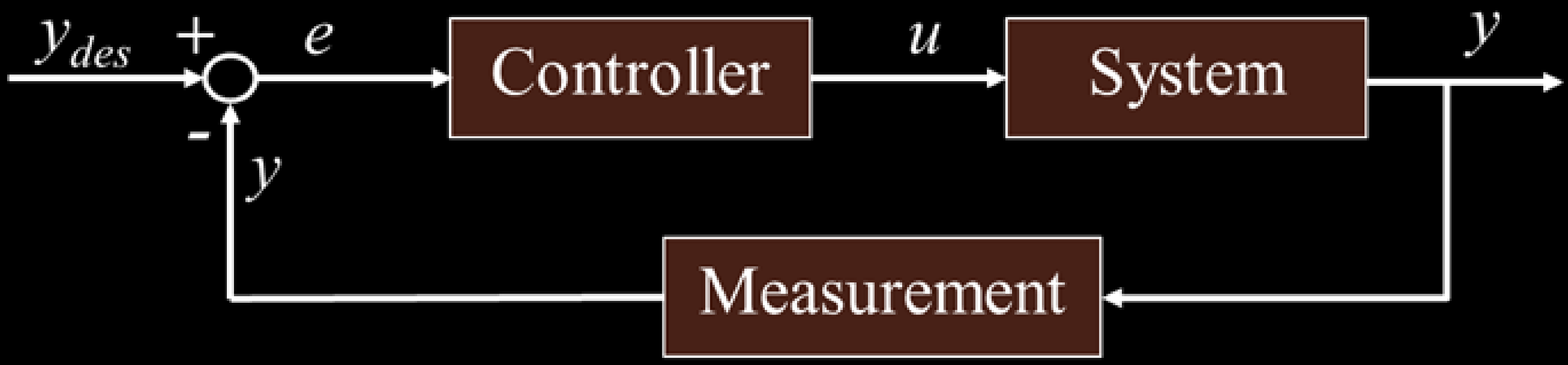
Procedure di apprendimento (singolo neurone)



$$error = y - y_{des}$$

$$y = \varphi\left(\sum \omega_i x + b_i\right) \qquad \omega_{new} = \omega_{old} - l_r\left(\frac{\delta\, error}{\delta\, \omega}\right)$$

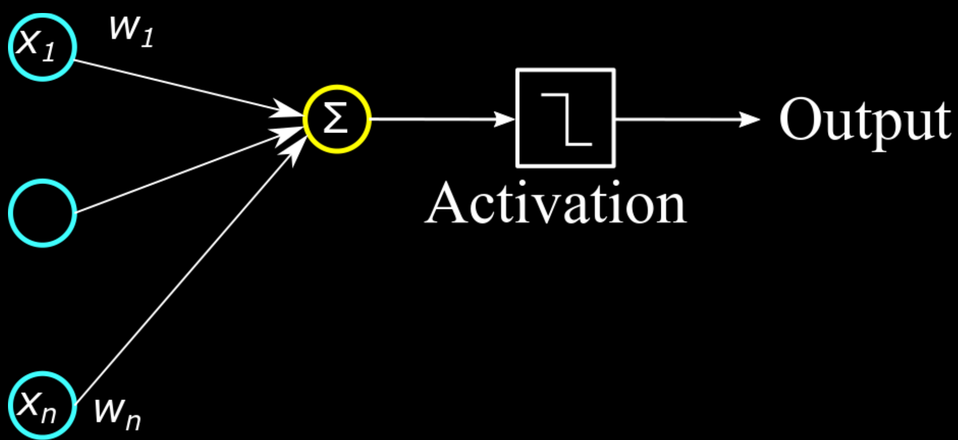
Analogia con sistemi di controllo



$$error = y - y_{des}$$

$$y = \varphi\left(\sum \omega_i x + b_i\right) \qquad \omega_{new} = \omega_{old} - l_r\left(\frac{\delta\, error}{\delta\, \omega}\right)$$

Procedure di apprendimento (singolo neurone)



Learning rate

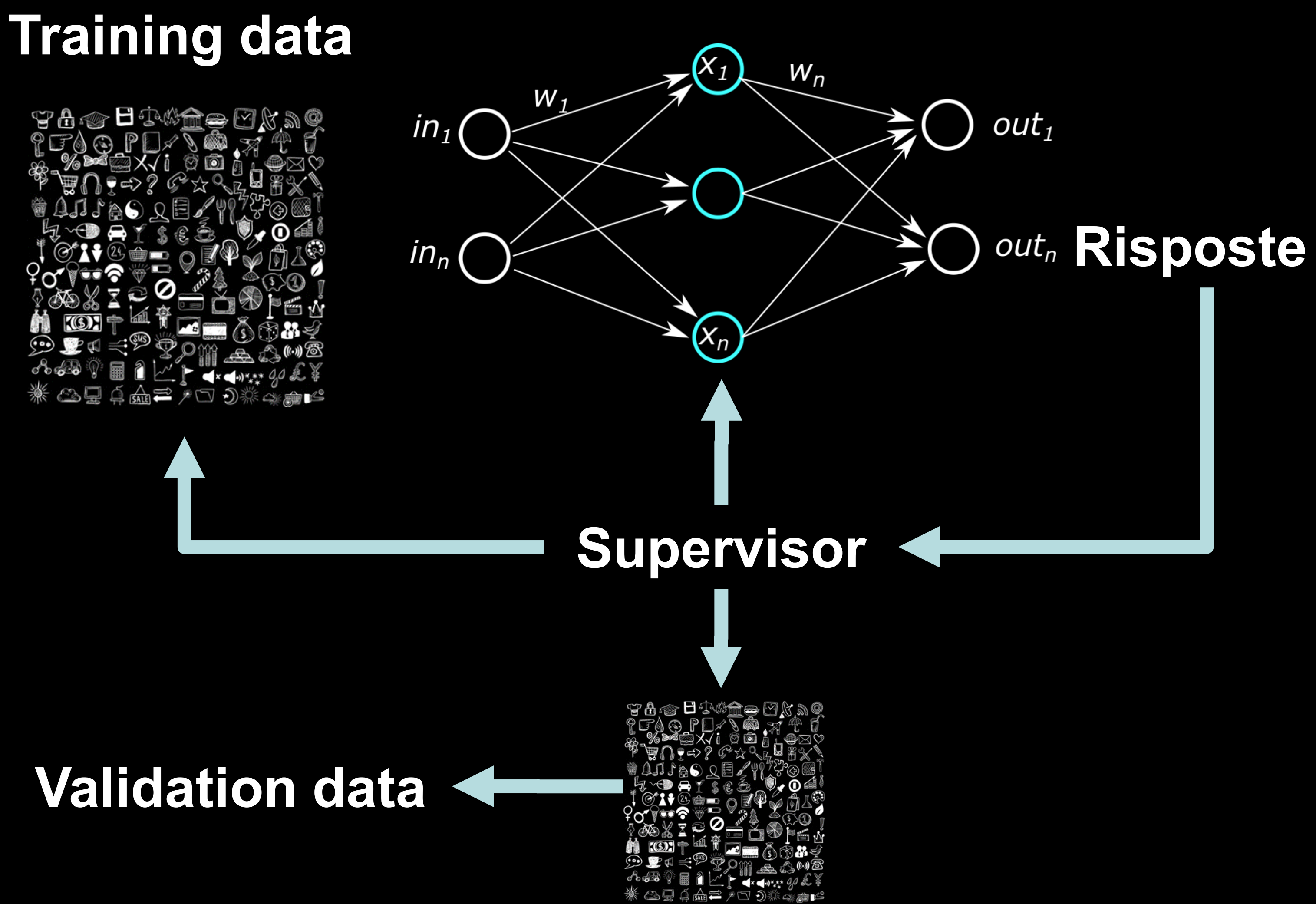
$$error = y - y_{des}$$

$$y = \varphi\left(\sum \omega_i x + b_i\right) \qquad \omega_{new} = \omega_{old} - \boxed{l_r}\left(\frac{\delta\, error}{\delta\, \omega}\right)$$

Procedure di apprendimento

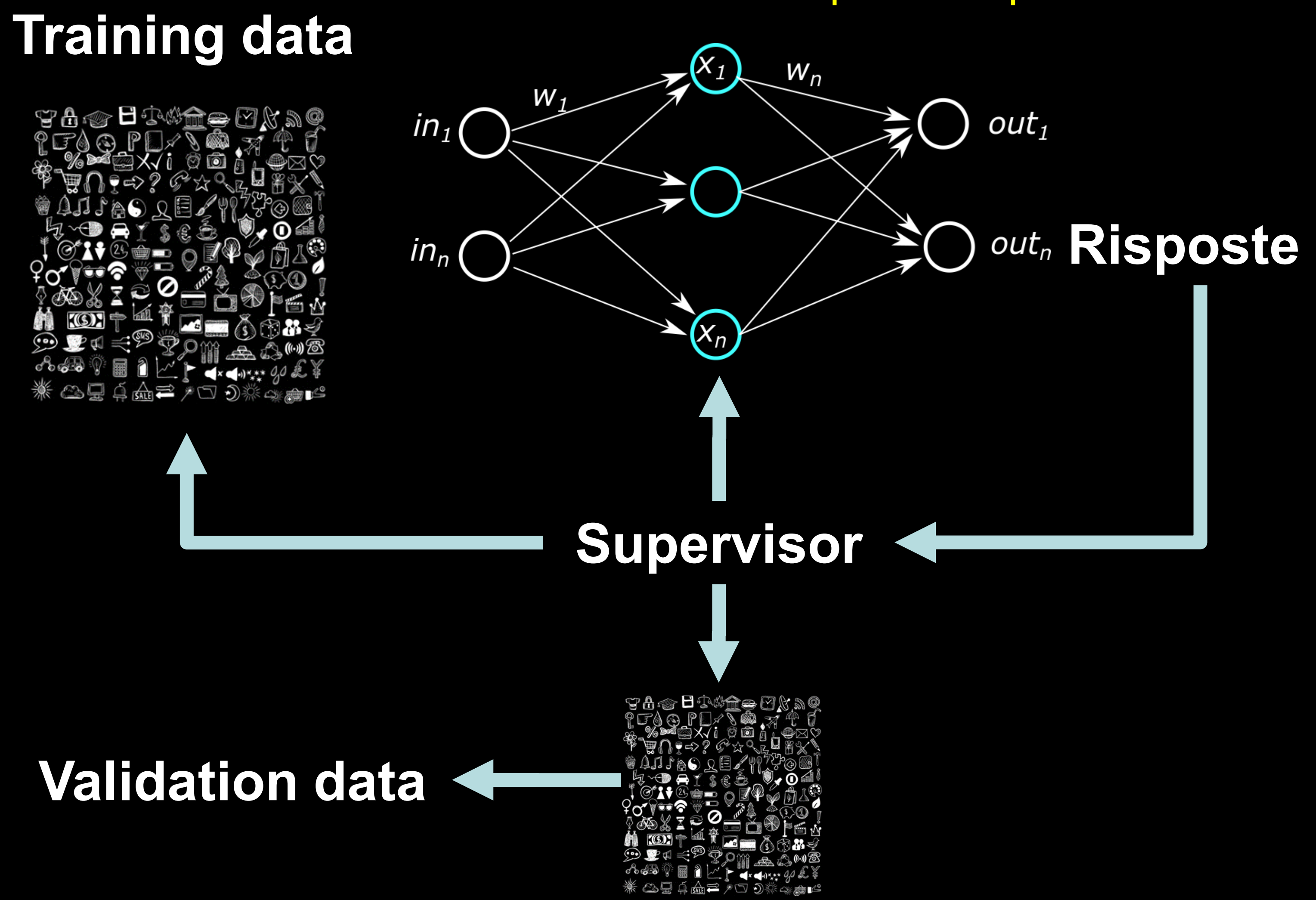
- ✓ Apprendimento supervisionato
- ✓ Apprendimento semi-supervisionato
- ✓ Apprendimento non supervisionato

Apprendimento supervisionato



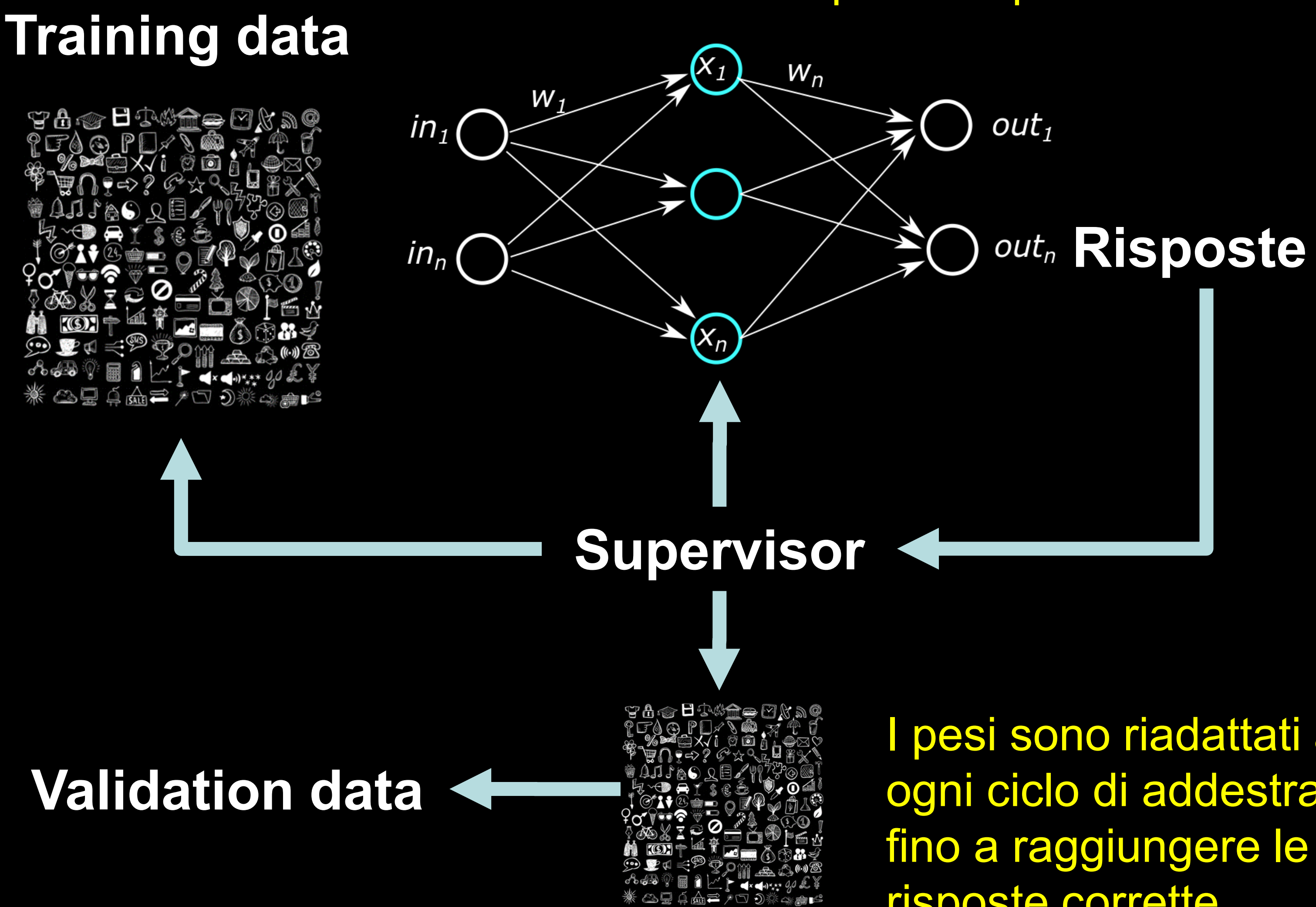
Apprendimento supervisionato

Richiede che per ogni esempio di training
che siano noti input e output desiderato



Apprendimento supervisionato

Richiede che per ogni esempio di training che siano noti input e output desiderato



I pesi sono riadattati ad ogni ciclo di addestramento fino a raggiungere le risposte corrette

Annotazione

L’annotazione dati, necessaria per ogni sistema supervisionato, è necessaria per gli algoritmi di machine learning e consiste in una operazione lunga e costosa.

Tecniche di annotazione:

- manuale
- in crowdsourcing
- basata sull’uso
- data-driven
- con AI API-driven

Annotazione manuale

I progettisti del sistema si occupano di annotare i dati e organizzarli

Annotazione manuale

I progettisti del sistema si occupano di annotare i dati e organizzarli

Vantaggi:

- Richiede basso sforzo amministrativo e di annotazione
- I progettisti riescono a capire meglio i dati aumentandone la qualità
- I progettisti possono scoprire risvolti interessanti guardando i dati che possono essere incorporati negli algoritmi

Svantaggi

- non è scalabile
- costoso in termini di tempo e denaro
- piccoli team vengono facilmente saturati

Annotazione in crowdsourcing

Il crowdsourcing è un metodo scalabile di annotazione, specifiche piattaforme pagano gli utenti per generare le annotazioni su task specifici

Annotazione in crowdsourcing

Il crowdsourcing è un metodo scalabile di annotazione, specifiche piattaforme pagano gli utenti per generare le annotazioni su task specifici

Vantaggi:

- basso costo
- scalabile
- rapido

Svantaggi:

- richiede l’implementazione di controlli di qualità
- richiede conoscenza sulle piattaforme e i metodi di crowdsourcing

Annotazione basata sull’uso

In questo caso i dati sono associati al loro task in maniera già strutturata (label implicita).

Annotazione basata sull'uso

In questo caso i dati sono associati al loro task in maniera già strutturata (label implicita).

Vantaggi:

- gratis o basso costo
- scalabile
- richiede un costo amministrativo basso (se i dati sono di buona qualità)

Svantaggi:

- rumore nei dati
- i dati richiedono pulitura, adattamenti e post/pre-processing

Annotazione data driven

Le annotazioni sono generate tramite regole associative su un subset (annotato manualmente) di dati e successivamente esportato su un dataset più grande sufficientemente vicino al dataset di partenza

Annotazione data driven

Le annotazioni sono generate tramite regole associative su un subset (annotato manualmente) di dati e successivamente esportato su un dataset più grande sufficientemente vicino al dataset di partenza

Vantaggi:

- gratis
- scalabile
- richiede un costo amministrativo basso (se i dati sono di buona qualità)

Svantaggi:

- massima accuratezza nelle annotazione irraggiungibile
- dati poco rappresentativi (limita il potere di generalizzazione)

Annotazione con AI-based API

Se vogliamo risolvere un task sufficientemente vicino ad un task già studiato (es. image classification) possiamo inviare delle query al sistema e usare le annotazioni del loro modello come ground truth

Annotazione con AI-based API

Se vogliamo risolvere un task sufficientemente vicino ad un task già studiato (es. image classification) possiamo inviare delle query al sistema e usare le annotazioni del loro modello come ground truth

Vantaggi:

- basso costo (minore dell'annotazione manuale)
- scalabile
- richiede un costo amministrativo basso (se i dati sono di buona qualità)

Svantaggi:

- una API potrebbe contenere delle imprecisioni inserendo rumore nei dati
- API-call a pagamento

Pipeline di annotazione

Combinazione di diversi metodi di annotazione per generare dei dataset completamente annotati, esempi di flussi sono:

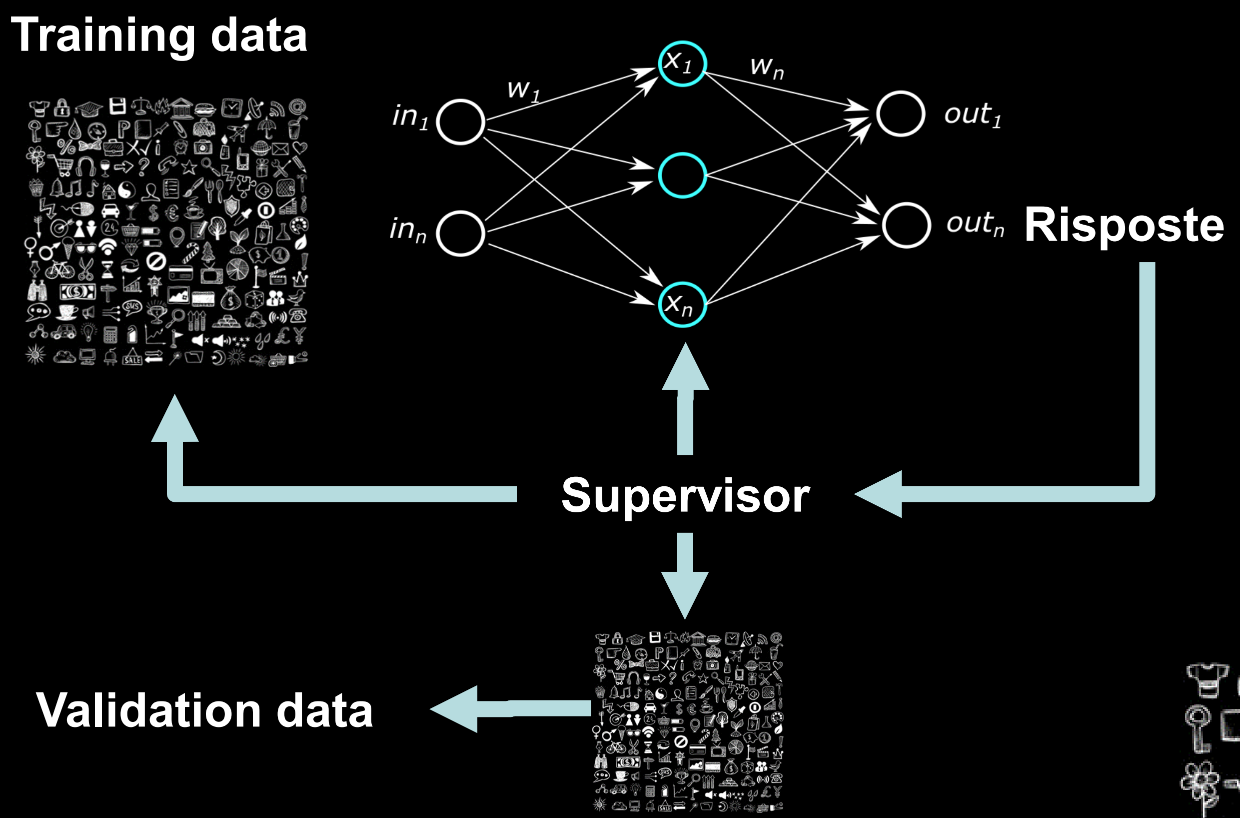
- Manuale | crowdsourcing
- API-driven => manuale | crowdsourcing
- Usage-based | data-driven => manuale | crowdsourcing
- Usage-based | data-driven => API-driven => manuale | crowdsourcing

Annotazione

è chiaro che queste ultime tecniche di annotazione aprono la strada ad altri sistemi di apprendimento che non sono supervisionati

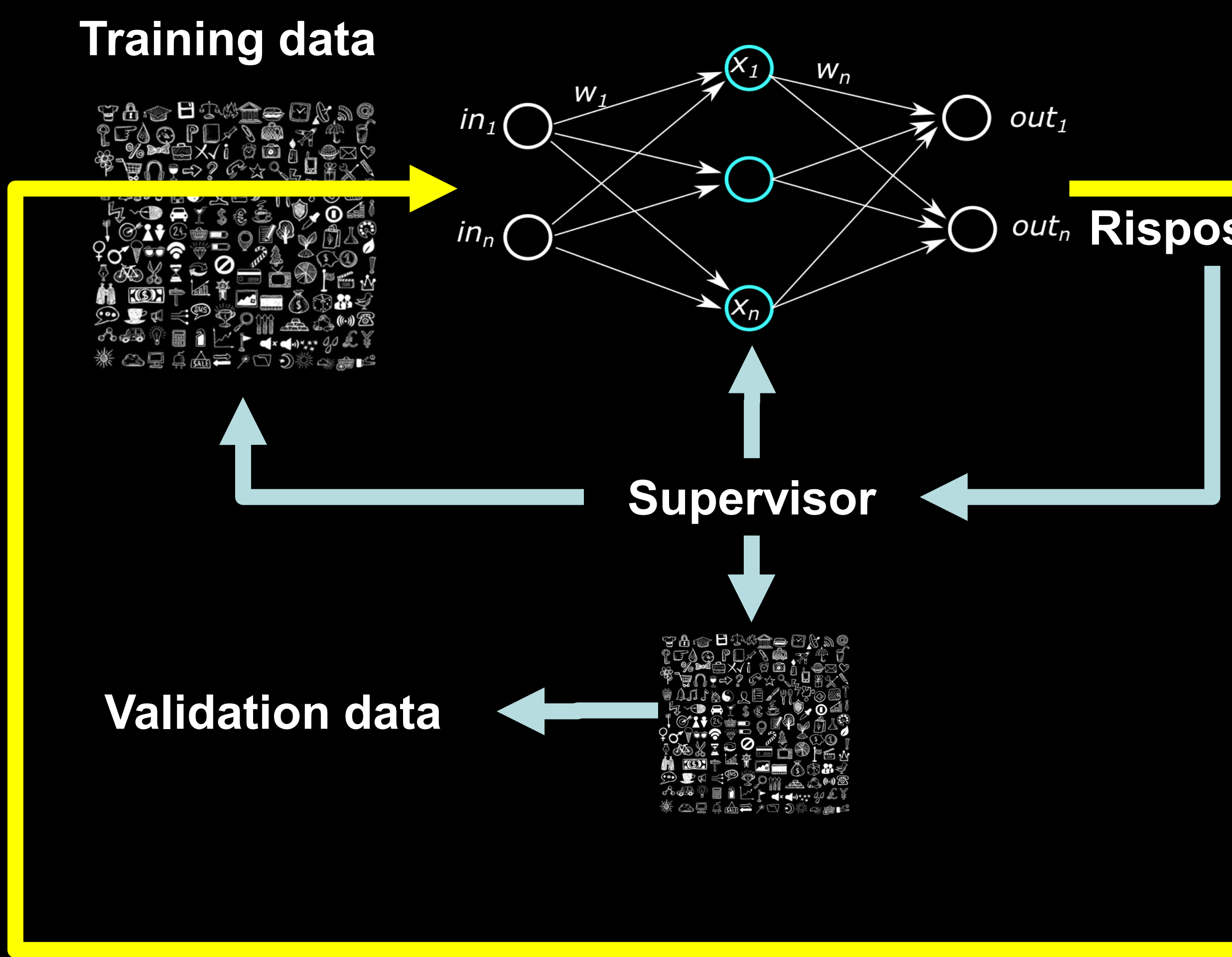
- Learning semi-supervisionato
- Weakly Supervised Learning
- Learning non-supervisionato
- Active learning

Semi-supervised learning



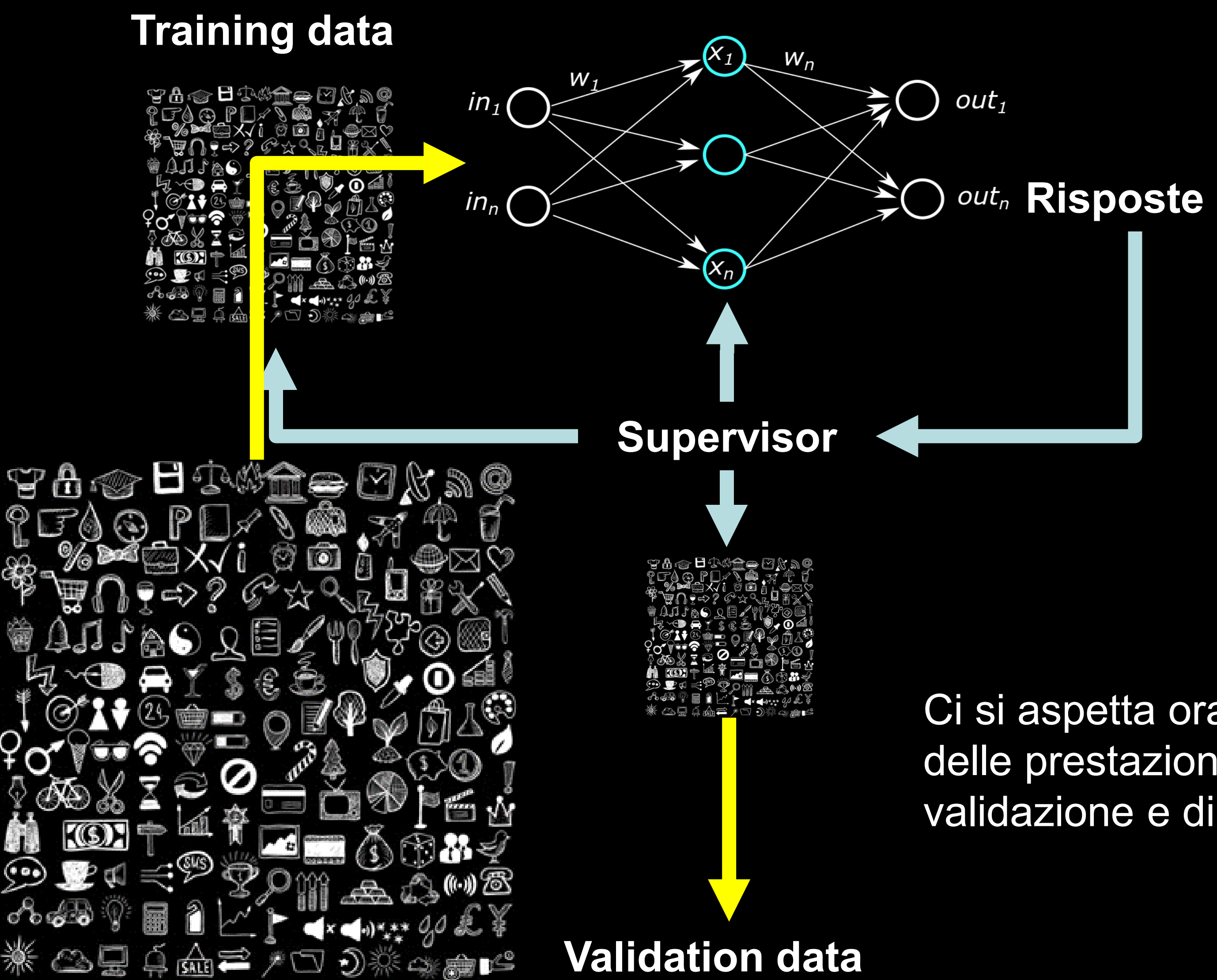
Supponiamo di avere a disposizione una enorme quantità di dati non annotati e di voler usare questi dati per astrarre nuova conoscenza

Semi-supervised learning



L’inferenza della rete sui dati non annotati genera delle “pseudo labels”

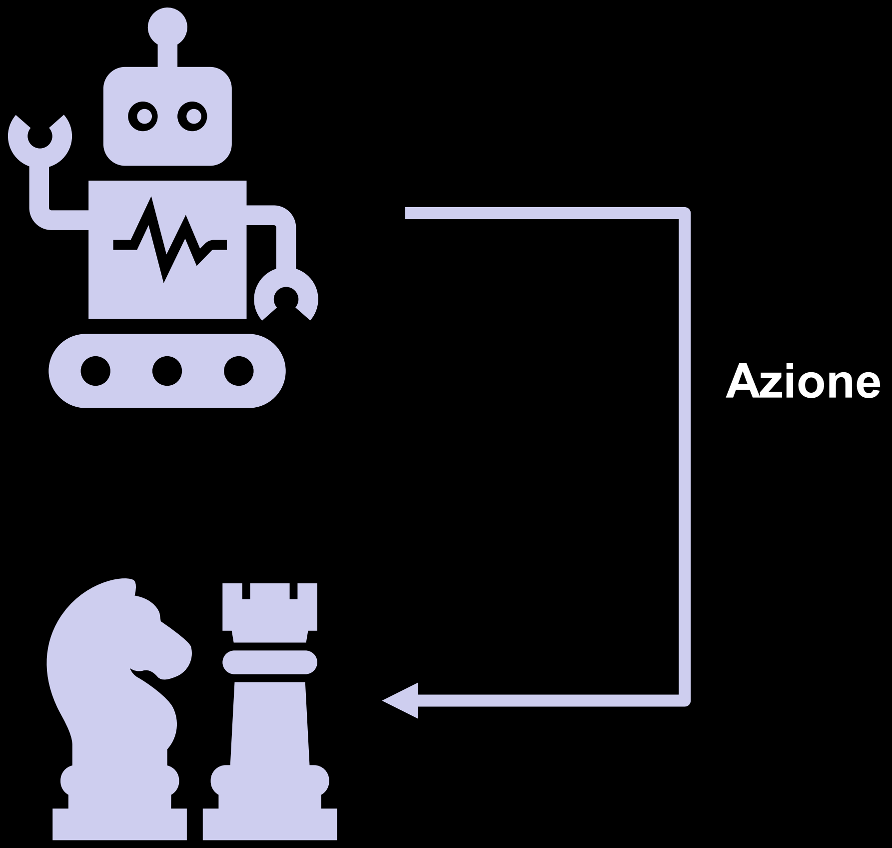
Semi-supervised learning



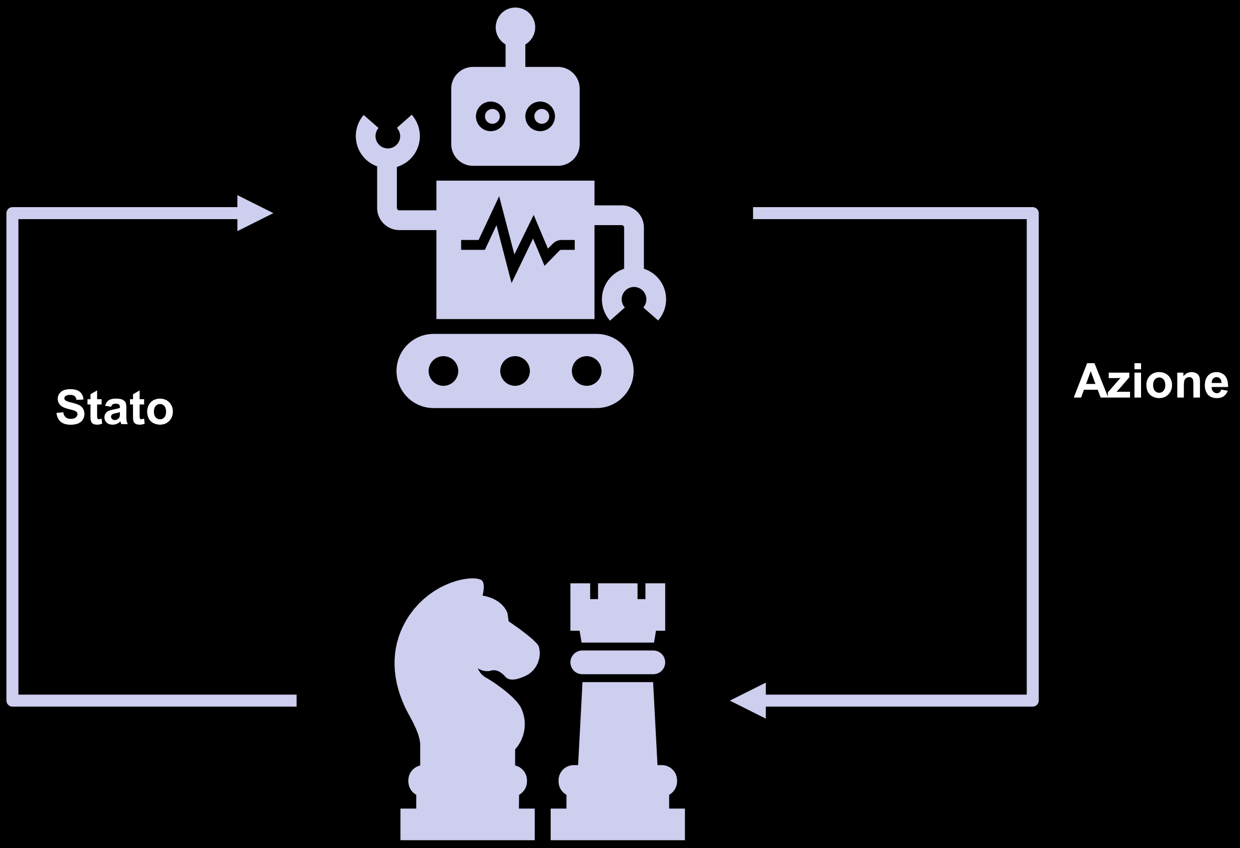
Ci si aspetta ora un miglioramento delle prestazioni sui dati di validazione e di test


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

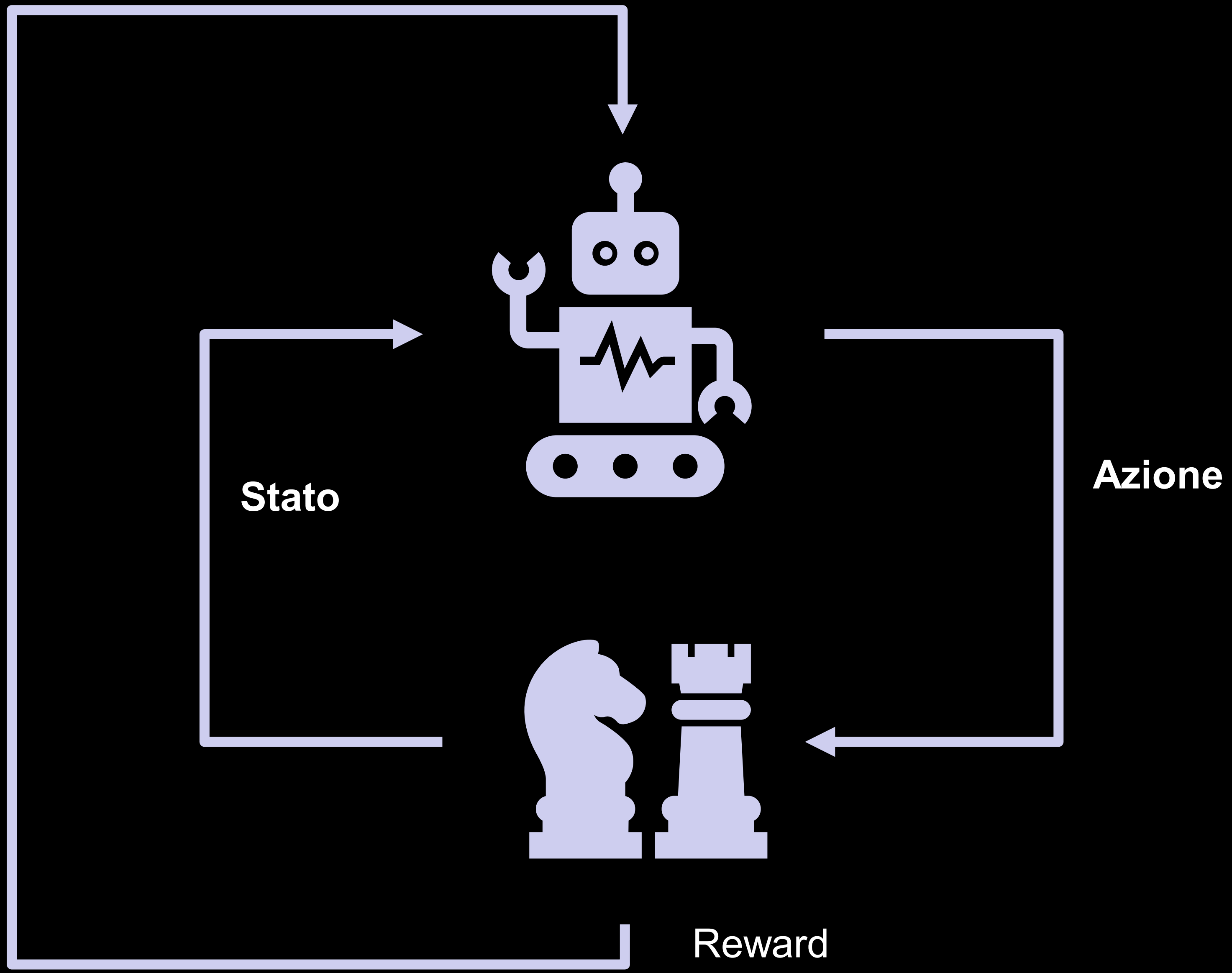
Reinfocement learning



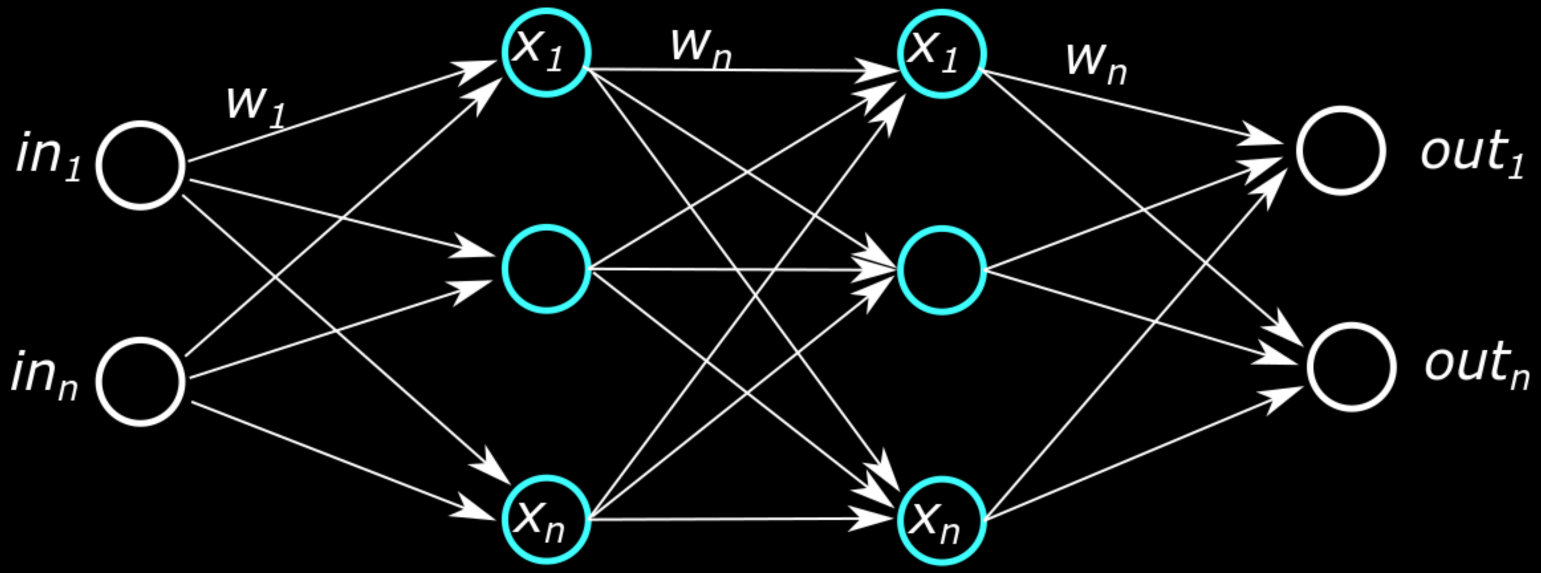
Reinfocement learning



Reinfocement learning



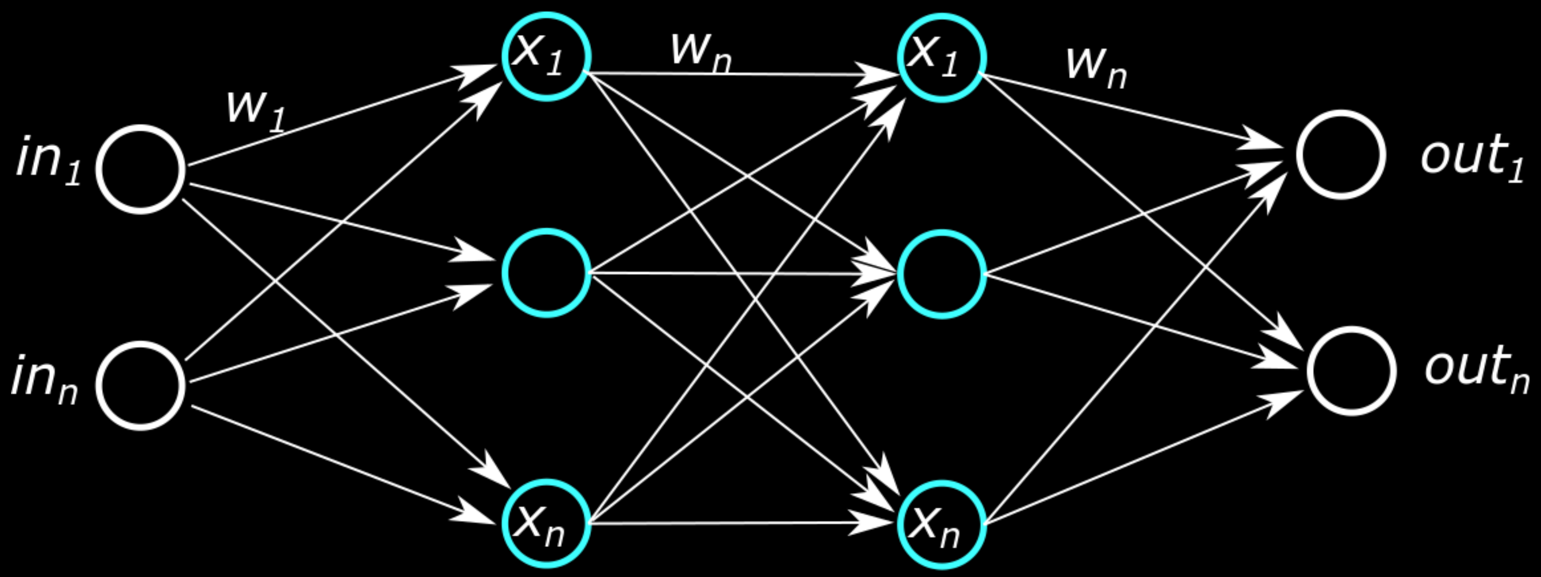
Parallelismo



Ogni neurone nello stesso livello è indipendente dagli altri

Parallelismo

NVIDIA CUDA

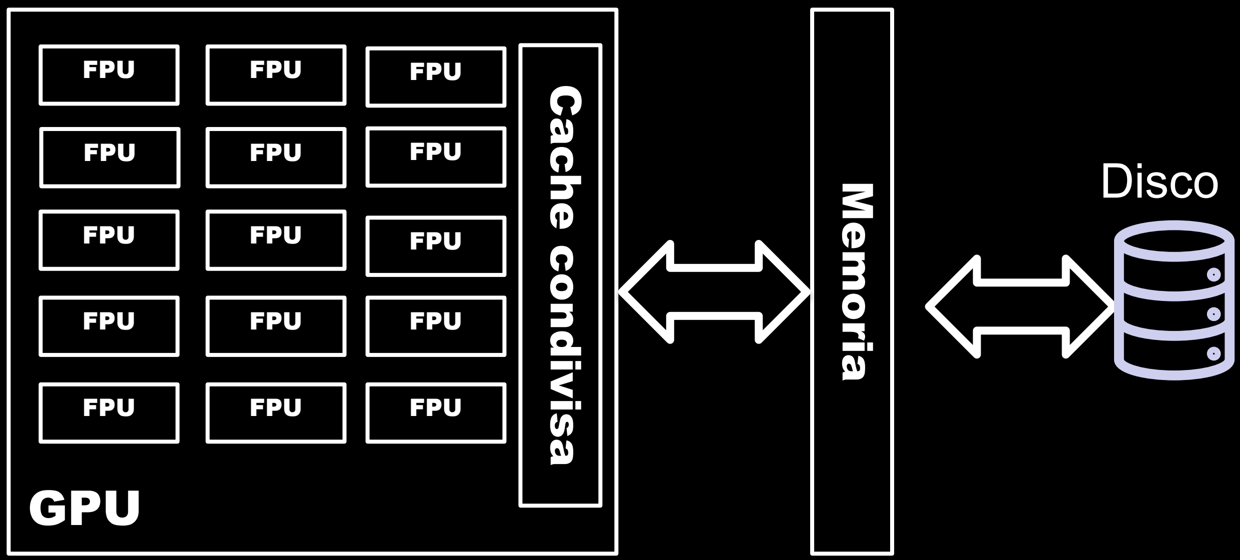


Ogni neurone nello stesso livello è indipendente dagli altri

Ogni livello dipende solo dal precedente livello

Parallelizzazione su CPU VS GPU

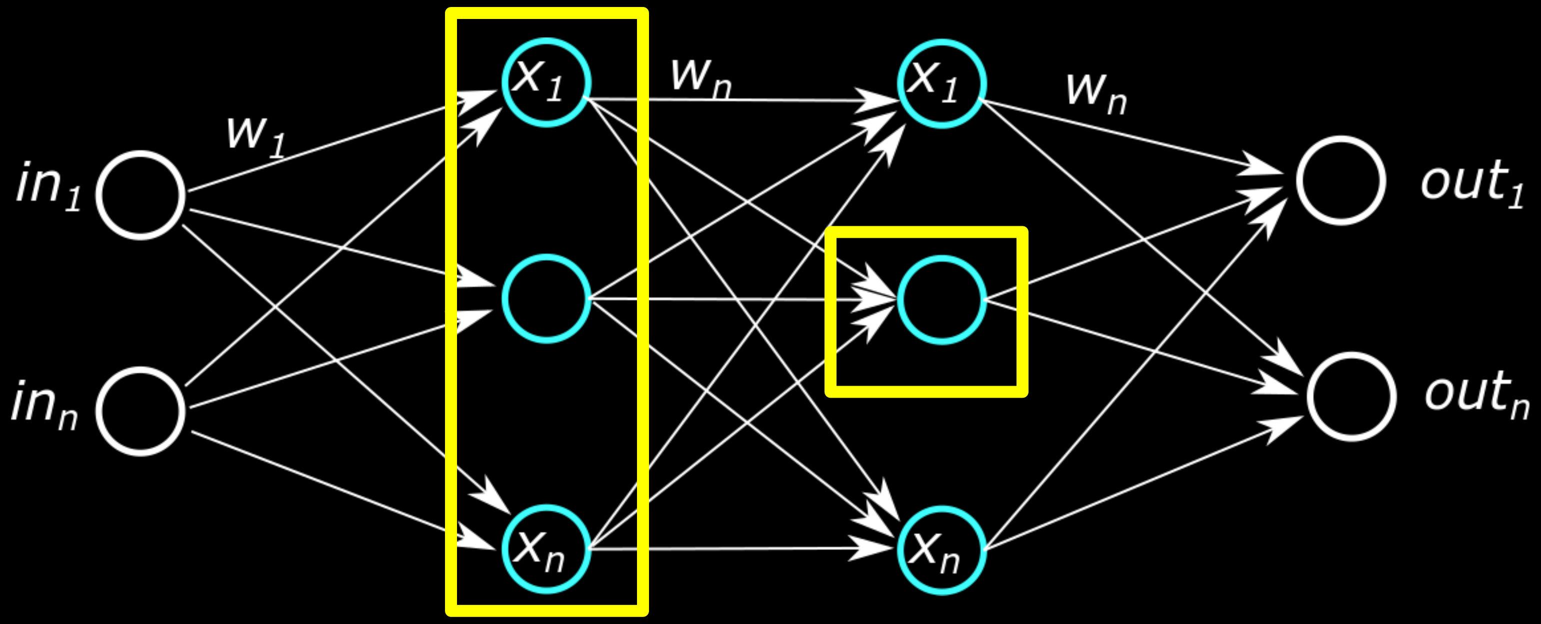
Entrambe hanno FPU cores che fanno somme e moltiplicazioni, la principale differenza è che le GPU fanno la stessa operazione (eseguono la stessa istruzione di basso livello) allo stesso momento su molticores diversi con input diversi



Obiettivo: alta larghezza di banda

Di fatto tutte le FPU eseguono la stessa istruzione macchina allo stesso tempo sulla cache condivisa

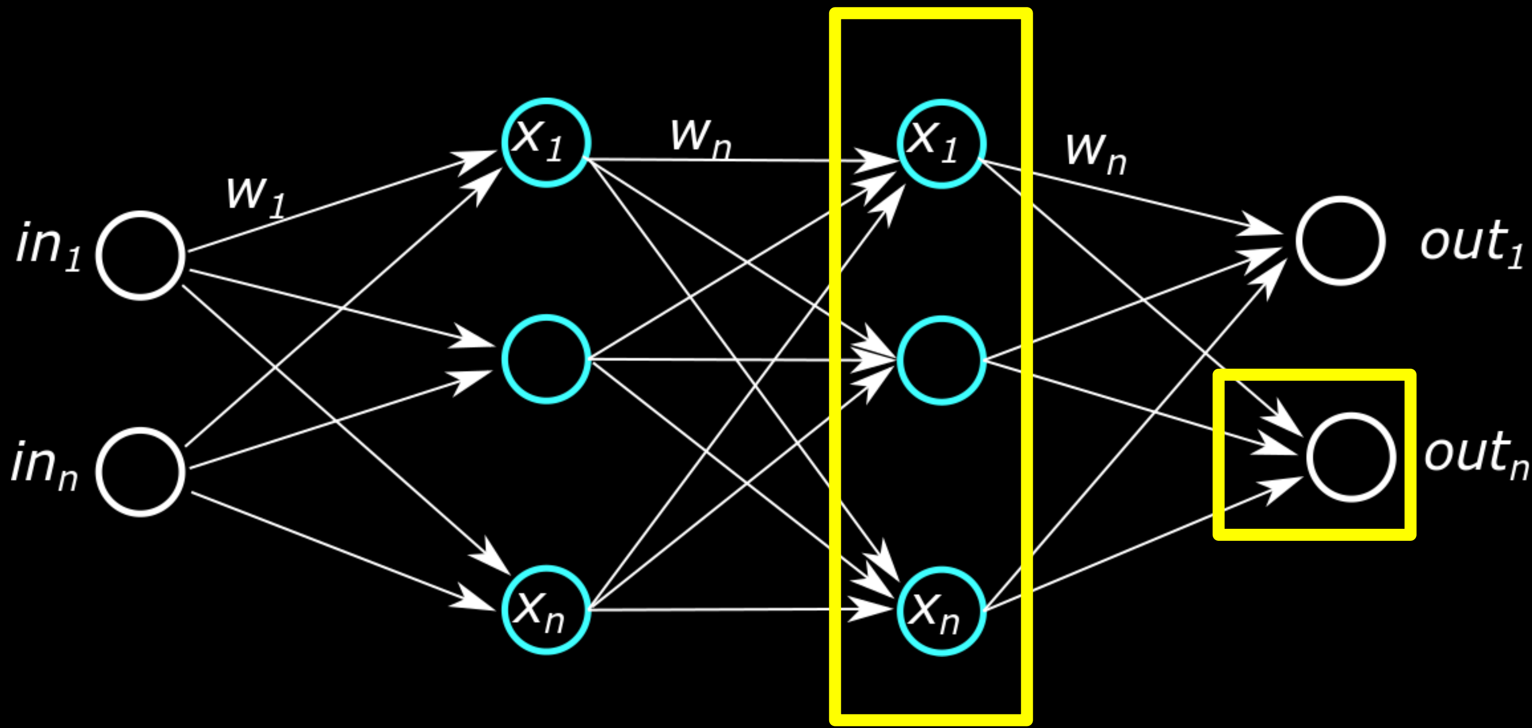
Parallelismo



Ogni neurone nello stesso livello è indipendente dagli altri

Ogni livello dipende solo dal precedente livello

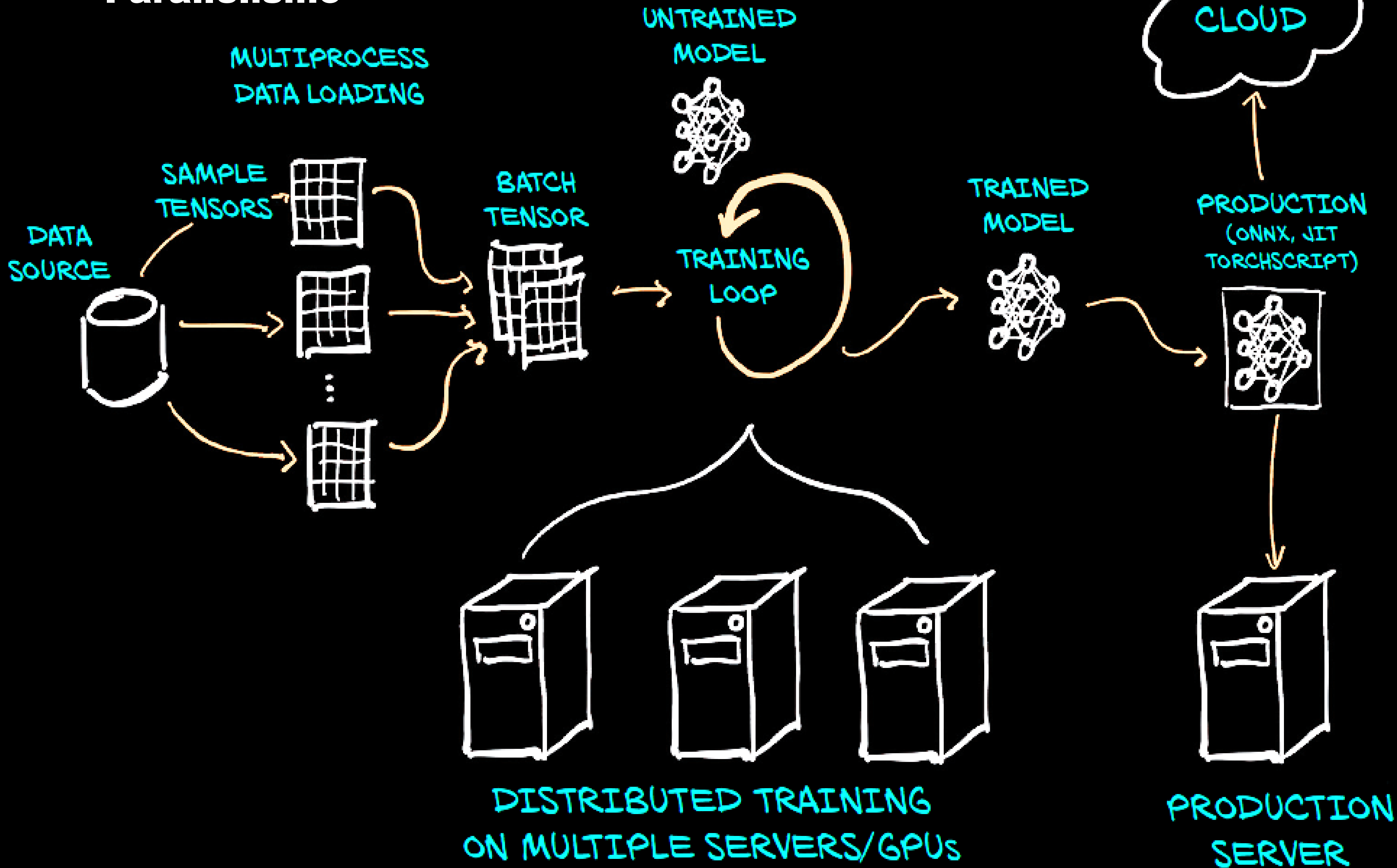
Parallelismo



Ogni neurone nello stesso livello è indipendente dagli altri

Ogni livello dipende solo dal precedente livello

Parallelismo



source: Eli Stevens, Luca Antiga, Thomas Viehmann, “Deep Learning With Pytorch: Build, Train, and Tune Neural Networks Using Python Tools” 2020.

Multilayer Feedforward Networks are Universal Approximators

Kurt Hornik

Technische Universität Wien

Maxwell Stinchcombe and Halbert White

University of California, San Diego

(*Received* 16 *September* 1988; *revised and accepted* 9 *March* 1989)

Abstract—*This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.*

source: Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer feedforward networks are universal approximators. Neural networks, 2(5), pp.359-366.

Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK

Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERT WHITE

University of California, San Diego

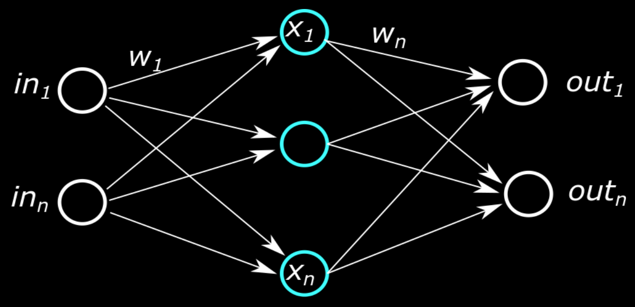
(*Received 16 September* 1988; *revised and accepted 9 March* 1989)

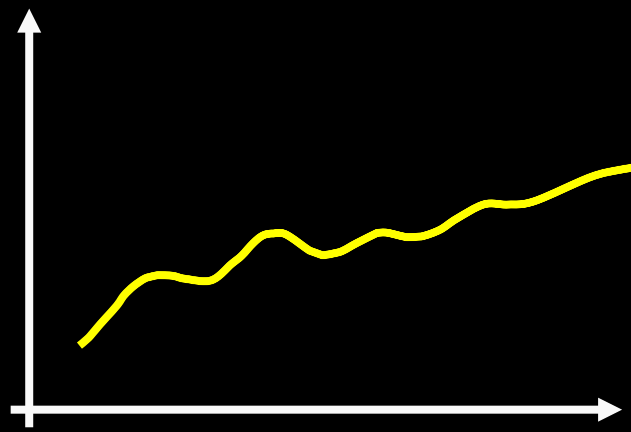
Abstract—*This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.*

Reti feedforward multilivello con almeno un livello nascosto aventi una arbitraria funzione di attivazione “squashing” sono in grado di approssimare qualunque funzione misurabile

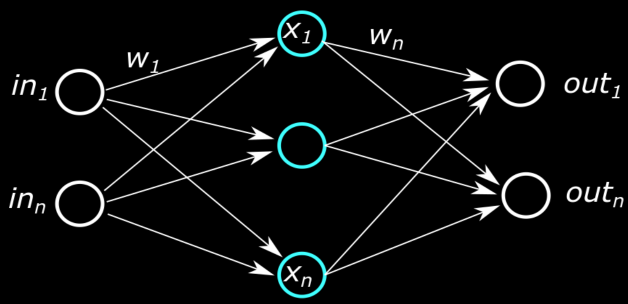
source: Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer feedforward networks are universal approximators. Neural networks, 2(5), pp.359-366.

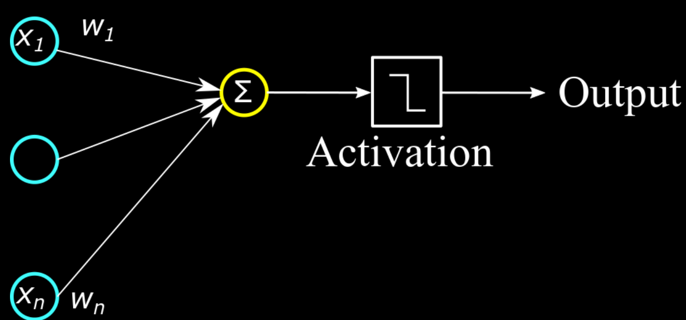
Teorema di approssimazione universale

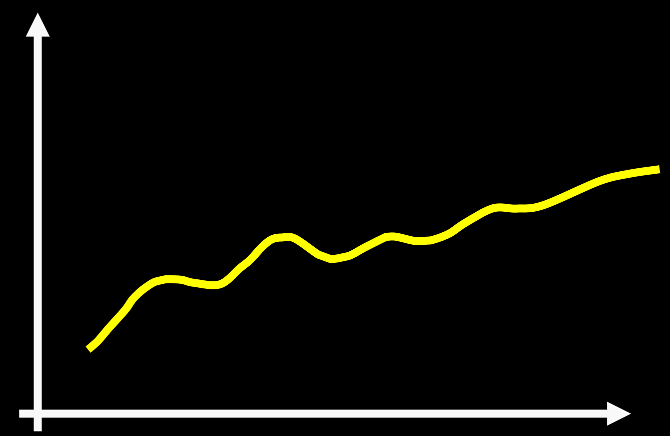




Teorema di approssimazione universale

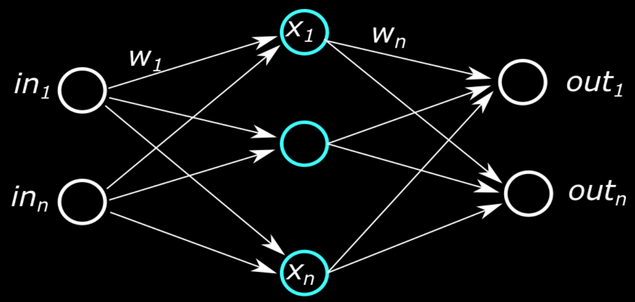


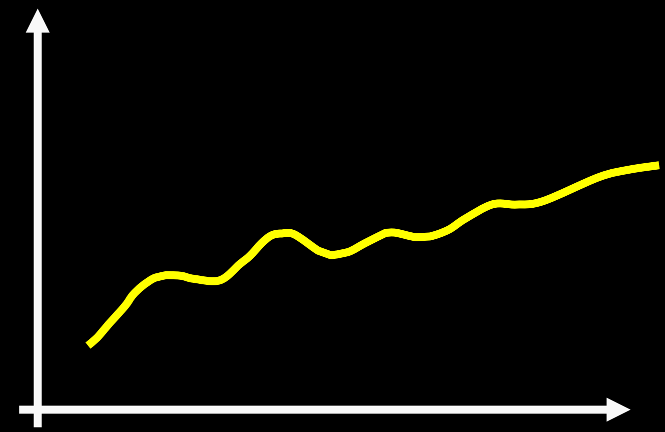


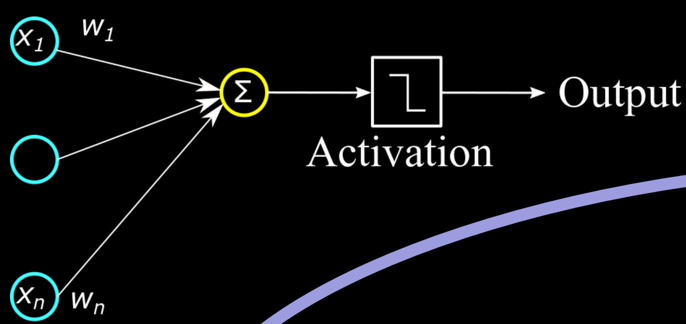


$$y = \sigma\left(\sum \omega_i x + b_i\right)$$

Teorema di approssimazione universale

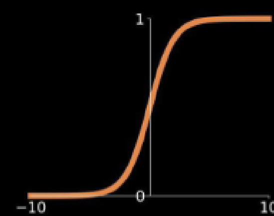






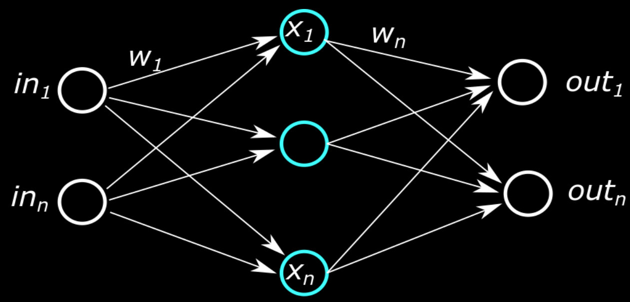
Sigmoid

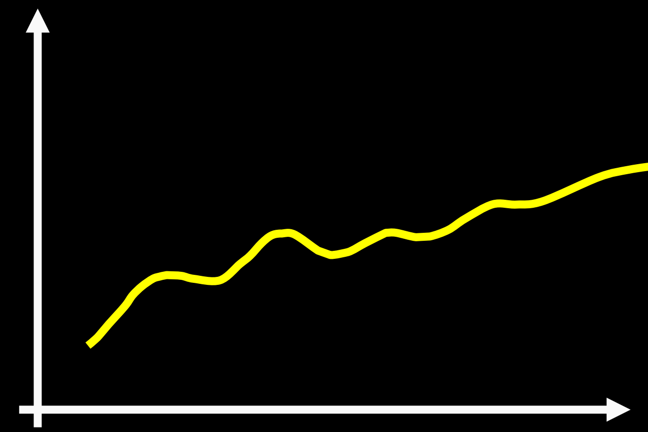
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

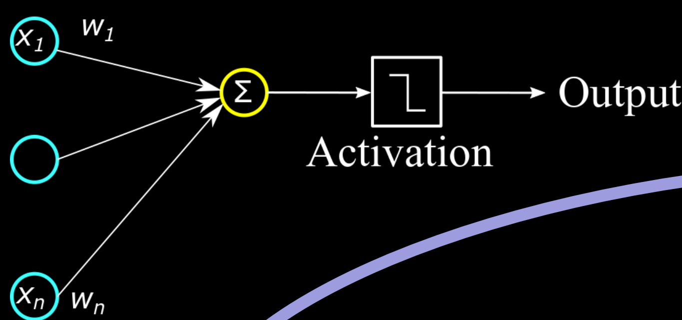


$$y = \sigma\left(\sum \omega_i x + b_i\right)$$

Teorema di approssimazione universale

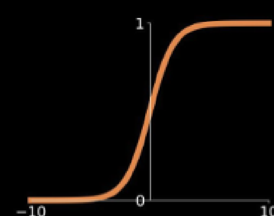


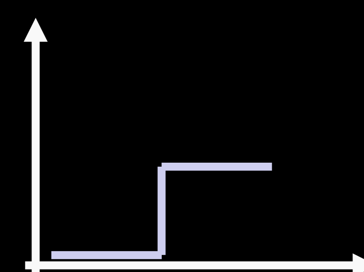




Sigmoid

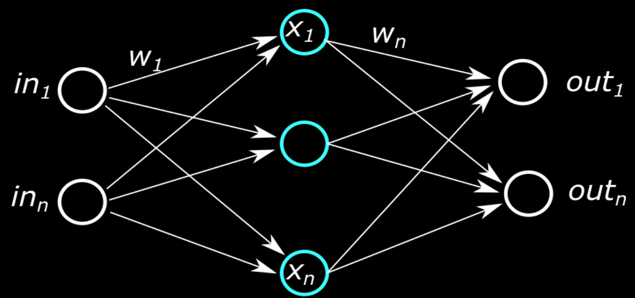
$\sigma(x) = \frac{1}{1+e^{-x}}$

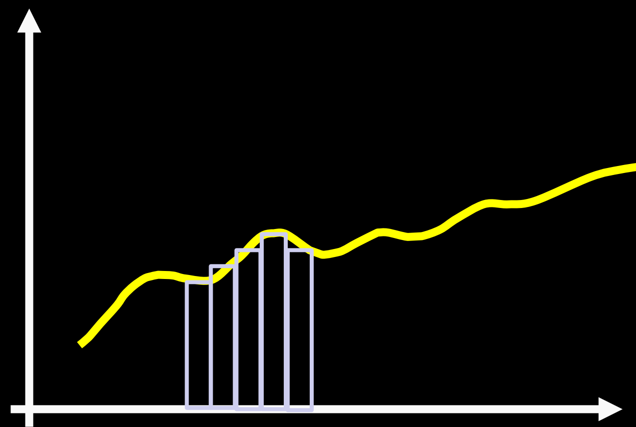


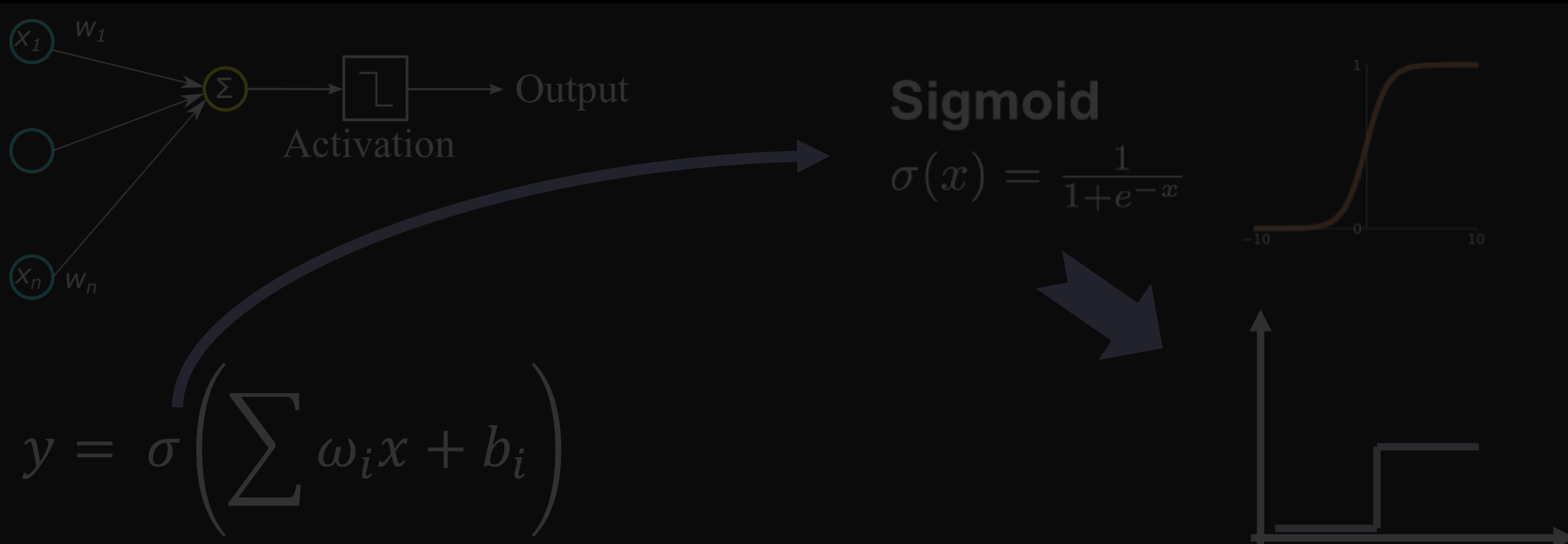


$$y = \sigma\left(\sum \omega_i x + b_i\right)$$

Teorema di approssimazione universale







Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK

Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERT WHITE

University of California, San Diego

available. These results establish multilayer feedforward networks as a class of universal approximators. As such, failures in applications can be attributed to inadequate learning, inadequate numbers of hidden units, or the presence of a stochastic rather than a deterministic relation between input and target. Our results do not address the issue of how many units are needed to attain a given degree of approximation.

source: Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer feedforward networks are universal approximators. Neural networks, 2(5), pp.359-366.

Multilayer Feedforward Networks are Universal Approximators

Kur' Hornik

Technische Universität Wien

Maxwell Stinchcombe and Halbek White

University of California, San Diego

available. These results establish multilayer feedforward networks as a class of universal approximators. As such, failures in applications can be attributed to inadequate learning, inadequate numbers of hidden units, or the presence of a stochastic rather than a deterministic relation between input and target. Our results do not address the issue of how many units are needed to attain a given degree of approximation.

Questi risultati stabiliscono che le reti neurali multilivello feedforward sono approssimatori universali. Ciò detto, ogni fallimento nelle loro applicazioni può essere attribuito a una procedura di apprendimento inadeguata, numero di livelli inadeguati, o la presenza di una relazione stocastica piuttosto che deterministica tra input e output

source: Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer feedforward networks are universal approximators. Neural networks, 2(5), pp.359-366.

No free lunch theorem

il nostro modello è una semplificazione del mondo reale

le semplificazioni sono basate su ipotesi (bias del modello)

le ipotesi falliscono in alcune situazioni

Conclusione: Non esiste un modello che funziona in tutti i casi possibili

No free lunch theorem

Il no free lunch theorem o «teorema dell'inesistenza del pranzo gratis» dice che, mediando tute le possibili distribuzioni di dati, ogni algoritmo di classificazione ha lo stesso margine di errore quando si classificano punti non precedentemente osservati

in altre parole

come il teorema di approssimazione universale, non esiste una rete migliore di qualunque altra e la differenza di prestazioni visibile sui dati di test è frutto di un bias

Teorema di approssimazione universale – Prova visuale

http://neuralnetworksanddeeplearning.com/chap4.html

Teorema di approssimazione universale

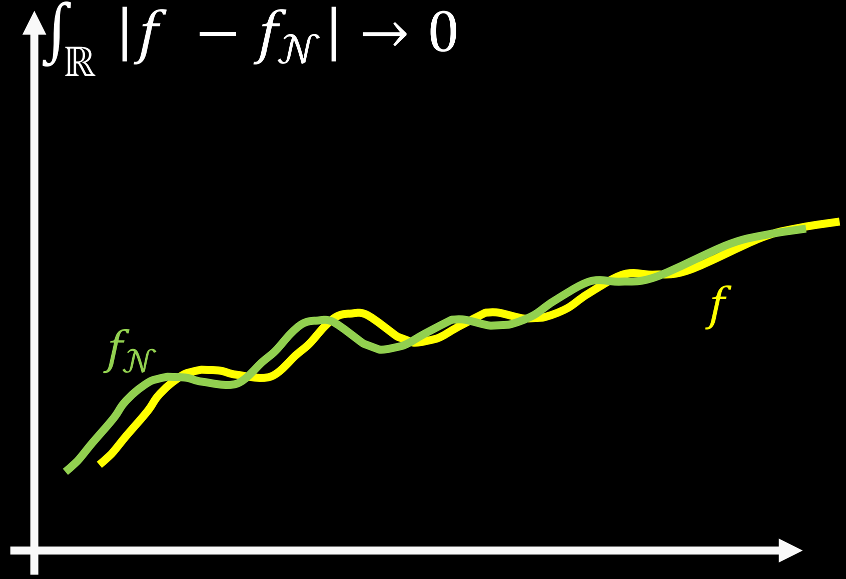
Una rete neurale può approssimare qualunque funzione:

$$\int_{\mathbb{R}} |f - f_{\mathcal{N}}| \to 0$$

Teorema di approssimazione universale

Una rete neurale può approssimare qualunque funzione:

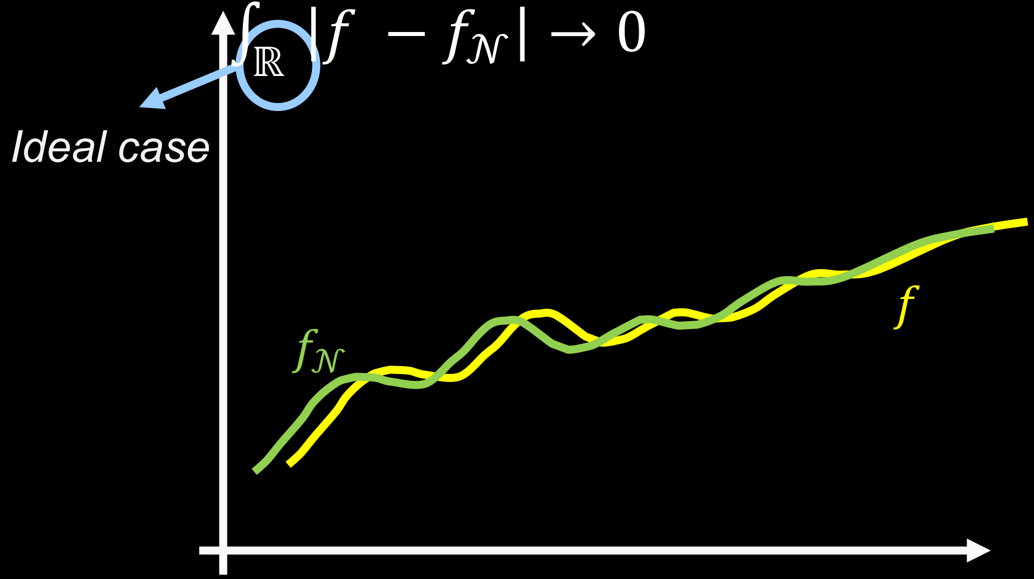
$$\int_{\mathbb{R}} |f - f_{\mathcal{N}}| \to 0$$



Teorema di approssimazione universale

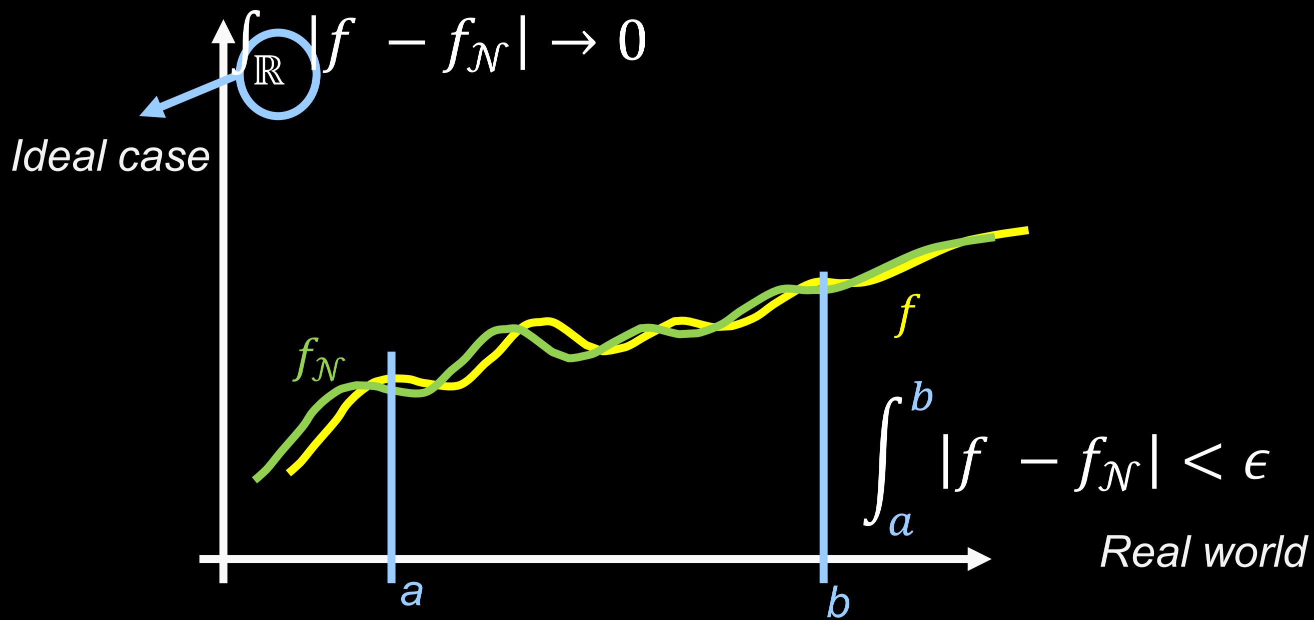
Una rete neurale può approssimare qualunque funzione:

$$\int_{\mathbb{R}} |f - f_{\mathcal{N}}| \rightarrow 0$$

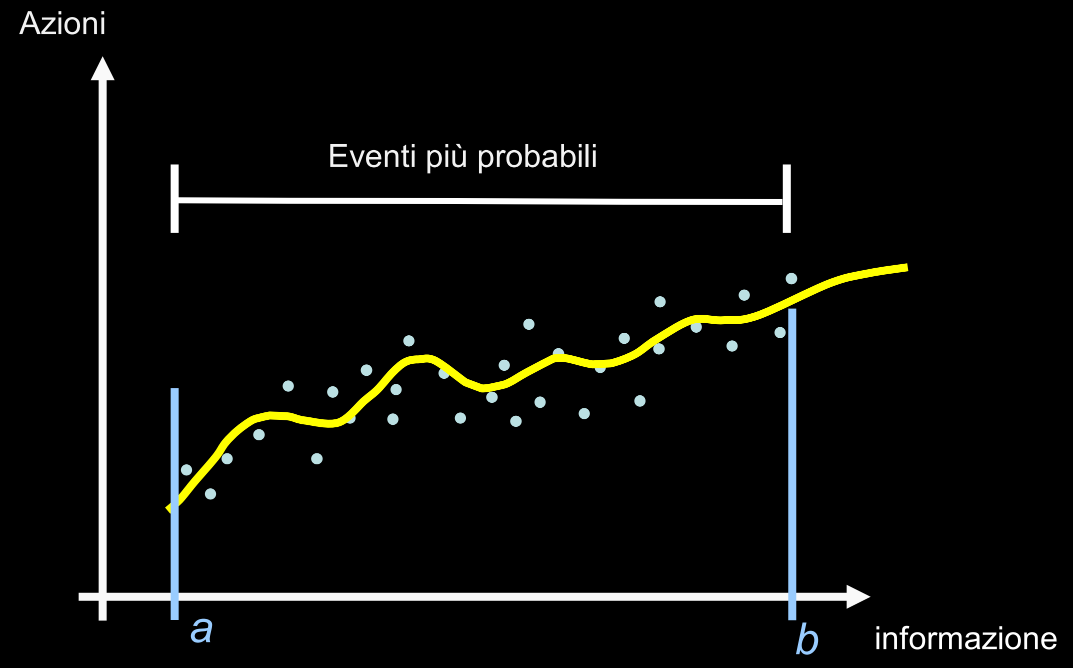


Teorema di approssimazione universale

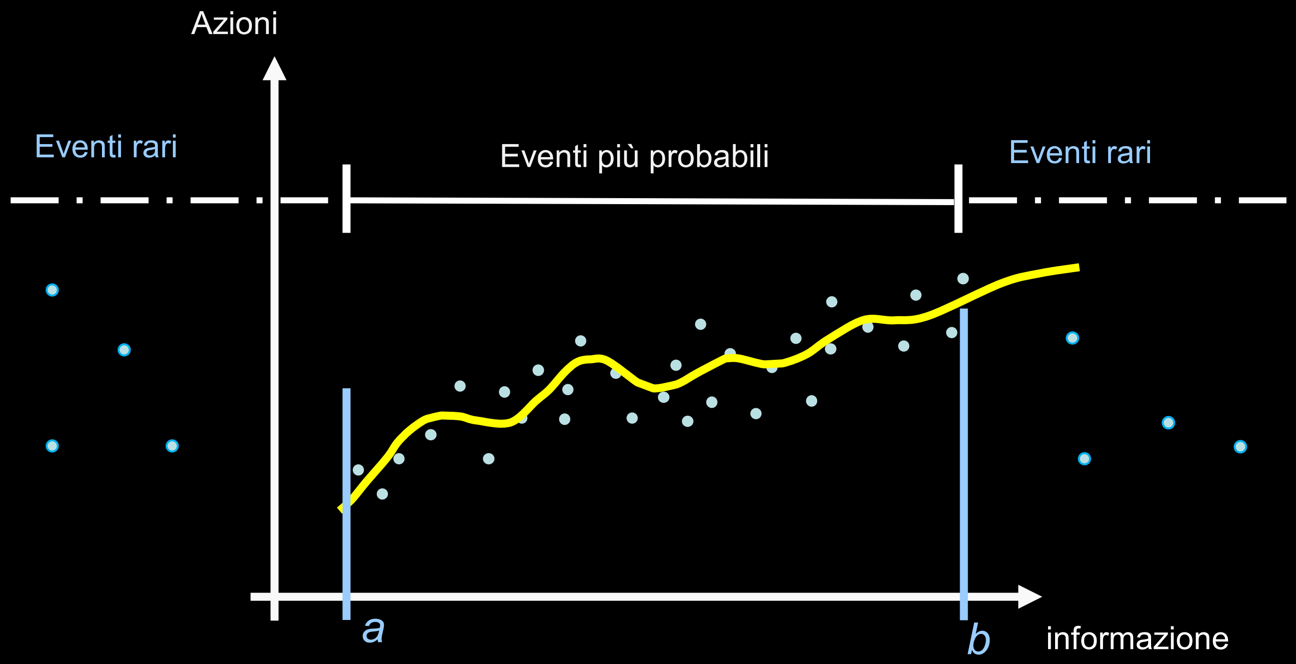
Una rete neurale può approssimare qualunque funzione:

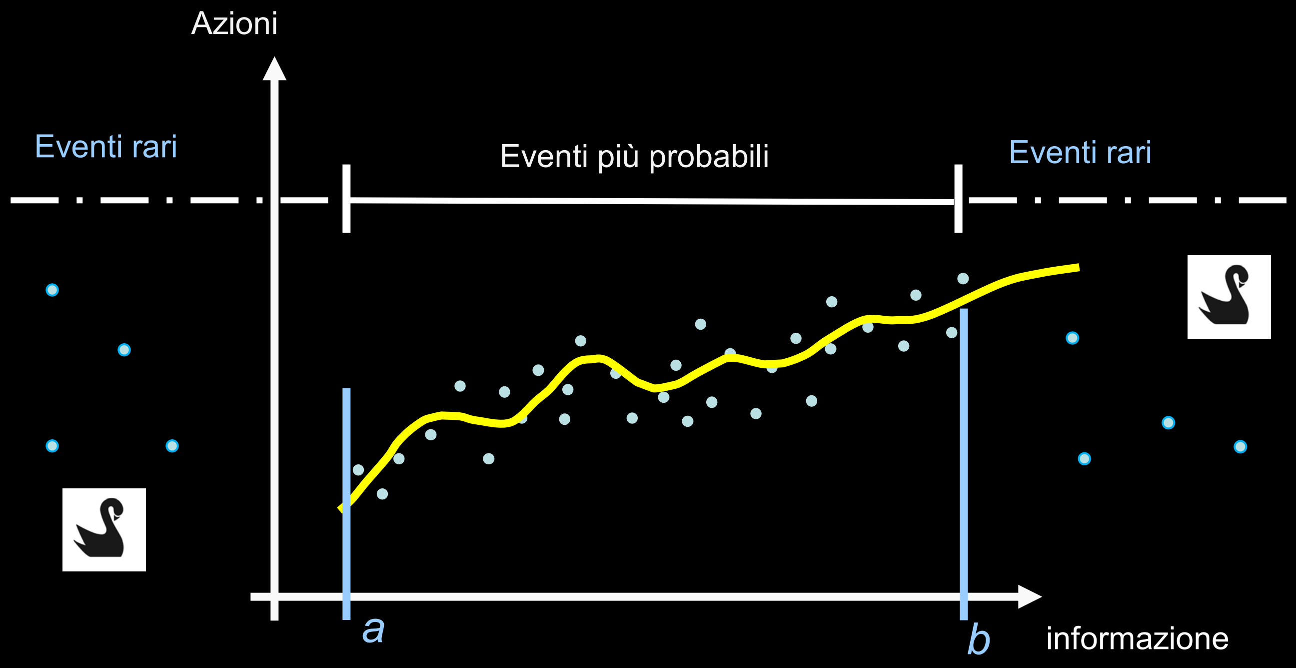


Teorema di approssimazione universale



Teorema di approssimazione universale

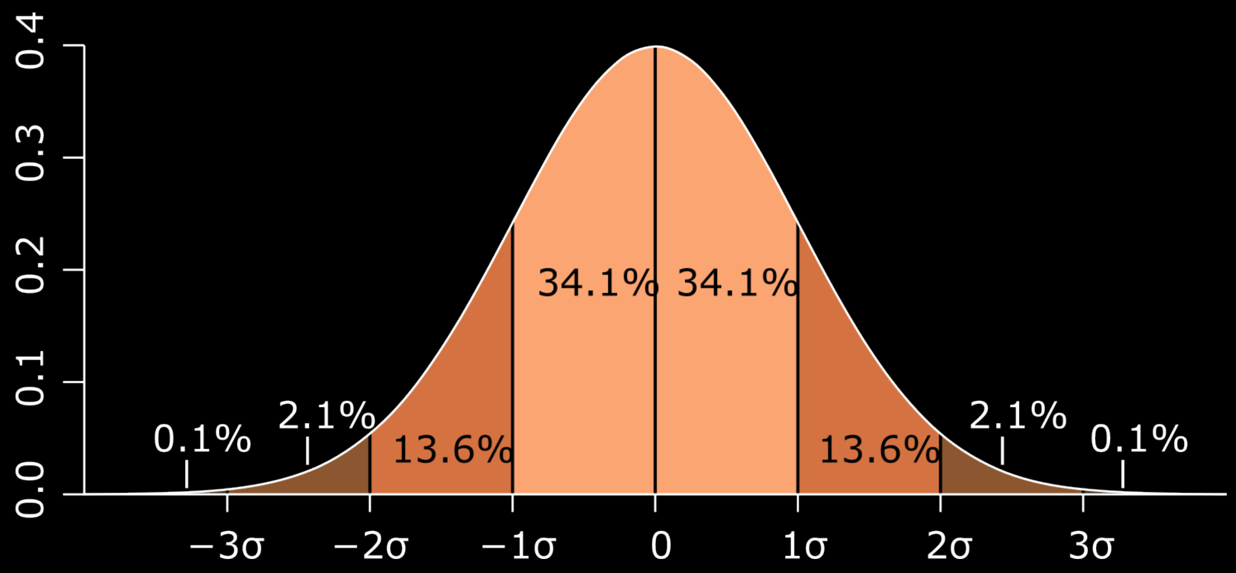




Azioni
Eventi rari
Eventi più probabili
Eventi rari
a
b
informazione

Analisi delle code delle distribuzioni

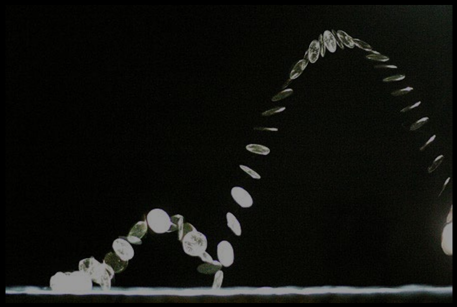
Il cigno nero è un simbolo che rappresenta un insieme di eventi improbabili che fanno parte della distribuzione di probabilità ma che influiscono fortemente sul sistema

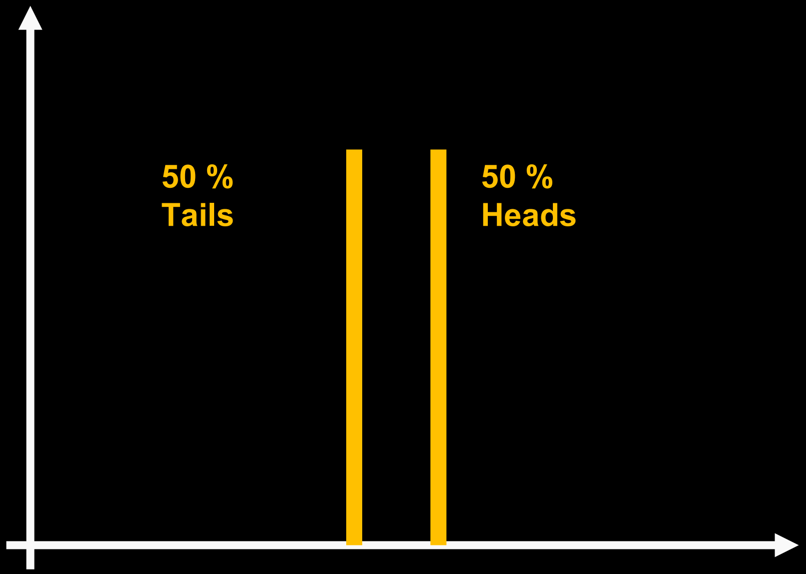


Analisi delle code delle distribuzioni

Esempi:

- Moneta

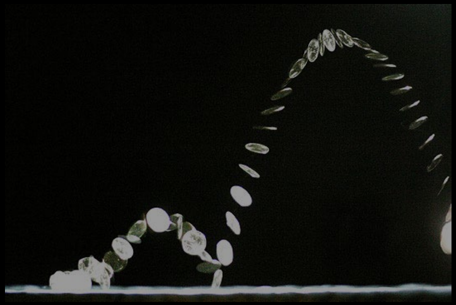


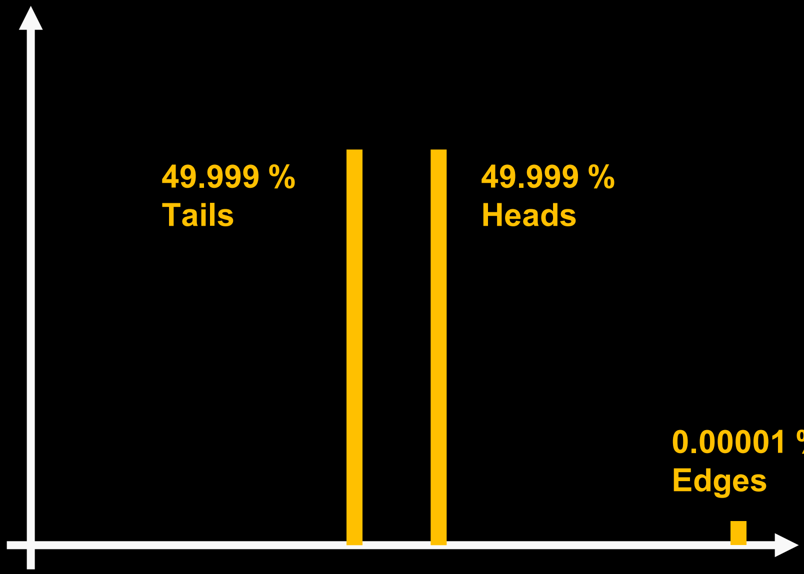


Analisi delle code delle distribuzioni

Esempi:

- Moneta

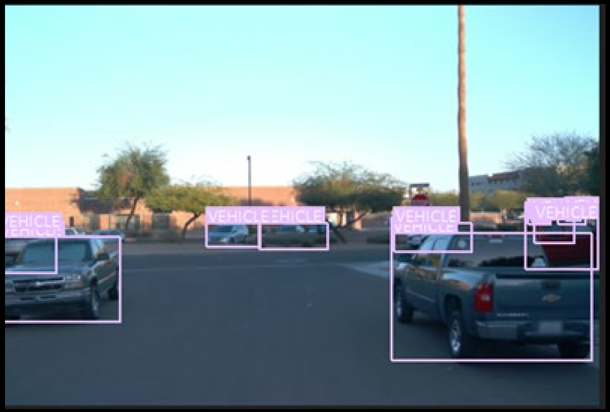




Analisi delle code delle distribuzioni

Esempi:

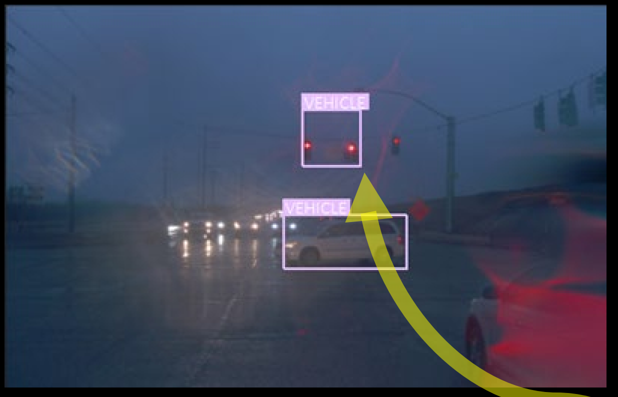
- Moneta
- Guida autonoma



Analisi delle code delle distribuzioni

Esempi:

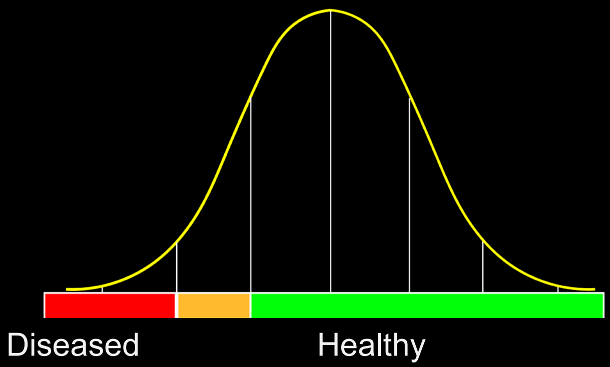
- Moneta
- Guida autonoma



Analisi delle code delle distribuzioni

Esempi:

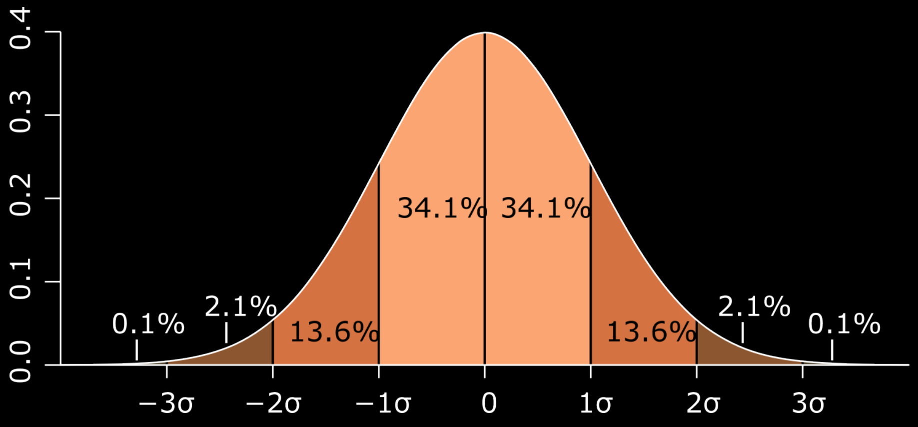
- Moneta
- Guida autonoma
- Sanità



Analisi delle code delle distribuzioni

Motivazione:

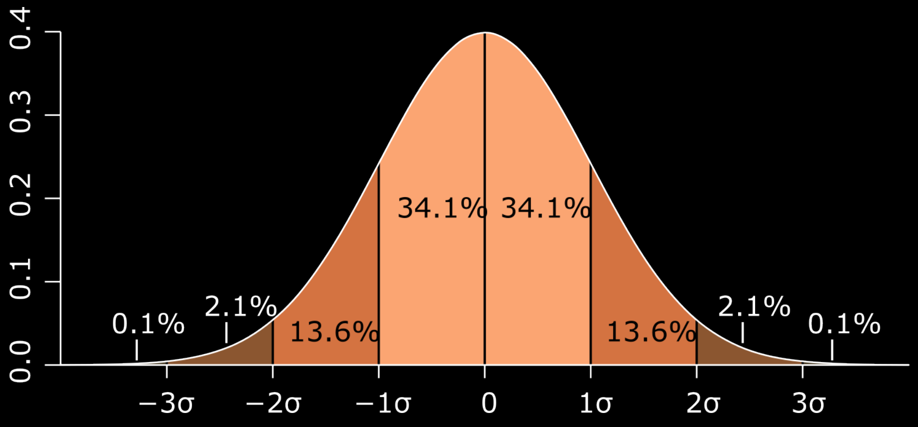
- Decrementare la probabilità di fallimento
- Incrementare l'efficacia dei sistemi basati su IA
- Costruire componenti critici (sicurezza)



Analisi delle code delle distribuzioni

Cosa possiamo fare?

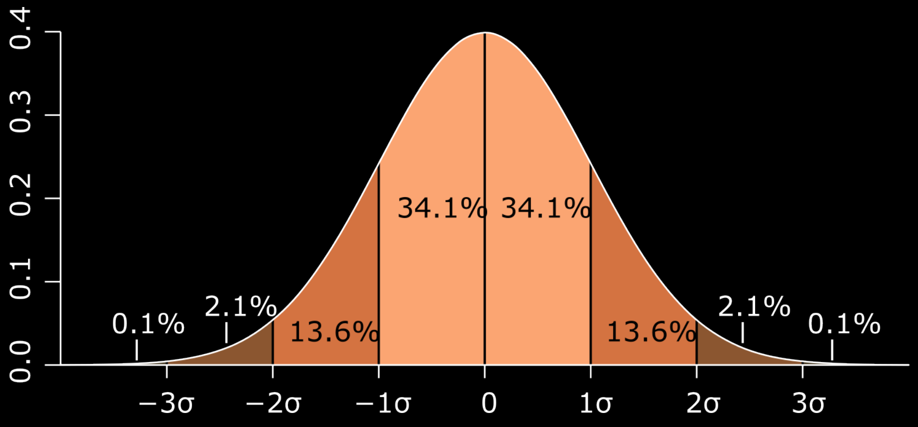
- Acquisire più informazione



Analisi delle code delle distribuzioni

Cosa possiamo fare?

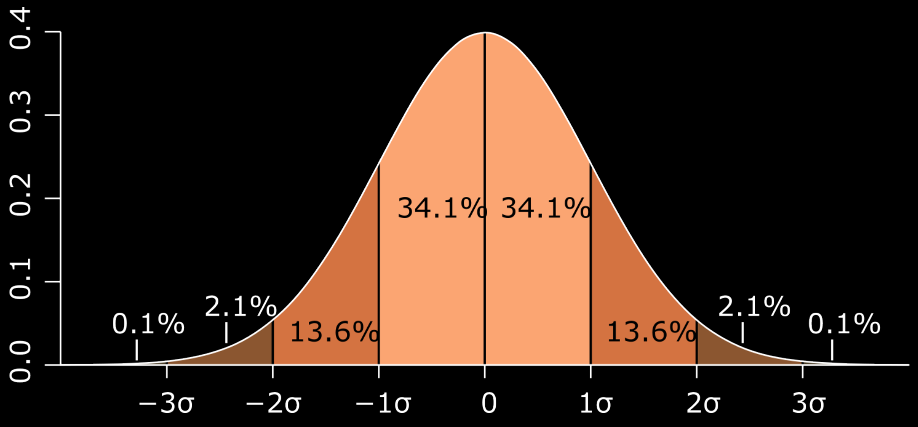
- Acquisire più informazione
- Adversarial learning



Analisi delle code delle distribuzioni

Cosa possiamo fare?

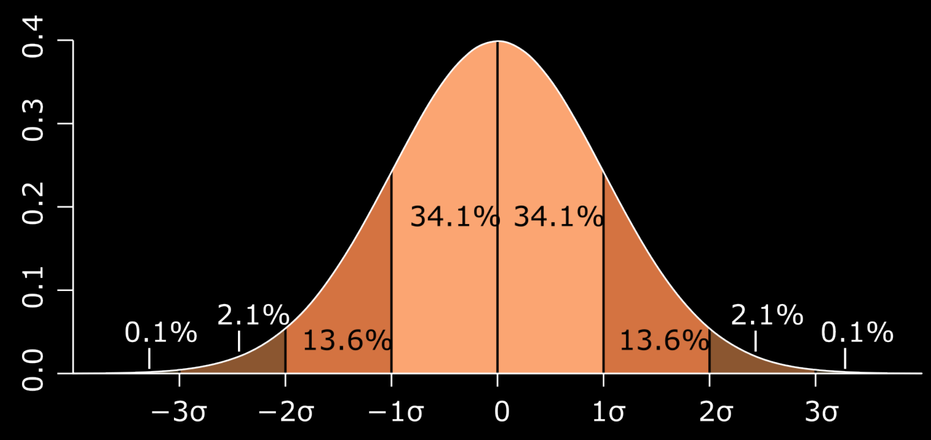
- Acquisire più informazione
- Adversarial learning
- Explainable AI



Analisi delle code delle distribuzioni

Cosa possiamo fare?:

- Acquisire più informazione
- Adversarial learning
- Explainable AI
- Connessioni simboliche



AI level 1

Speech recognition

Image recognition

Mechanical tasks

dog

AI level 1

Image recognition

Speech recognition

Mechanical tasks

dog $= f($  $)$

AI level 1

Speech recognition

Image recognition

Mechanical tasks

word Sound

AI level 1

Image recognition

Speech recognition

Mechanical tasks

$$\text{word} = f(\text{Sound})$$

AI level 1

Speech recognition

Image recognition

Mechanical tasks

Questa relazione non è ovvia
quando si parla di abilità cognitive ed
astrazione di conoscenza

AI level 1

Speech recognition

Image recognition

Mechanical tasks

Problem solving

Mathematical thinking

Comprehension

Active coversation

AI level 2

AI level 1

Speech recognition

Image recognition

Mechanical tasks

“Trying to understand perception by studying only neurons is like trying to understand bird flight by studying only feathers: it just cannot be done.

In order to understand bird flight, we have to understand aerodynamics; only then do the structure of feathers and the different shapes of birds‘ wings make sense”

David Marr,
Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.
(1982) H. Freeman and Co.