
Introduzione al Deep Learning

Massimiliano Ghiotto

Dottorando presso Dipartimento di Matematica,
Università di Pavia

In collaborazione con:

Raffaella Guglielmann¹



Table of Contents

- ▶ Introduzione
- ▶ Percettrone
- ▶ Deep Learning
- ▶ Applicazioni interessanti



UNIVERSITÀ DI PAVIA

Il Deep Learning é una la scienza che porta i computers a risolvere problemi senza programmarli esplicitamente. Il processo di apprendimento é ricavato dai data con cui si allena il modello.

Il Deep Learning é una la scienza che porta i computers a risolvere problemi senza programmarli esplicitamente. Il processo di apprendimento é ricavato dai data con cui si allena il modello.

- I primi modelli sviluppati sono chiamati percetroni e sono stati sviluppati nel 1957,
- il primo modello di Deep Learning é stato sviluppato nel 1989 (multi-percettrone),
- Deep Blue ha sconfitto Kasparov nel 1997, AlphaGo ha sconfitto Lee Sedol nel 2016.

Documentario interessante

1 Introduzione



Quando usare il Deep Learning?

1 Introduzione

Quando usarlo:

- Grandi quantità di dati,
- problema molto complesso,
- per velocizzare alcuni algoritmi.

Quando usare il Deep Learning?

1 Introduzione

Quando usarlo:

- Grandi quantità di dati,
- problema molto complesso,
- per velocizzare alcuni algoritmi.

Quando non usarlo:

- Piccole quantità di dati,
- problemi semplici,
- quando si cercano spiegazioni,
- quando si possono commettere errori.

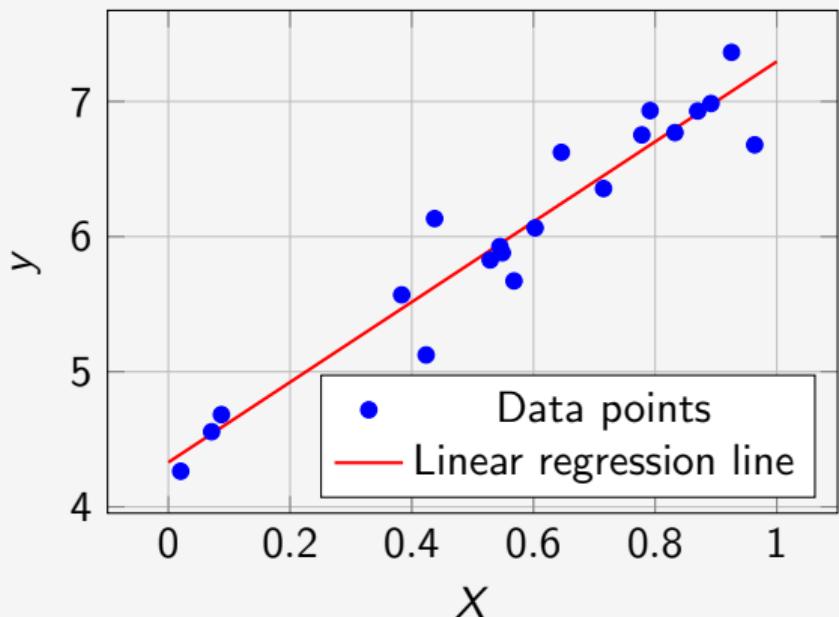
Table of Contents

- ▶ Introduzione
- ▶ Percettrone
- ▶ Deep Learning
- ▶ Applicazioni interessanti

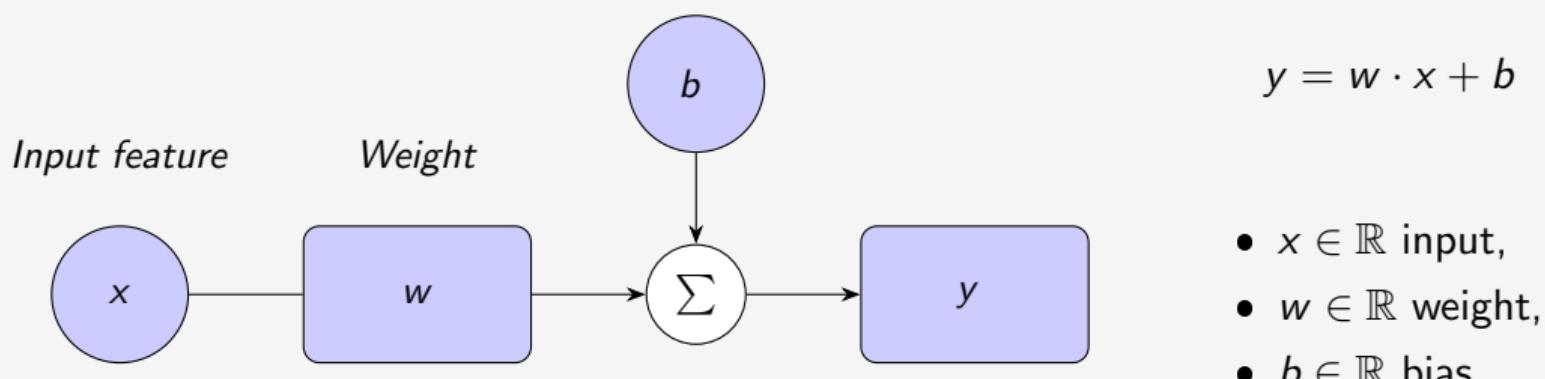


UNIVERSITÀ DI PAVIA

Linear Regression Example

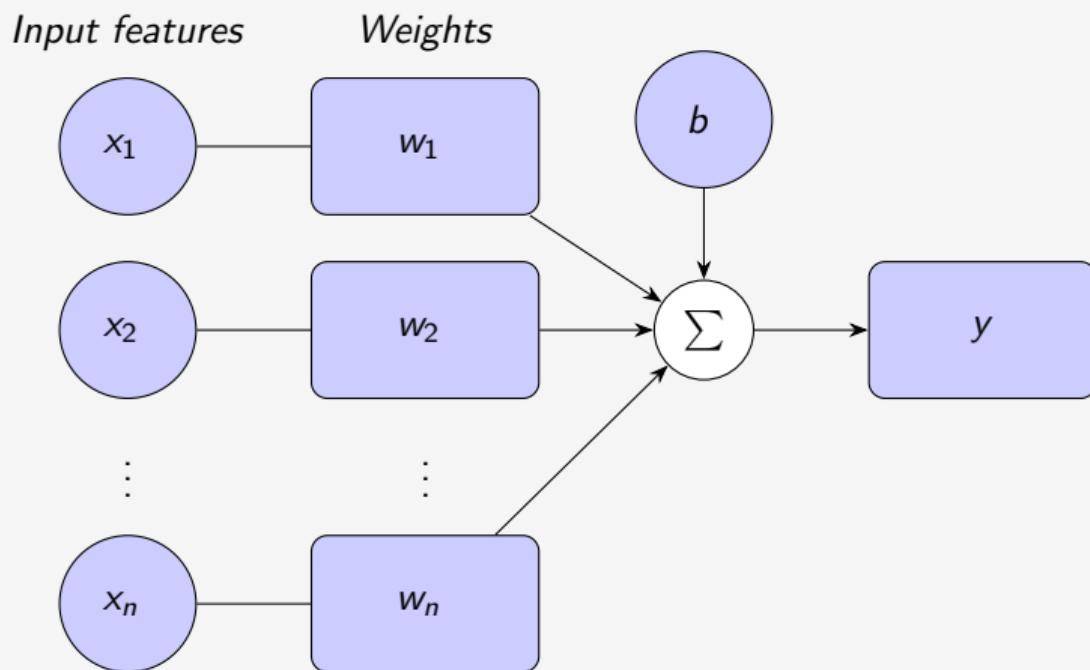


Dove $\{x_i\}_i \subset \mathbb{R}$ sono i dati di input e $\{y_i\}_i \subset \mathbb{R}$ sono i dati di output. Noi vogliamo trovare la retta $y = w \cdot x + b$ che meglio approssima i dati.



Regressione lineare

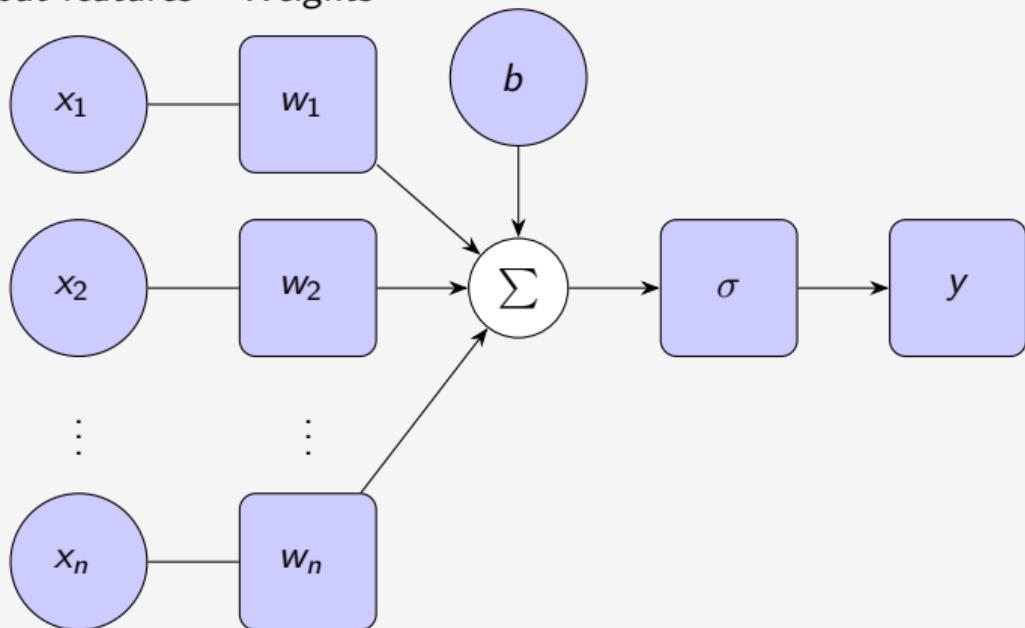
2 Percettrone



$$y = \mathbf{w} \cdot \mathbf{x} + b$$

- $\mathbf{x} \in \mathbb{R}^n$ inputs,
- $\mathbf{w} \in \mathbb{R}^n$ weights,
- $b \in \mathbb{R}$ bias.

Input features Weights



Consideriamo il problema
di classificazione.

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b),$$

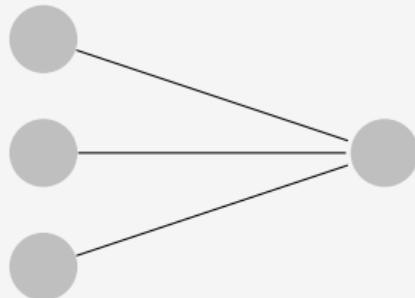
con $\sigma(z) = 1$ se $z \geq 0$ e
nulla altrove.

Chiamiamo $\hat{y}(\mathbf{x})$ la label corretta associata ad \mathbf{x} , l'algoritmo per trovare \mathbf{w} e b è:

- calcolare $y(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$,
- se $y(\mathbf{x}) = \hat{y}(\mathbf{x})$ no aggiornamento,
- se $y(\mathbf{x}) = 1$ e $\hat{y}(\mathbf{x}) = 0$ allora aggiorniamo $\mathbf{w} = \mathbf{w} + \eta \mathbf{x}$ e $b = b + \eta$.
- se $y(\mathbf{x}) = 0$ e $\hat{y}(\mathbf{x}) = 1$ allora aggiorniamo $\mathbf{w} = \mathbf{w} - \eta \mathbf{x}$ e $b = b - \eta$.

Chiamiamo $\hat{y}(\mathbf{x})$ la label corretta associata ad \mathbf{x} , l'algoritmo per trovare \mathbf{w} e b è:

- calcolare $y(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$,
- se $y(\mathbf{x}) = \hat{y}(\mathbf{x})$ no aggiornamento,
- se $y(\mathbf{x}) = 1$ e $\hat{y}(\mathbf{x}) = 0$ allora aggiorniamo $\mathbf{w} = \mathbf{w} + \eta \mathbf{x}$ e $b = b + \eta$.
- se $y(\mathbf{x}) = 0$ e $\hat{y}(\mathbf{x}) = 1$ allora aggiorniamo $\mathbf{w} = \mathbf{w} - \eta \mathbf{x}$ e $b = b - \eta$.



Ogni cerchio rappresenta un neurone, a cui è associato un bias. Ogni freccia rappresenta una connessione tra due neuroni a cui è associato un peso.

Notiamo che se $x_0, x_1 \in \{0, 1\}$ allora con il percettrone si puo' rappresentare le operazioni logiche di negazione, congiunzione e disgiunzione.

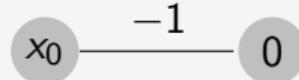


Figure: Negazione

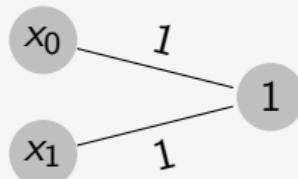


Figure: Congiunzione

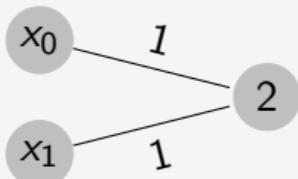


Figure: Disgiunzione

Notiamo che se $x_0, x_1 \in \{0, 1\}$ allora con il percettrone si puo' rappresentare le operazioni logiche di negazione, congiunzione e disgiunzione.

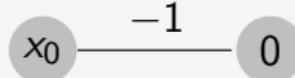


Figure: Negazione

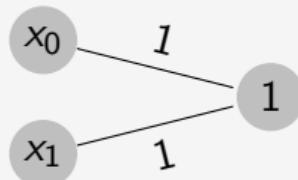


Figure: Congiunzione

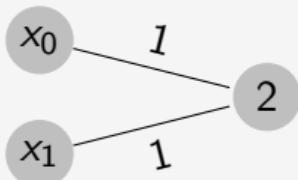


Figure: Disgiunzione

Ma non si puo' rappresentare la disgiunzione esclusiva (XOR).

Table of Contents

- ▶ Introduzione
- ▶ Percettrone
- ▶ Deep Learning
- ▶ Applicazioni interessanti



UNIVERSITÀ DI PAVIA

Multi-layer Perceptron

3 Deep Learning

Indicando lo XOR con \times vale che

$$x_0 \times x_1 = (x_0 \vee x_1) \wedge \neg(x_0 \wedge x_1) = (x_0 \vee x_1) \wedge (\neg x_0 \vee \neg x_1).$$

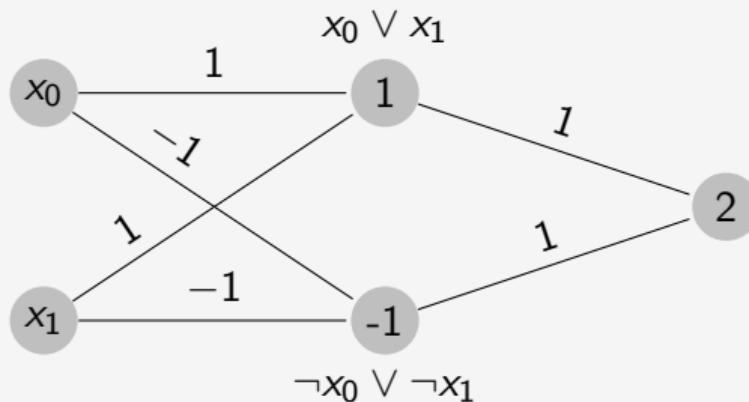
Multi-layer Perceptron

3 Deep Learning

Indicando lo XOR con \times vale che

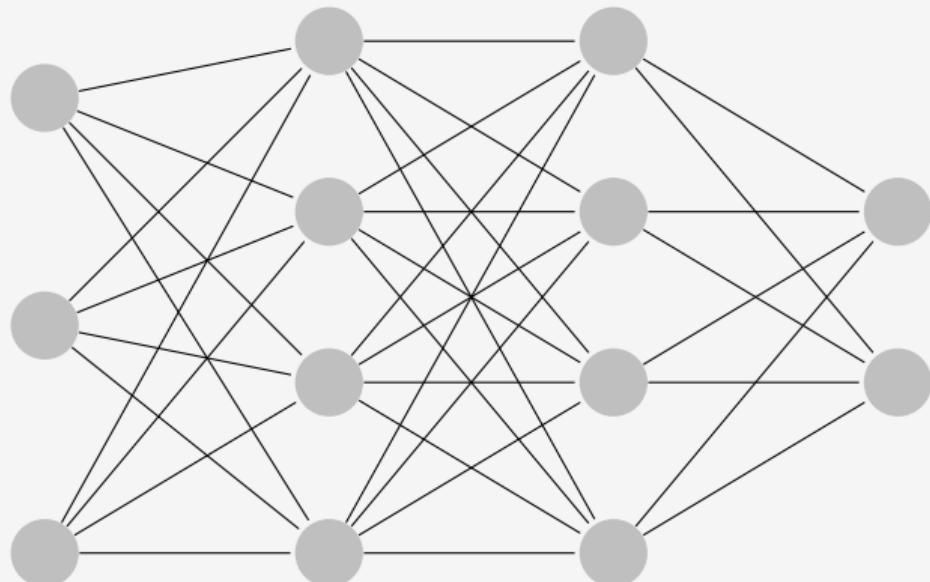
$$x_0 \times x_1 = (x_0 \vee x_1) \wedge \neg(x_0 \wedge x_1) = (x_0 \vee x_1) \wedge (\neg x_0 \vee \neg x_1).$$

Dunque lo possiamo rappresentare con



Multi-layer Perceptron

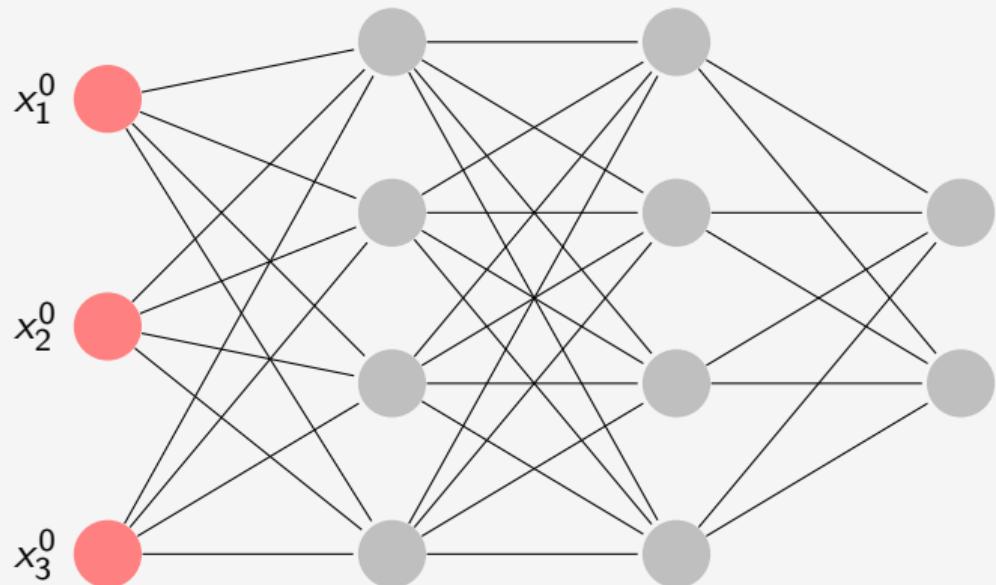
3 Deep Learning



Esempio di architettura di un Multi-layer Perceptron con 3 neuroni di input, 4 neuroni nel primo hidden layer, 4 neuroni nel secondo hidden layer e 2 neuroni di output.

Multi-layer Perceptron

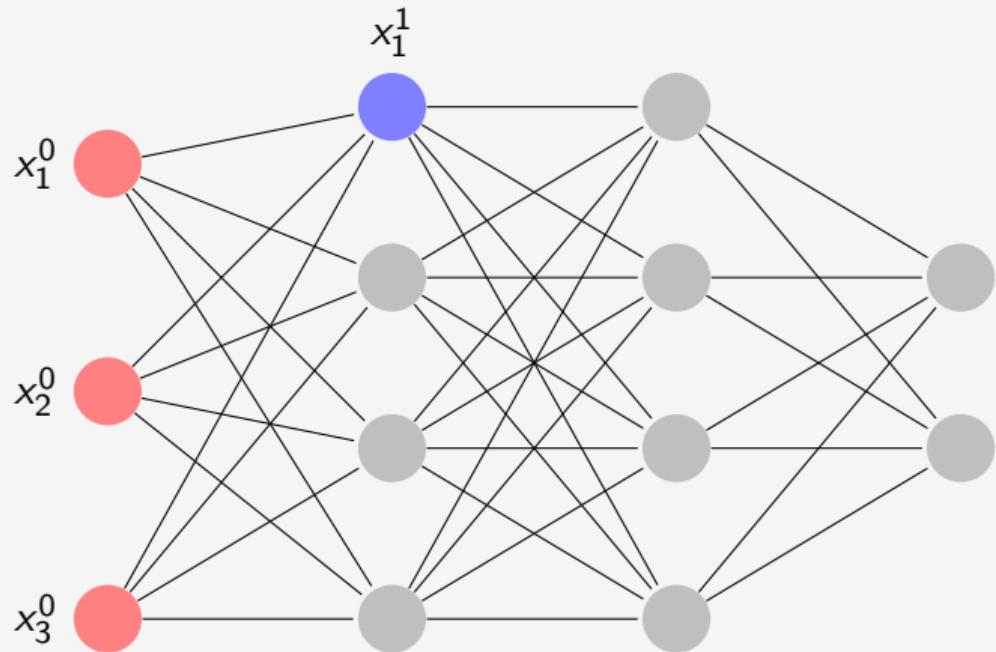
3 Deep Learning



Layer di input con 3 neuroni.

Multi-layer Perceptron

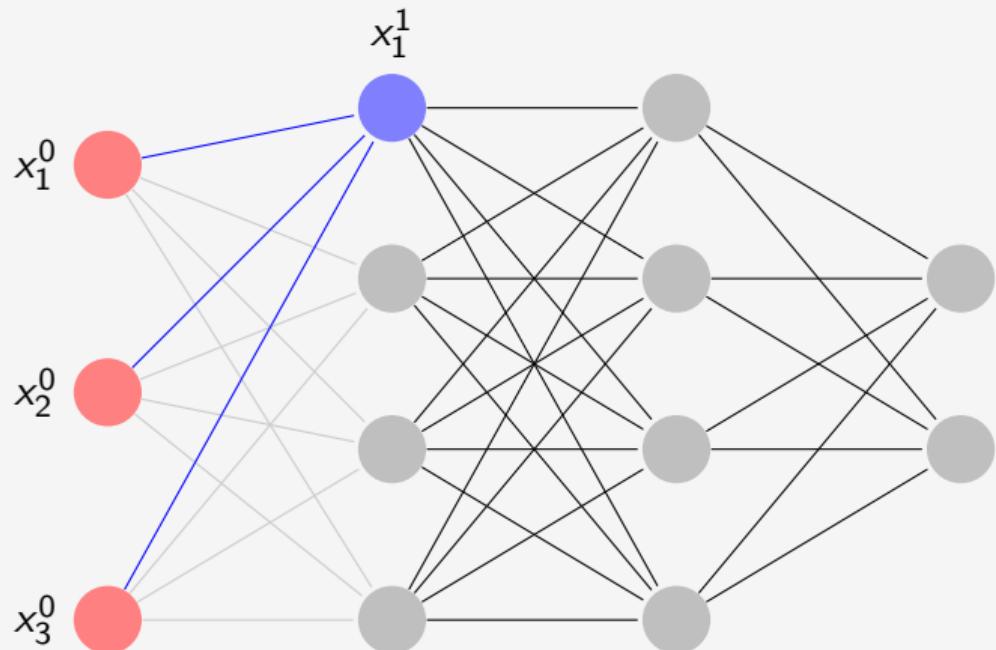
3 Deep Learning



Vogliamo definire il valore del neurone x_1^1 .

Multi-layer Perceptron

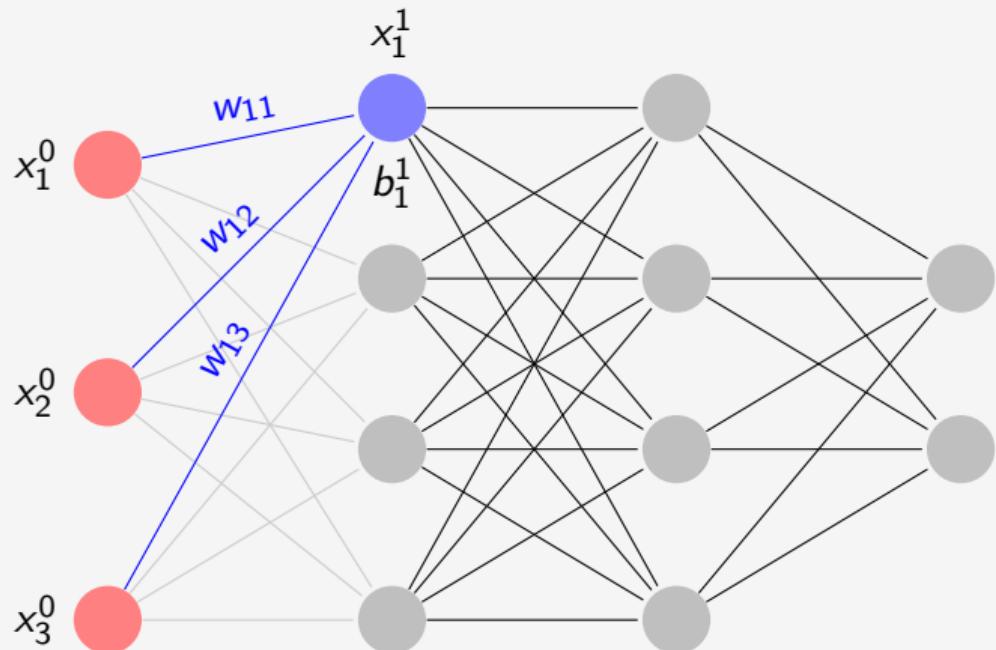
3 Deep Learning



Evidenziamo le connessioni con
i neuroni nel layer precedente.

Multi-layer Perceptron

3 Deep Learning

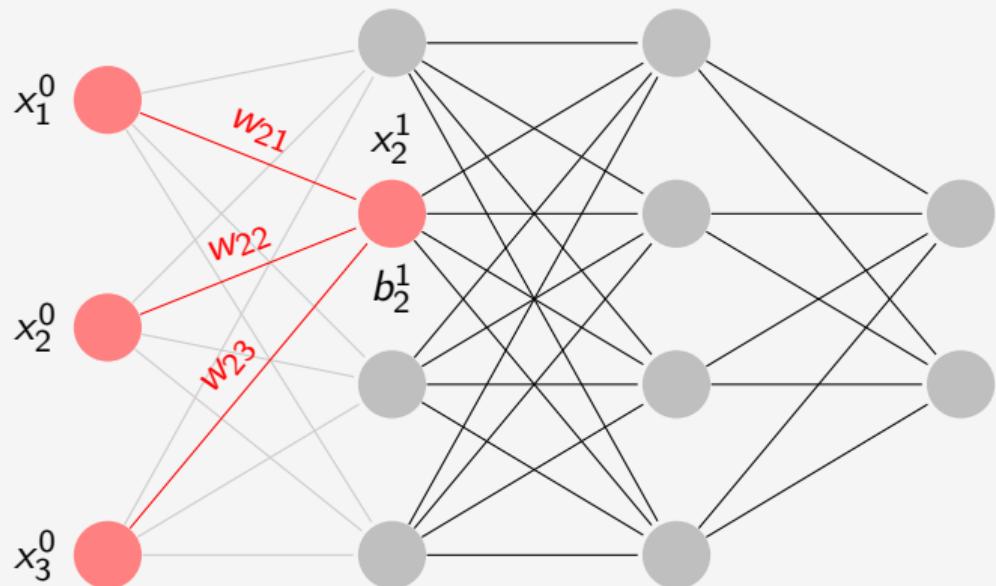


$$x_1^1 = \sigma \left(b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

Dove σ è una funzione di attivazione non lineare.

Multi-layer Perceptron

3 Deep Learning

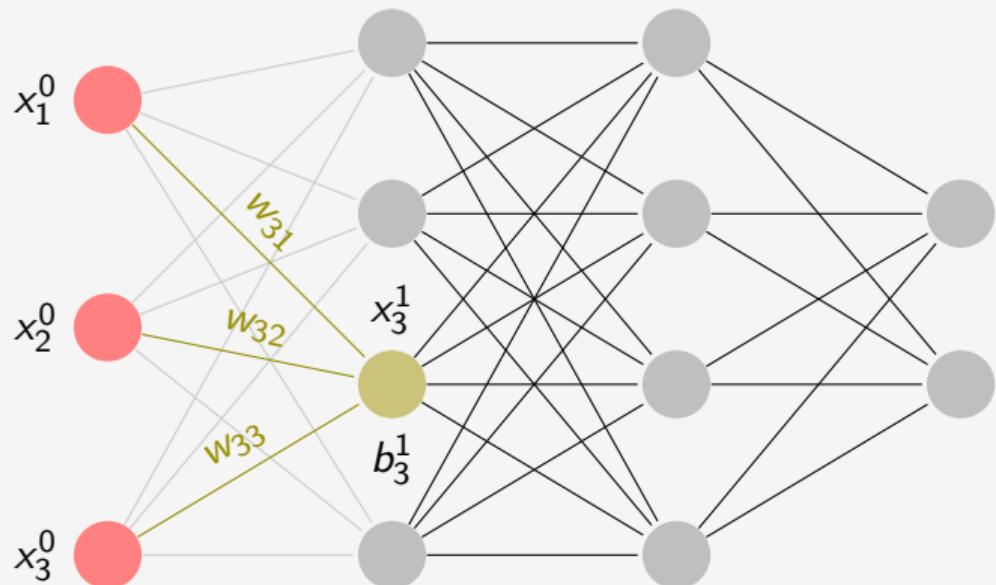


$$x_1^1 = \sigma \left(b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

$$x_2^1 = \sigma \left(b_2^1 + \sum_{i=1}^4 w_{2i} x_i^0 \right)$$

Multi-layer Perceptron

3 Deep Learning



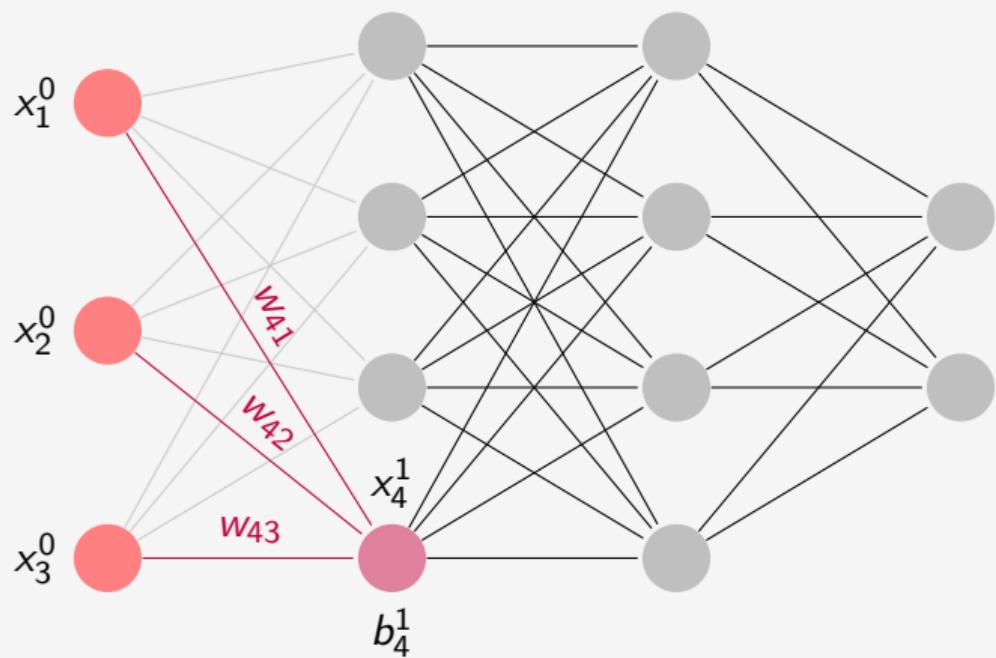
$$x_1^1 = \sigma \left(b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

$$x_2^1 = \sigma \left(b_2^1 + \sum_{i=1}^4 w_{2i} x_i^0 \right)$$

$$x_3^1 = \sigma \left(b_3^1 + \sum_{i=1}^4 w_{3i} x_i^0 \right)$$

Multi-layer Perceptron

3 Deep Learning



$$x_1^1 = \sigma \left(b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

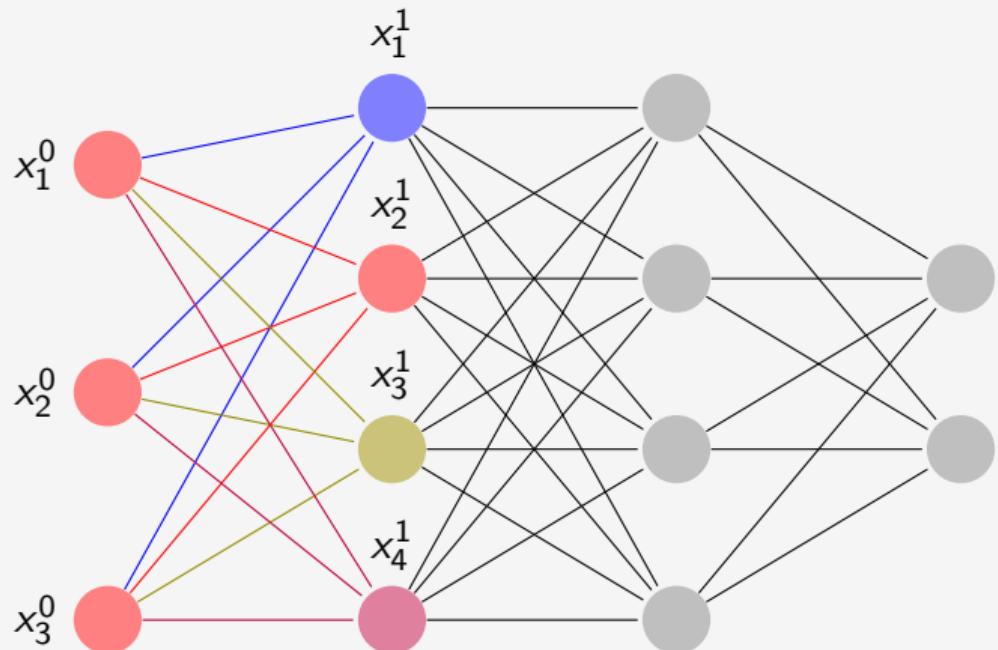
$$x_2^1 = \sigma \left(b_2^1 + \sum_{i=1}^4 w_{2i} x_i^0 \right)$$

$$x_3^1 = \sigma \left(b_3^1 + \sum_{i=1}^4 w_{3i} x_i^0 \right)$$

$$x_4^1 = \sigma \left(b_4^1 + \sum_{i=1}^2 w_{4i} x_i^0 \right)$$

Multi-layer Perceptron

3 Deep Learning



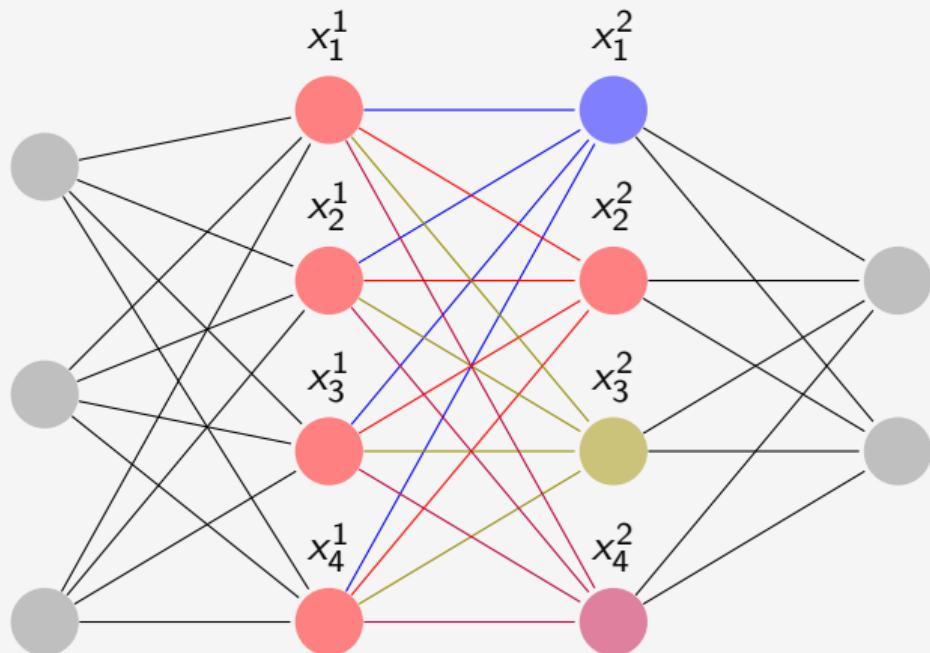
Dato $\mathbf{x}^0 \in \mathbb{R}^3$

$$\mathbf{x}^1 := \sigma(\mathbf{W}_1 \mathbf{x}^0 + \mathbf{b}_1) \in \mathbb{R}^4$$

con $\mathbf{W} \in \mathbb{R}^{4 \times 3}$, $\mathbf{b} \in \mathbb{R}^4$ e σ funzione di attivazione non lineare applicata componente per componente.

Multi-layer Perceptron

3 Deep Learning



Dato $\mathbf{x}^0 \in \mathbb{R}^3$

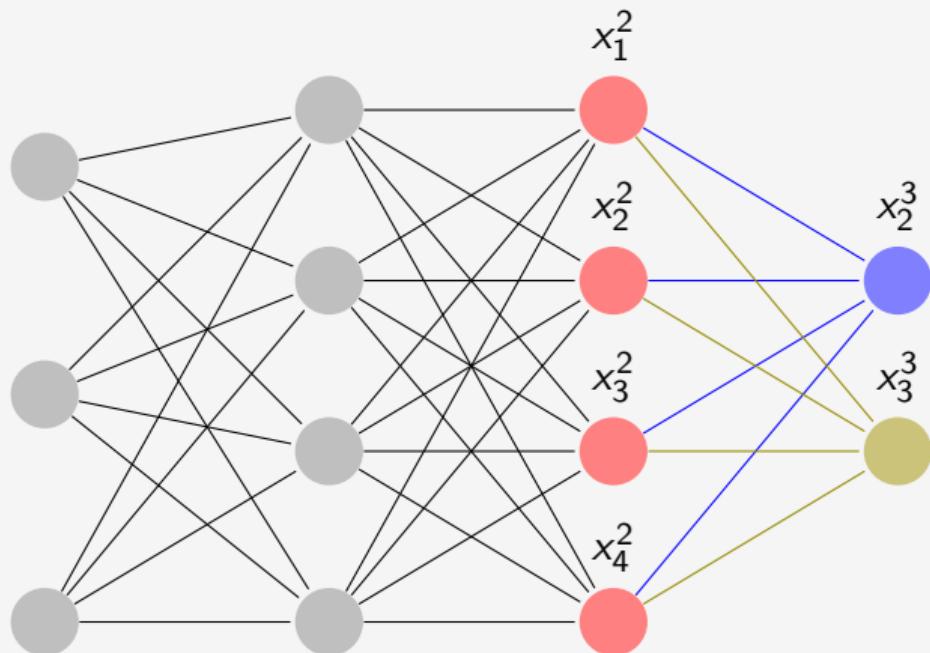
$$\mathbf{x}^1 := \sigma(\mathbf{W}_1 \mathbf{x}^0 + \mathbf{b}_1) \in \mathbb{R}^4$$

$$\mathbf{x}^2 := \sigma(\mathbf{W}_2 \mathbf{x}^1 + \mathbf{b}_2) \in \mathbb{R}^4$$

con $\mathbf{W}_2 \in \mathbb{R}^{4 \times 4}$ e $\mathbf{b} \in \mathbb{R}^4$.

Multi-layer Perceptron

3 Deep Learning



Dato $\mathbf{x}^0 \in \mathbb{R}^3$

$$\mathbf{x}^1 := \sigma(\mathbf{W}_1 \mathbf{x}^0 + \mathbf{b}_1) \in \mathbb{R}^4$$

$$\mathbf{x}^2 := \sigma(\mathbf{W}_2 \mathbf{x}^1 + \mathbf{b}_2) \in \mathbb{R}^4$$

$$\mathbf{x}^3 := \sigma(\mathbf{W}_3 \mathbf{x}^2 + \mathbf{b}_3) \in \mathbb{R}^2$$

con $\mathbf{W}_3 \in \mathbb{R}^{2 \times 4}$ e $\mathbf{b} \in \mathbb{R}^2$.

Definition

Un Multi-layer Perceptron (MLP) con input $\mathbf{x}^0 \in \mathbb{R}^{d_0}$, L hidden layers con d_1, \dots, d_L neuroni e output $\mathbf{x}^L \in \mathbb{R}^{d_L}$ è definito come

$$\mathbf{x}^1 := \sigma(\mathbf{W}_1 \mathbf{x}^0 + \mathbf{b}_1) \in \mathbb{R}^{d_1}$$

⋮

$$\mathbf{x}^L := \sigma(\mathbf{W}_L \mathbf{x}^{L-1} + \mathbf{b}_L) \in \mathbb{R}^{d_L}$$

dove $\mathbf{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ e $\mathbf{b}_\ell \in \mathbb{R}^{d_\ell}$ per $\ell = 1, \dots, L$ e σ è una funzione di attivazione non lineare applicata componente per componente.

Definito un MLP e scelta una funzione di costo $C(\mathbf{x}, \mathbf{y})$ con \mathbf{y} target, si vuole minimizzare C rispetto ai parametri \mathbf{W}_ℓ e \mathbf{b}_ℓ .

Definition

Il metodo del gradiente è un metodo iterativo per minimizzare una funzione $MLP(\mathbf{p})$ definita su uno spazio \mathbb{R}^d . Dato un punto iniziale $\mathbf{p}^{(0)}$, la sequenza $\mathbf{p}^{(t)}$ è definita come

$$\mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} - \eta \nabla f(\mathbf{p}^{(t)})$$

dove η è il learning rate.

Definito un MLP e scelta una funzione di costo $C(\mathbf{x}, \mathbf{y})$ con \mathbf{y} target, si vuole minimizzare C rispetto ai parametri \mathbf{W}_ℓ e \mathbf{b}_ℓ .

Definition

Il metodo del gradiente è un metodo iterativo per minimizzare una funzione $MLP(\mathbf{p})$ definita su uno spazio \mathbb{R}^d . Dato un punto iniziale $\mathbf{p}^{(0)}$, la sequenza $\mathbf{p}^{(t)}$ è definita come

$$\mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} - \eta \nabla f(\mathbf{p}^{(t)})$$

dove η è il learning rate.



Definition

Una funzione continua $f : \mathbb{R} \rightarrow \mathbb{R}$ si dice sigmoidale se

$$\lim_{x \rightarrow -\infty} f(x) = 0 \quad \text{e} \quad \lim_{x \rightarrow +\infty} f(x) = 1.$$

Definition

Una funzione continua $f : \mathbb{R} \rightarrow \mathbb{R}$ si dice sigmoidale se

$$\lim_{x \rightarrow -\infty} f(x) = 0 \quad \text{e} \quad \lim_{x \rightarrow +\infty} f(x) = 1.$$

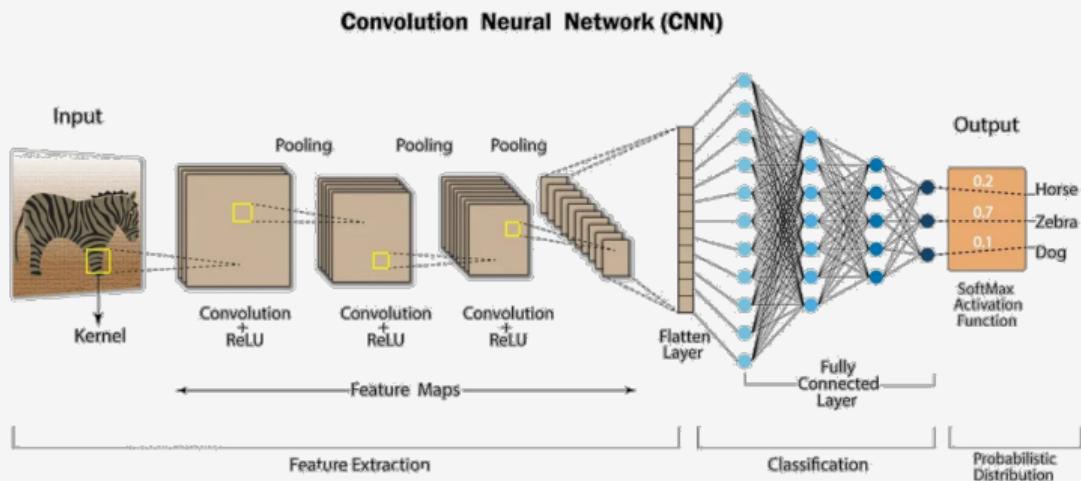
Theorem

Sia $K \subset \mathbb{R}^d$ chiuso e limitato e con $d \in \mathbb{N}$ e $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ funzione sigmoidale allora $MLP(\sigma, d, 2)$ è denso in $C(K)$.

Questo significa che per ogni $f \in C(K)$ e $\varepsilon > 0$ esiste una rete neurale $f_\theta \in MLP(\sigma, d, 2)$ tale che $\|f - f_\theta\|_\infty < \varepsilon$.

Convolutional Neural Network (CNN)

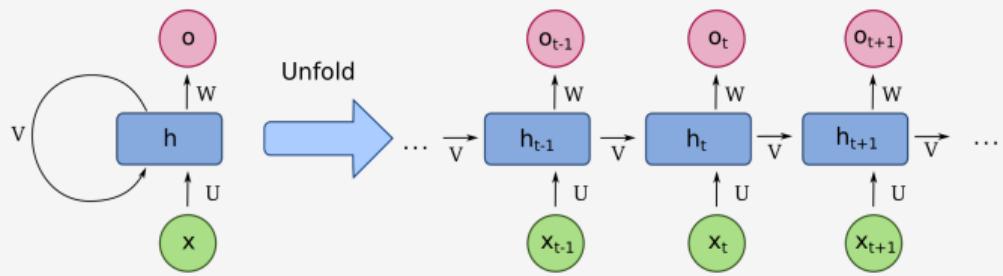
3 Deep Learning



Reti neurali convoluzionali
per l'applicazione alle
immagini.

Recurrent Neural Network (RNN)

3 Deep Learning

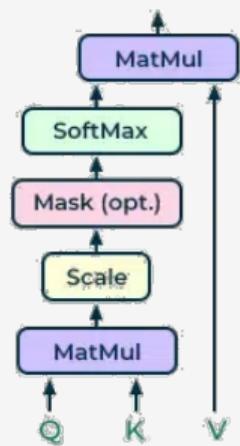


Reti neurali ricorrenti per l'applicazione al linguaggio naturale ed ai segnali, o più in generale alle quantità dipendenti dal tempo.

Transformer

3 Deep Learning

Scaled Dot-Product Attention



Multi-Head Attention

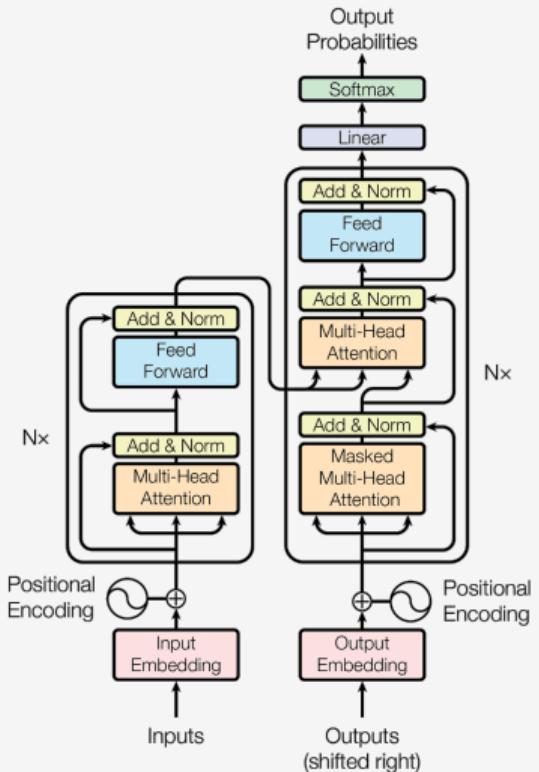
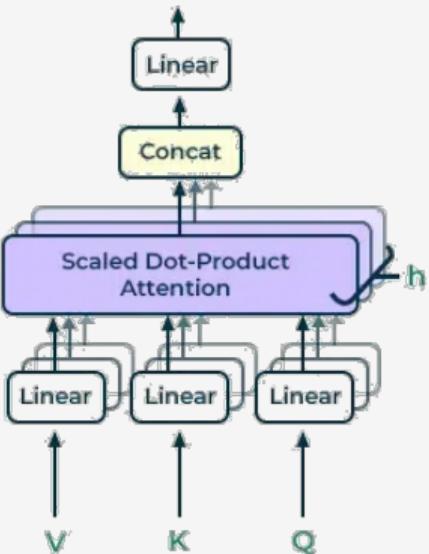


Table of Contents

- ▶ Introduzione
- ▶ Percettrone
- ▶ Deep Learning
- ▶ Applicazioni interessanti

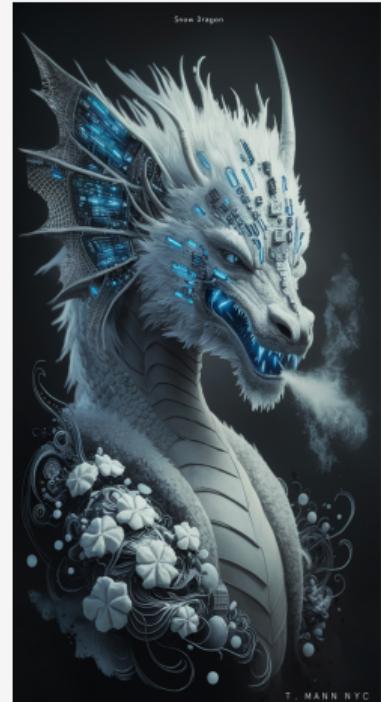


UNIVERSITÀ DI PAVIA

Applicazioni alle immagini

4 Applicazioni interessanti

Siti per generazione d'immagini: [DALL-E3](#)  [DALL-E](#), [Midjourney](#)  e [Ideogram](#) .



Applicazioni ai video

4 Applicazioni interessanti

Massimiliano

Ghiotto



UNIVERSITÀ
DI PAVIA

Deep learning per la generazione di video [SORA](#)  o per modificare video anche in tempo reale [NVIDIA Broadcast](#) .





- Modelli per la generazione di testo come [GPT-4](#) GPT-4 o [Copilot](#) .
- Esistono modelli open-source come [LLaMA-3](#)  scaricabile gratuitamente dal loro sito.
- Potete trovare modelli pre-addestrati e datasets su [Hugging Face](#) oppure su [kaggle](#) .

- Modelli per la generazione di testo come [GPT-4](#)  [GPT-4](#) o [Copilot](#) .
- Esistono modelli open-source come [LLaMA-3](#)   scaricabile gratuitamente dal loro sito.
- Potete trovare modelli pre-addestrati e datasets su [Hugging Face](#)  oppure su [kaggle](#) .

Per allenare Chat-GPT3, formata da 175 miliardi di parametri, hanno impiegato 1023 Nvidia A100 per 34 giorni spendendo 5 milioni di dollari in corrente.

- Modelli per la generazione di testo come [GPT-4](#)  GPT-4 o [Copilot](#) .
- Esistono modelli open-source come [LLaMA-3](#)  scaricabile gratuitamente dal loro sito.
- Potete trovare modelli pre-addestrati e datasets su [Hugging Face](#)  oppure su [kaggle](#) .

Per allenare Chat-GPT3, formata da 175 miliardi di parametri, hanno impiegato 1023 Nvidia A100 per 34 giorni spendendo 5 milioni di dollari in corrente.

Anche per scaricare il modello più piccolo di LLaMA 3 8B sono necessari 20 GB di VRAM e per il modello LLaMA 3 70B sono necessari 140GB di spazio e 160GB di VRAM in FP16.

Deep Learning - Talent Week 18/06/2024

Grazie per l'attenzione!



UNIVERSITÀ DI PAVIA