Introduzione a AI, ML e DL
Regressione

Rimini – 03/11/2021

Alessia Angeli

Studente di dottorato in Data Science and Computation

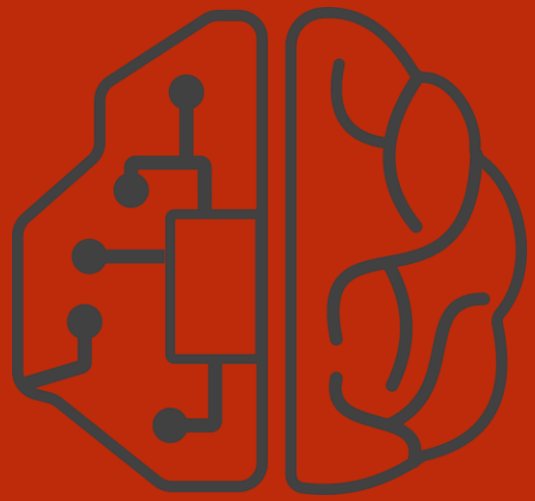
Dipartimento di Informatica – Scienza e Ingegneria

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

VARLAB: VIRTUAL AND AUGMENTED REALITY LAB

Contatti

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



Artificial Intelligence, Machine e Deep Learning

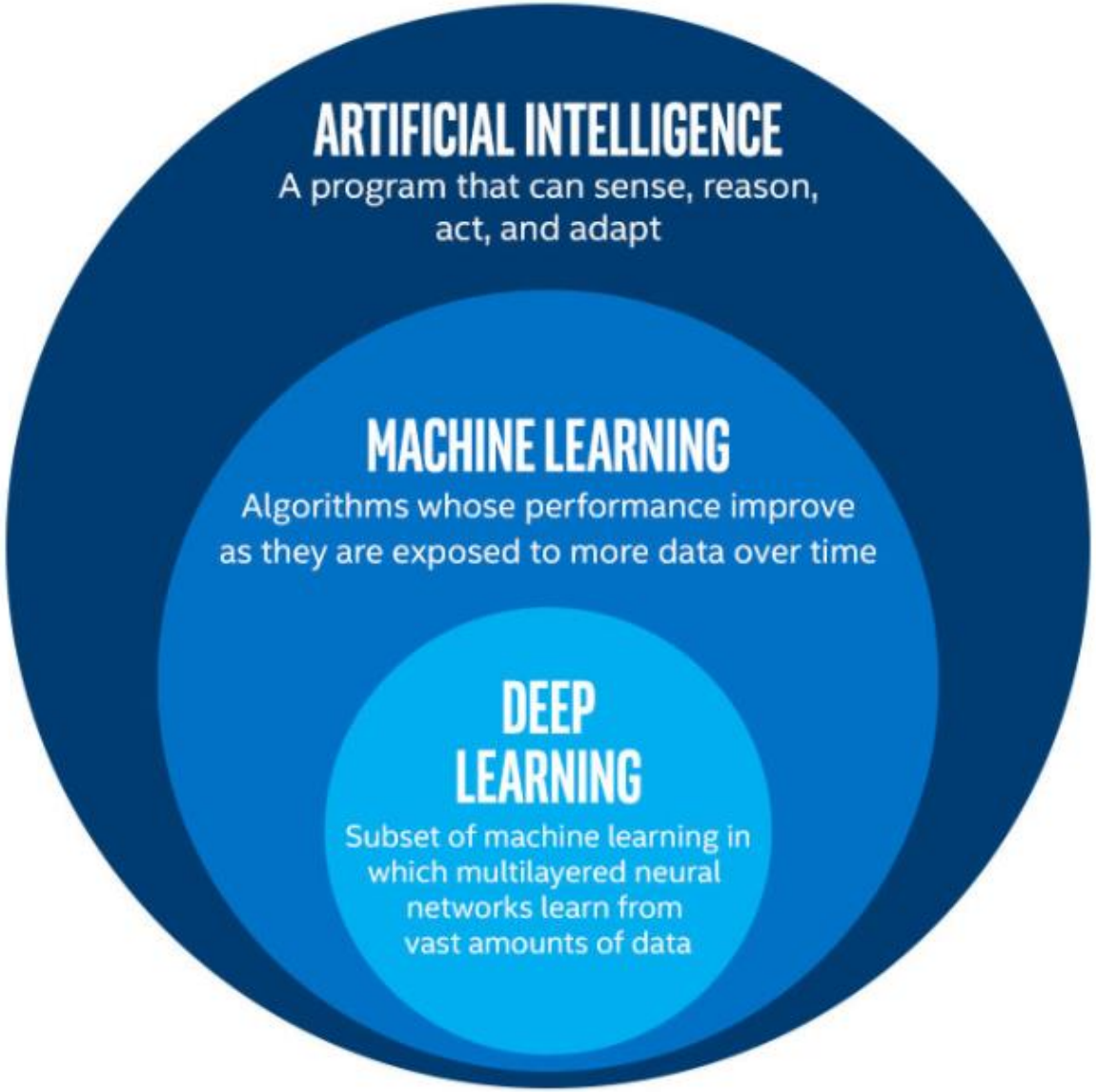
Alessia Angeli

Studente di dottorato in Data Science and Computation

Dipartimento di Informatica – Scienza e Ingegneria

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Artificial Intelligence (AI), Machine Learning (ML) e Deep Learning (DL)



https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55

Machine Learning – Target

Problemi difficili da risolvere con algoritmi «classici»:

- **Problemi di regressione**;

  (e.g., previsione costi, previsione numero unità vendute)

- **Problemi di classificazione**;

  (e.g., email spam/no spam, sentiment analysis, transazioni fraudolenti/corrette)

- **Riconoscimento di oggetti in immagini**;
- **Riconoscimento di linguaggio/musica**;
- **Data Mining**.

  (e.g., individuazione di clusters)

Machine Learning – Caratteristiche

- Bassa conoscenza di informazioni (a priori);
- Dati con elevato numero di attributi (input features);
- Elevato volume di dati per il training;
- Adattabilità.

https://medium.com/@lokeshpara17/tiny-imagenet-using-pytorch-42a3f2ee3c9d

https://www.image-net.org/

Machine Learning – Learning Algorithm

- Definire un **modello** per il problema da risolvere: il modello dipende da un insieme di **parametri**;
- Definire una metrica per misurare le performance: una **misura di errore** per valutare il modello;
- Allenare il modello (aggiornare i parametri), per minimizzare l'errore, sui dati di training.

UN ALGORITMO DI LEARNING E' UN PROCESSO DI OTTIMIZZAZIONE

Machine Learning – Parametri ed iperparametri

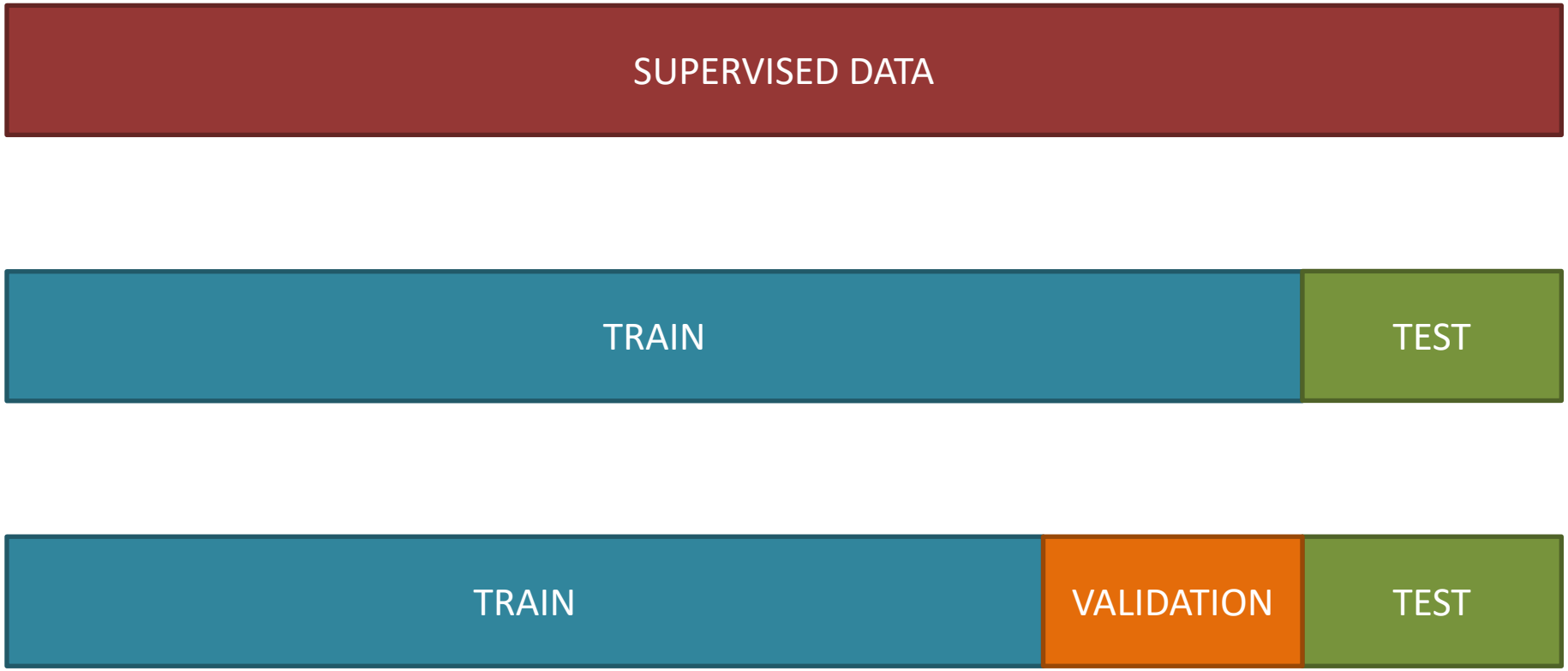
Parametri

Con parametri di un modello di machine e/o deep learning si intendono i parametri il cui valore cambia durante l'allenamento del modello. Il valore dei parametri cambia quindi in base ai dati del training set su «istruzione» dell'algoritmo utilizzato durante l'allenamento per minimizzare la funzione di errore.

Iperparametri

Con iperparametri di un modello di machine e/o deep learning si intendono (se presenti) i parametri il cui valore viene deciso dal programmatore prima dell'allenamento del modello. Il programmatore può poi decidere di utilizzare parte dei dati che ha a disposizione (non appartenenti al training set) per formare un altro insieme di dati (validation set) ed utilizzarlo per ottimizzare la scelta degli iperparametri.

Machine Learning – Training, Validation e Test set



ATTENZIONE ALL'INTERSEZIONE TRA GLI INSIEMI: DEVE ESSERE VUOTA!

Machine Learning – Diverse tipologie di problemi

SUPERVISED LEARNING – Inputs + outputs (labels)

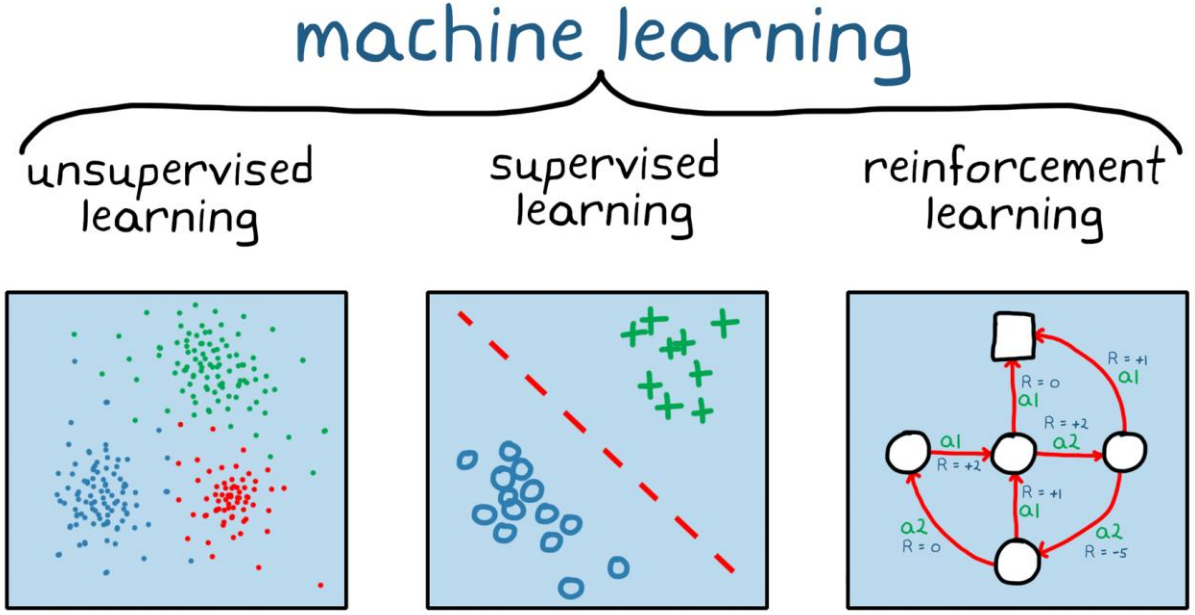
- Regressione;
- Classificazione.

UNSUPERVISED LEARNING – Input (no labels)

- Clustering;
- …

REINFORCEMENT LEARNING – Action and rewards

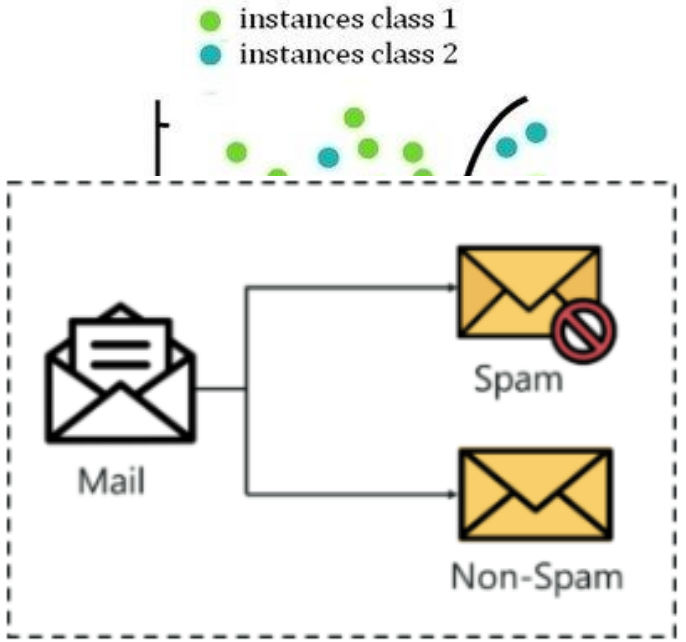
(non ne parleremo in questo corso)



https://it.mathworks.com/discovery/reinforcement-learning.html

Machine Learning – Supervised learning

Regressione VS Classificazione

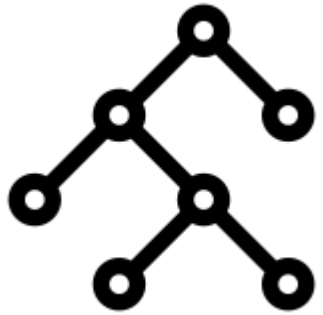


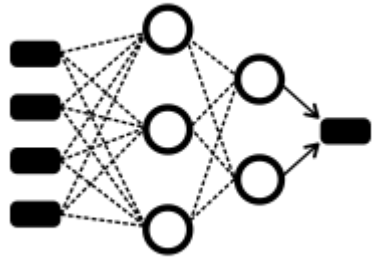
https://www.researchgate.net/publication/350993856_Machine_learning_approaches_to_rank_news_feed_updates_on_social_media/figures?lo=1

Machine Learning – Diverse techiche

Diverse tecniche per definire i **modelli**:

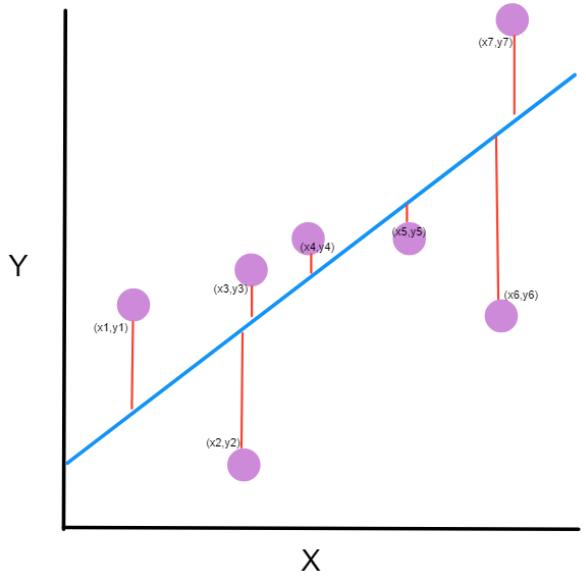
- Modelli lineari;
- Modelli ad albero;
- Reti neurali;
- …

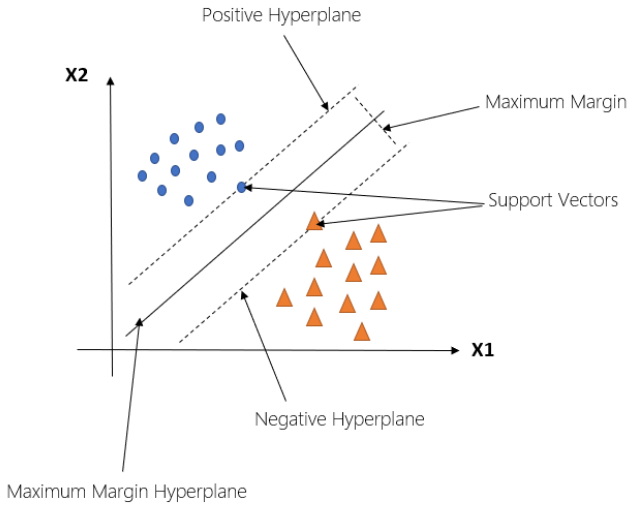




Diverse tecniche per definire **funzioni di errore**:

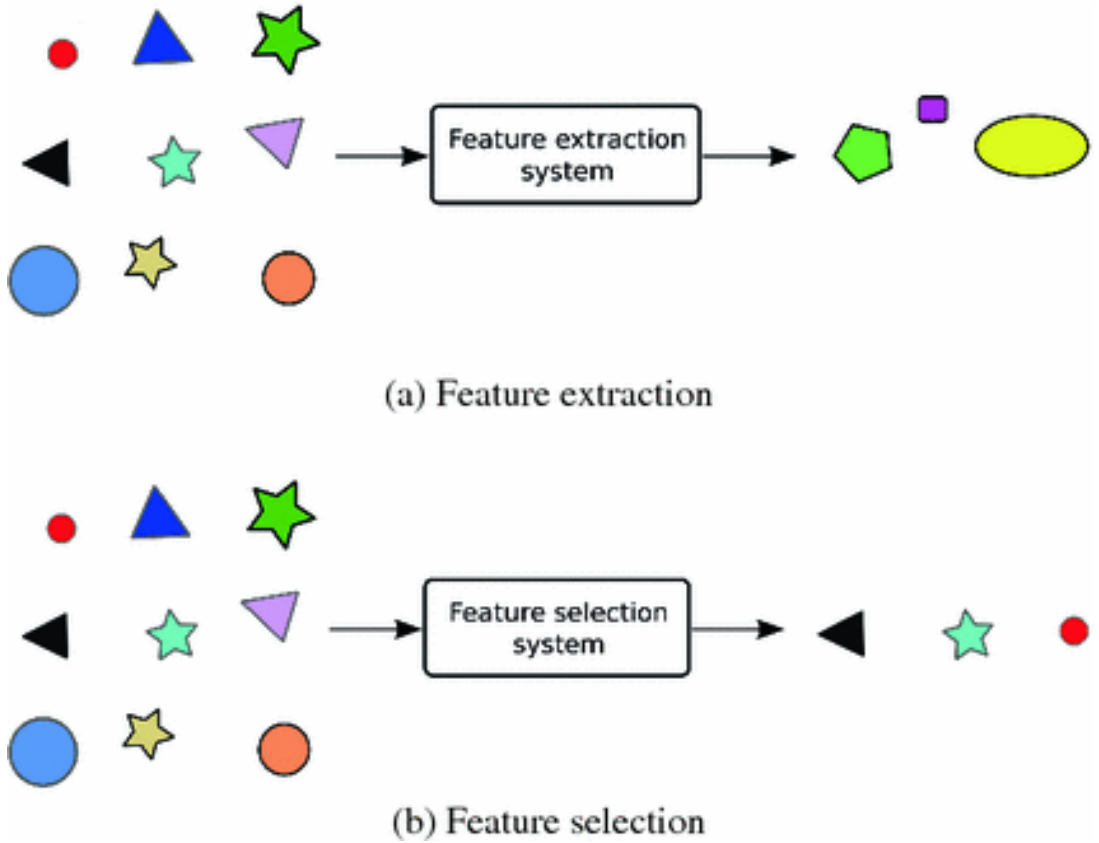
- Minini quadrati;
- Margine massimo;
- Distanza cosenica;
- Funzione logistica;
- …





Machine Learning – Features

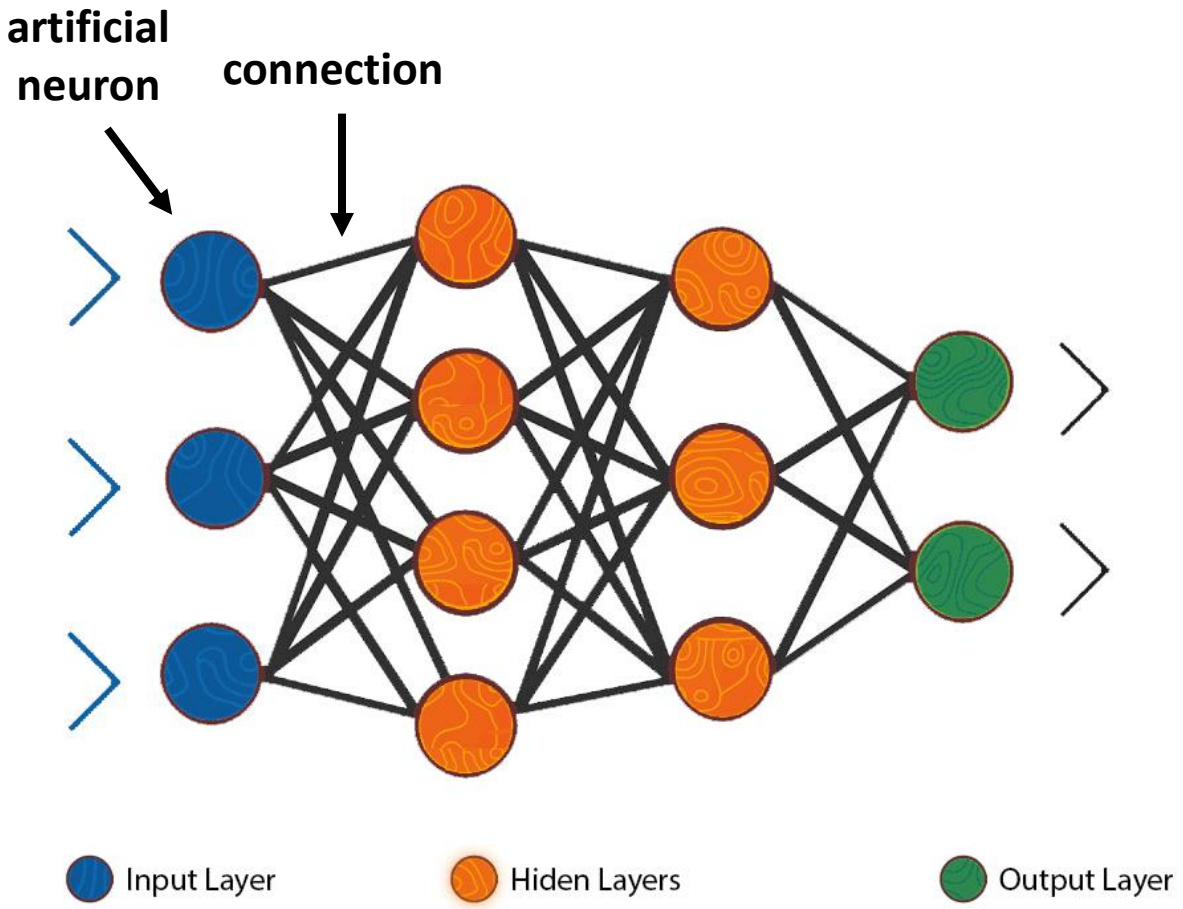
- Una qualsiasi informazione relativa ad un dato viene chiamata **feature**;
- Le features sono gli inputs in un processo/algoritmo di learning;
- Gli algoritmi di learning sono altamente sensibili alla scelta delle features;
- Scegliere delle «buone» features (**feature selection** e/o **feature extraction**) può essere molto difficile.



https://link.springer.com/chapter/10.1007/978-3-319-90080-3_2

Dal Machine al Deep Learning – Rete Neurale

- Una rete neurale è un insieme di neuroni collegati tra loro;
- Tali neuroni sono organizzati in layers: input layer, hidden layer(s), output layer;
- Se c'è 1 unico hidden layer la rete neurale viene detta **shallow**, se ce ne sono multipli (>1), invece, viene detta **deep**;
- Ogni neurone prende multipli inputs in ingresso e produce un singolo output in uscita (che può essere passato come input ad altri neuroni).

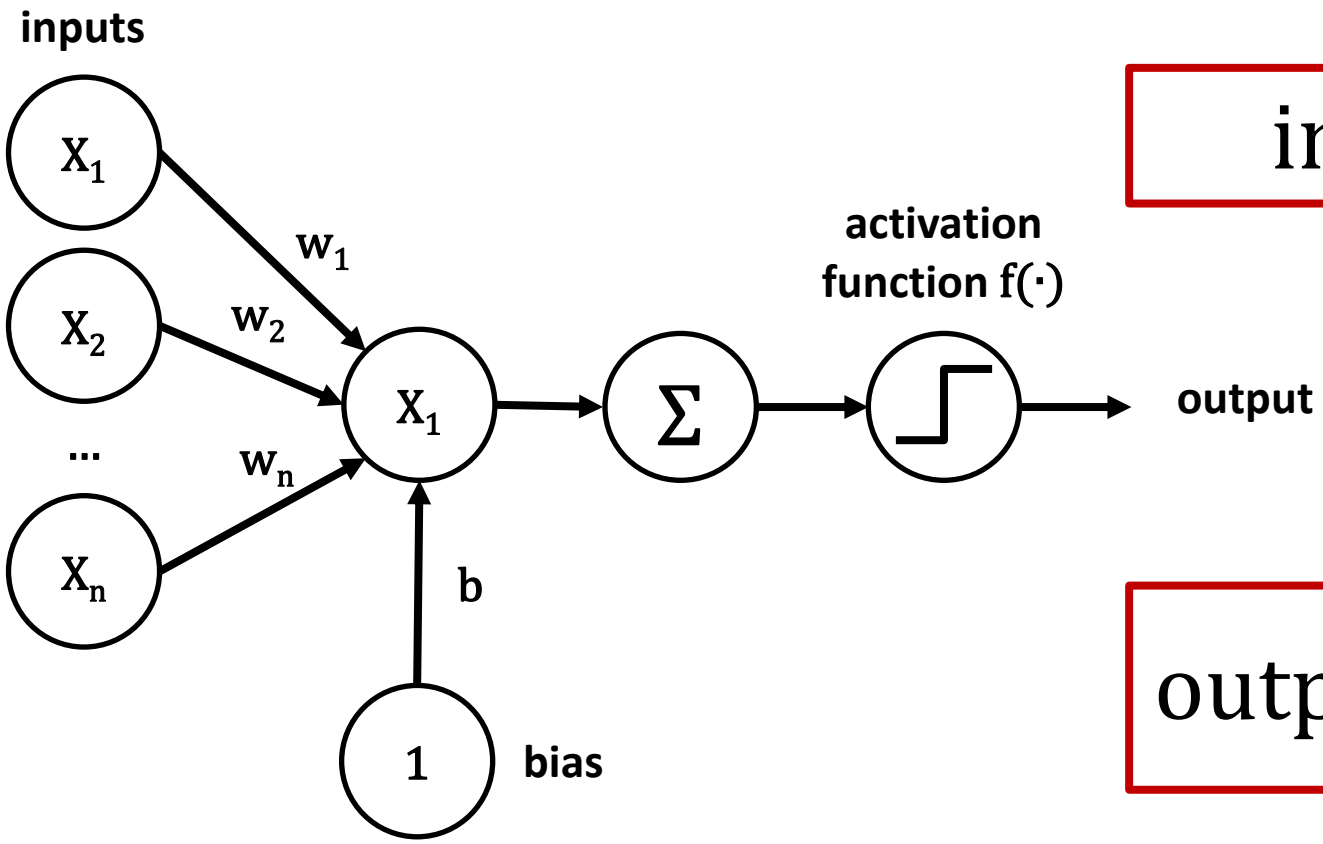


Feed-forward neural network

NOTA: Esistono altri tipi di reti neurali (e.g., CNN, RNN).

Dal Machine al Deep Learning – Rete Neurale

Singolo neurone



$$\text{inputs} = [x_1, x_2, \dots, x_n]$$

$$\text{output} = f\left(\sum_{i=1}^{n}(w_i x_i + b)\right)$$

NOTA: l'obiettivo dell'activation function, funzione non lineare, (ne esistono diverse) è quello di introdurre un meccanismo di thresholding (non vedremo i dettagli in questo corso).

Funzione di errore – Verso il minimo…

ESEMPIO – **Aggiornamento dei pesi $w = [w_1, w_2, \dots, w_n]$ e b**

L'obiettivo è minimizzare la funzione di errore scelta $L(w, b)$ (sul training set), aggiornando i parametri $w = [w_1, w_2, \dots, w_n]$ e b.

Ci sono diversi algoritmi per eseguire questa operazione, però tutti si basano sull'algoritmo di discesa del gradiente **Gradient Descent (GD)**. Per le reti neurali questo viene poi combinato con l'algoritmo di **Back Propagation** (non lo vedremo).

$$\nabla_w L = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \dots, \frac{\partial L}{\partial w_n}\right]$$

$$\nabla_b L = \left[\frac{\partial L}{\partial b}\right]$$

Il gradiente di una funzione è il vettore delle derivate parziali (in questo caso rispetto a w e b) della funzione stessa e punta nella direzione (direzione e verso) di massima salita (in termini di valore) della funzione.

Funzione di errore – Verso il minimo…

ESEMPIO – **Aggiornamento dei pesi $w = [w_1, w_2, \dots, w_n]$ e b**

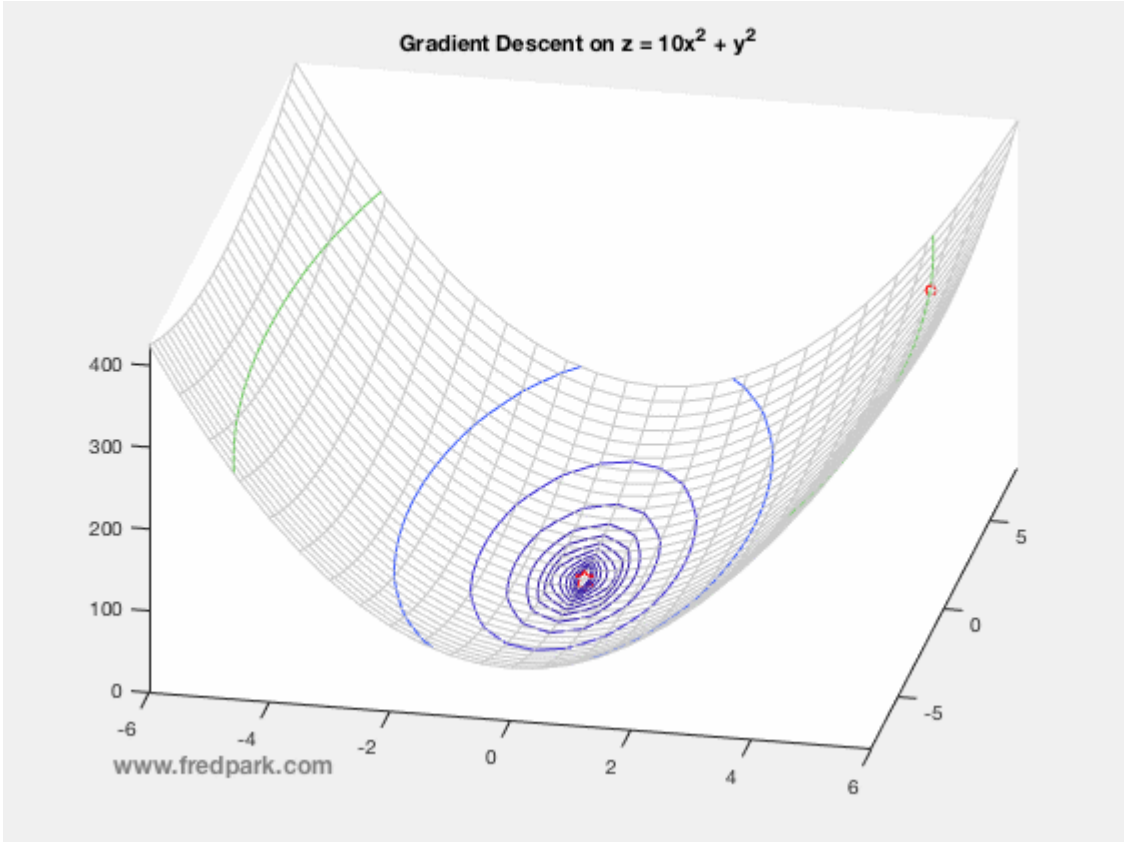
E' possibile raggiungere una configurazione ottimale per i parametri (w e b) procedendo iterativamente, step by step, nella stessa direzione e verso opposto del vettore gradiente:

$$w = w - r\nabla_w L$$

$$b = b - r\nabla_b L$$

dove r è il parametro (iperparametro) di **learning rate**.

NOTA: questo non garantisce il raggiungimento del minimo globale per ogni funzione di errore considerata (e.g., raggiungimento di un minimo locale, di un plateau) (non vedremo i dettagli in questo corso).



Machine Learning Vs Deep Learning

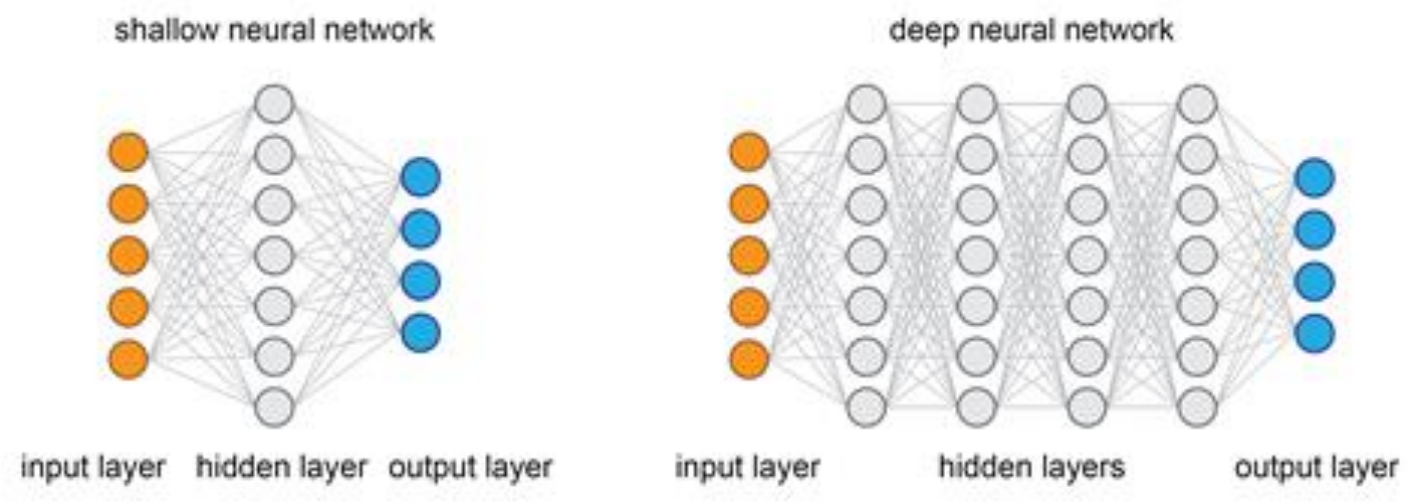
Machine learning approach

- Calcolare delle features efficaci «a mano» e applicare di seguito un algoritmo di learning (e.g., linear regression, random forest).

Deep learning approach

- Fornire i raw data (dati grezzi) e lasciare al modello il compito di calcolare features efficaci. Da tenere presente che operazioni come normalizzazione e/o standardizzazione dati sono spesso comunque da fare.

Una rete neurale viene chiamata **deep** quando ha multipli hidden layers (>1). In particolare, ogni hidden layer calcola nuove features dal layer precedente.



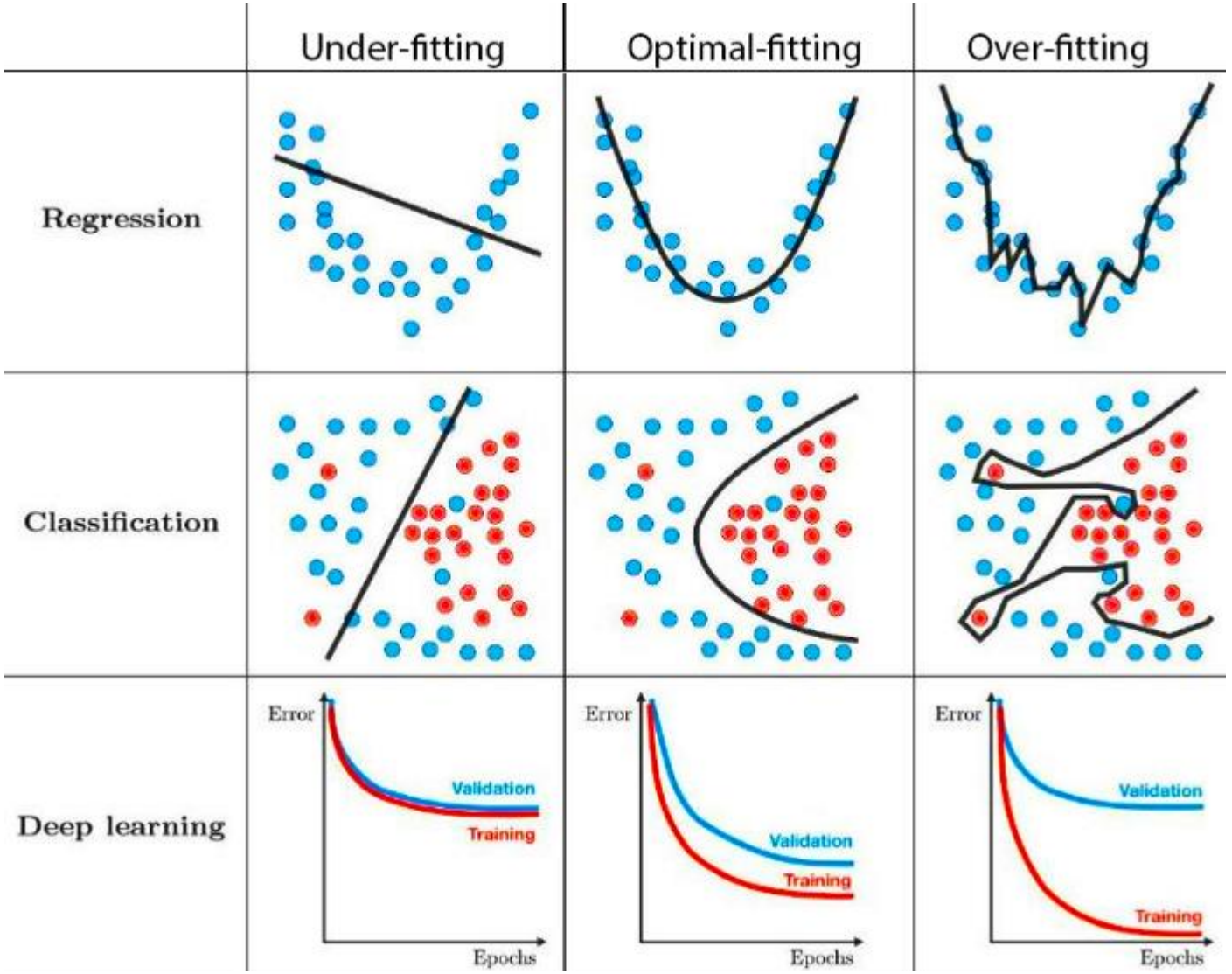
Underfitting e Overfitting

Underfitting

- Il modello è troppo semplice e non riesce a rappresentare le informazioni presenti nei dati.

Overfitting

- Il modello è troppo complesso e specializzato sulle caratteristiche dei dati del training set. Potrebbe quindi non essere in grado di generalizzare sui dati del test set.



https://towardsdatascience.com/techniques-for-handling-underfitting-and-overfitting-in-machine-learning-348daa2380b9

Bias e Variance

Bias

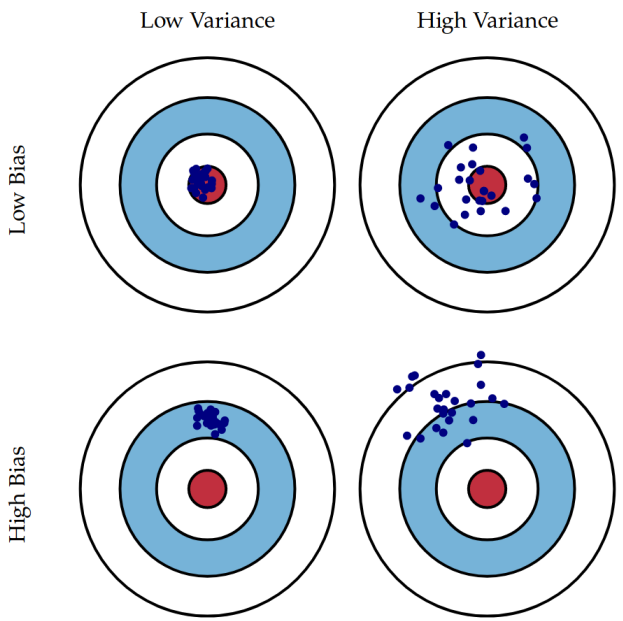
- E' la differenza tra la media delle previsioni del modello e il valore corretto che si sta cercando di prevedere.
- Un modello con bias elevato presta poca attenzione ai dati del training set e semplifica troppo il fenomeno (underfitting).
- Come risultato un modello con bias elevato tende sempre ad avere errori elevati sul training set.

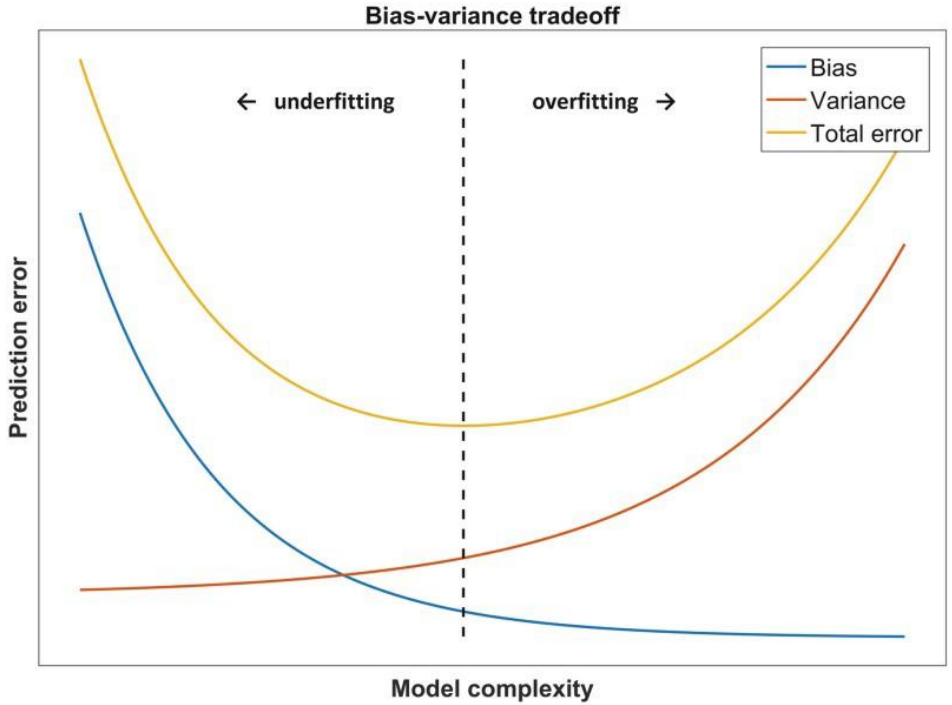
Variance

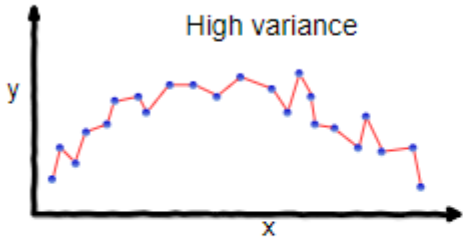
- E' la variabilità della previsione del modello per un certo dato o un valore che ci indica la «sparsità» dei dati.
- Un modello con variance elevata è troppo specializzato sui dati del training set e non generalizza su dati mai visti prima, come i dati del test set (overfitting).
- Come risultato un modello con variance elevata tende sempre ad avere buone performance sul training set ma errori elevati sul test set.

Bias e Variance – Trade-off

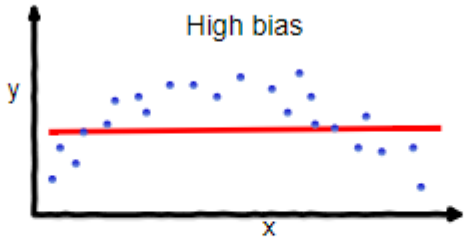
La «soluzione» per un modello ottimale è quella di cercare un **trade-off** tra bias e variance (variando i valori degli iperparametri del modello).



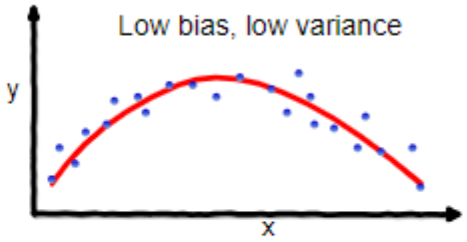




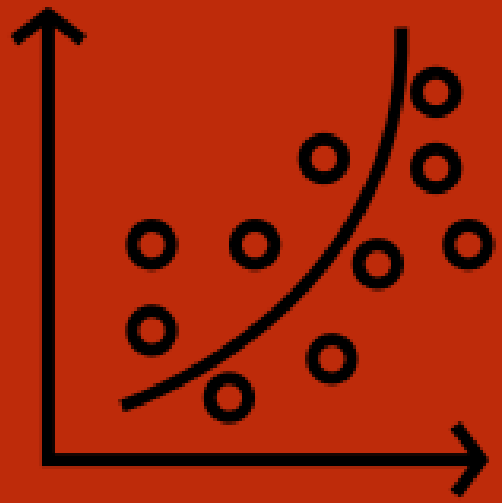
overfitting



underfitting



Good balance



Regressione

Alessia Angeli

Studente di dottorato in Data Science and Computation

Dipartimento di Informatica – Scienza e Ingegneria

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Univariate
Linear Regression

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Univariate Linear Regression – Il problema

ESEMPIO

Per iniziare, consideriamo un problema semplice:

«cercare di prevedere il prezzo di una casa in base alla metratura abitabile»

	sqft_living	price
0	1180	221900.0
1	2570	538000.0
2	770	180000.0
3	1960	604000.0
4	1680	510000.0
5	5420	1225000.0
6	1715	257500.0
7	1060	291850.0
8	1780	229500.0
9	1890	323000.0

Questo è un problema:

- Di **supervised learning**: il target è presente (il prezzo);
- Di **regressione**: cerchiamo di prevedere un valore reale (il prezzo).

Univariate Linear Regression – Notazione e terminologia

- m: # di samples nel training set

  21613

- x: variabile di input (feature)

  sqft_living

- y: variabile di output (target)

  price

- (x,y): singolo sample (singola riga del dataset)

  e.g., (x,y) = (1180,221900)

- $(x^{(i)},y^{(i)})$: i-esimo sample (i-esima riga del dataset)

  e.g., $(x^{(3)},y^{(3)})$ = (770,180000)

	sqft_living	price
0	1180	221900.0
1	2570	538000.0
2	770	180000.0
3	1960	604000.0
4	1680	510000.0
5	5420	1225000.0
6	1715	257500.0
7	1060	291850.0
8	1780	229500.0
9	1890	323000.0

Univariate Linear Regression – L'ipotesi

Nella regressione lineare univariata l'ipotesi h viene rappresentata come:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

- Dove $\theta = [\theta_0, \theta_1]$ rappresentano i parametri del modello;
- Cerchiamo quindi di prevedere y come funzione lineare di x.

Questo modello di ML viene chiamato Univariate Linear Regression poiché:

- Abbiamo una sola variabile indipendente x (**univariate**);
- Cerchiamo di predire la variabile dipendente (y) come funzione lineare (**linear**) della variabile indipendente (x);
- Cerchiamo di prevedere un valore reale (**regression**).

Univariate Linear Regression – Recap

- Training set

	sqft_living	price
0	1180	221900.0
1	2570	538000.0
2	770	180000.0
3	1960	604000.0
4	1680	510000.0
...	...	...

- Ipotesi $h_\theta(x) = \theta_0 + \theta_1 x$
- Parametri $\theta = [\theta_0, \theta_1]$

Ma… Come scegliamo i parametri?

Univariate Linear Regression – Funzione di errore

Per scegliere i parametri ottimali dobbiamo cercare di minimizzare l'errore, quindi ci troviamo di fronte ad un problema di minimizzazione, in cui:

- Vogliamo minimizzare (su $\theta = [\theta_0, \theta_1]$) la differenza tra $h(x)$ e y;
- Vogliamo minimizzare tale differenza per ogni sample (item) del training set.

<u>Un modo</u> per fare questo è:

- Considerare, per ogni sample, la differenze tra $h(x)$ e x;
- Calcolarne il quadrato (così non dobbiamo più preoccuparci del segno);
- Calcolare la media delle quantità ottenute;
- Per questioni matematiche (non stiamo a guardarle qui) moltiplicare per $^1/_2$;

Quindi:

$$min_{\theta_0,\theta_1} \frac{1}{2}\frac{1}{n}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Minimization of a function – Funzione di errore e Gradient Descent (GD)

$$\boxed{min_{\theta_0,\theta_1}} \frac{1}{2}\frac{1}{n}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Problema di minimo:

- Si procede applicando l'algoritmo di GD introdotto prima;
- Viene calcolato il gradiente della funzione di errore ed aggiornati i parametri dopo avere fissato il parametro (iperparametro) di learning rate.

NOTA: non è l'obiettivo di questo corso fornire tutti i dettagli matematici, cerchiamo di comprendere al meglio i concetti e le diverse tipologie di problemi/algoritmi collegati al mondo del machine e deep learning.

Multivariate Linear Regression

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Multivariate Linear Regression – Il problema

Una linear regression con più variabili indipendenti (variabili di input per il modello) viene chiamata Multivariate Linear Regression

Univariate Linear Regression

Avere 1 unica variabile (feature) da usare (in input) per predire il «prezzo della casa»

Vs

Multivariate Linear Regression

Avere un numero >1 di variabili (features) da usare (in input) per predire il «prezzo della casa»

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15	year
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955	0	98178	47.5112	-122.257	1340	5650	2014
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951	1991	98125	47.7210	-122.319	1690	7639	2014
2	180000.0	2	1.00	770	10000	1.0	0	0	3	6	770	0	1933	0	98028	47.7379	-122.233	2720	8062	2015
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965	0	98136	47.5208	-122.393	1360	5000	2014
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987	0	98074	47.6168	-122.045	1800	7503	2015

Multivariate Linear Regression – L'ipotesi

Nella regressione lineare multivariata l'ipotesi h viene rappresentata come:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n \longrightarrow h_\theta(x) = \theta^T x$$

- Dove $\theta = [\theta_0, \theta_1, \theta_2, \dots, \theta_n]$ rappresentano i parametri del modello;
- Cerchiamo quindi di prevedere y come funzione lineare di $x = [x_1, x_2, \dots, x_n]$.

Questo modello di ML viene chiamato Multivariate Linear Regression poiché:

- Abbiamo n (n>1) variabili indipendenti $x_1, x_2, \dots, x_n$ (**multivariate**);
- Cerchiamo di predire la variabile dipendente (y) come funzione lineare (**linear**) delle variabili indipendente ($x_1, x_2, \dots, x_n$).
- Cerchiamo di prevedere un valore reale (**regression**).

Multivariate Linear Regression – Funzione di errore

Analoga al caso Univariate Linear Regression però:

- Minimizzare rispetto a $\theta = [\theta_0, \theta_1, \theta_2, \ldots, \theta_n]$;
- Adesso X è un vettore $X = [X_1, X_2, \ldots, X_n]$.

$$min_\theta \frac{1}{2}\frac{1}{n}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

NOTA: adesso $X^{(i)}$ è un vettore!

ESEMPIO

$x^{(3)}$ = [2,1.00,770,10000,1.0,0,0,3,6,770,0,1933,0,98028,47.7379,-122.233,2720,8062,2015]

Multivariate Linear Regression – Feature scaling

Assiucurarsi che tutte le features abbiano tutte scala simile (i.e., tutte le features hanno un range di valori possibili simile).

In generale:

- Feature: X
- Range valori: (a,b)

$$x_i \rightarrow \frac{x_i}{b}$$

- Range valori dopo **feature scaling** (0,1).

NOTA: più in generale con il features scaling si cerca di portare tutte le feature ad avere un range di [-1,1] dato che in generale si possono avere anche features con valori negativi.

Non è tanto il valore assoluto del range che importa, ma il fatto che i range delle diverse variabili (features) abbiano valore simile!

Multivariate Linear Regression – Mean normalization

Un'altra operazione che spesso è bene applicare alle features prima di un algoritmo di learning, oltre al feature scaling, è la **mean normalization**.

In questo caso, data una feature, questa viene divisa per l'ampiezza del range di valori assunti (o per la deviazione standard), ma prima gli viene sottratta la media dei valori assunti. Questo con l'obiettivo di portare tutte le features ad avere distribuzione con media «vicino» a zero.

In generale:

- Feature: X
- Range valori: (a,b)
- Media dei valori X nel training set: μ

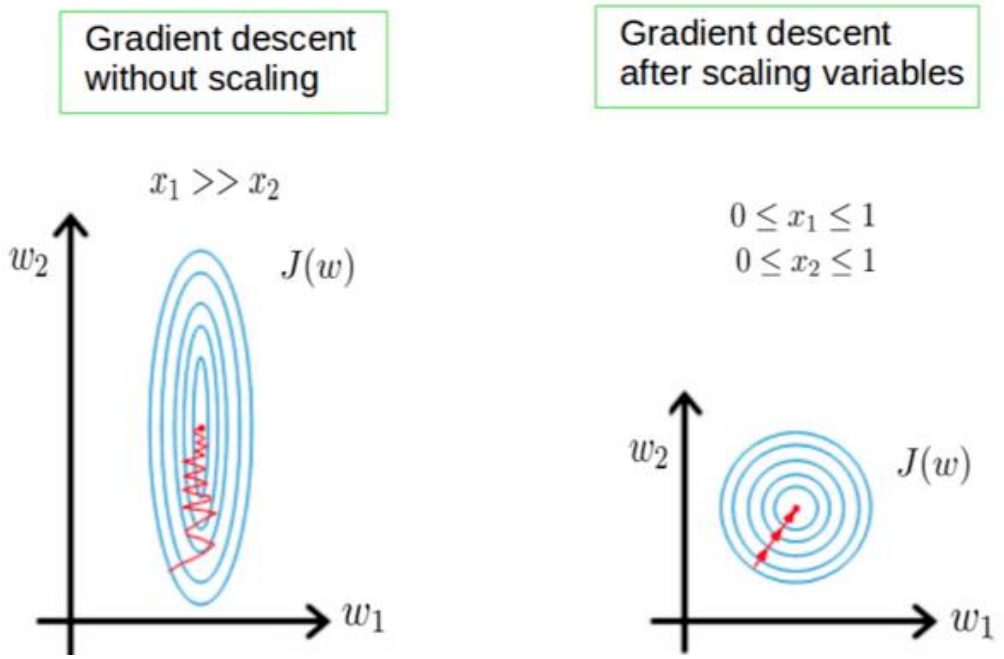
$$x_i \rightarrow \frac{x_i - \mu}{b - a}$$

$b - a$ → deviazione standard

Multivariate Linear Regression – Feature scaling and mean normalization

- **Feature scaling** and **mean normalization** sono semplici operazioni anche se può non essere intuitivo perché questo calcolo ci possa aiutare con gli algoritmi di machine e deep learning.

- Con queste operazioni si può rendere più veloce (meno iterazioni) la convergenza (ipotizzando di essere in un caso in cui converge) dell'algoritmo GD (non vedremo i dettagli in questo corso).



https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

Polynomial Regression

Polynomial Regression – Il problema

Collegandoci all'idea di «scegliere le proprie features», passiamo alla Polynomial Regression.

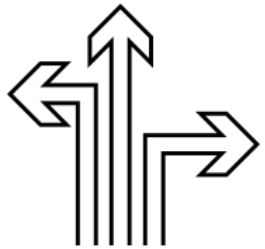
In questo caso possiamo analizzare problemi con una (caso univariato) o più (caso multivariato) variabili indipendenti (variabili di input per il modello).

Con la Polynomial Regression si cerca di trovare una **funzione polinomiale** (e.g., quadratica, cubica, di ordine k) che segua l'andamento dei nostri dati (fit).

Detto questo, abbiamo una vasta scelta di opzioni su come combinare tra loro le variabili di input (features):

- Moltiplicare tra loro due features;
- Calcolare il quadrato di una feature, il cubo, la potenza k-esima;
- Calcolare la radice quadrata di una feature, la radice cubica, la radice k-esima;
- …

Un po' disorientante! Non trovate?



Polinomial Regression – L'ipotesi

Nella regressione polinomiale l'ipotesi h viene rappresentata come:

$$h_\theta(x) = p(x)$$

dove:

- $p(x)$ polinomio in $x = [x_1, x_2, \ldots, x_n]$ (possibile anche n = 1);
- $\theta = [\theta_0, \theta_1, \theta_2, \ldots, \theta_n]$ rappresentano i parametri del modello;
- Cerchiamo quindi di prevedere y come funzione polinomiale di $x = [x_1, x_2, \ldots, x_n]$.

Questo modello di ML viene chiamato Polynomial Regression poiché:

- Cerchiamo di predire la variabile dipendente (y) come funzione polinomiale (**polynomial**) della variabile indipendente (x);
- Cerchiamo di prevedere un valore reale (**regression**).

Polinomial Regression – L'ipotesi

- Come facciamo a capire come scegliere questa funzione?

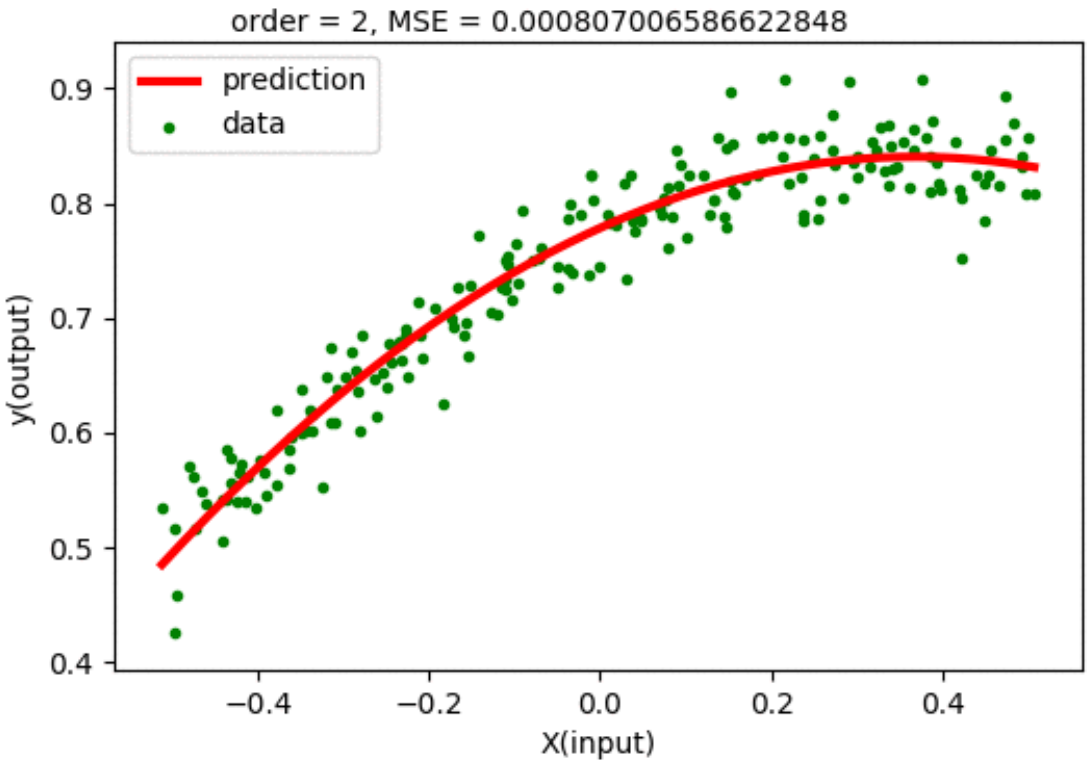
Ci sono algoritmi che in automatico scelgono le features «più adatte» (rispetto a tutte le features disponibili) per cercare di risolvere un determinato problema.

- La questione fondamentale è che:

Potendo considerare diverse funzioni si ha la possibilità di trovare una curva che segua meglio l'andamento dei dati considerati rispetto ad una qualsiasi retta portando quindi a costruire un modello migliore in relazione al problema che si sta analizzando.

NOTA: Attenzione a underfitting e overfitting!

ESEMPIO



Caso univariato
features = 1

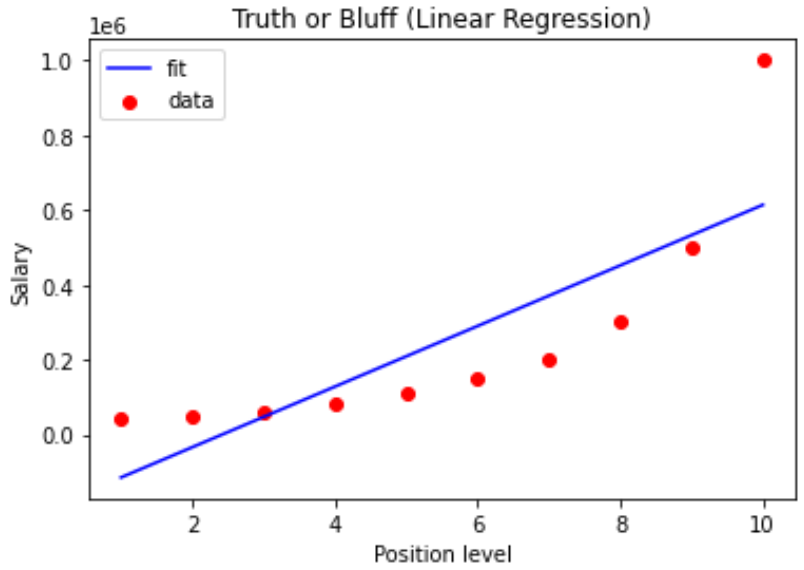
Polinomial Regression – Funzione di errore

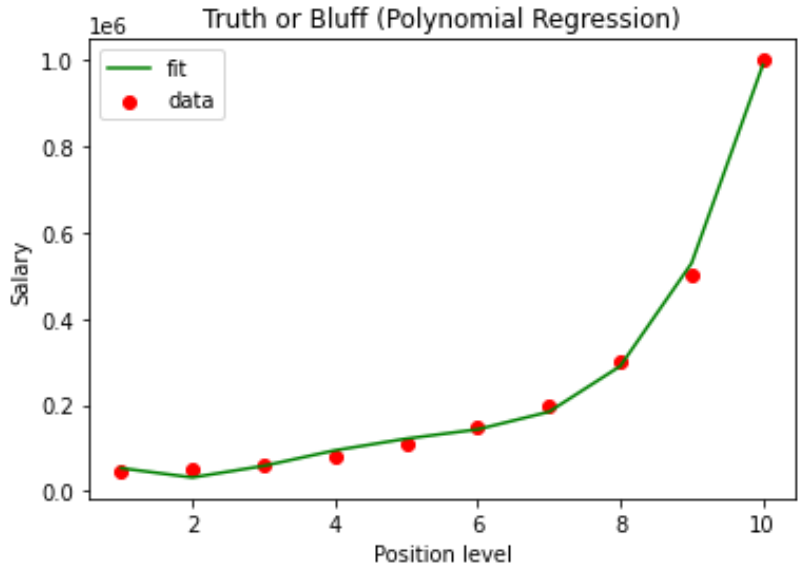
Analogo al caso della Univariate Linear Regression o della Multivariate Linear Regression in base al numero di variabili indipendenti (features in input per il modello).

ESEMPIO

- **Notebook 1 – Caso Studio Esempio**

	years_experience	salary
0	1	45000
1	2	50000
2	3	60000
3	4	80000
4	5	110000
5	6	150000
6	7	200000
7	8	300000
8	9	500000
9	10	1000000





https://towardsdatascience.com/machine-learning-polynomial-regression-with-python-5328e4e8a386

Multi Layer Perceptron (MLP)

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Multi Layer Perceptron

Con Multi Layer Perceptron (MLP) viene indicata la classe di feed-forward neural networks (le Artificial Neural Networks (ANN) più «classiche»).

Un modello MLP può essere shallow o deep (in base al numero di hidden layers).

Il numero di neuroni in ogni layer viene deciso da chi implementa il modello (il numero ottimale dipende dai dati e dal problema e non c'è un metodo analitico per calcolarlo esattamente).

Per quanto riguarda il funzionamento del modello MLP ci si rifà a quanto detto per le reti neurali (non aggiungeremo altri dettagli e/o specifiche tecniche in questo corso).

In quanto rete neurale, il modello MLP lavora «bene» training set contenente molti dati e può lavorare bene anche con un numero elevato di features in input.

Multi Layer Perceptron – Il problema

ESEMPIO

In questo caso consideriamo lo stesso problema affrontato con la Multivariate Linear Regression.

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15	year
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955	0	98178	47.5112	-122.257	1340	5650	2014
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951	1991	98125	47.7210	-122.319	1690	7639	2014
2	180000.0	2	1.00	770	10000	1.0	0	0	3	6	770	0	1933	0	98028	47.7379	-122.233	2720	8062	2015
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965	0	98136	47.5208	-122.393	1360	5000	2014
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987	0	98074	47.6168	-122.045	1800	7503	2015

Per quanta riguarda il modello considereremo quello implementato di default dalla funzione *sklearn.neural_network.MLPRegressor()* della libreria **scikit-learn** di Python, dato che non ci occupiamo dell'ottimizzazione degli iperparametri di un modello (e le reti neurali sono modelli con molti di iperparametri).

Per questo modello non aggiungiamo ulteriori dettagli, vedremo direttamente il codice nel Notebook di esercitazione.

Regressione – MAE, MSE e R^2

Alessia Angeli

Studente di dottorato in Data Science and Computation

Dipartimento di Informatica – Scienza e Ingegneria

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Regressione – Funzione di errore MAE

L'errore medio assoluto, Mean Absolute Error (MAE), si calcola considerando, per ogni elemento del test set, il valore assoluto della differenza tra il valore attuale ed il valore predetto dal modello, e poi facendo la media tra queste quantità.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \qquad \text{dove } N = \text{numero degli elementi del test set}$$

Per come è definito, il MAE non può assumere valori negativi.

VANTAGGIO: considerando il valore assoluto tutti gli errori vengono pesati sulla stessa scala lineare (tutti gli errori vengono «trattati allo stesso modo»).

SVANTAGGIO: se sono presenti previsioni outliers queste potrebbero alzare di molto il valore del MSE dando un'informazione «alterata» delle performance del modello.

https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3

Regressione – Funzione di errore MSE

L'errore medio quadratico, Mean Squared Error (MSE) è forse la funzione di errore più semplice e più utilizzata per problemi di regressione. Il MSE si calcola considerando, per ogni elemento del test set, il quadrato della differenza tra il valore attuale ed il valore predetto dal modello, e poi facendo la media tra queste quantità.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \quad \text{dove } N = \text{numero degli elementi del test set}$$

Per come è definito, il MSE non può assumere valori negativi.

VANTAGGIO: il MSE assicura che, rispetto al training set, non ci siano previsioni outliers con errori elevati in quanto il MSE dà un peso maggiore a questi errori (con il quadrato della differenza).

SVANTAGGIO: se il modello considerato effettua una singola previsione outlier, il MSE ingrandisce l'errore anche se, in molti casi, non ci si preoccuperebbe di questi valori (pochi e isolati) avendo l'obiettivo di un modello che raggiunga buoni risultati sulla maggioranza dei dati.

https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3

Regressione – Coefficiente di determinazione delle previsioni

Il coefficient di determinazione delle previsioni (R^2) è una misura di quanto I dati osservati vengono replicati dal modello. La possiamo quindi considerare una misure della bontà delle previsioni del modello considerate.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

dove

- $SS_{res} = \sum_i (y_i - \hat{y}_i)^2$
- $SS_{tot} = \sum_i (y_i - \bar{y})^2$

- y_i dati osservati
- $\hat{y}_i$ dati predetti
- $\bar{y}$ media dati osservati

Più il valore di R^2 è vicino ad 1 e più il modello è efficace per il problema considerato.

Hands-On

- Notebook 1 – Caso Studio House Price

ML, DL e Regressione – NOTE

Alessia Angeli

Studente di dottorato in Data Science and Computation

Dipartimento di Informatica – Scienza e Ingegneria

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

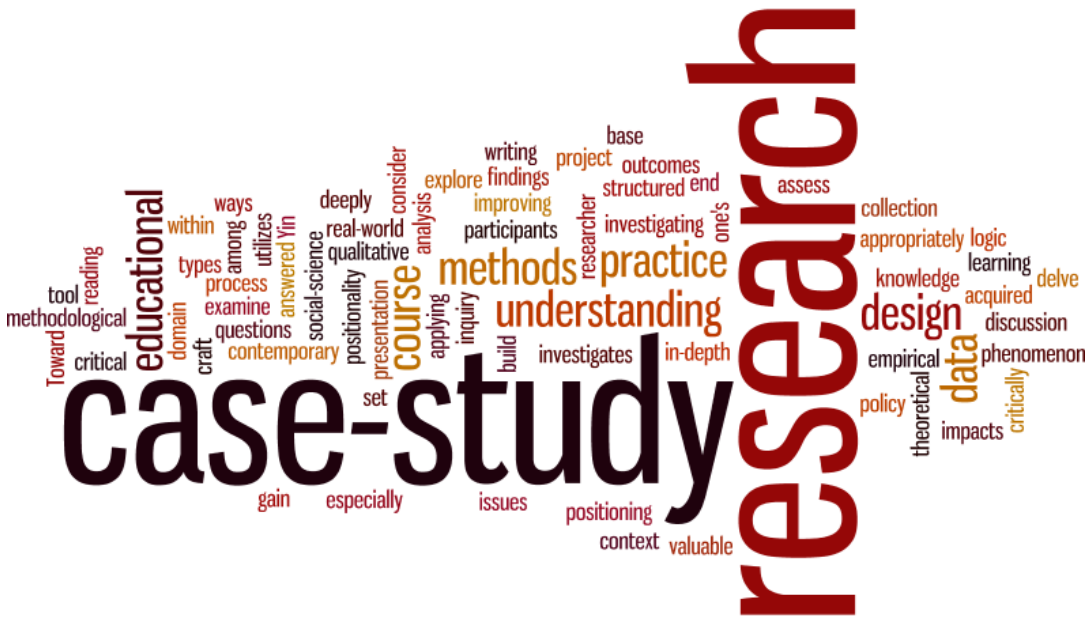
NOTA

- In questa lezione abbiamo visto alcuni modelli di ML e DL che si possono applicare per cercare di risolvere problemi di regressione.

- Importante sottolineare che questi non sono gli unici.

- Esistono altri modelli (e.g., Support Vector Regression, Random Forest Regression, Gradient Boosting Regression) che si possono applicare per cercare di risolvere questo tipo di problemi.

Continue to study and research…



NOTA

- Come avete visto dai Notebooks di esercitazioni per implementare modelli di ML e DL non è necessario scrivere il modello matematico from-scratch.

- Esistono infatti molte librerie di Python (e.g., statsmodels, scikit-learn, PyTorch, TensorFlow, Keras) con modelli già implementati (più o meno ad alto livello).

- Nei Notebooks di esercitazione per questa lezione, in particolare, abbiamo utilizzato funzioni di Scikit-learn.

Riferimenti

Corso «Machine Learning», Andrea Asperti, Università di Bologna (2018).

Corso «Applied Machine Learning», Daniele Bonacorsi, Università di Bologna (2019).

Sitografia presente slide per slide.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Alessia Angeli

Dipartimento di Informatica – Scienza e Ingegneria

alessia.angeli2@unibo.it