Sapere utile

IFOA
Istituto Formazione Operatori Aziendali

BIG DATA e Analisi dei Dati

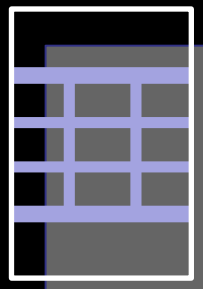
Mauro Bellone,
Robotics and AI researcher

bellonemauro@gmail.com
www.maurobellone.com

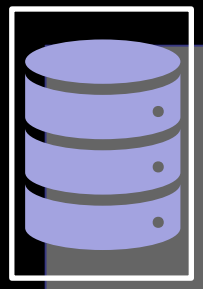
Obiettivo

- ✓ Regressione lineare e logistica
- ✓ Regressione - tutorial python
- ✓ Clusterizzazione per dati non annotati
- ✓ Clusterizzazione - tutorial python

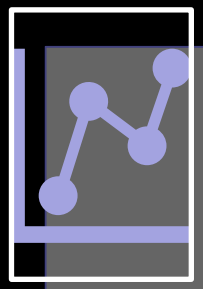
Task di data mining



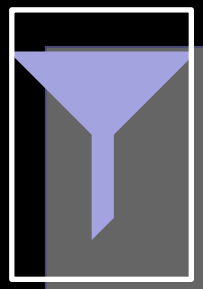
Clusterizzazione



Classificazione



Regressione



Aggregazione

Cluster Hadoop VS cloud service

- ✓ Forniscono piattaforme già integrate con numerose funzionalità
- ✓ Costo variabile in base all'utilizzo
- ✓ Alta flessibilità e adattabilità
- ✓ NO costo hardware
- ✓ NO costo di manutenzione (hardware e software)

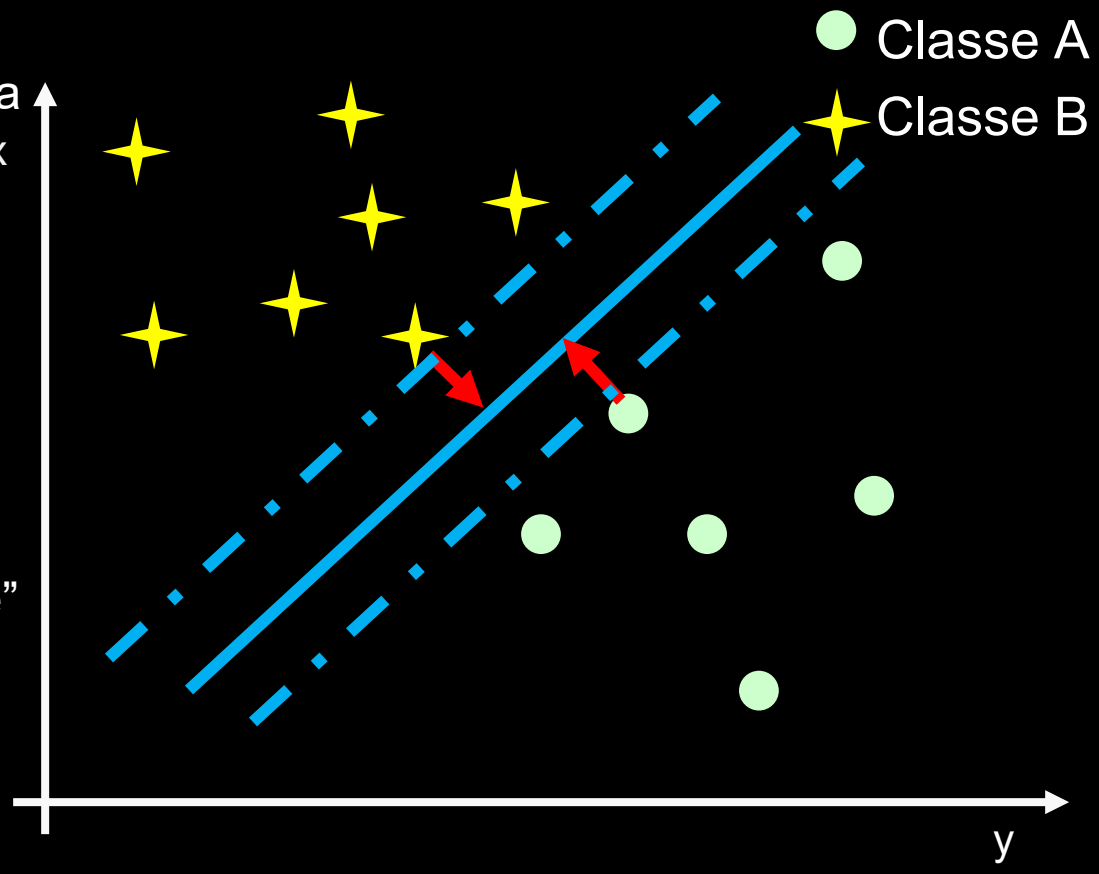
Support vector machines

Il support vector machine o SVM è un algoritmo di apprendimento che ordina i dati disponibili in categorie

In termini più tecnici si cerca la retta che massimizza la distanza tra i primi dati delle classi detti “support vectors”

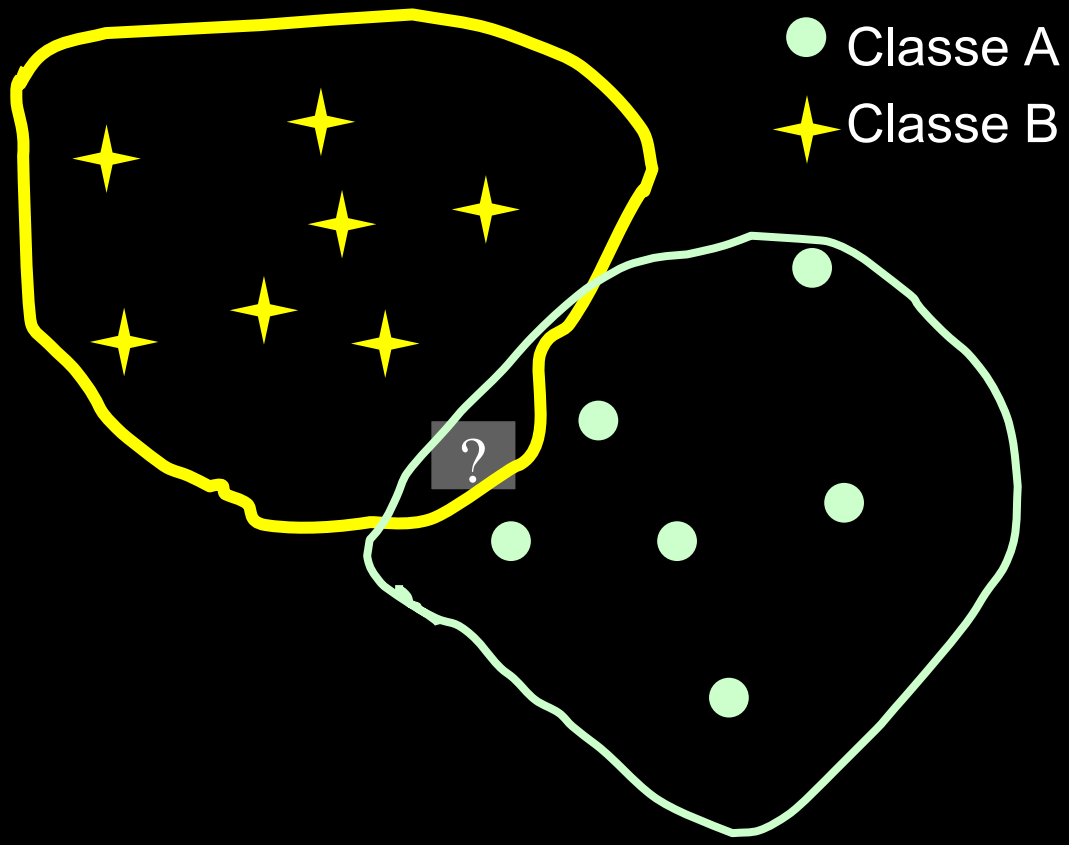
D+ e d- sono le distanze tra il punto delle classi più vicino alla retta di separazione.

La somma è chiamata “margine”



k-nearest neighbor k-NN

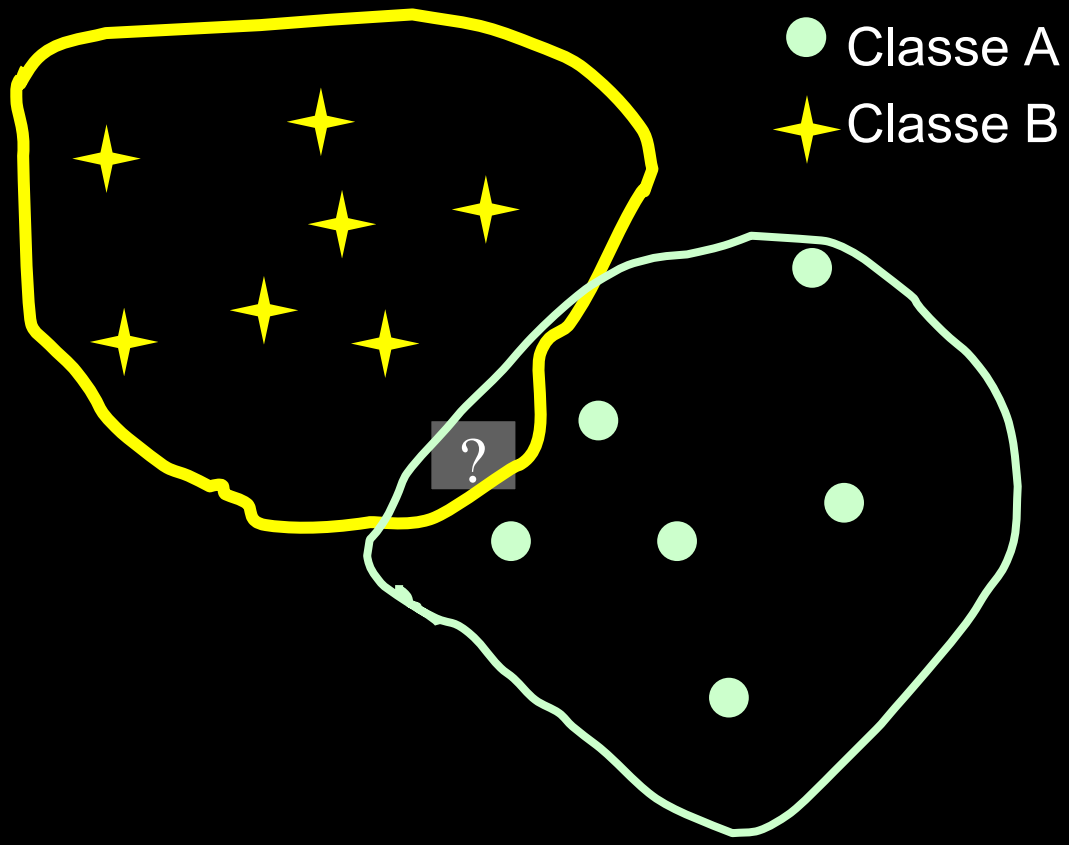
L’algoritmo k-nearest neighbors (k-NN) è un metodo di classificazone non parametrica.



k-nearest neighbor k-NN

L’algoritmo k-nearest neighbors (k-NN) è un metodo di classificazone non parametrica.

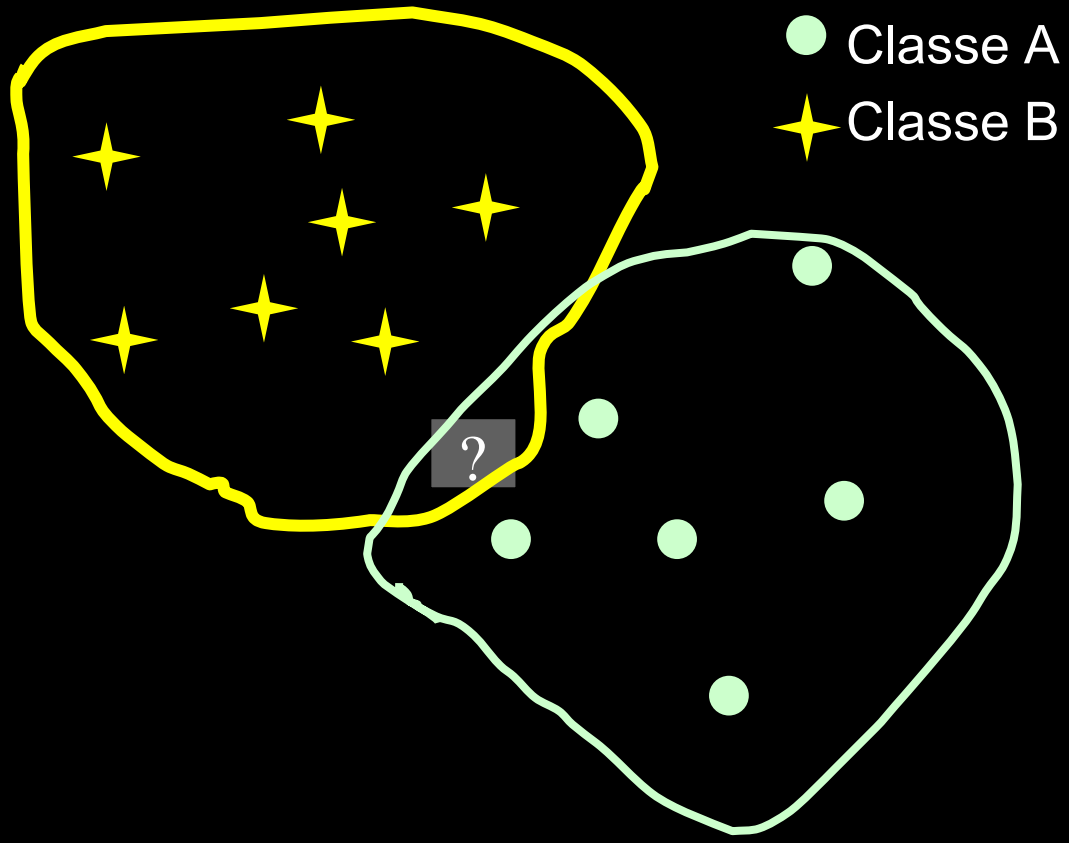
Può usato per classificazione e regressione



k-nearest neighbor k-NN

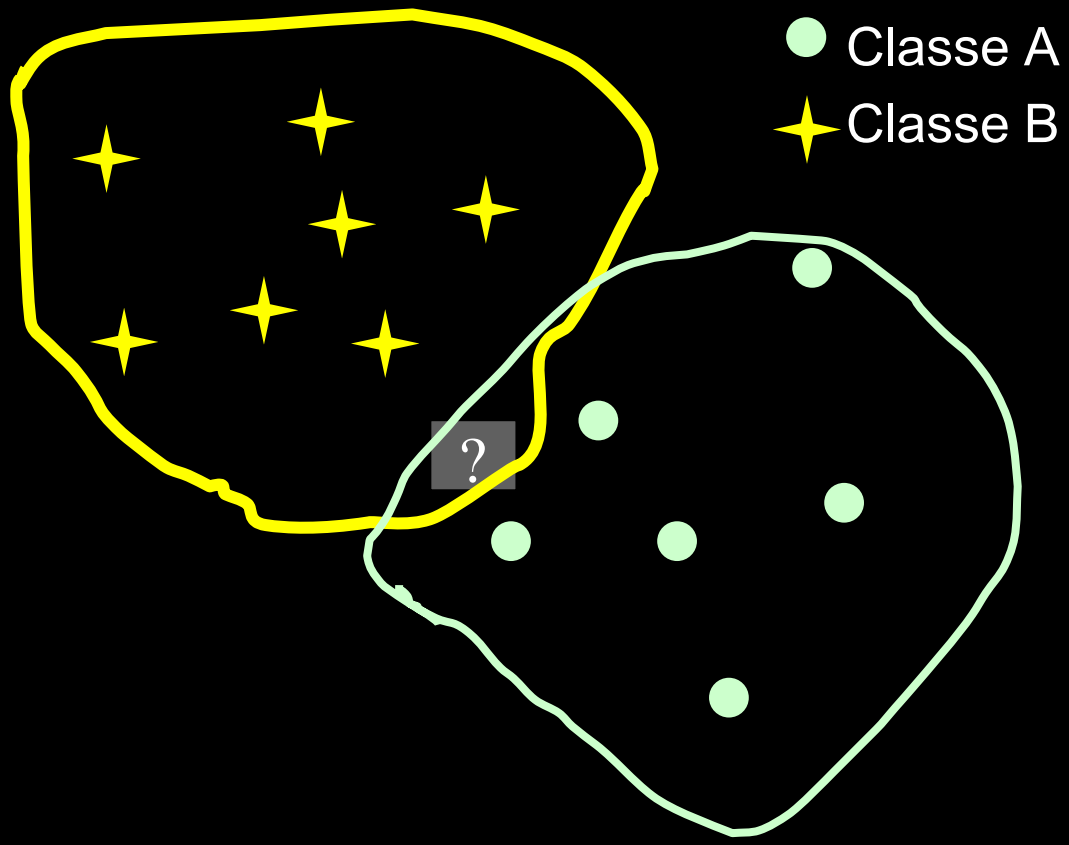
Nella classificazione l'output è l'appartenenza ad una classe.

Un oggetto è classificato in base alla pluralità di voti di tutti i neighbors ai quali è associato



k-nearest neighbor k-NN

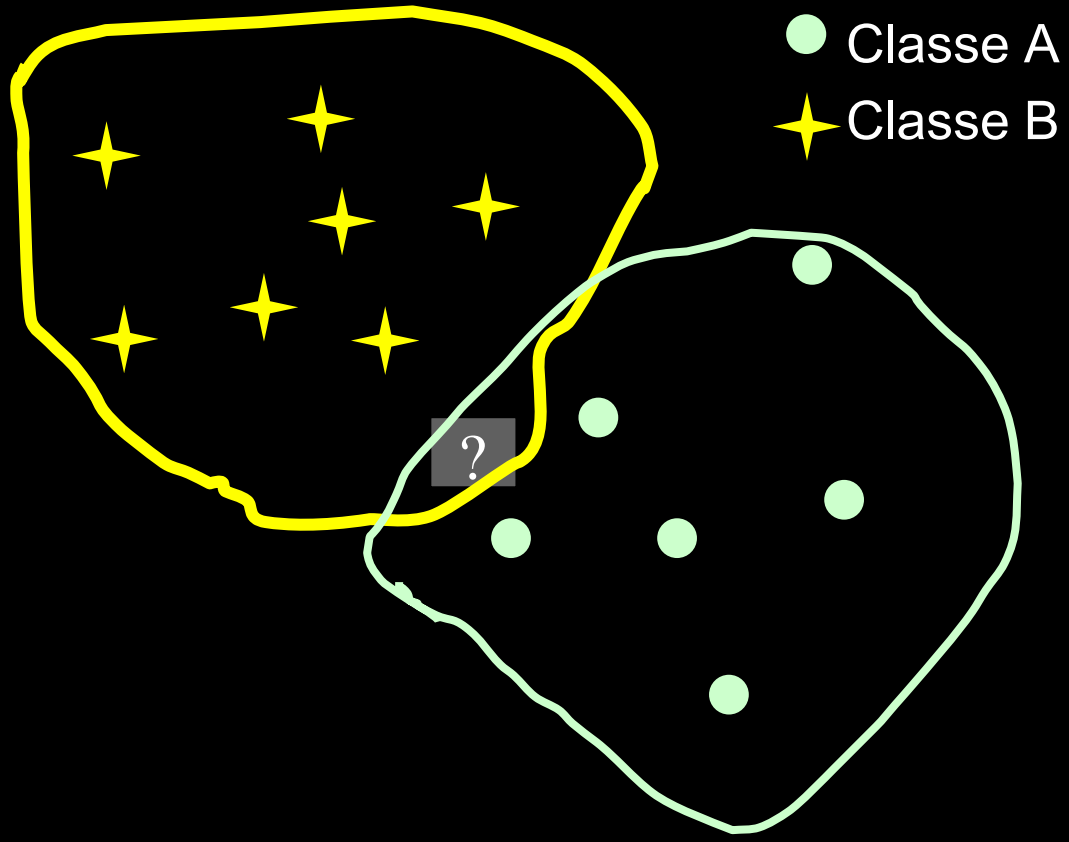
Nella regressione k-NN l'output è una proprietà dell'oggetto, tipicamente la media delle distanze dai k-vicini



k-nearest neighbor k-NN

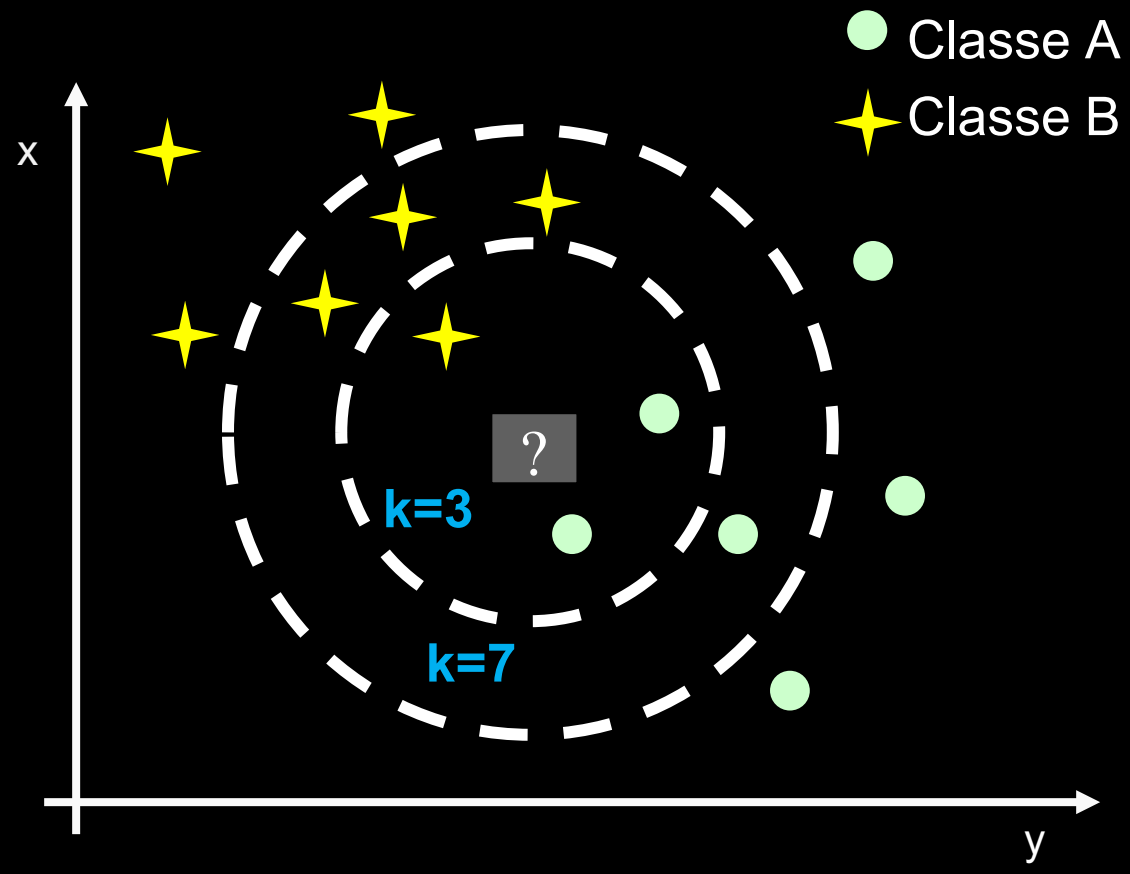
k-NN memorizza tutti i casi disponibili e effettua una classificazione o una regressione in base ad una misura di somiglianza (es. distanza)

Ogni punto è classificato in base alla classe dei punti più vicini



k-nearest neighbor k-NN

k è il parametro che definisce il numero di vicini da considerare per effettuare la decisione

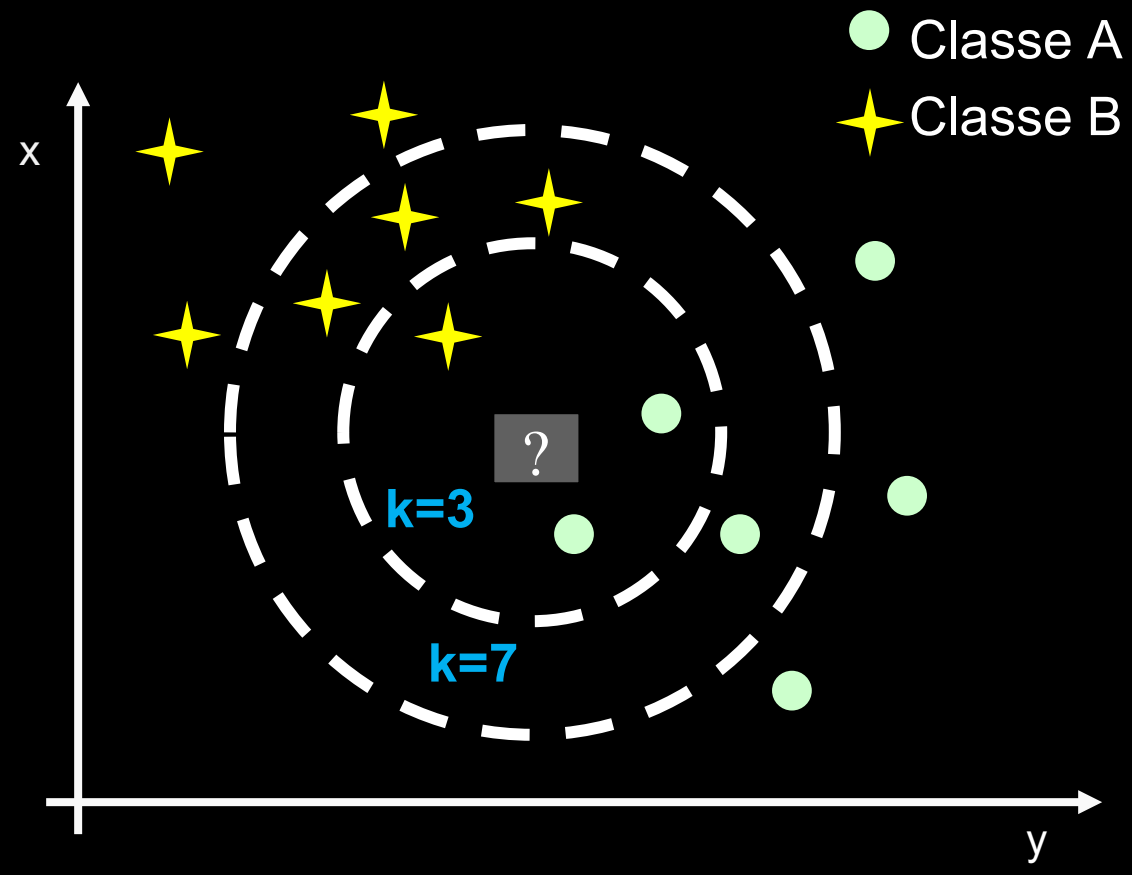


k-nearest neighbor k-NN

k è il parametro che definisce il numero di vicini da considerare per effettuare la decisione

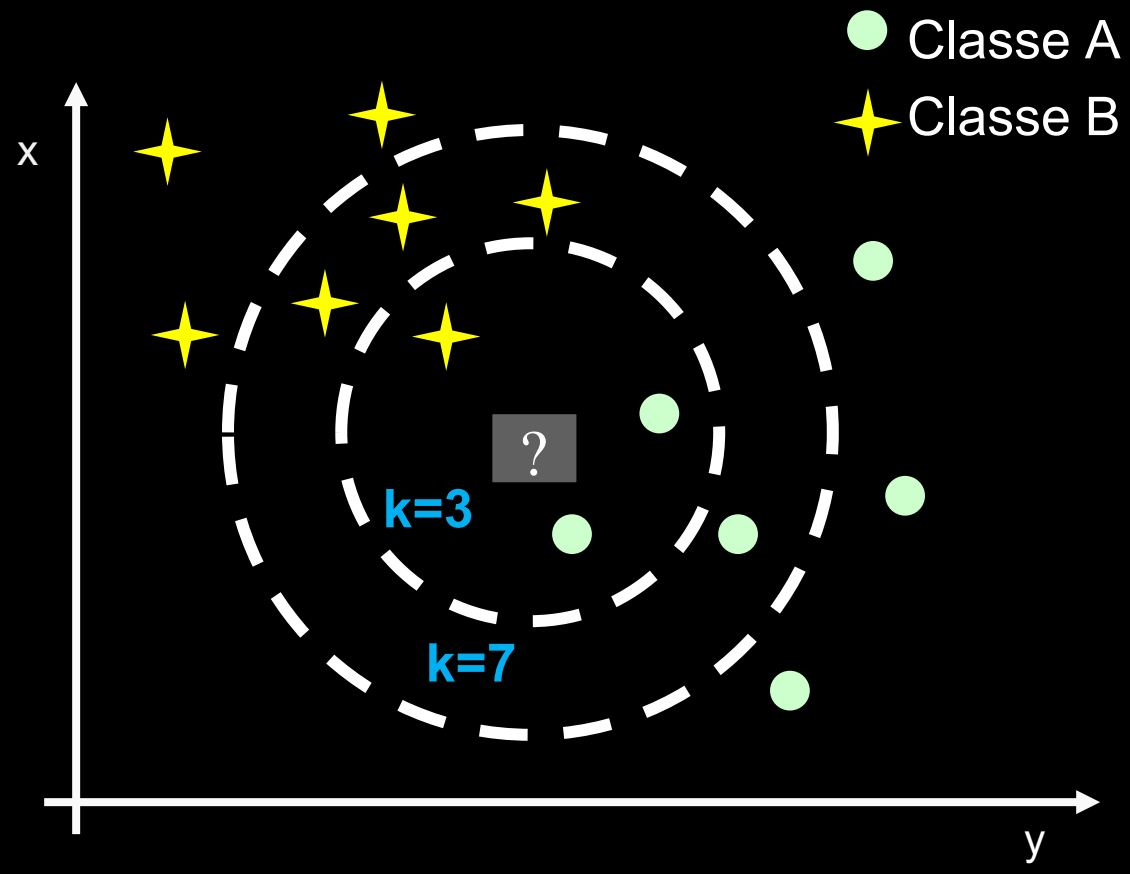
Nel caso in figura:

- se k=3 ➔ ? = Classe A
- se k=7 ➔ ? = Classe B



k-nearest neighbor k-NN

$k = \sqrt{n}$ dove *n* è il numero di campioni disponibili



k-nearest neighbor k-NN – pseudocode

input:

```
X dati, Y annotazioni, N_data #esempi, k soglia NN
c punto da classificare
```

output:

```
C classe di appartenenza
```

k-nearest neighbor k-NN – pseudocode

input:

```
 X dati, Y annotazioni, N_data #esempi, k soglia NN
 c punto da classificare
```

output:

```
 C classe di appartenenza
```

```
for i←1 to N_data do
  d([X,Y],c)=distanza([X,Y], c)
end
```

calcola la distanza Euclidea tra il dato da classificare e tutti i punti dell'insieme (gli esempi delle classi).

k-nearest neighbor k-NN – pseudocode

input:

```
 X dati, Y annotazioni, N_data #esempi, k soglia NN
 c punto da classificare
```

output:

```
  C classe di appartenenza
```

```
for i←1 to N_data do
  d([X,Y],c)=distanza([X,Y], c)
end
NN(c) = select(d([X,Y],c,k)
```

Selezioniamo k esempi dal nostro insieme di esempi noti

k-nearest neighbor k-NN – pseudocode

input:

```
 X dati, Y annotazioni, N_data #esempi, k soglia NN
 c punto da classificare
```

output:

```
  C classe di appartenenza
```

```
for i←1 to N_data do
  d([X,Y],c)=distanza([X,Y], c)
end
NN(c) = select(d([X,Y],c,k)
```

Selezioniamo k esempi dal nostro insieme di esempi noti

Controlliamo la distanza tra i *k* esempi e il nostro nuovo dato per dare una stima sulla classe di appartenenza

k-nearest neighbor k-NN – pseudocode

input:

```
 X dati, Y annotazioni, N_data #esempi, k soglia NN
 c punto da classificare
```

output:

```
 C classe di appartenenza
```

```
for i←1 to N_data do
  d([X,Y],c)=distanza([X,Y], c)
end
NN(c) = select(d([X,Y],c,k)
somma(annotazioni in N(c))
```

Cerchiamo la classe più commune tra gli esempi selezionati

k-nearest neighbor k-NN – pseudocode

input:

```
 X dati, Y annotazioni, N_data #esempi, k soglia NN
 c punto da classificare
```

output:

```
 C classe di appartenenza
```

```
for i←1 to N_data do
  d([X,Y],c)=distanza([X,Y], c)
end
NN(c) = select(d([X,Y],c,k)
somma(annotazioni in N(c))
C = argmax(somma(annotazione))
```

Cerchiamo la classe più commune tra gli esempi selezionati

k-nearest neighbor k-NN – pseudocode

input:

```
 X dati, Y annotazioni, N_data #esempi, k soglia NN
 c punto da classificare
```

output:

```
  C classe di appartenenza

for i←1 to N_data do
   d([X,Y],c)=distanza([X,Y], c)
end
NN(c) = select(d([X,Y],c,k)
somma(annotazioni in N(c))
C = argmax(somma(annotazione))
```

SVM tutorial

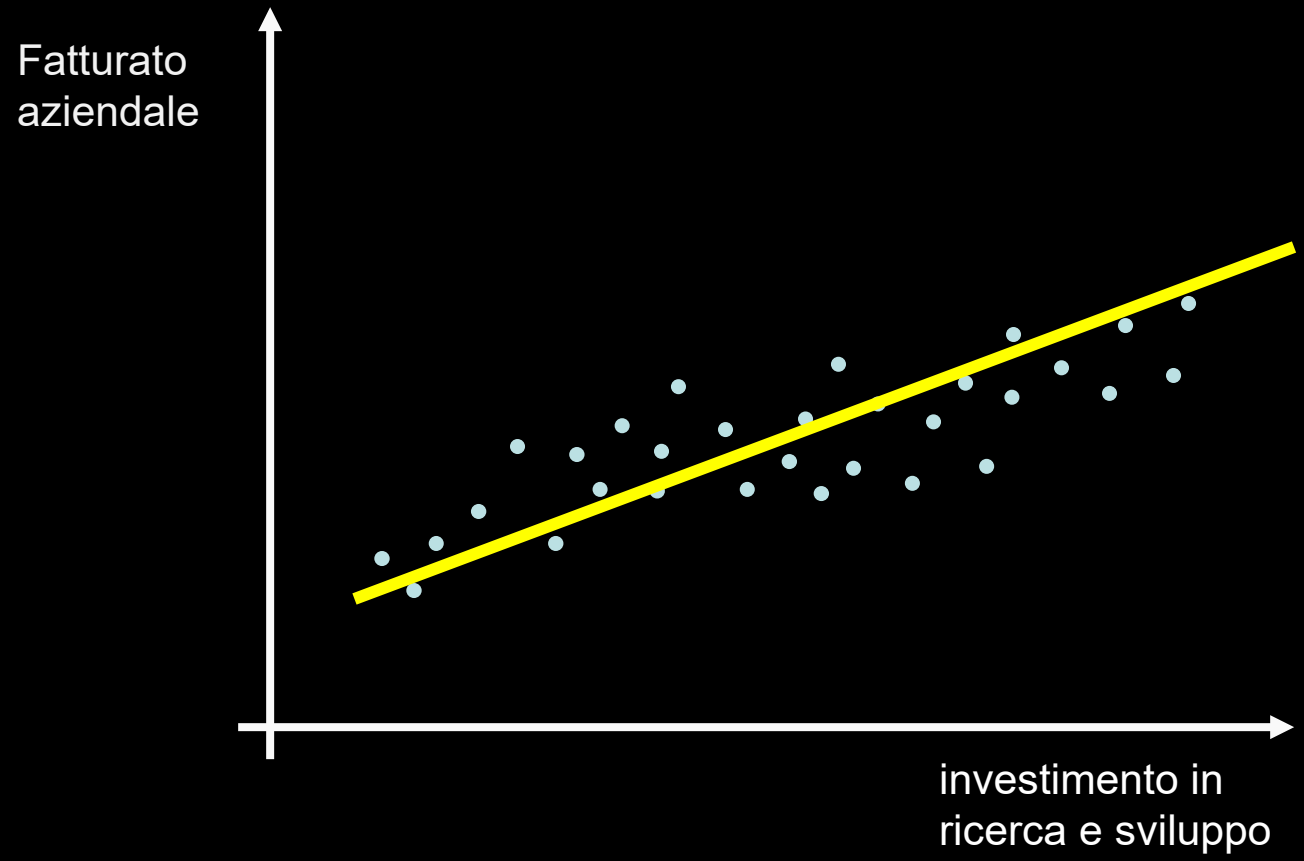
Codice

k-NN tutorial

Codice

Regressione

Esempio1: regressione fatturato aziendale contro investimento in ricerca e sviluppo



Regressione

- ✓ Variabile indipendente X: Variabile il cui valore non cambia come risultato (effetto) della variabile dipendente

- ✓ Variabile dipendente Y: Variabile che cambia se come risultato (effetto) della variabile indipendente ----- Correlazione causa effetto!

Regressione

- ✓ Variabile indipendente X: Variabile il cui valore non cambia come risultato (effetto) della variabile dipendente

- ✓ Variabile dipendente Y: Variabile che cambia se come risultato (effetto) della variabile indipendente ----- Correlazione causa effetto!

Per il nostro esempio, non è possible controllare il fatturato aziendale a causa di eventi esterni, ma l'investimento in ricerca e sviluppo ha sicuramente un impatto positivo

Applicazioni

La regressione consiste nella generazione di un modello statistico usato per predire relazioni tra una variabile dipendente e una variabile indipendente

Esamina 2 fattori:

1. Quale variabile è significativa per predire l'output della variabile dipendente

2. Quanto è significativo la retta di regressione per fare una effettiva predizione con la massima accuratezza possibile

Categorie

- ✓ Regressione lineare semplice – Molti dati e si stima una retta di minima distanza

Categorie

- ✓ Regressione lineare semplice – Molti dati e si stima una retta di minima distanza
- ✓ Regressione lineare multipla - Variabili multiple (più uscite)

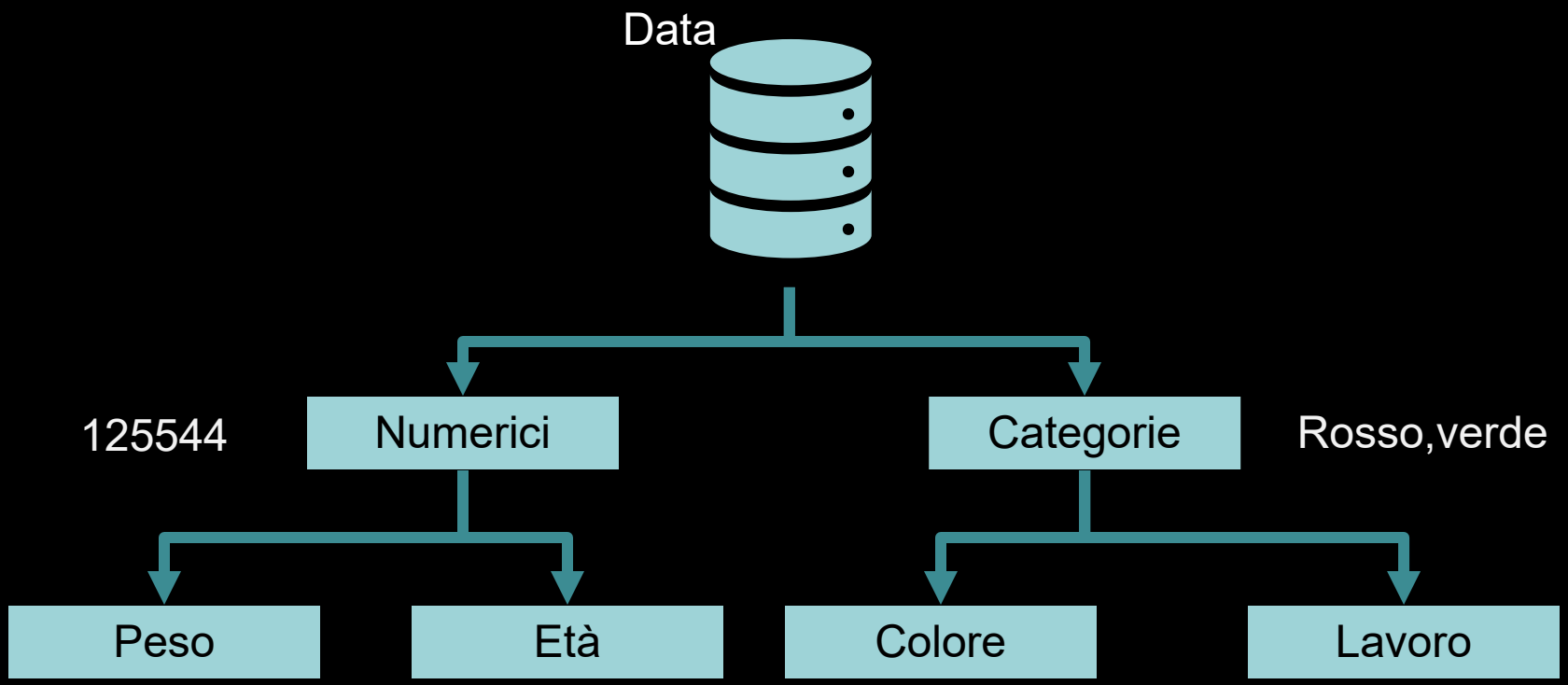
Categorie

- ✓ Regressione lineare semplice – Molti dati e si stima una retta di minima distanza
- ✓ Regressione lineare multipla - Variabili multiple (più uscite)
- ✓ Regressione polinomiale – Invece di stimare una retta si stima una curva di grado polinomiale

Regressione lineare: applicazioni

- ✓ Crescita economica
- ✓ Prezzi
- ✓ Vendite

Valori numeri e categorici



Regressione lineare

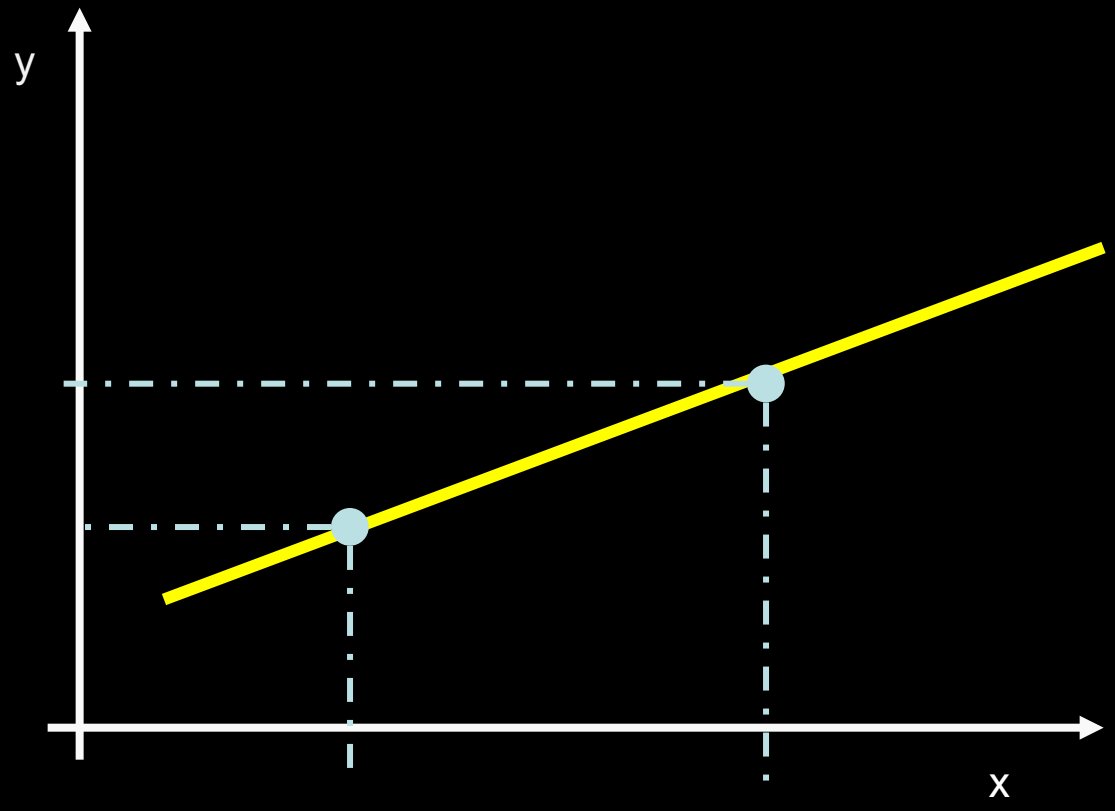
La forma più semplice di regressione è la regressione su una retta, stimare quindi coefficiente angolare e quota della retta

$$y = mx + c$$

Regressione lineare

La forma più semplice di regressione è la regressione su una retta, stimare quindi coefficiente angolare e quota della retta

$$y = mx + c$$



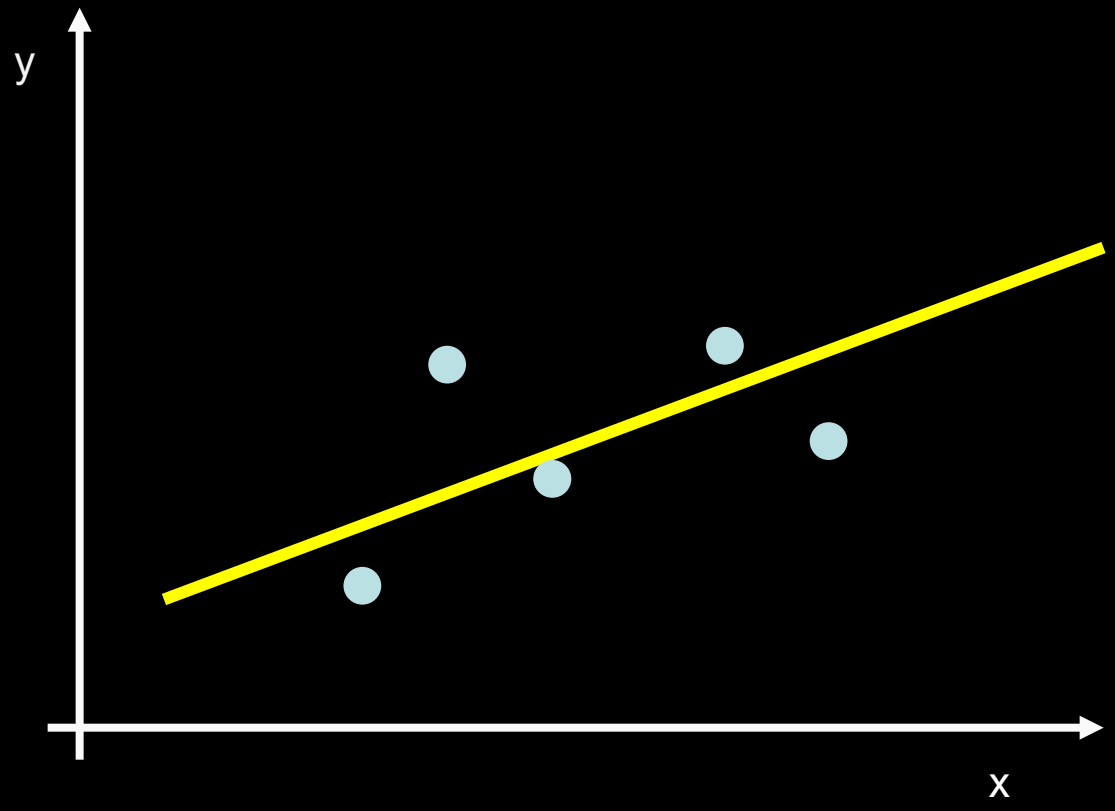
y = variabile dipendente

x = variabile indipendente

Regressione lineare

La forma più semplice di regressione è la regressione su una retta, stimare quindi coefficiente angolare e quota della retta

$$y = mx + c$$



y = variabile dipendente

x = variabile indipendente

Regressione lineare

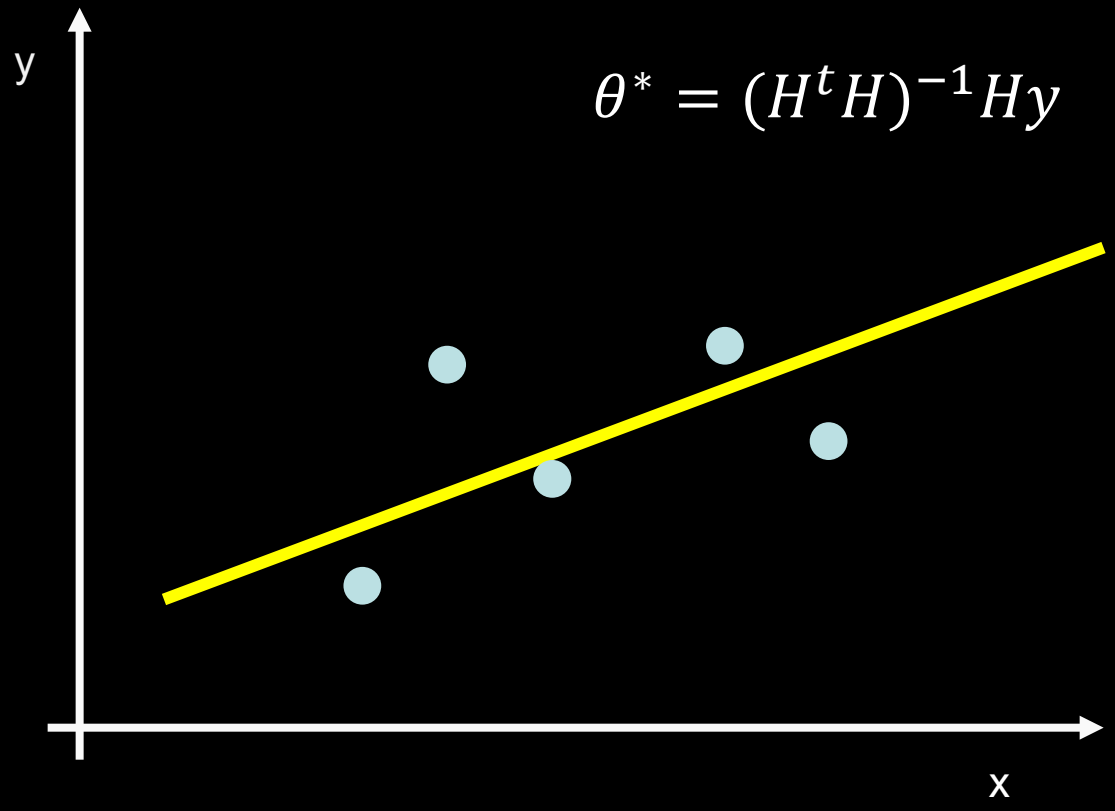
La forma più semplice di regressione è la regressione su una retta, stimare quindi coefficiente angolare e quota della retta

$$y = mx + c$$

$$\theta^* = (H^t H)^{-1} H y$$

y = variabile dipendente

x = variabile indipendente



Regressione lineare

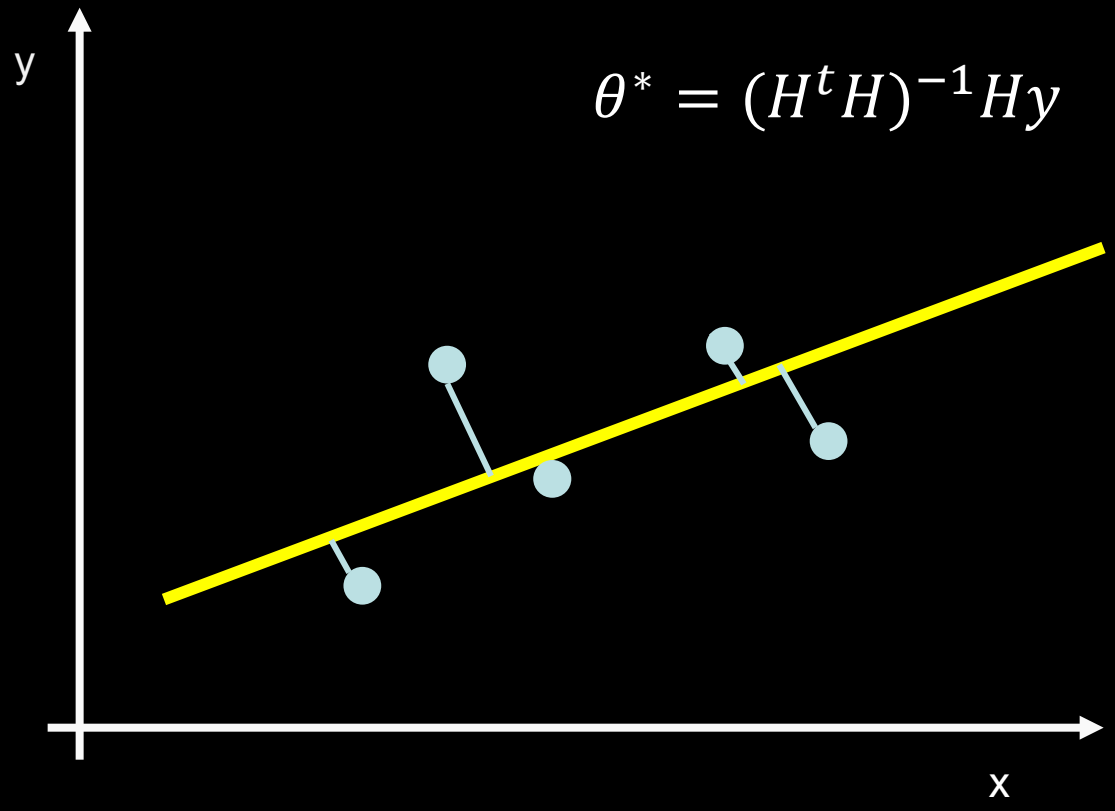
La forma più semplice di regressione è la regressione su una retta, stimare quindi coefficiente angolare e quota della retta

$$y = mx + c$$

$$\theta^* = (H^t H)^{-1} H y$$

y = variabile dipendente

x = variabile indipendente

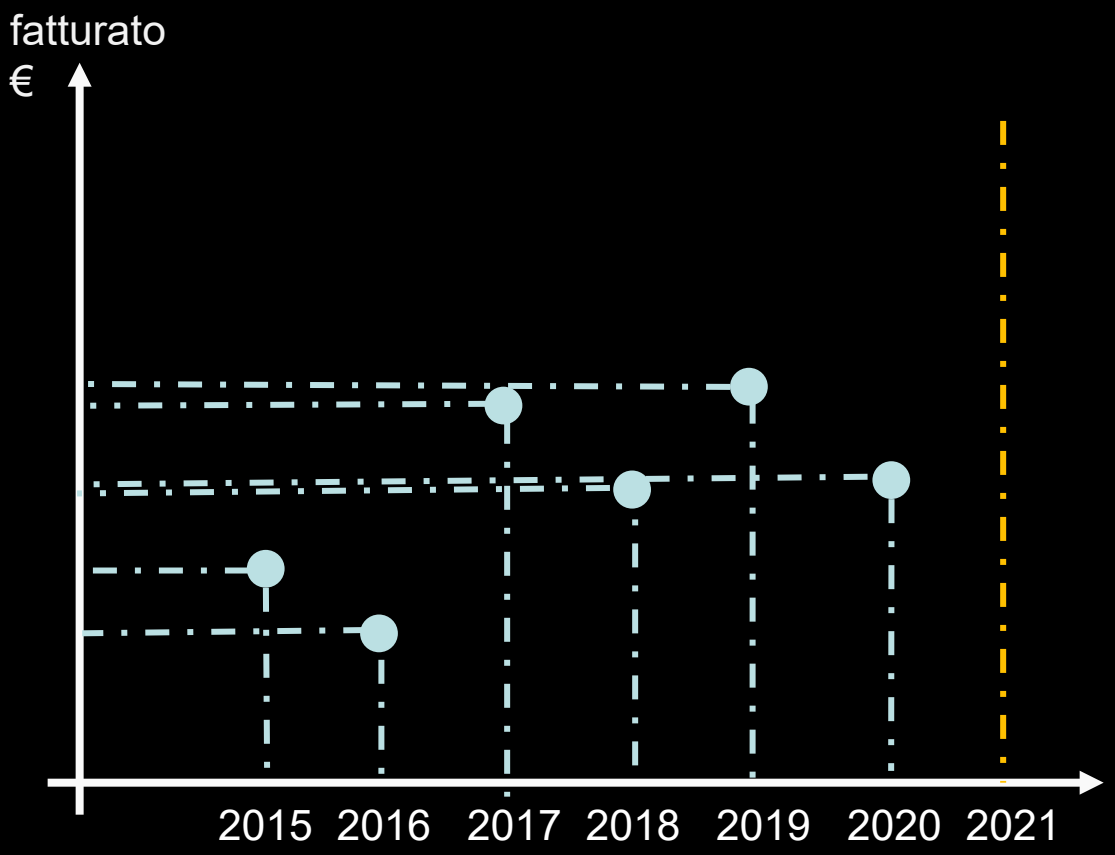


Regressione lineare

Supponiamo che i nostril dati siano anno – vendite, quindi ho il numero di vendite per ogni dato anno e voglio prevedere quale sarà la vendita nell’anno successivo

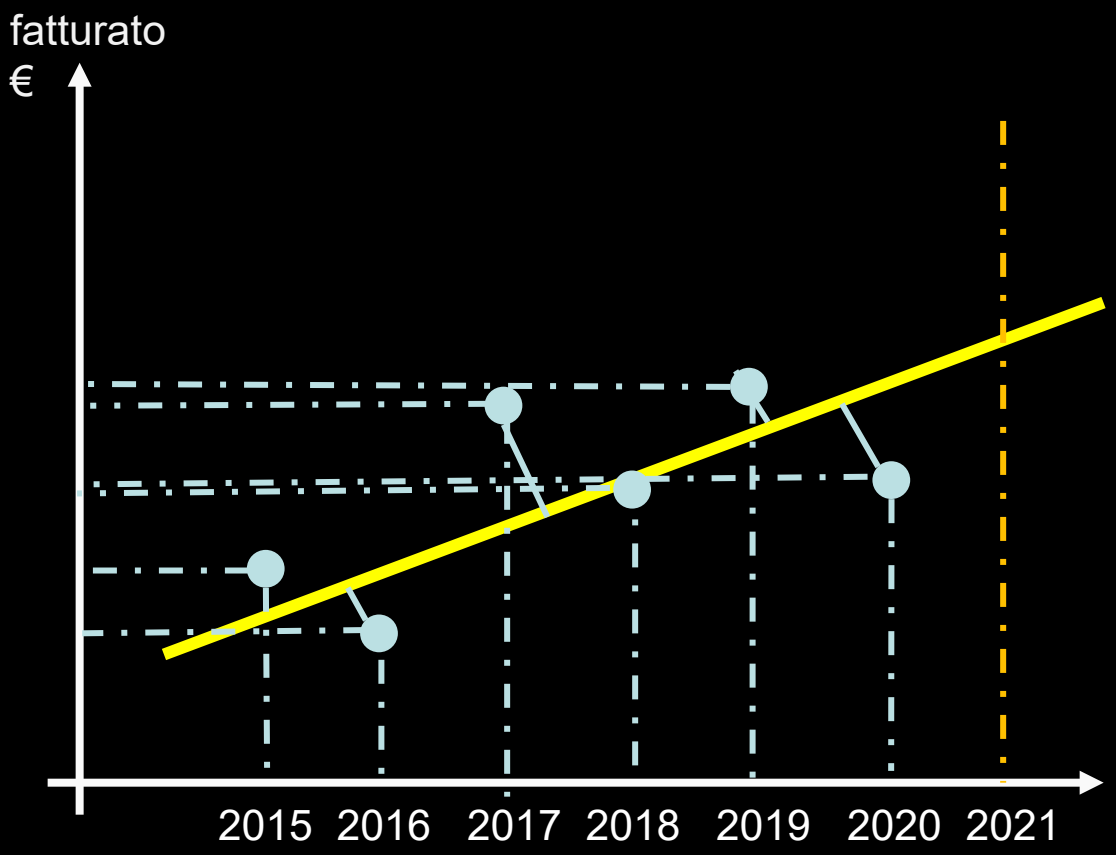
Regressione lineare

Supponiamo che i nostril dati siano anno – vendite, quindi ho il numero di vendite per ogni dato anno e voglio prevedere quale sarà la vendita nell'anno successivo



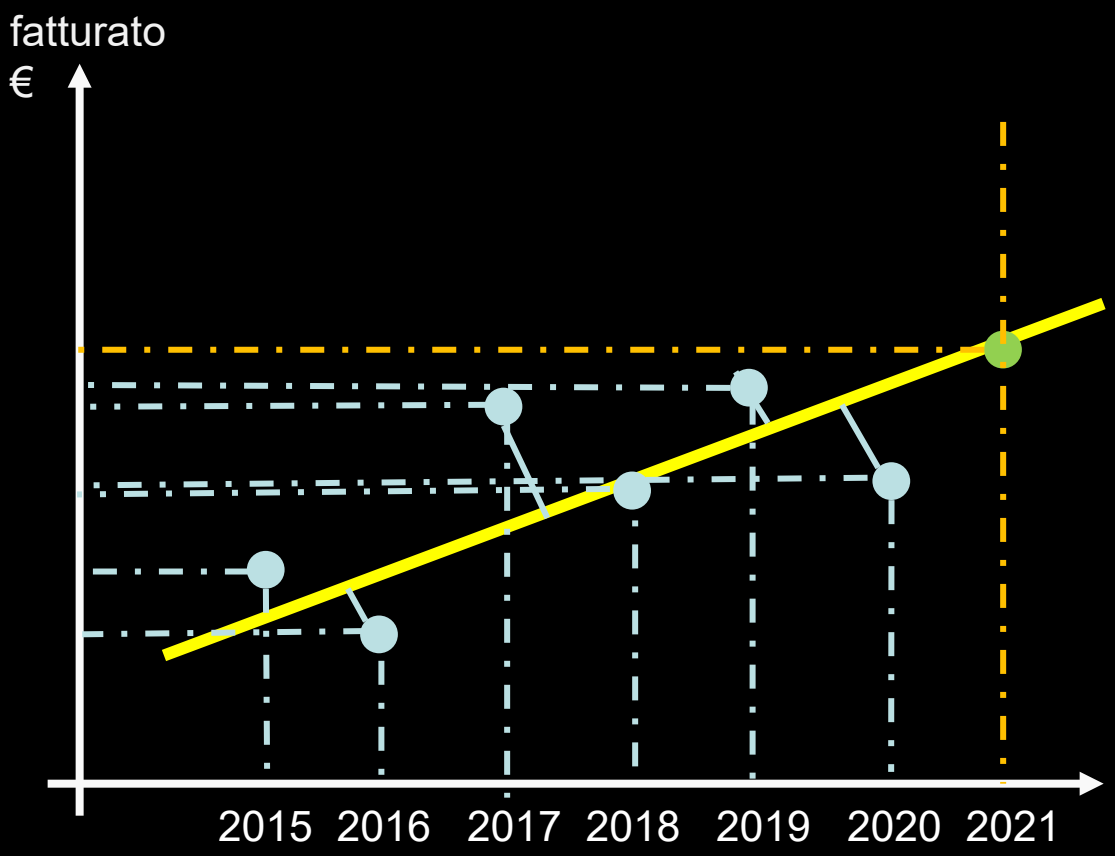
Regressione lineare

Supponiamo che i nostril dati siano anno – vendite, quindi ho il numero di vendite per ogni dato anno e voglio prevedere quale sarà la vendita nell'anno successivo



Regressione lineare

Supponiamo che i nostril dati siano anno – vendite, quindi ho il numero di vendite per ogni dato anno e voglio prevedere quale sarà la vendita nell'anno successivo



R^2 coefficiente di determinazione statistica

il coefficiente di determinazione statistica R^2 indica il legame tra i dati e la correttezza di un modello generato

$$R^2 = 1 - \frac{RSS}{TSS}$$

Dove:

TSS = devianza totale

$$RSS = \sum (x_i - E[x])^2$$

RSS = devianza residua

$$RSS = \sum (x_i - \widehat{x_i})^2$$

con x_i, $\widehat{x_i}$ rispettivamente le osservazioni e la stima del modello

R^2 coefficiente di determinazione statistica

il coefficiente di determinazione statistica R^2 indica il legame tra i dati e la correttezza di un modello generato

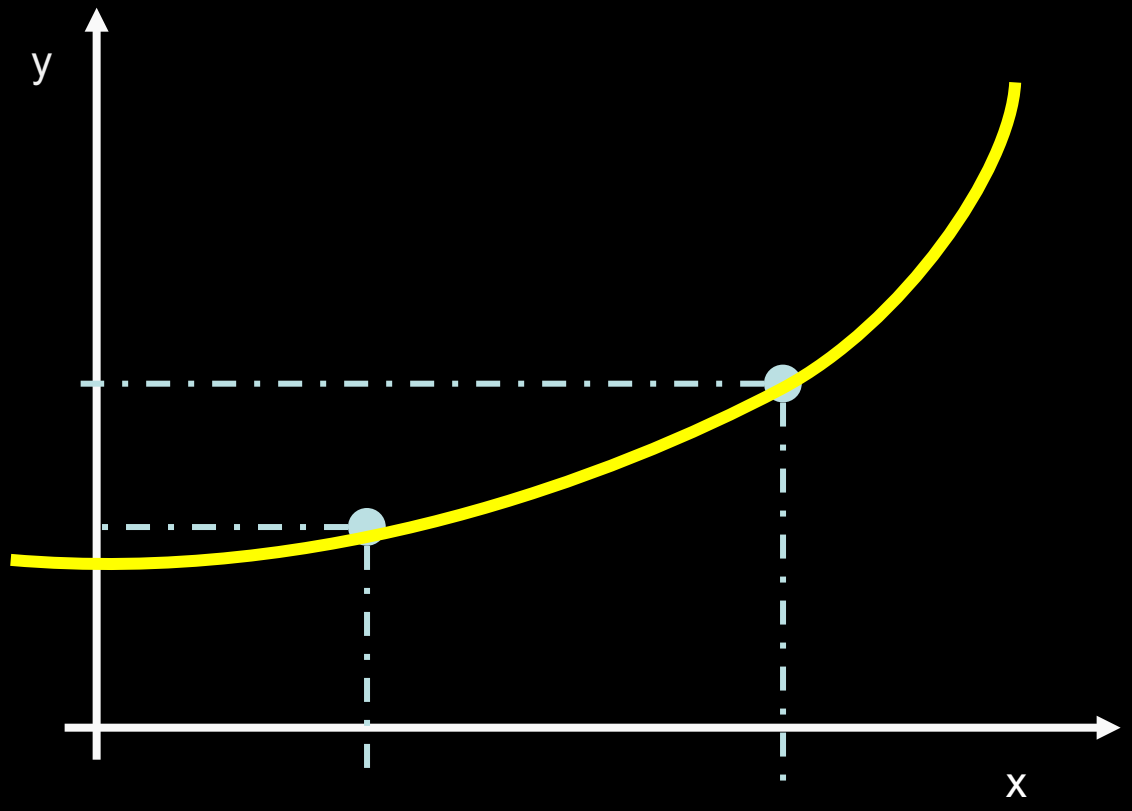
$$R^2 = 1 - \frac{RSS}{TSS}$$

- $R^2 \approx 1$ Significa che le previsioni del modello sono attendibili
- $R^2 \approx 0$ Significa che le previsioni del modello NON sono attendibili

Regressione polinomiale

Regressione su un polinomio di grado noto *n,* stimare i coefficienti a_n, …, a_1, a_o

$$y = a_n x^n + a_{n-1} x^{n-1} + \ ... + a_1 x + a_0$$



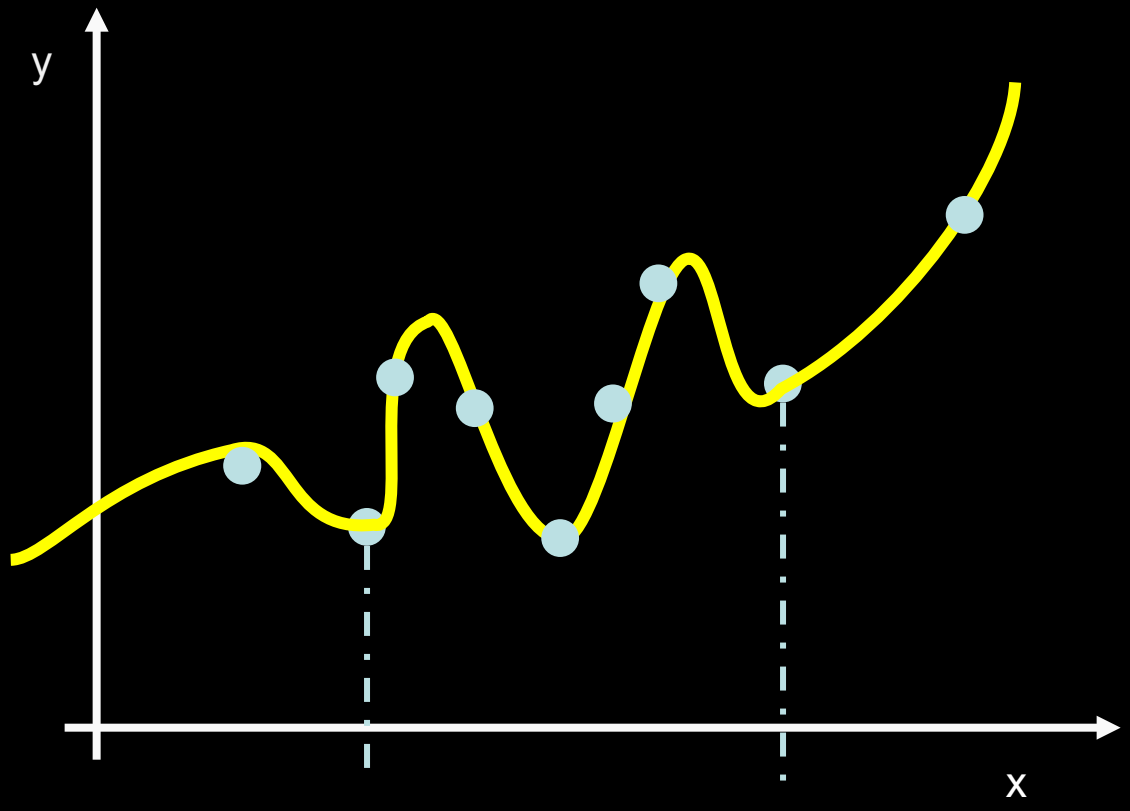
y = variabile dipendente

x = variabile indipendente

Regressione polinomiale

Regressione su un polinomio di grado noto *n,* stimare i coefficienti a_n, …, a_1, a_o

$$y = a_n x^n + a_{n-1} x^{n-1} + \ ... + a_1 x + a_0$$

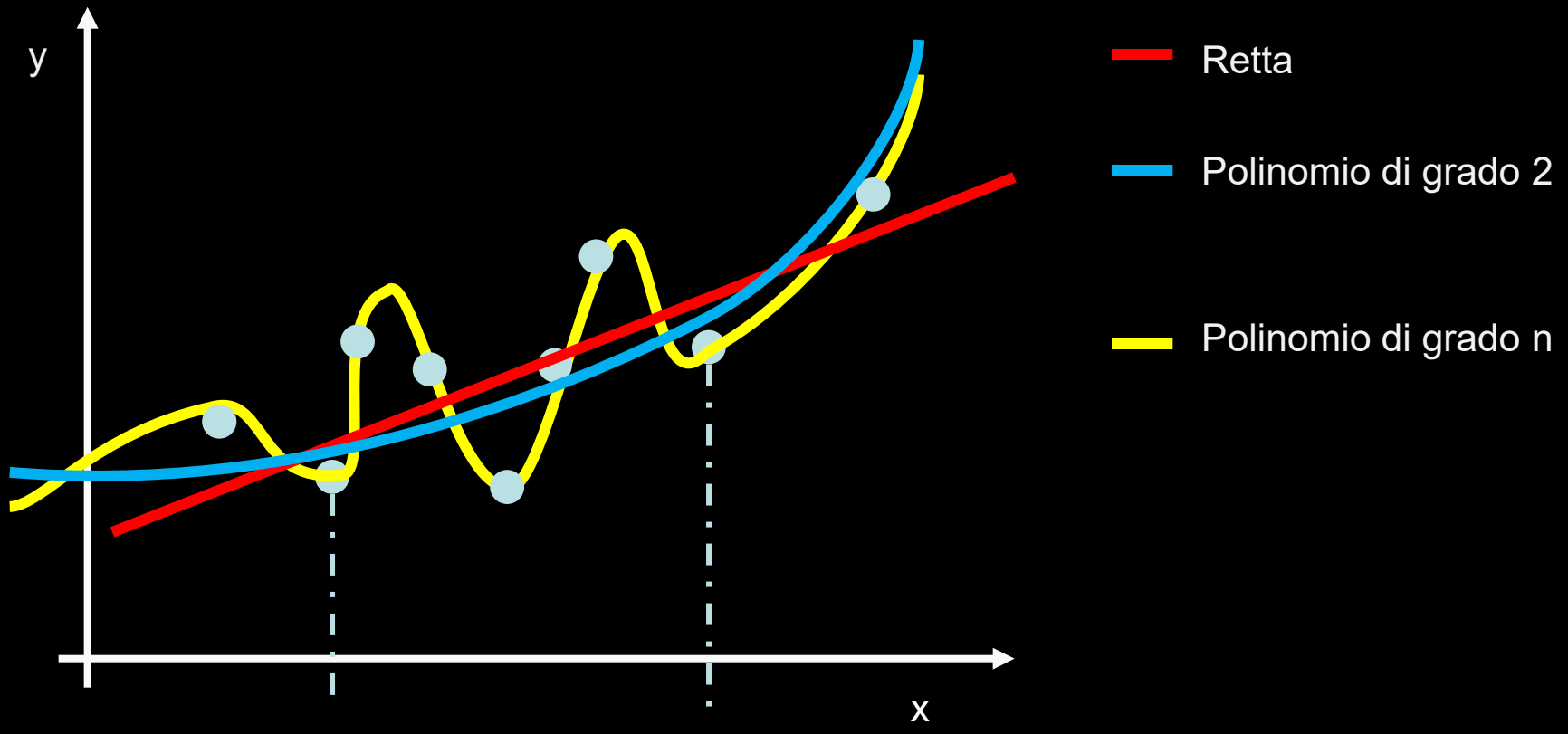


Aumentando il grado della polinomiale a piacere, è possibile approssimare i dati in maniera sempre più accurata

Regressione polinomiale

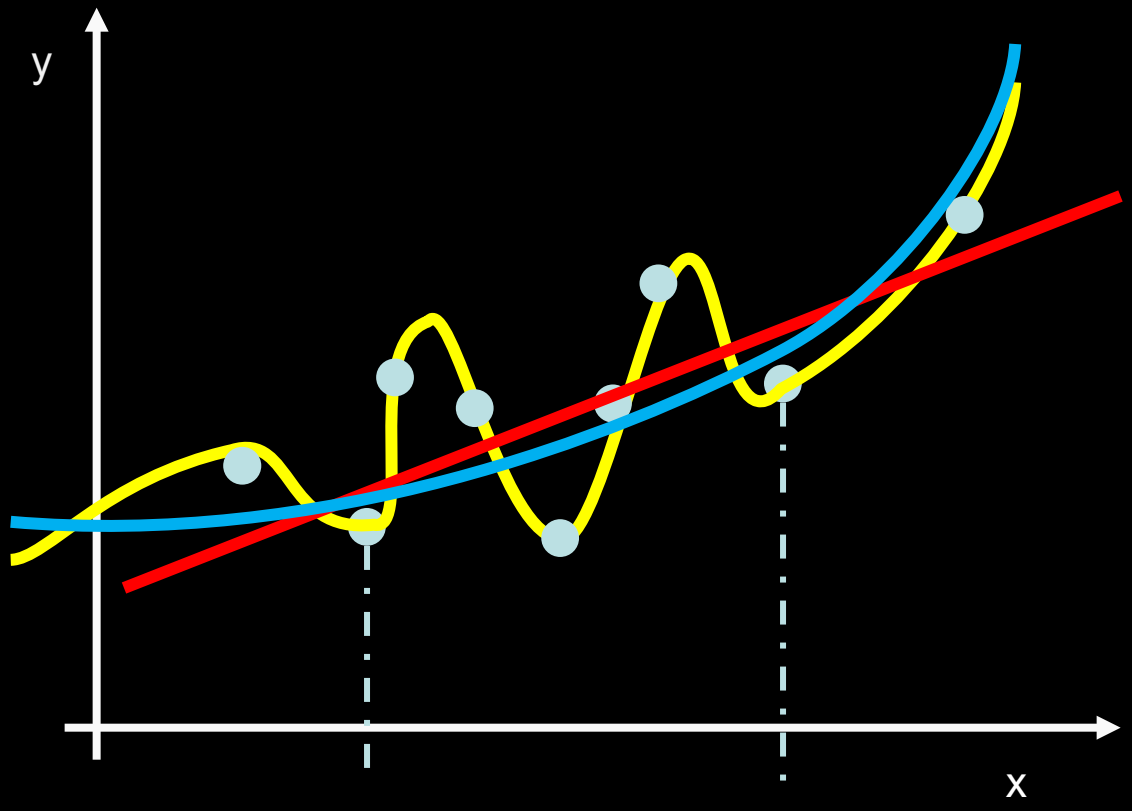
Regressione su un polinomio di grado noto *n,* stimare i coefficienti a_n, …, a_1, a_o

$$y = a_n x^n + a_{n-1} x^{n-1} + \ \ldots + a_1 x + a_0$$



Overfitting e underfitting

Aumentando a piacere il grado della polinomiale posso descrivere (fittare) meglio i dati, tuttavia questo non è sempre consigliabile



Retta (underfitting)

Polinomio di grado 2 (fitting giusto)

Polinomio di grado n (overfitting)

Overfitting e underfitting

Aumentando a piacere il grado della polinomiale posso descrivere (fittare) meglio i dati, tuttavia questo non è sempre consigliabile

In caso di underfitting:

- si aumenta il grado del modello

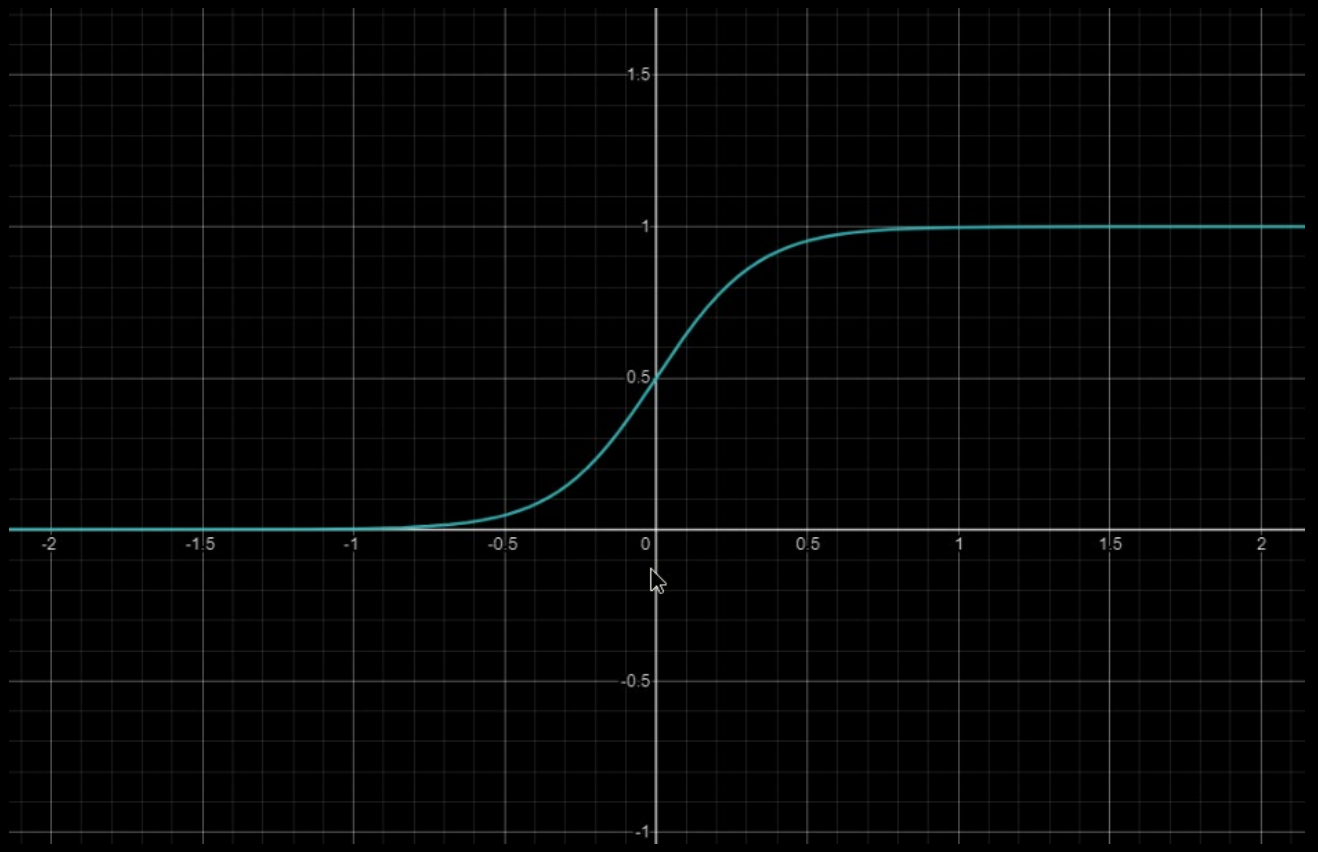
In caso di overfitting (problema più ricorrente, difficile da vedere e risolvere):

- si diminuisce il grado del modello
- si aumenta la dimensione del dataset (se possibile)

Regressione lineare e polinomiale

- ✓ Tutorial

Funzione sigmoide



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

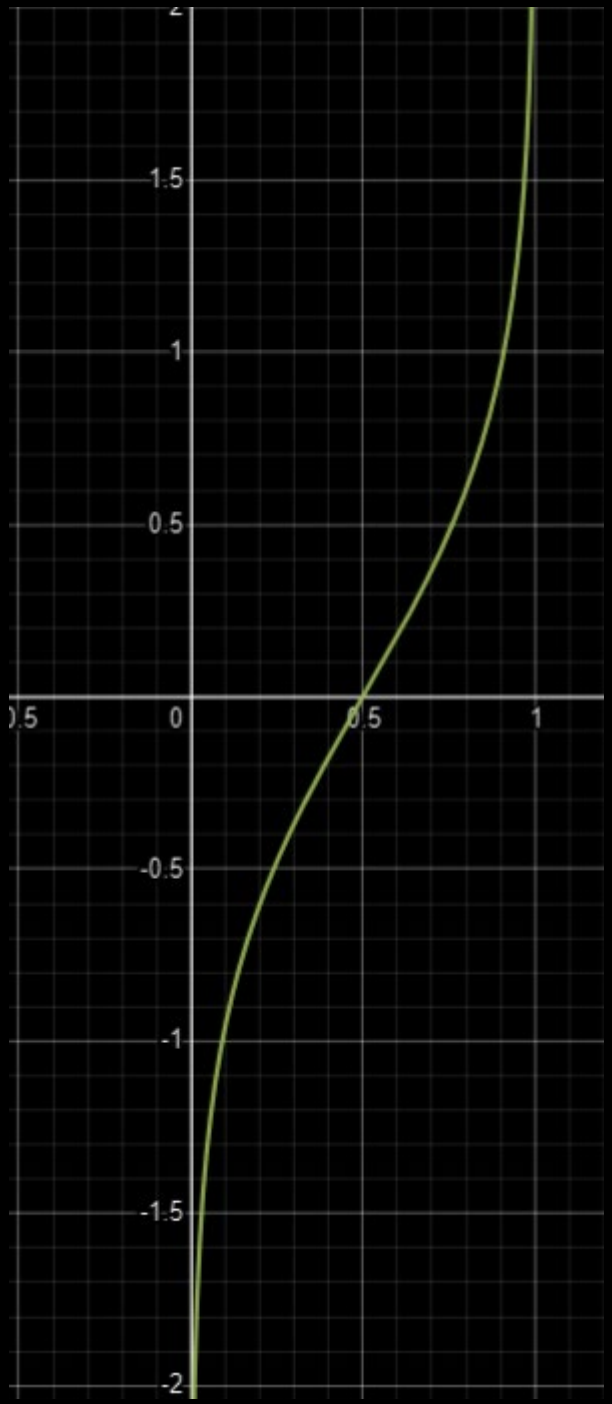
Check: https://www.desmos.com/calculator

Logit

$$f(p) = \log \frac{p}{1-p}$$

Il logit è una funzione logaritmica che mappa l'intervallo di probabilità [0,1] in [-inf,+inf]

E' la funzione inversa della sigmoide



Regressione logistica

Consideriamo una variabile indipendente *y* e un insieme di variabili dipendenti *x=(x_1,x_2, … x_r)* dove *r* è il numero di predittori (o input).

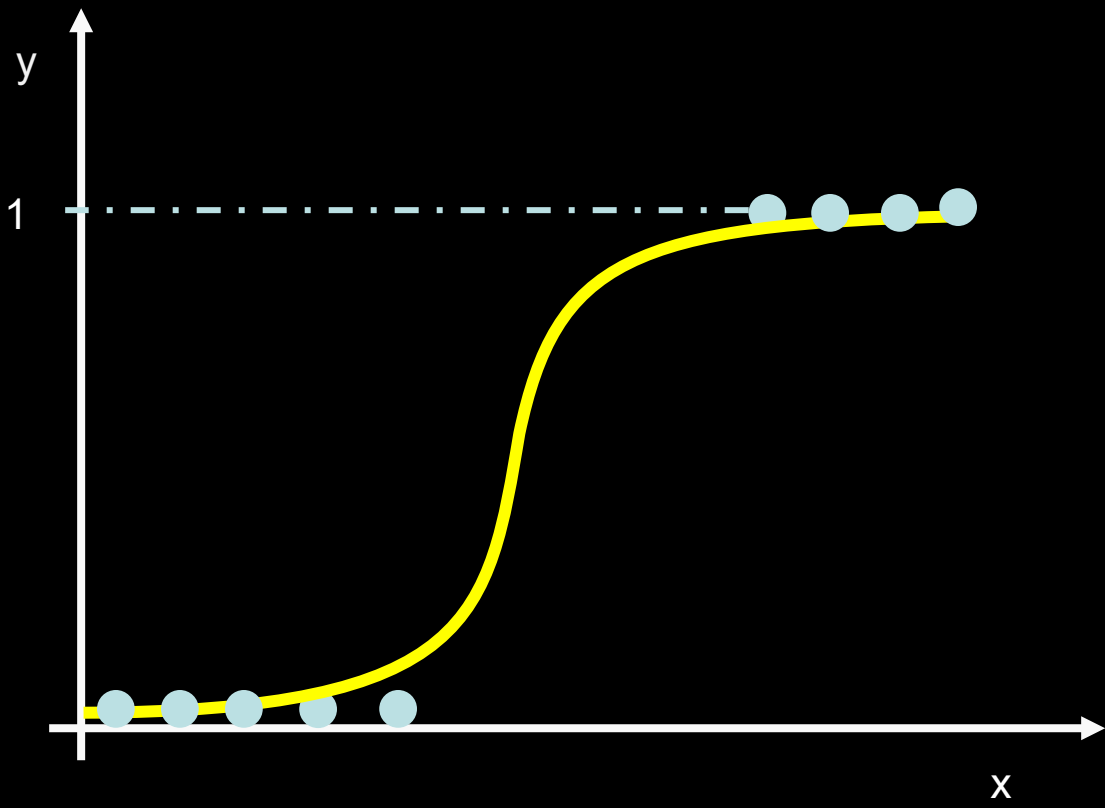
L'obiettivo è quello di implementare una funzione di regressione logistica tale che la predizione sia più vicina possibile alla risposta del sistema.

Regressione logistica

Consideriamo una variabile indipendente *y* e un insieme di variabili dipendenti *x=(x_1,x_2, … x_r)* dove *r* è il numero di predittori (o input).

L'obiettivo è quello di implementare una funzione di regressione logistica tale che la predizione sia più vicina possibile alla risposta del sistema.

è importante ricordare che l'uscita della funzione deve essere sempre binario (0 o 1)



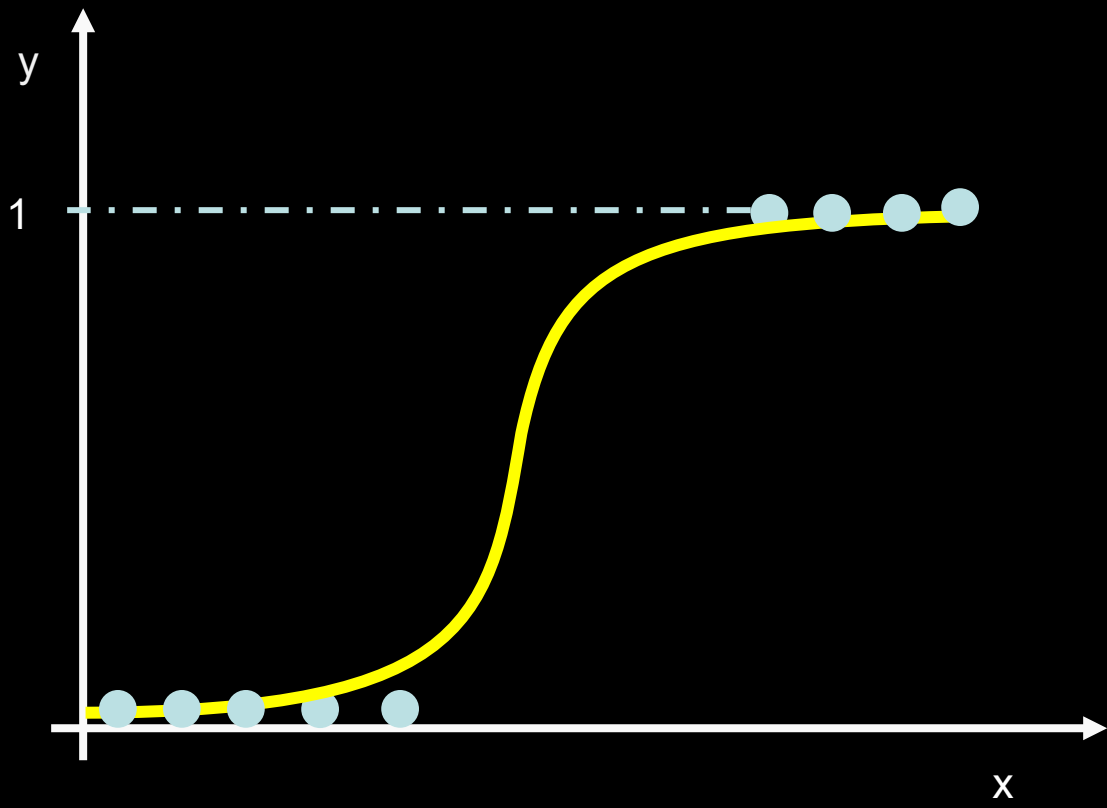
Regressione logistica

Consideriamo una variabile indipendente *y* e un insieme di variabili dipendenti *x=(x_1,x_2, … x_r)* dove *r* è il numero di predittori (o input).

L'obiettivo è quello di implementare una funzione di regressione logistica tale che la predizione sia più vicina possibile alla risposta del sistema.

è importante ricordare che l'uscita della funzione deve essere sempre binario (0 o 1)

Una volta nota questa funzione può essere usata per predire nuovi output



Regressione logistica

La regressione logistica è di fatto un classificatore lineare che usa una funzione lineare del tipo:

$$f(x) = a_o + a_1 x_1 + \cdots + a_n x_n$$

I parametri a_i sono detti stimatori

Regressione logistica

La regressione logistica è di fatto un classificatore lineare che usa una funzione lineare del tipo:

$$f(x) = a_o + a_1 x_1 + \cdots + a_n x_n$$

I parametri a_i sono detti stimatori.

La funzione di regressione logistica e una probabilità *p(x)* espressa come una sigmoide:

$$f(x): p(x) = \frac{1}{1 + e^{-f(x)}}$$

In tal modo la funzione è tipicamente vicina a 0 o 1 ed interpretata come la probabilità di appartenere alla classe o a quella contraria

Regressione logistica

Il problema è quindi quello di determinare i pesi a_i tali da fittare i nostri dati di input in funzione delle nostre osservazioni in oputput

Regressione logistica

Il problema è quindi quello di determinare i pesi a_i tali da fittare i nostri dati di input in funzione delle nostre osservazioni in oputput

Per farlo si usa sempre la risoluzione di un problema di ottimizzazione, con una funzione del tipo funzione di verosimiglianza logistica anche nota come LLF log-likelihood function

Regressione logistica

Il problema è quindi quello di determinare i pesi a_i tali da fittare i nostri dati di input in funzione delle nostre osservazioni in oputput

Per farlo si usa sempre la risoluzione di un problema di ottimizzazione, con una funzione del tipo funzione di verosimiglianza logistica anche nota come LLF log-likelihood function

e' chiaro che, per come è definita, la funzione avrà in uscita sempre un valore continuo, quindi tipicamente si inserisce una soglia a 0.5 tale per cui se il valore è > o < scelgo una classe o la sua opposta

Regressione lineare vs regressione logistica

Regressione lineare, stimo m e c tali che:

$$y = mx + c$$

Regressione logistica, stimo m e c tali che:

$$f(x) = \frac{1}{1 + e^{-(mx+c)}}$$

Regressione lineare vs regressione logistica

	Regressione lineare	Regressione logistica
Definizione	Predizione variabili continue	Predizione delle categorie
Tipo var	Variabile dipendente continua	Variabile dipendente categorica
Medodo di stima	Errore quadratico	Massima verosimiglianza
Equazione	Retta	Logaritmo della probabilità
Best fit	Retta	Sigmoide
Relazione	Richiesta una relazione lineare tra variabile dipendente e indipendente	La relazione lineare non è richiesta
Output	Valore reale predetto	Valore binario [0,1]
Applicazioni	previsioni	classificazione

Regressione logistica

Tutorial

Clusterizzazione

La clusterizzazione è il processo di divisione di un dataset in sottogruppi che presentano una similarità

Tale che

- punti nello stesso gruppo siano più simili possibile (massima verosimiglianza)

- punti in gruppi diversi sono più DISimili possible (minima verosimiglianza)

Clusterizzazione

La separazione dei dati deve essere:

- Significativa, quindi deve espandere la conoscenza in un certo dominio, es. es. identificazione gruppi di pazienti per risposta ad un farmaco

- Utile, far parte di un processo, es. raggruppamento di clienti per campagne pubblicitarie dedicate

Tipologie di clusterizzazione

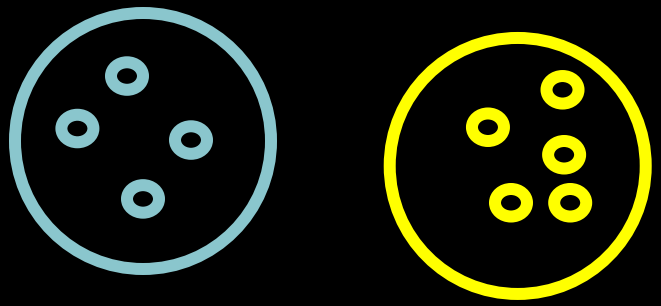
- Clusterizzazione esclusiva o ripartiva (partitional or exclusive clustering)
- Clusterizzazione sovrapponente (Overlapping clustering)
- Clusterizzazione gerarchica

Clusterizzazione esclusiva

Si dividono oggetti in insiemi esclusivi (no sovrapposizioni), queste tecniche richiedono l'indicazione del numero *k* di clusters nel quale dividere i dati

I punti possono appartenere solo ad un cluster

es. K-means clustering

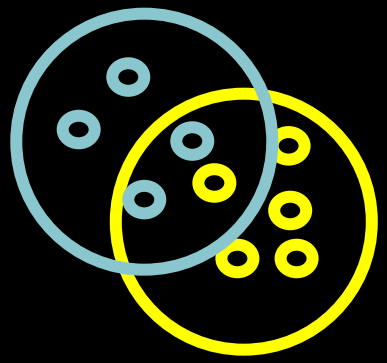


Overlapping clustering

I gruppi vengono divisi in insiemi NON esclusivi, tipicamente i bordi si sovrappongono (soft clustering o density clustering).

La probabilità di appartenere ad una data classe è massima nel centro della classe.

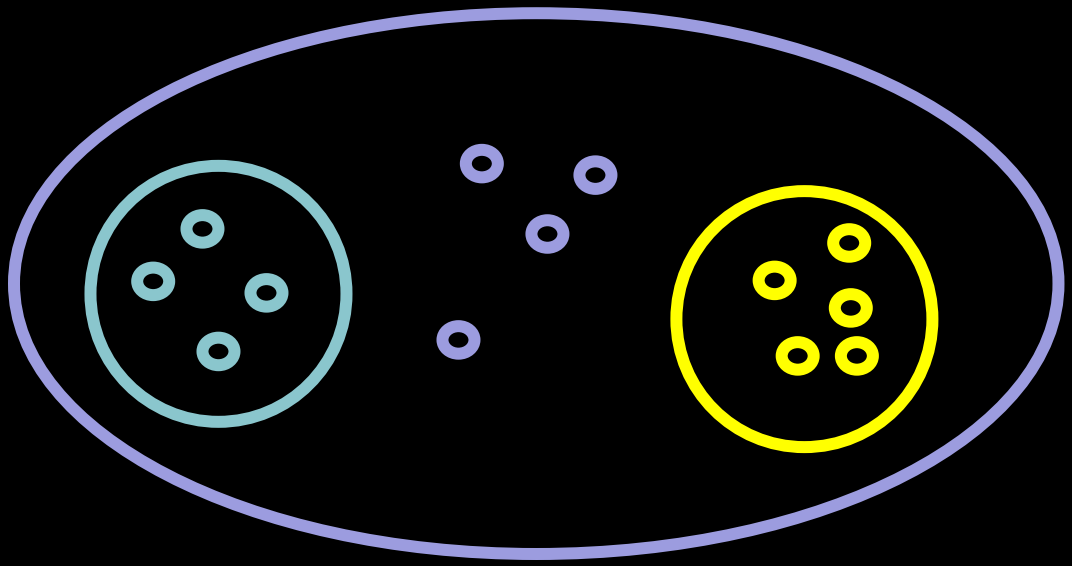
es. Fuzzy/C-means clustering



Clusterizzazione gerarchica

Gruppi di dati vengono raggruppati per componenti esclusive e raggruppati per altre componenti al fine di creare insiemi e sottoinsiemi (modalità albero)

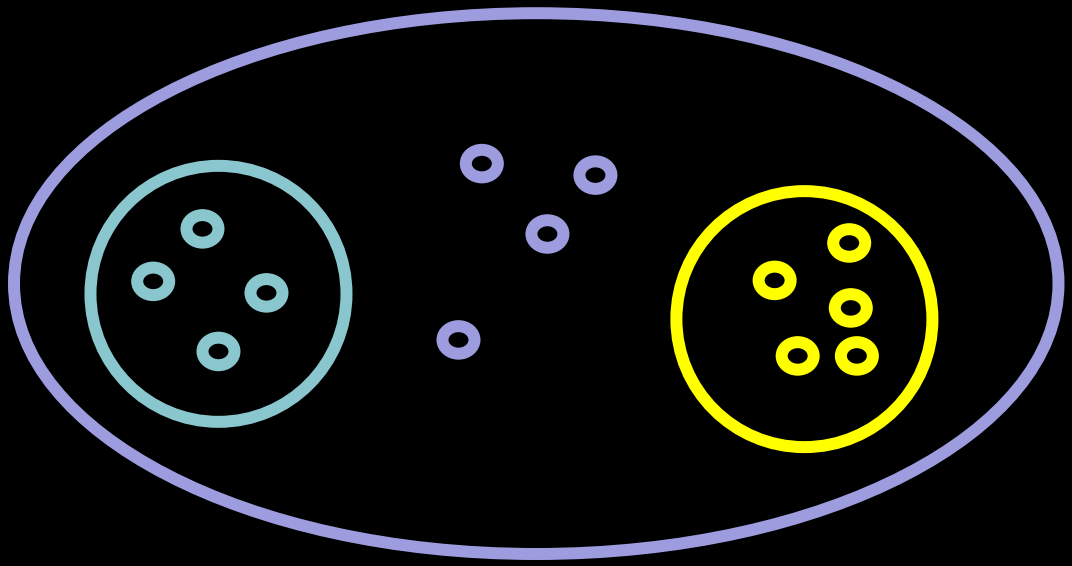
Es. Classificare vestiario come, magliette, pantaloni, camicie (base) – vestiti rossi, bianchi, neri (livello superiore)



Clusterizzazione gerarchica

Gruppi di dati vengono raggruppati per componenti esclusive e raggruppati per altre componenti al fine di creare insiemi e sottoinsiemi (modalità albero)

- ✓ Gli insiemi possono sono esclusivi nella profondità dell'albero (clustering divisivo)
- ✓ Gli insiemi possono essere inclusivi nella profondità dell'albero (clustering agglomerativo)



k-means clustering

E’ un algoritmo di clusterizzazione finalizzato a raggruppare elementi dello stesso tipo in *k* gruppi

k-means clustering

E’ un algoritmo di clusterizzazione finalizzato a raggruppare elementi dello stesso tipo in *k* gruppi

Algoritmo – k-means clustering

1 – selezionare il numero di clusters da identificare *k*

2 – Selezionare *k* punti random (centroidi del cluster)

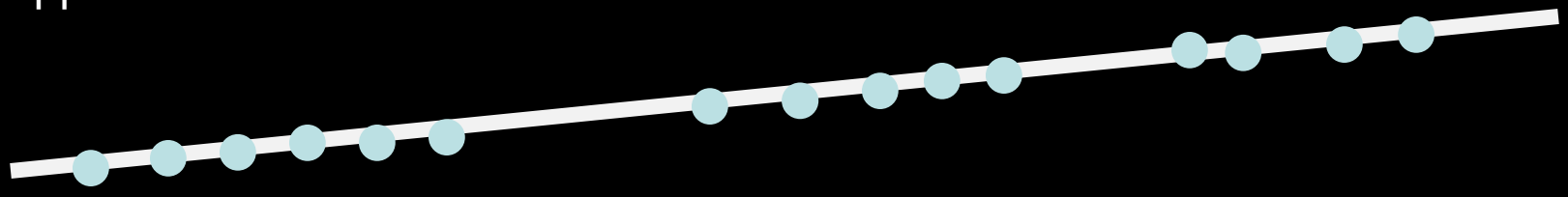
while (i centroidi cambiano)

- 3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters
- 4 – assegnare il primo punto al centroide più vicino
- 5 – calcolare la media tra il primo punto non assegnato e il centroide assegnato (questo punto sarà il nuovo centroide)

Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare ***k = 3***

Dati una serie di punti distribuiti lungo una retta, vogliamo determinare 3 gruppi

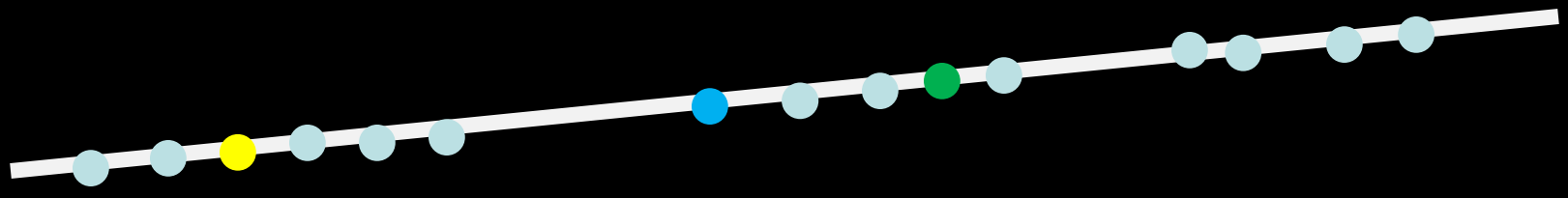


Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k = 3*

2 – Selezionare *k* punti random (centroidi del cluster)

I primi 3 punti random saranno i primi centroidi delle classi da individuare, quindi gli assegno 3 colori diversi



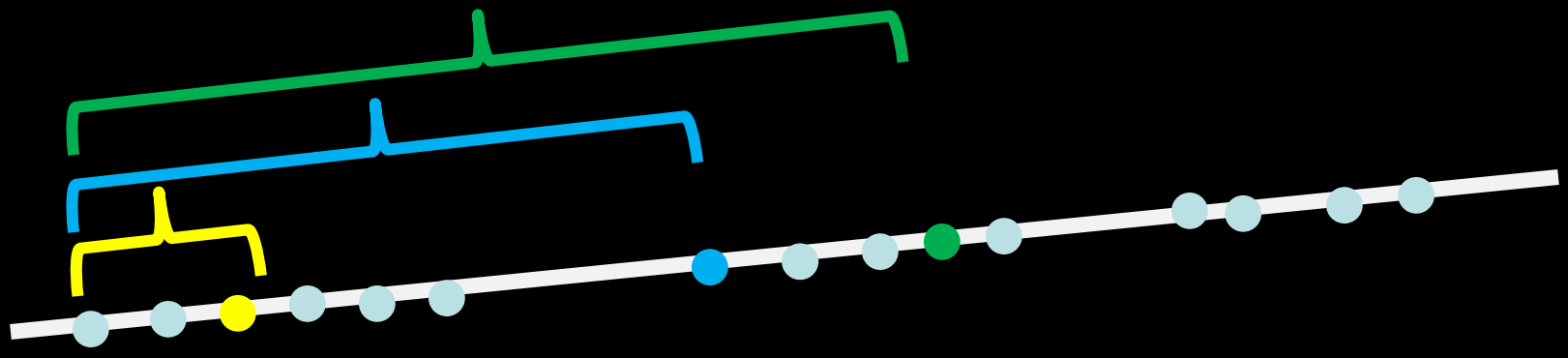
Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters



Algoritmo k-means clustering - Esempio

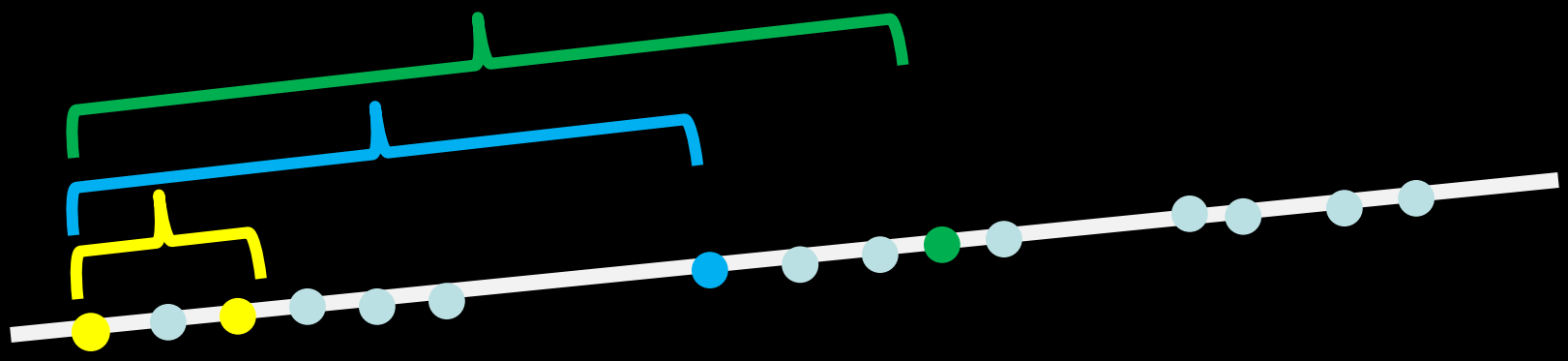
1 – selezionare il numero di clusters da identificare *k*

2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino



Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

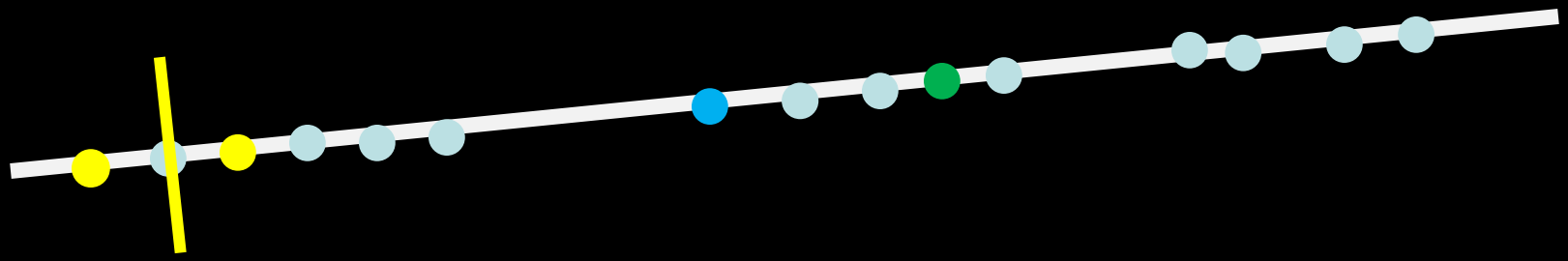
2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino

5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

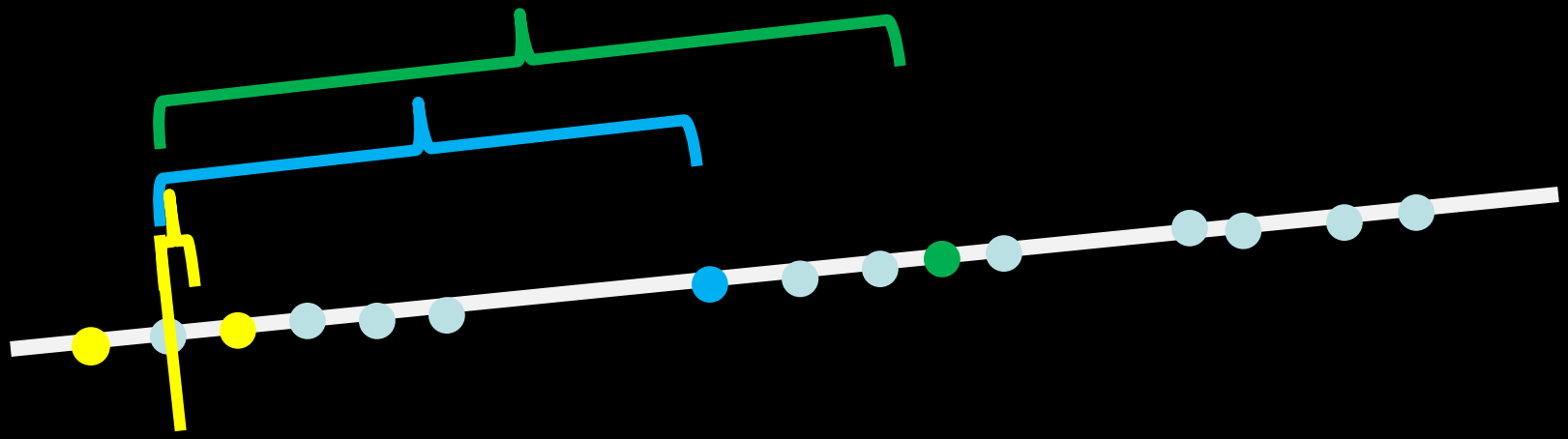
2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino

5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

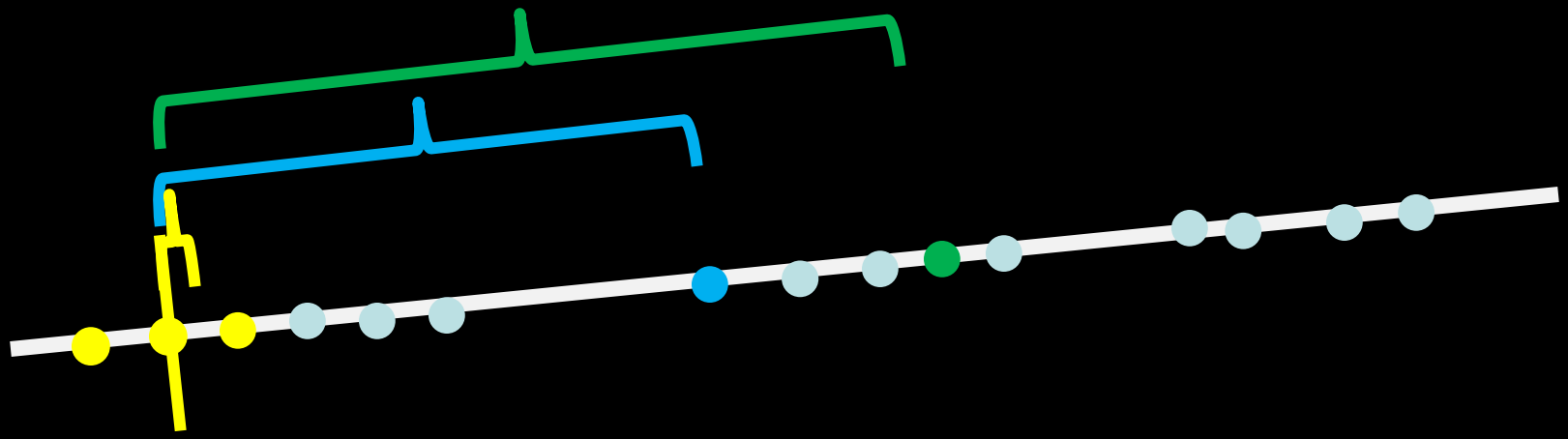
2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino

5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

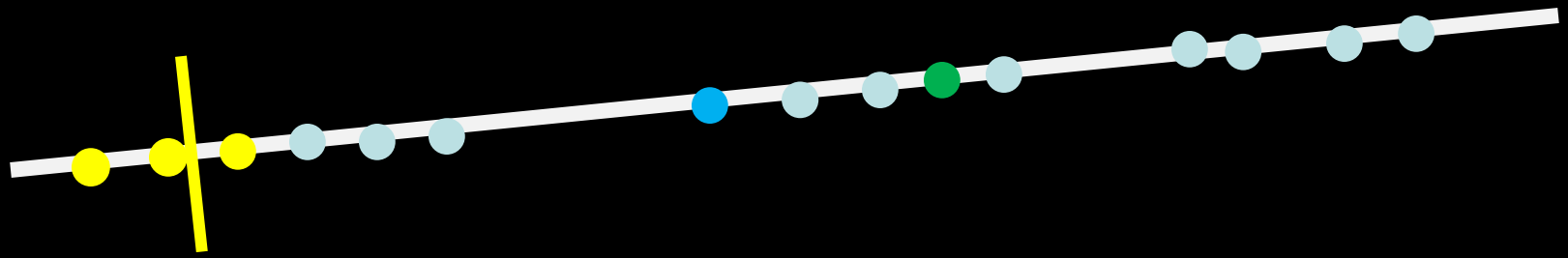
2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino

5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

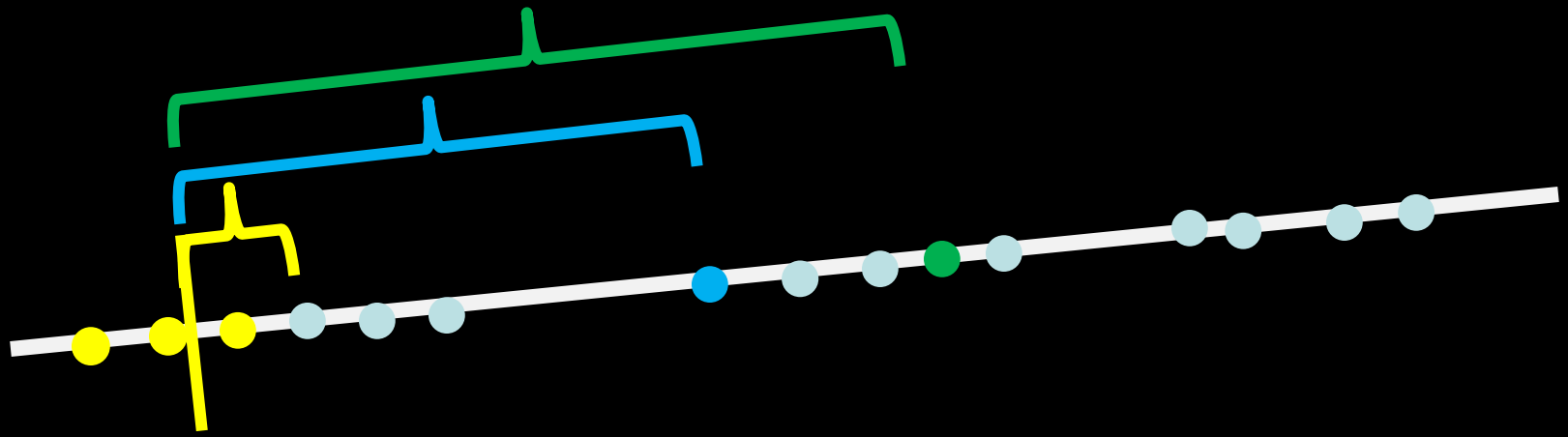
2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino

5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

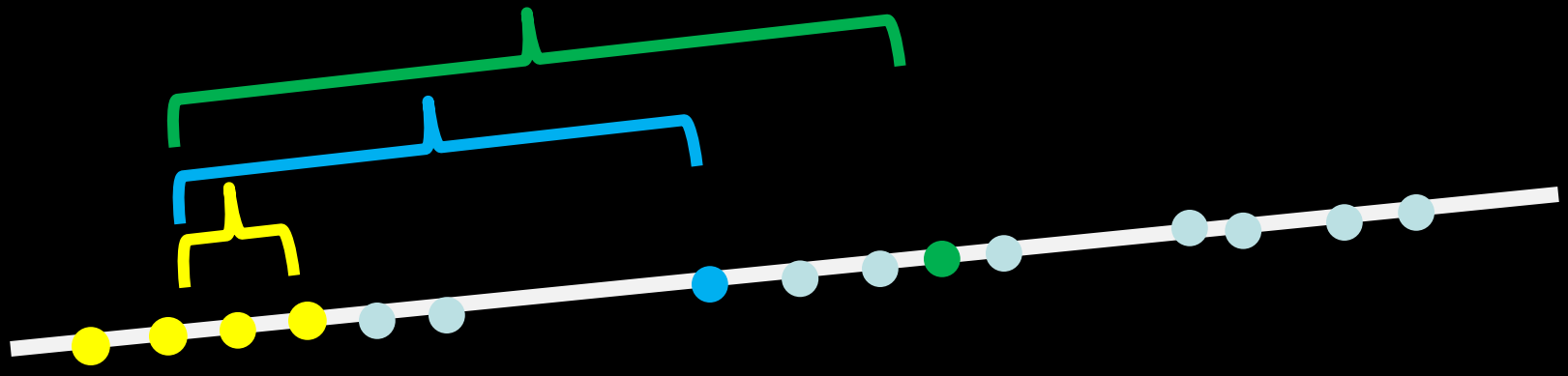
2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino

5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

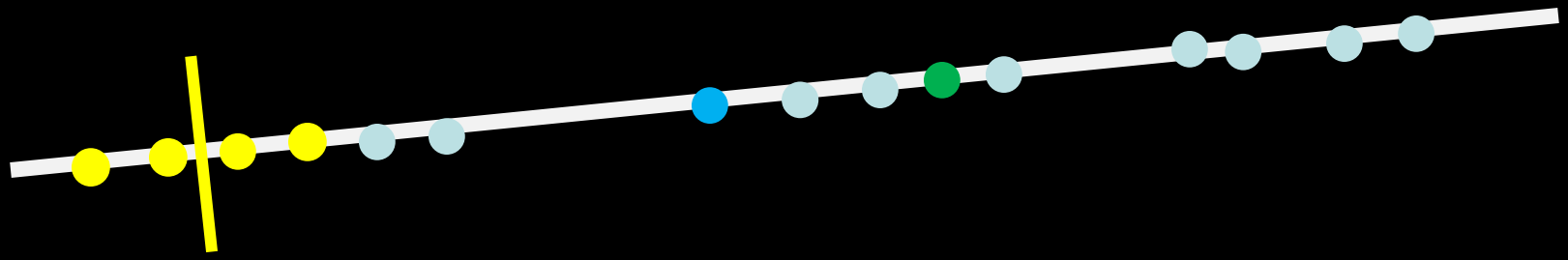
2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino

5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

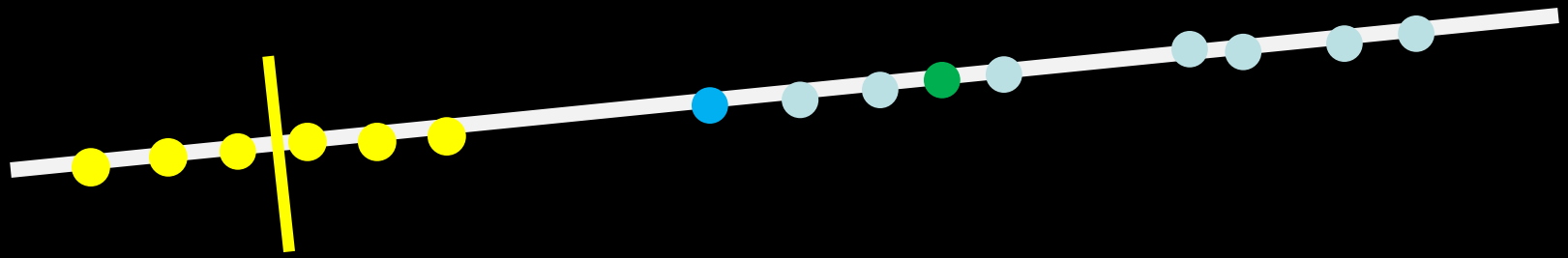
2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino

5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



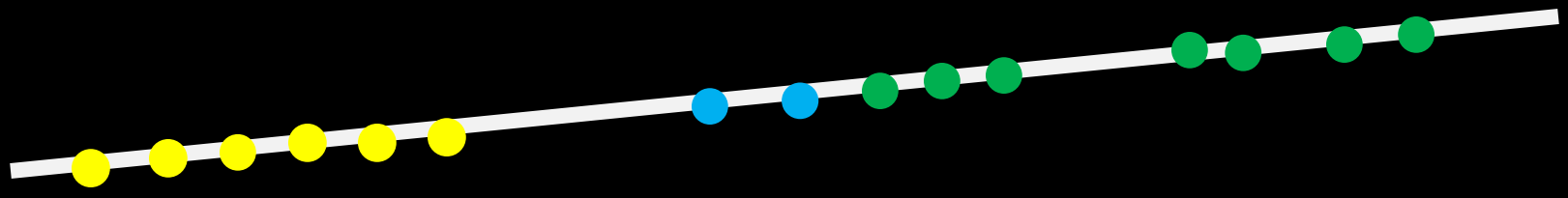
Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare *k*

2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

- 3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters
- 4 – assegnare il primo punto al centroide più vicino
- 5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



Algoritmo k-means clustering - Esempio

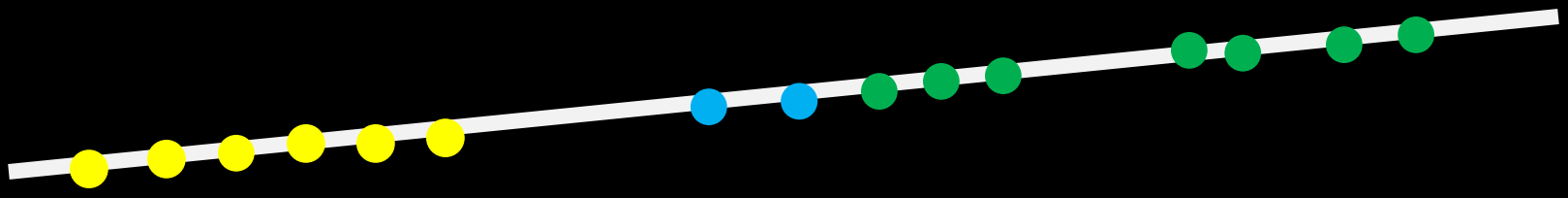
1 – selezionare il numero di clusters da identificare *k*

2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

- 3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters
- 4 – assegnare il primo punto al centroide più vicino
- 5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)

NON E' UNA DIVISIONE OTTIMA



Algoritmo k-means clustering - Esempio

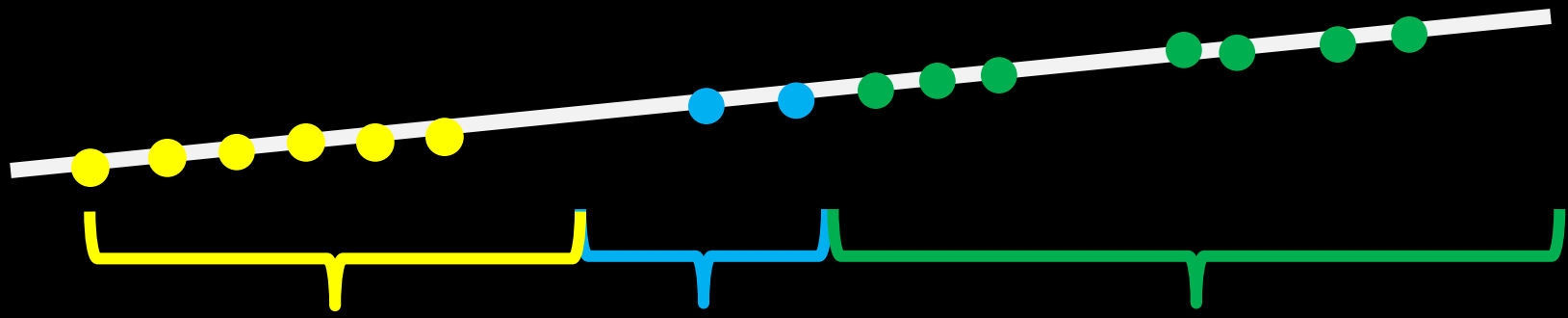
1 – selezionare il numero di clusters da identificare *k*

2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

- 3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters
- 4 – assegnare il primo punto al centroide più vicino
- 5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)

Si misura la varianza delle distribuzioni nelle classi con l'obiettivo di minimizzare la variazianza



Algoritmo k-means clustering - Esempio

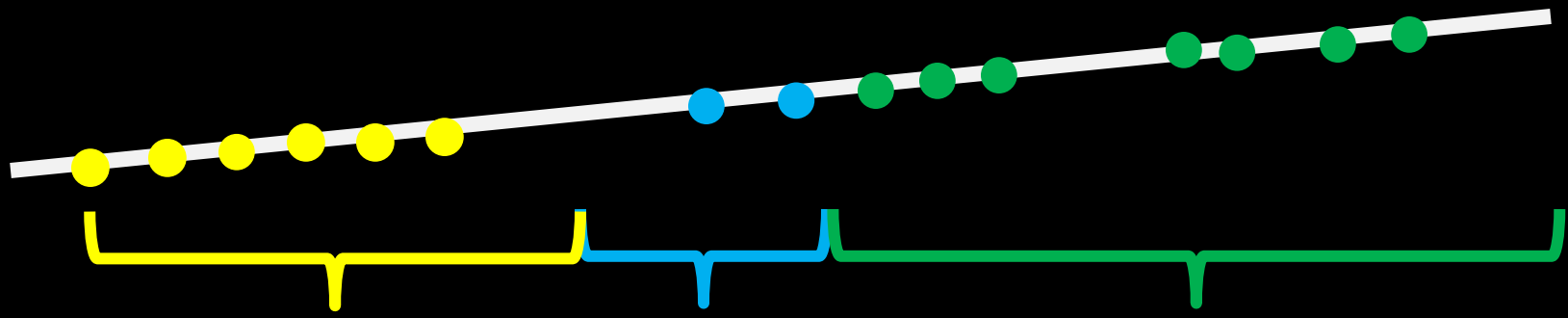
1 – selezionare il numero di clusters da identificare *k*

2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

- 3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters
- 4 – assegnare il primo punto al centroide più vicino
- 5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)

Reiteriamo l'algoritmo scegliendo dei punti random iniziali diversi



Algoritmo k-means clustering - Esempio

1 – selezionare il numero di clusters da identificare k

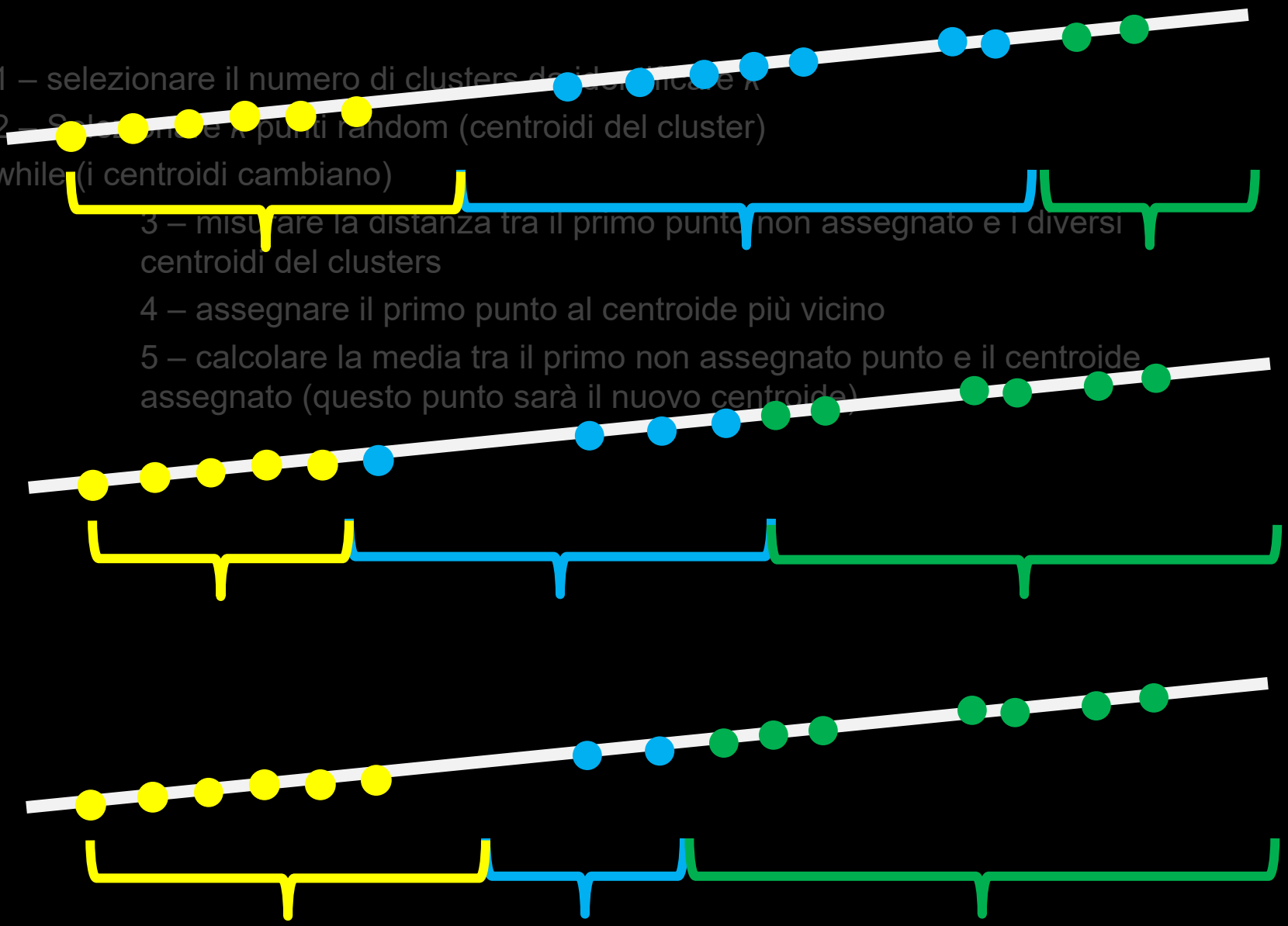
2 – Selezionare k punti random (centroidi del cluster)

while (i centroidi cambiano)

3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters

4 – assegnare il primo punto al centroide più vicino

5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)



Algoritmo k-means clustering - Esempio

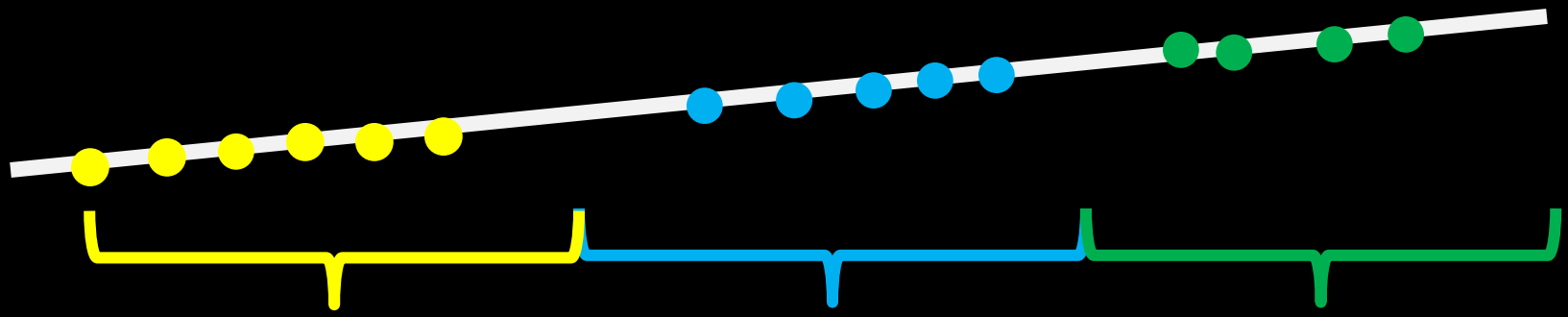
1 – selezionare il numero di clusters da identificare *k*

2 – Selezionare *k* punti random (centroidi del cluster)

while (i centroidi cambiano)

- 3 – misurare la distanza tra il primo punto non assegnato e i diversi centroidi del clusters
- 4 – assegnare il primo punto al centroide più vicino
- 5 – calcolare la media tra il primo non assegnato punto e il centroide assegnato (questo punto sarà il nuovo centroide)

Al crescere del numero di iterazioni si trova la soluzione a varianza minima



k-means clustering

Tutorial

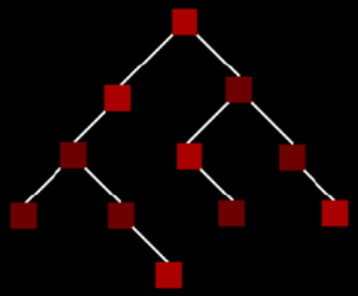
Confronto

	Alberi decisionali	Regole associative	k-NN	Reti neurali
Regressione	SI	NO	SI	SI
Classificazione	SI	SI	SI	SI
Astrazione	Bassa	Bassa	Media	Alta
Robustezza	Bassa	Bassa	Media	Alta
Complessità	Bassa	Bassa	Media	Alta
Comprensione	Alta	Alta	Media	Bassa
Debugging	Bassa	Bassa	Media	Impossibile

Libraries ecosystem

TensorFlow

Keras



mlpack