

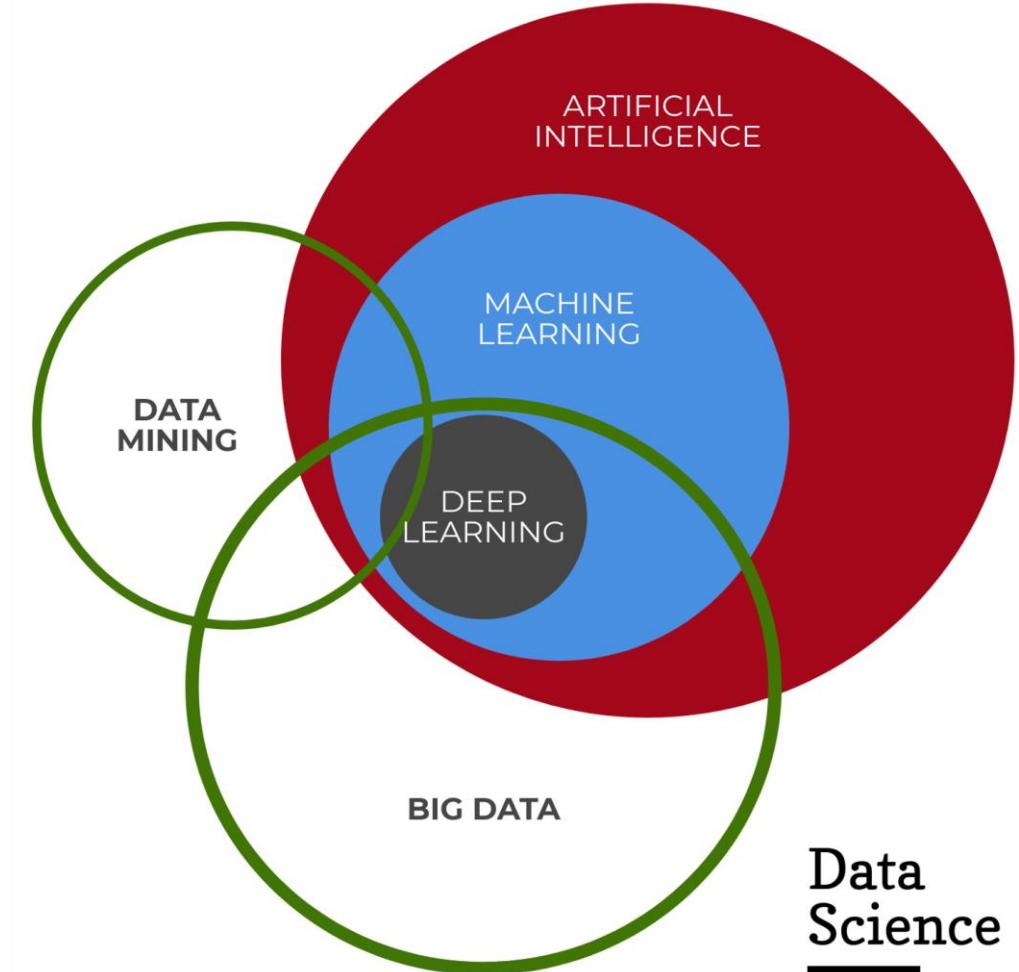
Machine learning e modelli di classificazione di dati biomedici

Christian Salvatore
Scuola Universitaria Superiore IUSS Pavia

Machine learning

Machine learning is the subfield
of **computer science** that gives
**"computers the ability to learn
without being explicitly programmed"**

Arthur Samuel, 1959



How is ML different from classical statistics?

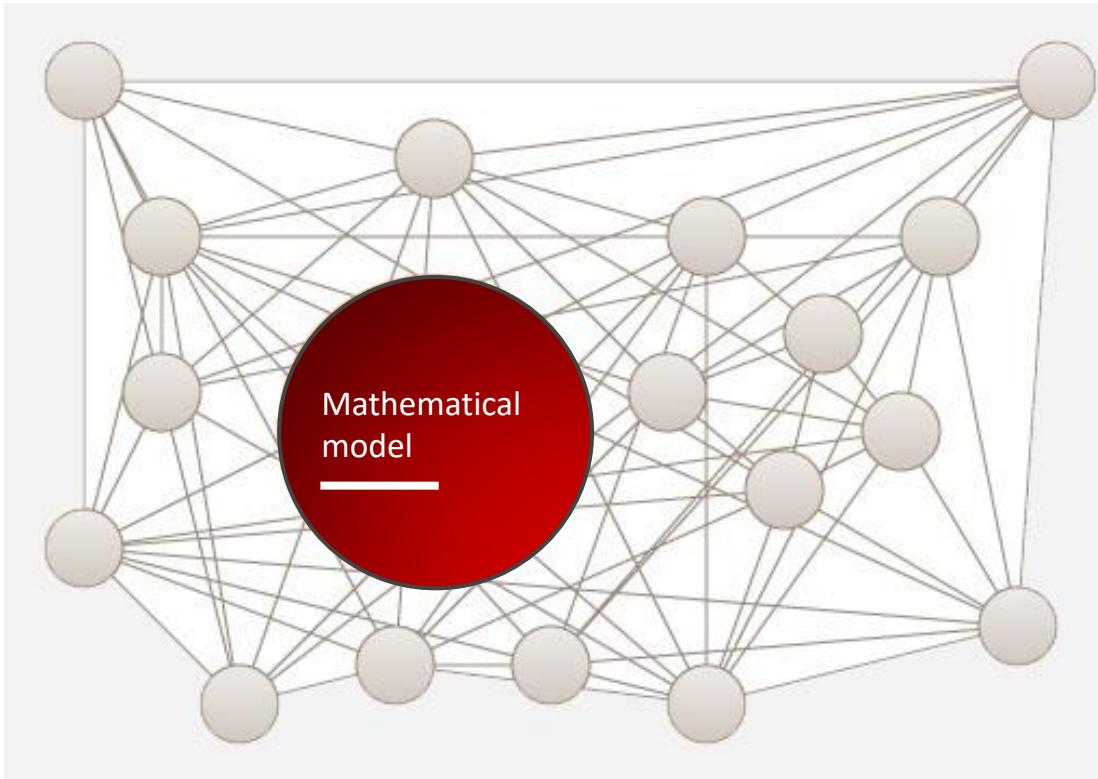
*It is focused on
classification rather than
inference*

Distribution-free approach

*High-dimensional
problems*

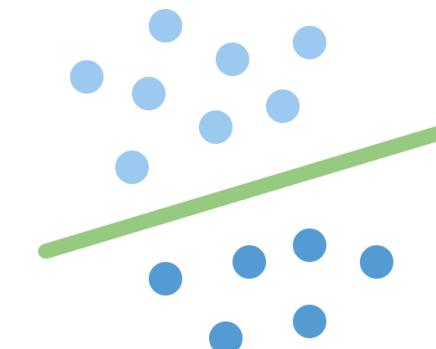
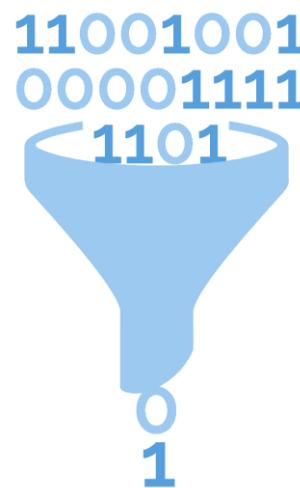
*Algorithmic and
computational aspects play a
central role*

Machine learning



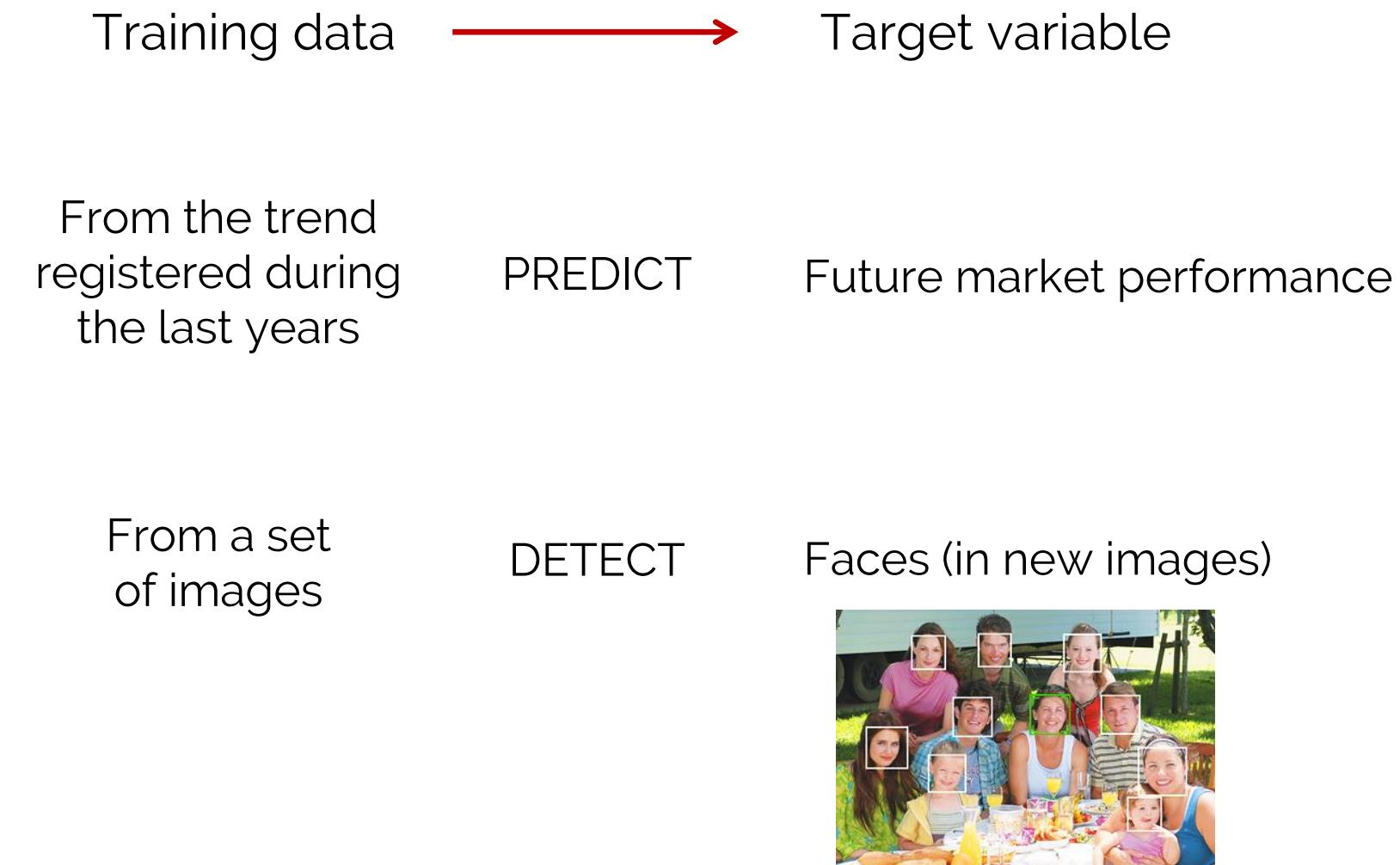
Designing mathematical models able to

Machine learning

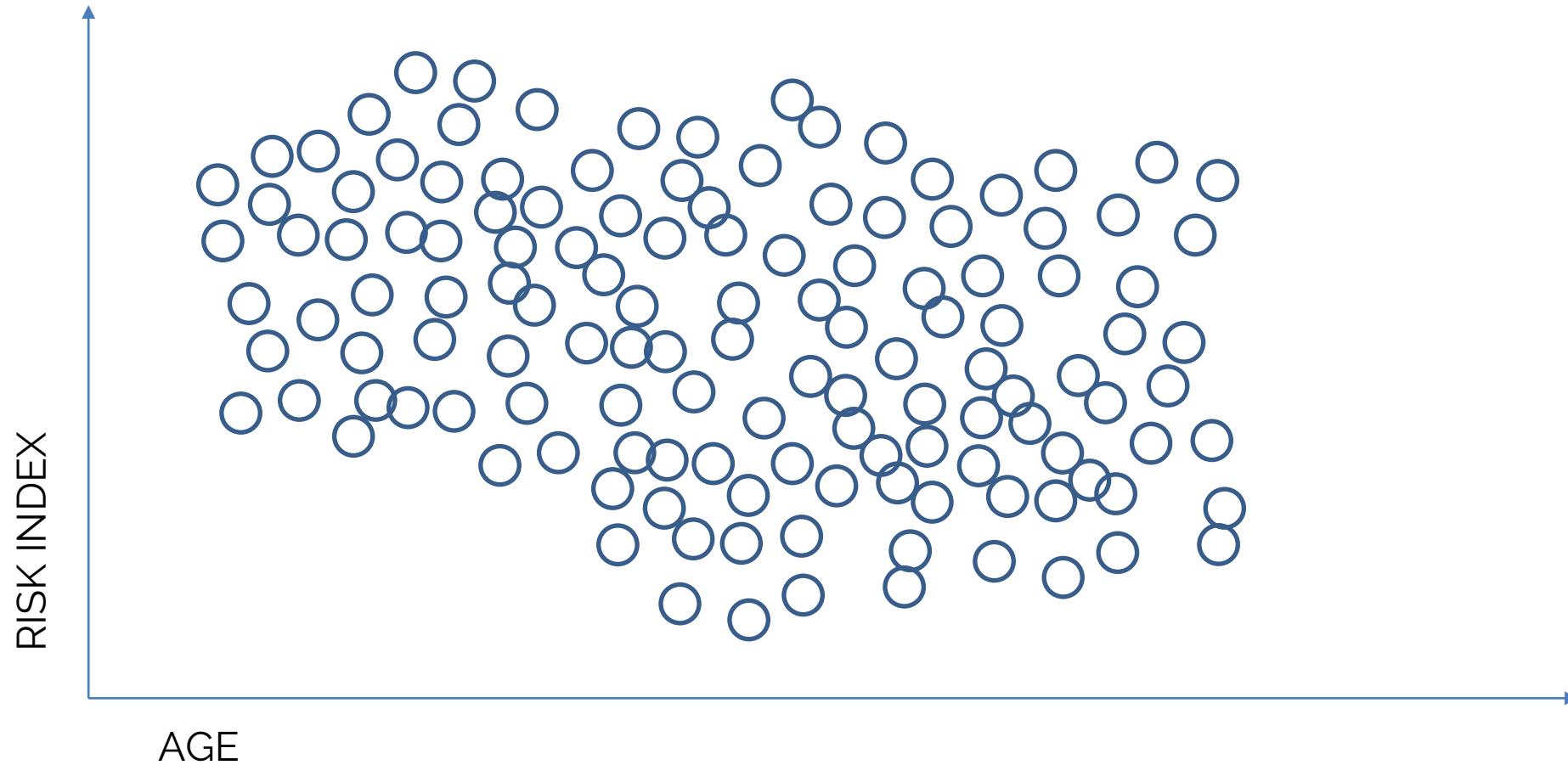


1. identify hidden patterns in data
2. handle and summarize a great quantity of data into a model
3. use that model to perform automatic classification

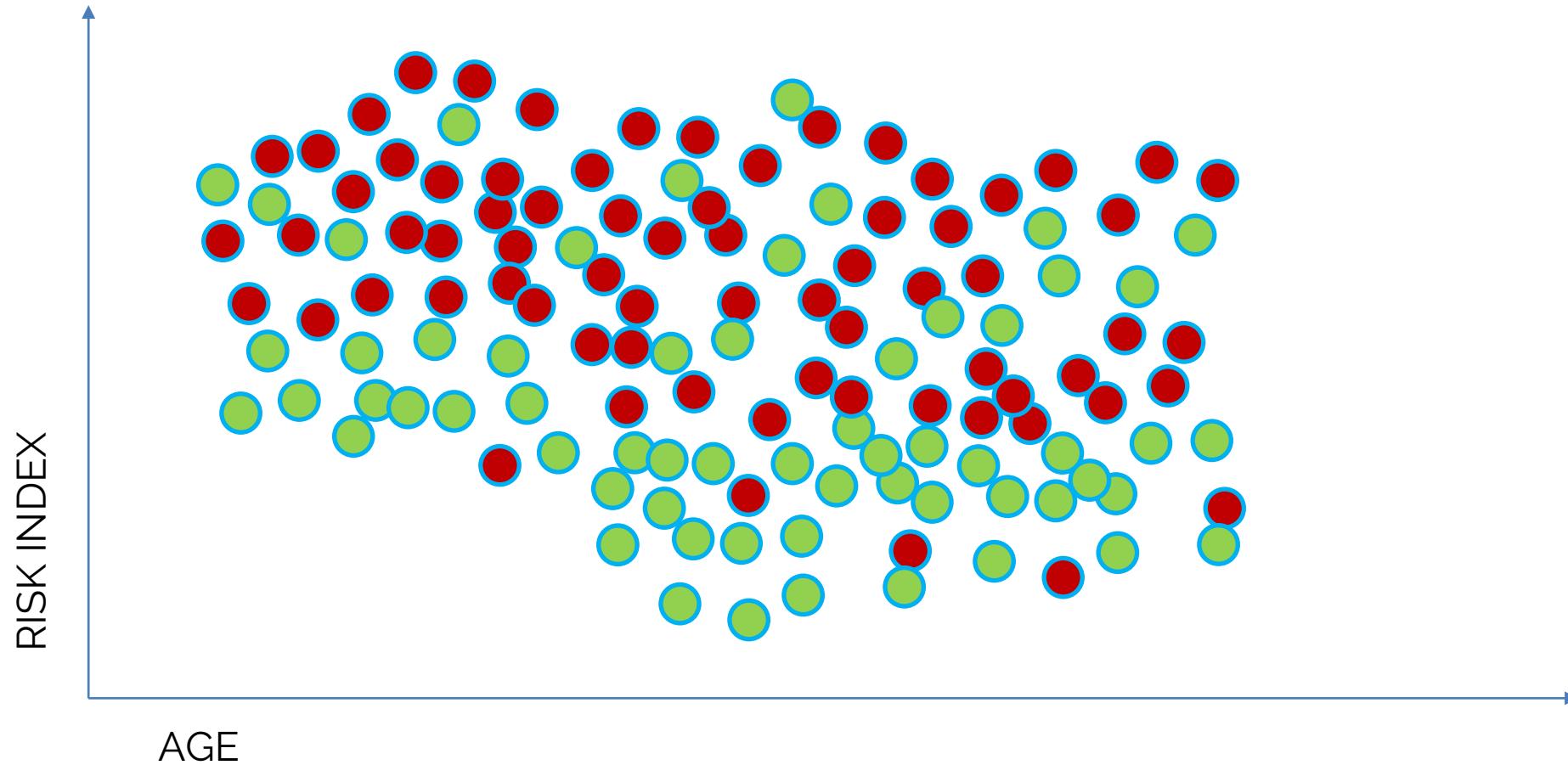
Machine learning



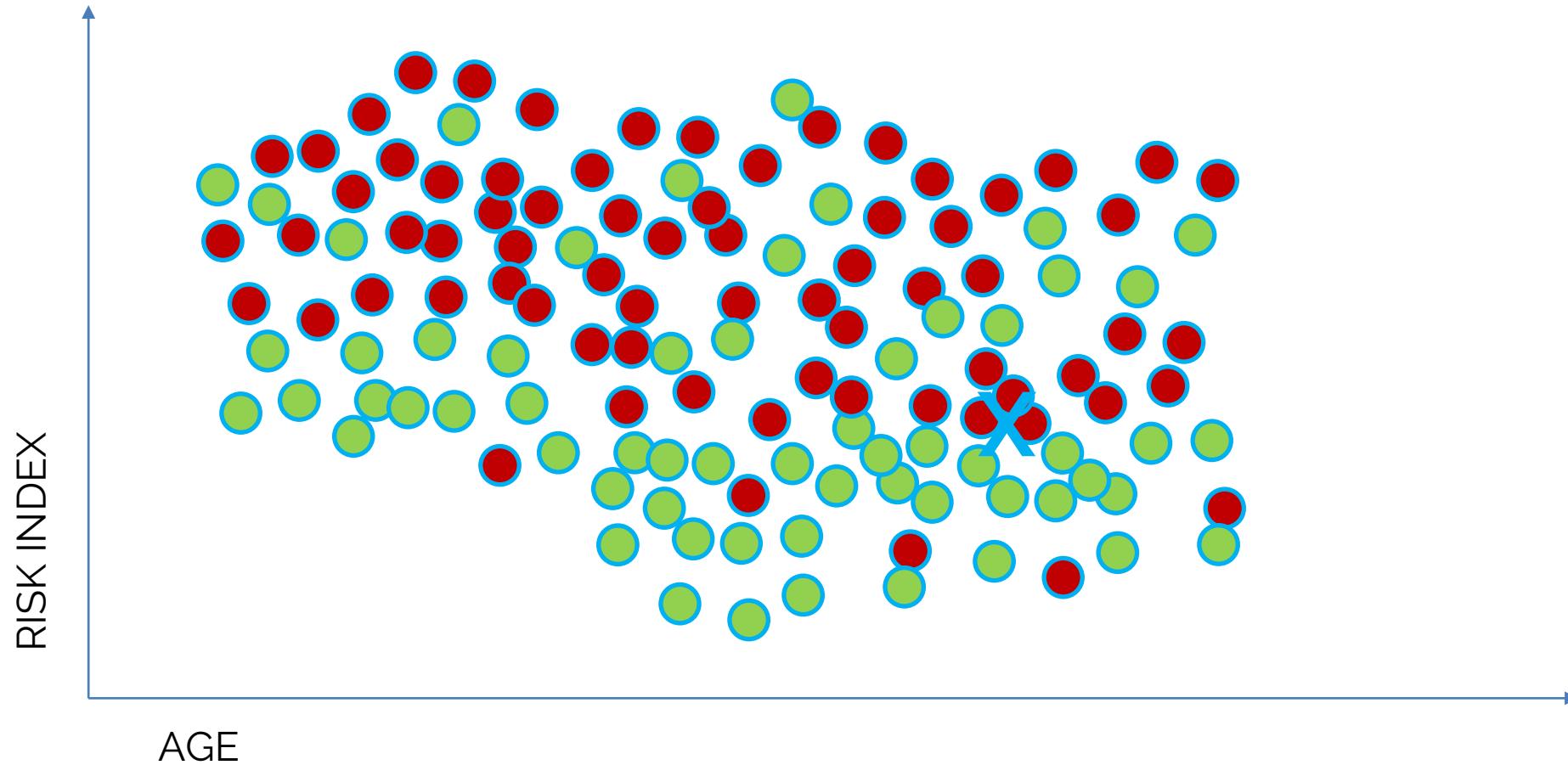
Classification



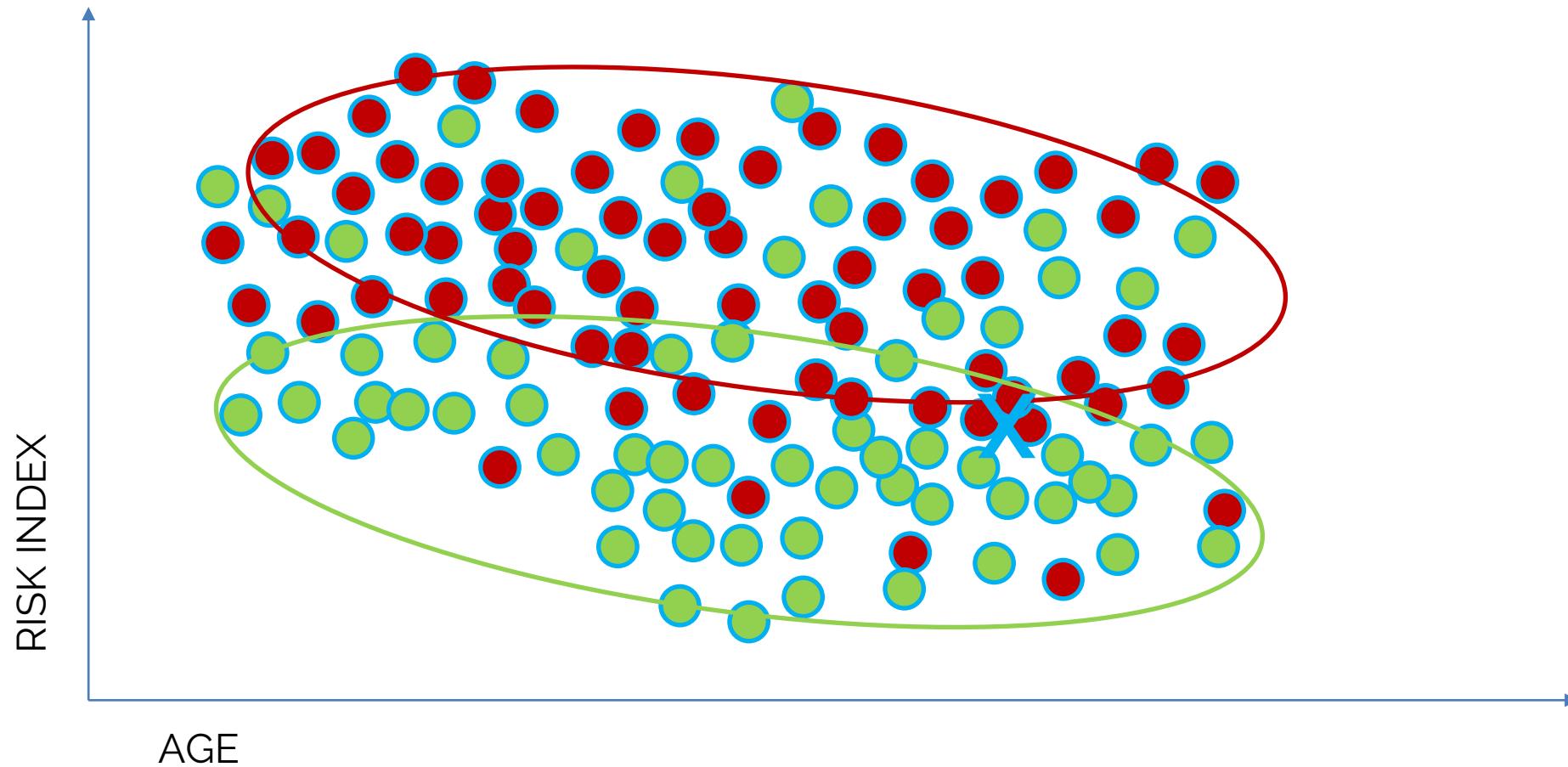
Classification



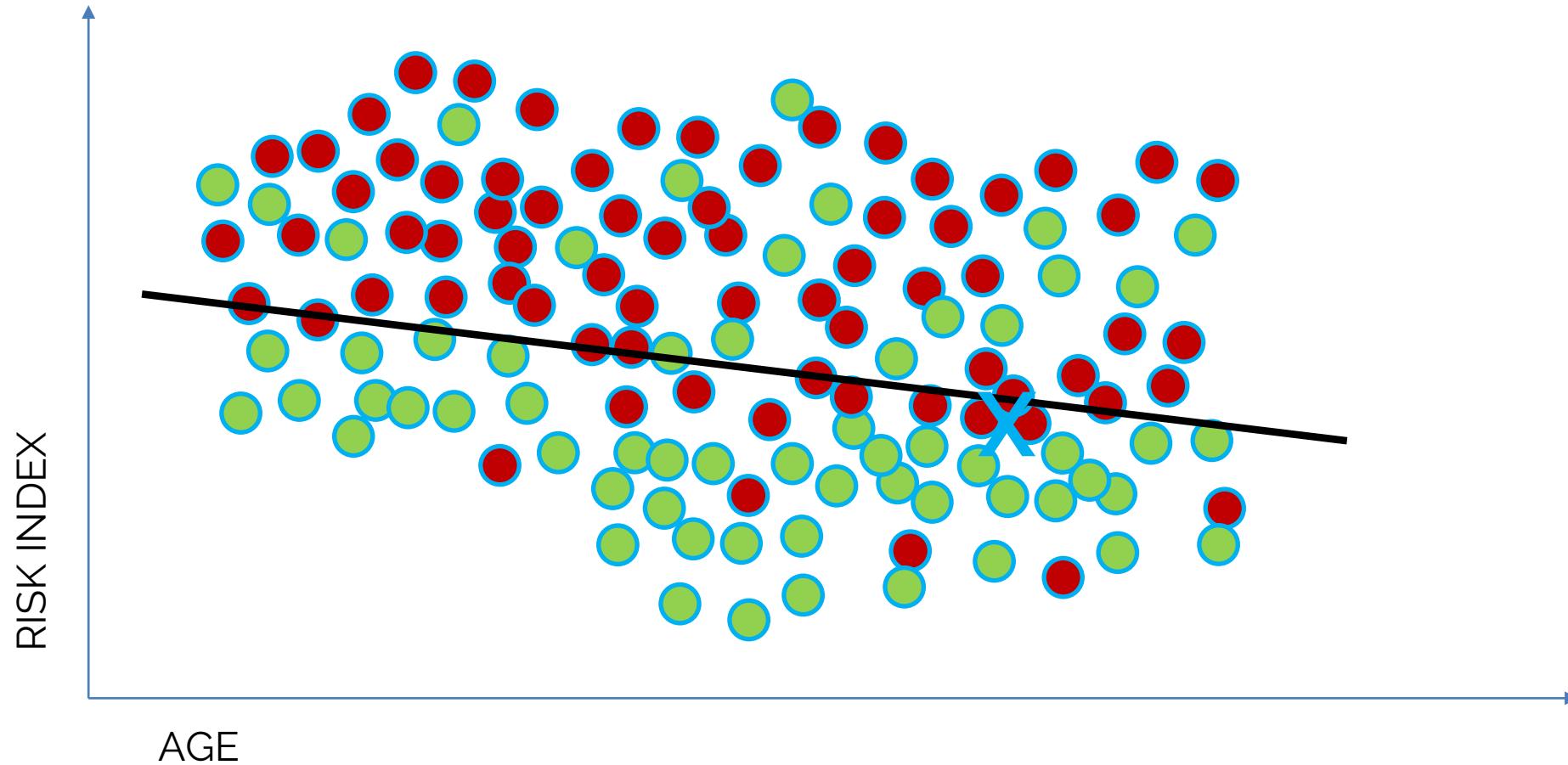
Classification



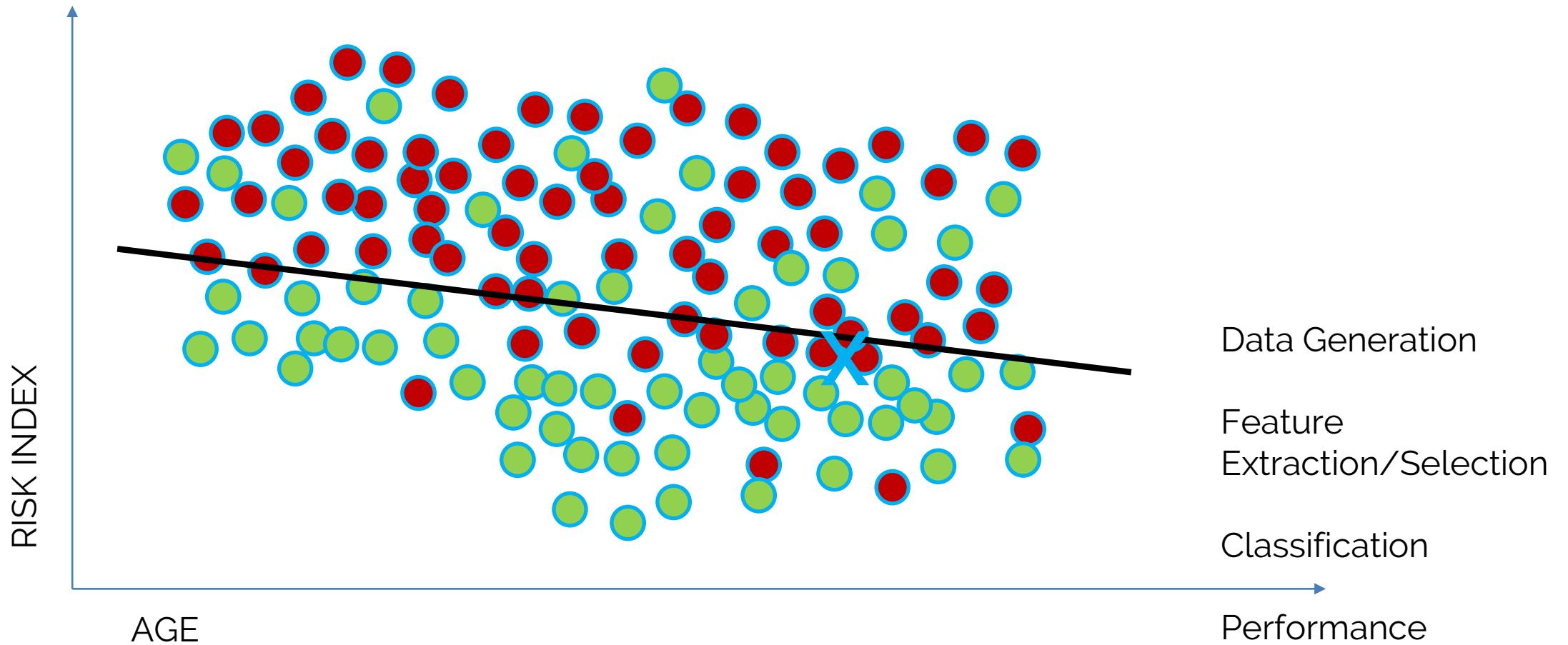
Classification



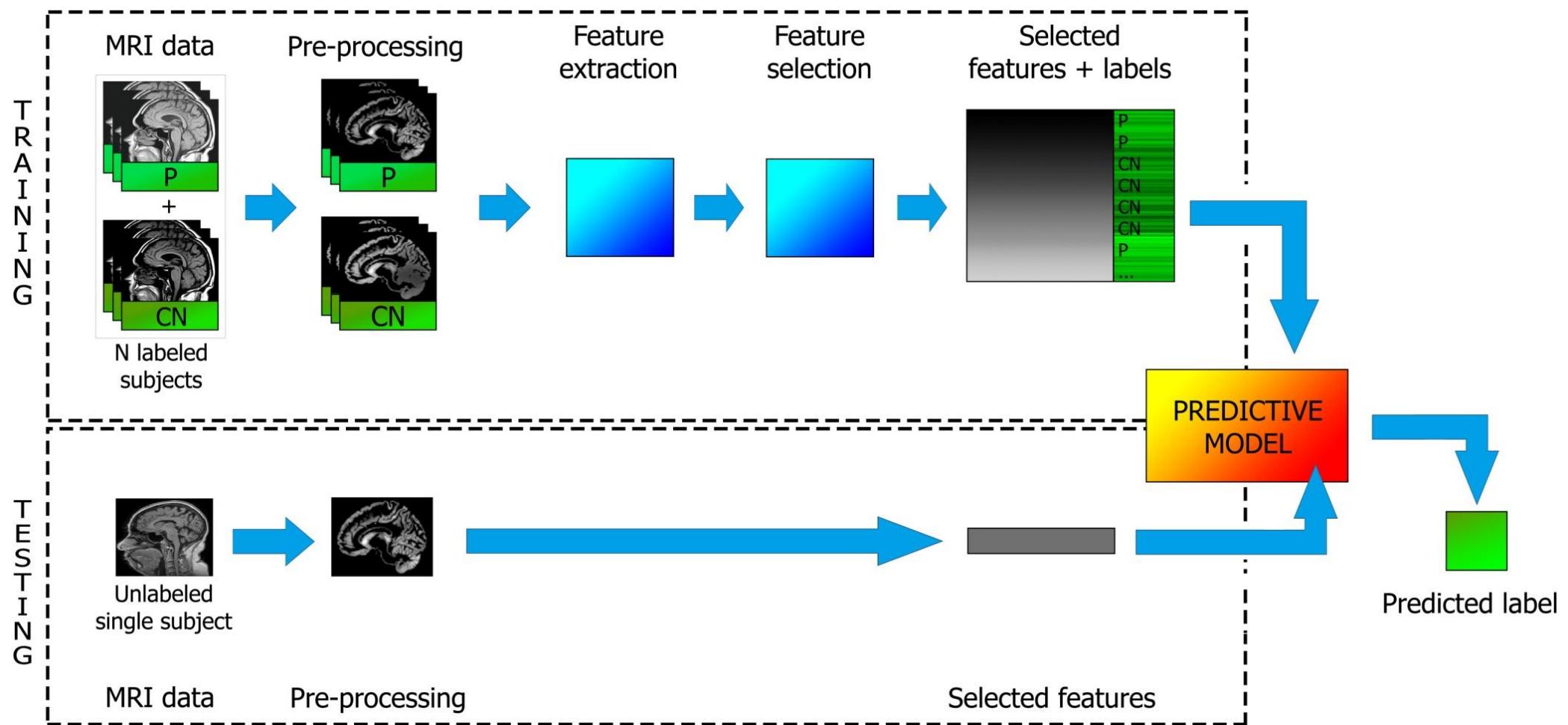
Classification



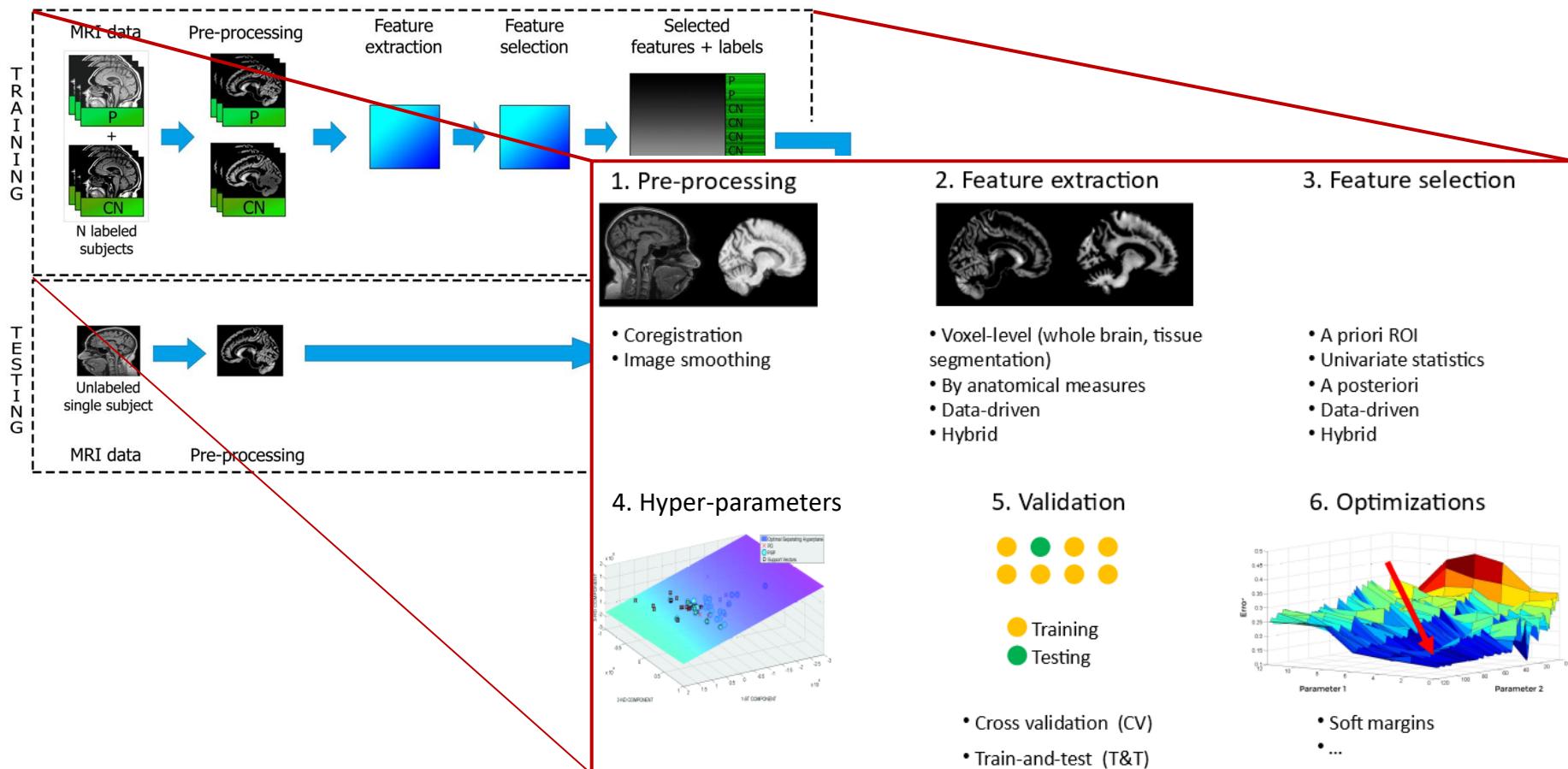
Classification



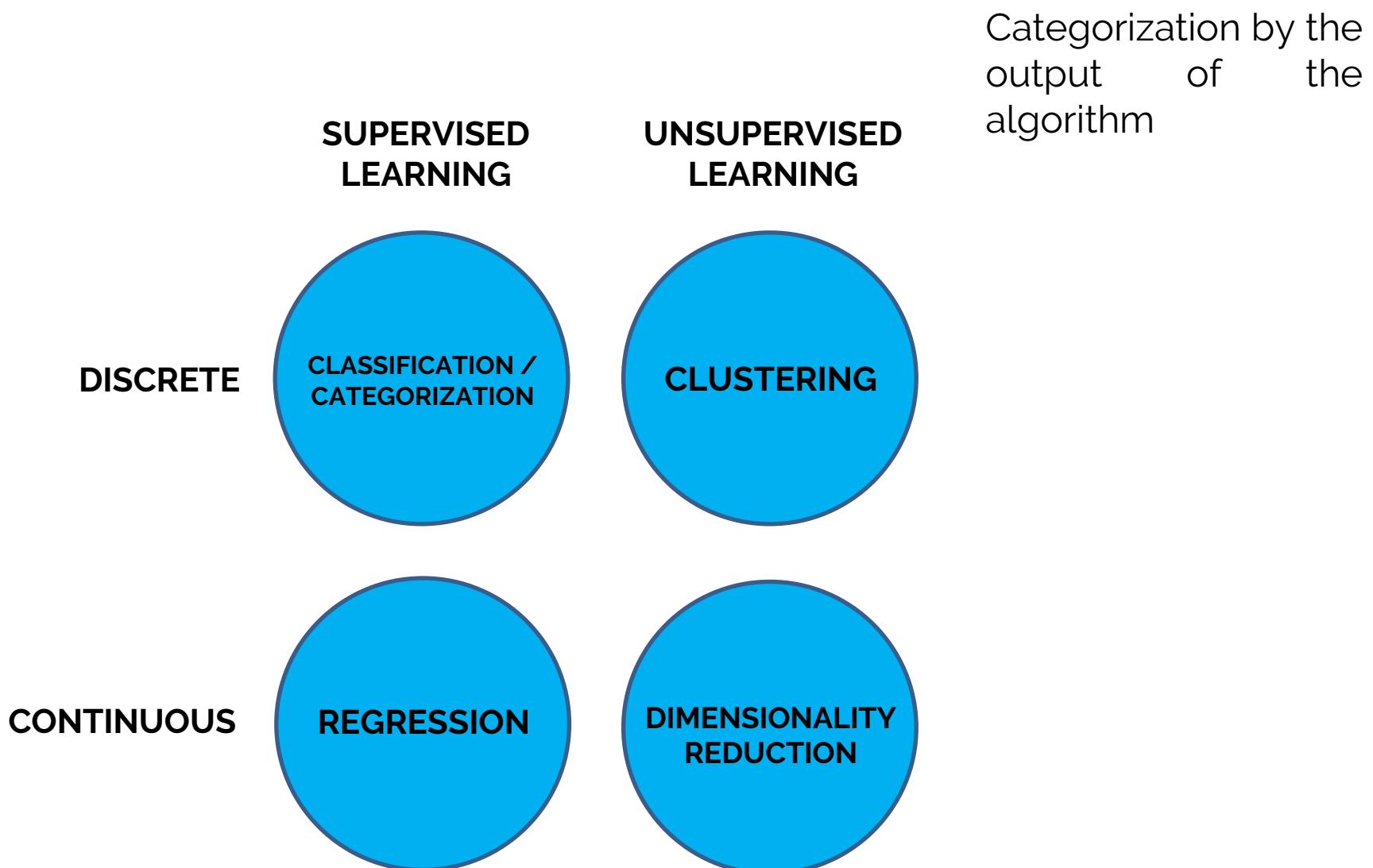
Machine Learning Applied to Medical Data



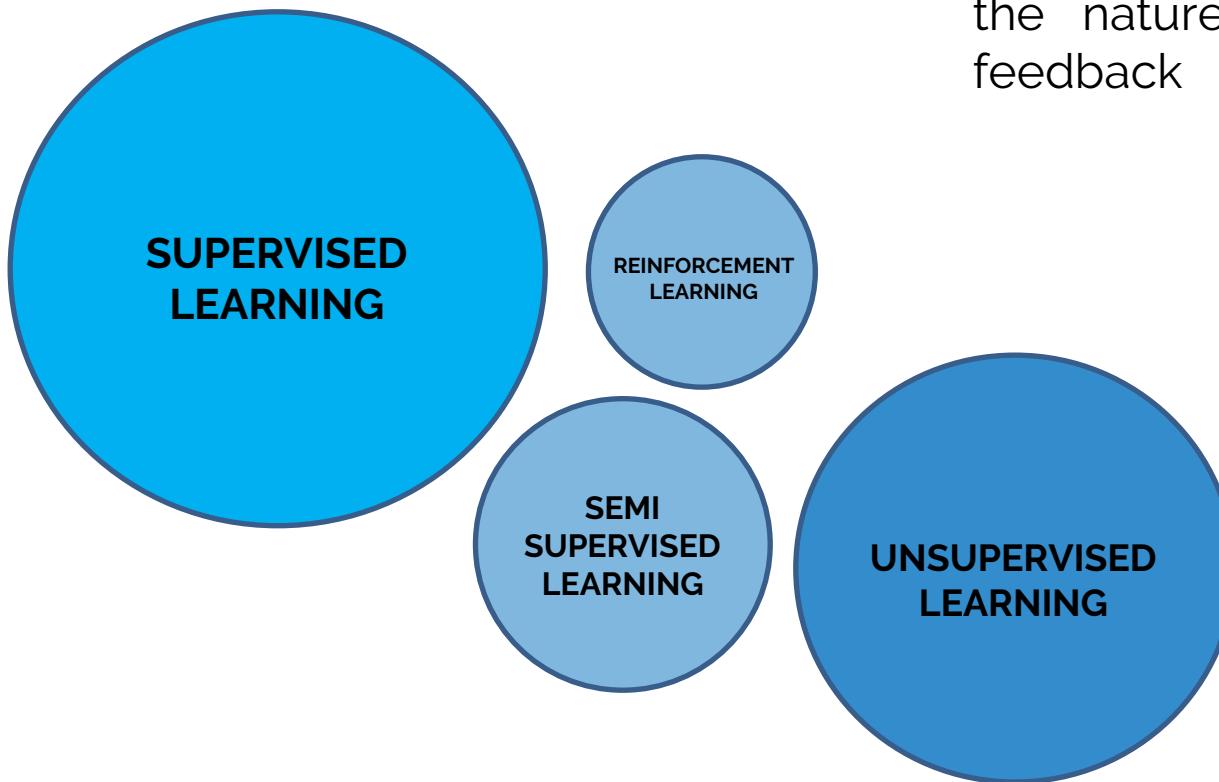
Machine Learning Applied to Medical Data



Machine learning



Machine learning



FEATURE EXTRACTION

Feature Extraction | Obiettivo

Eseguire un mapping dallo spazio iniziale a uno spazio di dimensione inferiore R^k , con $k < d$

Operare in spazi a dimensionalità inferiore

- rende più semplice addestrare algoritmi di machine learning (richiede meno dati per l'addestramento)
- scartando dati ridondanti (informazioni correlate) e rumorosi si migliorano anche le prestazioni e le si rendono più robuste

Obiettivo è scartare le informazioni non rilevanti o meno rilevanti per il problema di interesse

Ridurre la dimensionalità non significa scartare alcune dimensioni e salvarne altre, ma combinare le dimensioni in modo opportuno.

Feature Extraction | Main Techniques

Analisi delle Componenti Principali - Principal Component Analysis (PCA)

- trasformazione non-supervisionata che esegue un mapping lineare delle dimensioni con l'obiettivo di preservare al massimo l'informazione dei pattern (nota anche come Karhunen-Loeve -KL- transform)

(FEATURE SELECTION)

Analisi delle Discriminanti Lineari, Linear Discriminant Analysis (LDA)

- trasformazione supervisionata di mapping lineare

Feature Extraction | Principal Components Analysis

Dato un training set $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1 \dots n$, siano

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1 \dots n} \mathbf{x}_i \quad \text{il vettore medio} \in \mathbb{R}^d$$

$$\Sigma = \frac{1}{n-1} \sum_{i=1 \dots n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^t \quad \text{la matrice di covarianza} \in \mathbb{R}^{d \times d}$$

allora per un dato k ($k < d$, $k < n$, $k > 0$), lo spazio k dimensionale (Sx, Φ_k)

è univocamente definito dal vettore medio e dalla matrice di proiezione $\Phi_k \in \mathbb{R}^{d \times k}$

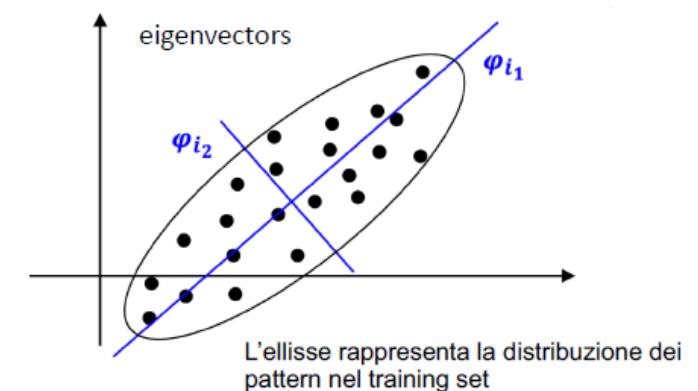
le cui colonne sono costituite dagli autovettori di Σ corrispondenti ai k più grandi autovalori

$$\Phi_k = [\varphi_{i_1}, \varphi_{i_2} \dots \varphi_{i_k}] \text{ con } \lambda_{i_1} \geq \lambda_{i_2} \geq \dots \lambda_{i_k} \geq \dots \lambda_{i_d}$$

φ_{ir} autovettore di Σ corrispondente all'autovalore λ_{ir} $r = 1 \dots d$

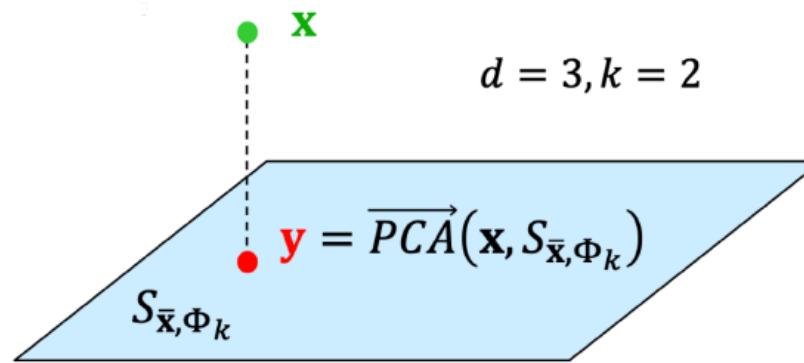
I primi k autovettori sono detti componenti principali (PC)

φ_{i1} indica la direzione di maggior varianza nel training set



Feature Extraction | Principal Components Analysis

$\Re_d \rightarrow \Re_k$



Proiezione Una volta determinato lo spazio PCA, la proiezione di un pattern \mathbf{x} su tale spazio è semplicemente la proiezione geometrica del vettore \mathbf{x} sull'iperpiano che definisce lo spazio. In realtà la vera proiezione geometrica è un vettore che ha la stessa dimensionalità del vettore originale mentre in questo contesto indichiamo con proiezione il vettore (ridotto) nello spazio PCA. Matematicamente questa operazione è eseguita come prodotto della matrice di proiezione trasposta per il pattern \mathbf{x} al quale è preventivamente sottratta la media.

$$\overrightarrow{PCA}(\mathbf{x}, S_{\bar{\mathbf{x}}, \Phi_k}) = \Phi_k^t (\mathbf{x} - \bar{\mathbf{x}})$$

$\Re_k \rightarrow \Re_d$

Retroproiezione Dato un vettore \mathbf{y} nello spazio PCA, la sua retro-proiezione verso lo spazio originale si ottiene moltiplicando il vettore per la matrice di proiezione e sommando il vettore medio. Questa trasformazione non sposta spazialmente il vettore, che giace ancora sullo spazio PCA, ma opera un cambiamento di coordinate che ne permette la codifica in termini delle d componenti dello spazio originale.

$$\overleftarrow{PCA}(\mathbf{y}, S_{\bar{\mathbf{x}}, \Phi_k}) = \Phi_k \mathbf{y} + \bar{\mathbf{x}}$$

Feature Extraction | Principal Components Analysis

Se l'obiettivo è quello di scartare informazione inutile e dati correlati mantenendo gran parte del contenuto informativo si può scegliere k nel modo seguente:

Fissata una percentuale t del contenuto informativo che si vuole preservare (es. $t = 95\%$) si sceglie il minimo valore di k per cui la somma dei più grandi k autovalori è maggiore o uguale a t rispetto alla somma di tutti gli autovalori

Considerando gli autovalori ordinati in ordine decrescente:

$$k = \arg \min_z \left\{ \frac{\sum_{i=1 \dots z} \lambda_i}{\sum_{i=1 \dots d} \lambda_i} \geq t \right\}$$

Poiché gli autovalori denotano la varianza lungo i diversi assi, il rapporto nella formula indica la varianza conservata rispetto alla varianza totale

La scelta di k è obbligata ad esempio per la visualizzazione 2D o 3D dei dati ($k=2, 3$)

Feature Extraction | Principal Components Analysis

Per d elevato (tipico nel caso di immagini, audio, ecc.) la matrice di covarianza può essere molto grande

Esempio per $d = 16384$, $\Sigma \in \mathbb{R}^{16384 \times 16384}$ oltre 268 milioni di valori !

E' più conveniente calcolare la matrice di proiezione attraverso la decomposizione ai valori singolari (Single Value Decomposition, SVD) della matrice rettangolare degli n pattern centralizzati $\mathbf{X} \in \mathbb{R}^{d \times n}$, $n \ll d$ senza passare per la matrice di covarianza

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix}$$

SVD per $n \ll d$: $\mathbf{X} = \mathbf{U}\Gamma\mathbf{V}^t$, con $\mathbf{U} \in \mathbb{R}^{d \times n}$ ortonormale unitaria, $\Gamma \in \mathbb{R}^{n \times n}$ diagonale, $\mathbf{V} \in \mathbb{R}^{n \times n}$ ortonormale

Gli elementi di Γ sono i valori singolari di \mathbf{X} , gli elementi di \mathbf{U} sono i vettori singolari sinistri di \mathbf{X} e gli elementi di \mathbf{V} sono i vettori singolari destri di \mathbf{X} . Si verifica che:

I vettori singolari di sinistra di \mathbf{X} sono gli autovettori di $\mathbf{X}\mathbf{X}^t$, i vettori singolari di destra di \mathbf{X} sono gli autovettori di $\mathbf{X}^t\mathbf{X}$, i vettori singolari non nulli di \mathbf{X} (che si trovano sulla diagonale principale di Γ) sono le radici quadrate degli autovalori non nulli di $\mathbf{X}\mathbf{X}^t$ e $\mathbf{X}^t\mathbf{X}$

$$\Sigma = \frac{1}{n} \mathbf{X}\mathbf{X}^t = \frac{1}{n} \mathbf{U}\Gamma\mathbf{V}^t\mathbf{V}\Gamma\mathbf{U}^t = \frac{1}{n} \mathbf{U}\Gamma^2\mathbf{U}^t$$

Gli autovettori e gli autovalori di Σ possono dunque essere ottenuti dalle colonne di \mathbf{U} (vettori singolari sinistri di \mathbf{X}) e corrispondenti elementi diagonali di Γ^2 (valori singolari al quadrato di \mathbf{X}).

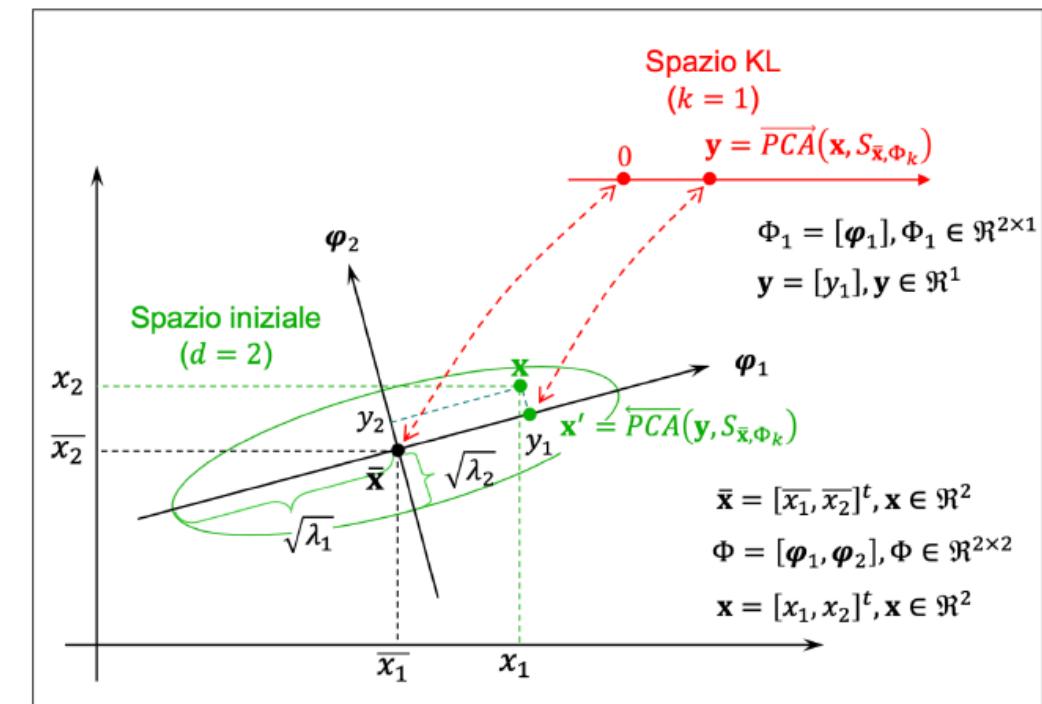
Feature Extraction | Principal Components Analysis

L'ellisse rappresenta la distribuzione dei pattern nel training set

φ_1 e φ_2 sono gli autovettori della matrice di covarianza

Gli autovalori λ_1 e λ_2 sono le varianze della distribuzione lungo gli assi φ_1 e φ_2 .

y_1 e y_2 sono le proiezioni di x sugli assi φ_1 e φ_2 .



se λ_2 è piccolo, x può essere approssimato con x' (retroproiezione di y) senza perdite significative di informazione

Feature Extraction | Principal Components Analysis

$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n}$$

$$s = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n - 1)}}$$

$$s^2 = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n - 1)}$$

$$var(X) = \frac{\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})}{(n - 1)}$$

$$cov(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n - 1)}$$

$$C = \begin{pmatrix} cov(x, x) & cov(x, y) & cov(x, z) \\ cov(y, x) & cov(y, y) & cov(y, z) \\ cov(z, x) & cov(z, y) & cov(z, z) \end{pmatrix}$$

Eigenvectors

Eigenvalues

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

Step 3: Calculate the covariance matrix

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Step 5: Choosing components and forming a feature vector

Step 5: Deriving the new data set

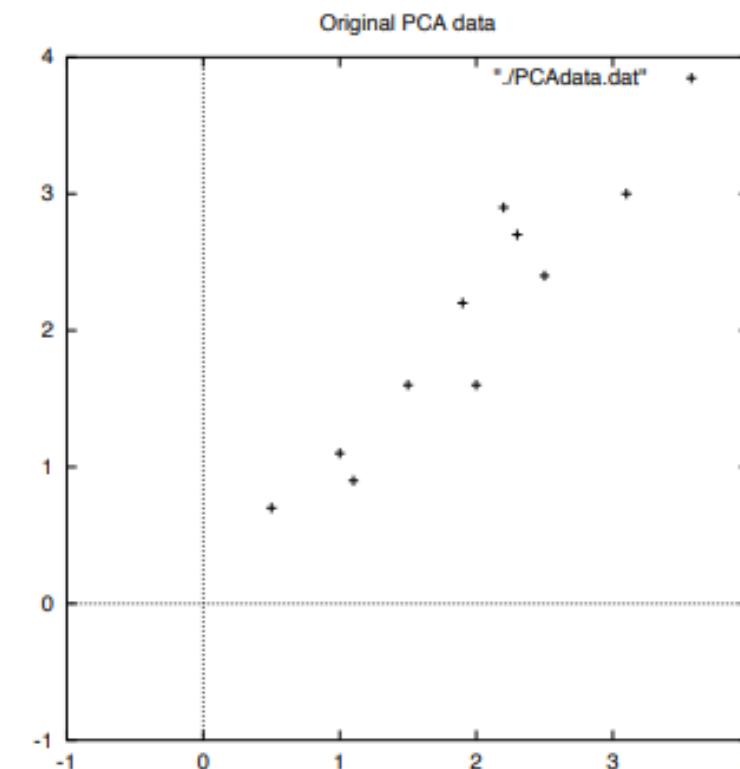
A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Data =



A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

x	y	x	y
2.5	2.4	.69	.49
0.5	0.7	-1.31	-1.21
2.2	2.9	.39	.99
1.9	2.2	.09	.29
Data =	3.1	1.29	1.09
	2.3	.49	.79
	2	.19	-.31
	1	-.81	-.81
	1.5	-.31	-.31
	1.1	-.71	-1.01

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

Step 3: Calculate the covariance matrix

	<i>x</i>	<i>y</i>
DataAdjust =	.69	.49
	-1.31	-1.21
	.39	.99
	.09	.29
	1.29	1.09
	.49	.79
	.19	-.31
	-.81	-.81
	-.31	-.31
	-.71	-1.01

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

Step 3: Calculate the covariance matrix

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

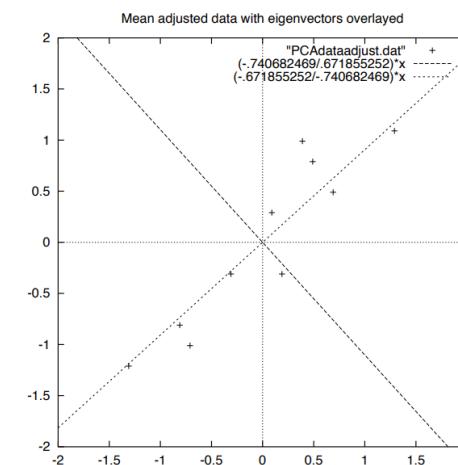


Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlaid on top.

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

Step 3: Calculate the covariance matrix

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

	x	y
	.69	.49
	-1.31	-1.21
	.39	.99
	.09	.29
DataAdjust =	1.29	1.09
	.49	.79
	.19	-.31
	-.81	-.81
	-.31	-.31
	-.71	-1.01

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

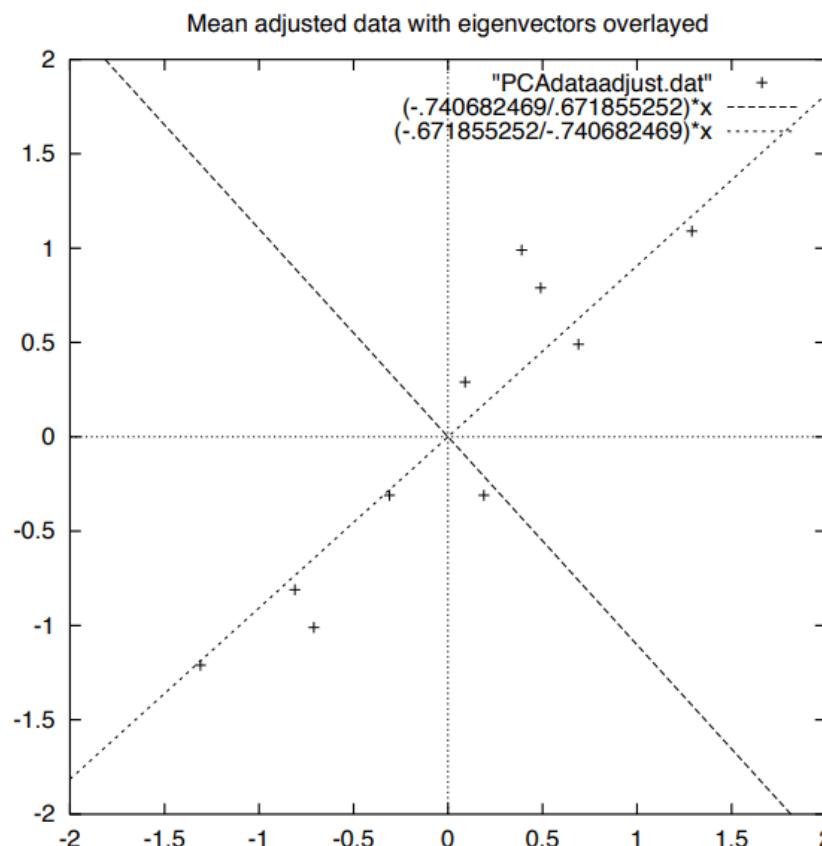


Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

Step 3: Calculate the covariance matrix

$$\text{FeatureVector} = (eig_1 \ eig_2 \ eig_3 \ \dots \ eig_n)$$

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Step 5: Choosing components and forming a feature vector

	x	y
	.69	.49
	-1.31	-1.21
	.39	.99
	.09	.29
DataAdjust =	1.29	1.09
	.49	.79
	.19	-.31
	-.81	-.81
	-.31	-.31
	-.71	-1.01

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

Step 3: Calculate the covariance matrix

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Step 5: Choosing components and forming a feature vector

Step 5: Deriving the new data set

$$\text{FinalData} = \text{RowFeatureVector} \times \text{RowDataAdjust},$$

where *RowFeatureVector* is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and *RowDataAdjust* is the mean-adjusted data *transposed*, ie. the data items are in each column, with each row holding a separate dimension.

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

Step 3: Calculate the covariance matrix

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Step 5: Choosing components and forming a feature vector

Step 5: Deriving the new data set

$$\text{FinalData} = \text{RowFeatureVector} \times \text{RowDataAdjust},$$

where *RowFeatureVector* is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and *RowDataAdjust* is the mean-adjusted data *transposed*, ie. the data items are in each column, with each row holding a separate dimension.

	x	y
	-.827970186	-.175115307
	1.77758033	.142857227
	-.992197494	.384374989
	-.274210416	.130417207
Transformed Data=	-1.67580142	-.209498461
	-.912949103	.175282444
	.0991094375	-.349824698
	1.14457216	.0464172582
	.438046137	.0177646297
	1.22382056	-.162675287

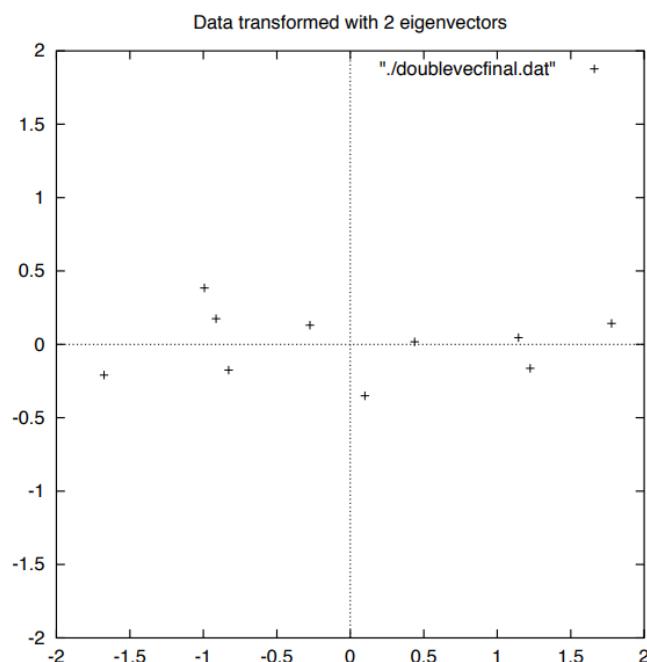


Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

Step 3: Calculate the covariance matrix

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Step 5: Choosing components and forming a feature vector

Step 5: Deriving the new data set

Transformed Data (Single eigenvector)

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

$$\text{FinalData} = \text{RowFeatureVector} \times \text{RowDataAdjust},$$

where *RowFeatureVector* is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and *RowDataAdjust* is the mean-adjusted data *transposed*, ie. the data items are in each column, with each row holding a separate dimension.

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

$$FinalData = RowFeatureVector \times RowDataAdjust$$

Step 2: Subtract the mean

$$RowDataAdjust = RowFeatureVector^{-1} \times FinalData$$

Step 3: Calculate the covariance matrix

$$RowDataAdjust = RowFeatureVector^T \times FinalData$$

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

only true if the elements of the matrix are all unit eigenvectors

Step 5: Choosing components and forming a feature vector

$$RowDataAdjust = RowFeatureVector^T \times FinalData$$

Getting the old data back

$$RowOriginalData = (RowFeatureVector^T \times FinalData) + OriginalMean$$

A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis

Step 1: Get some data

Step 2: Subtract the mean

Step 3: Calculate the covariance matrix

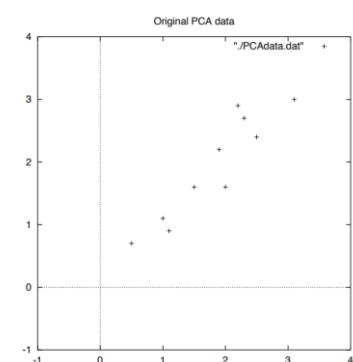
Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Step 5: Choosing components and forming a feature vector

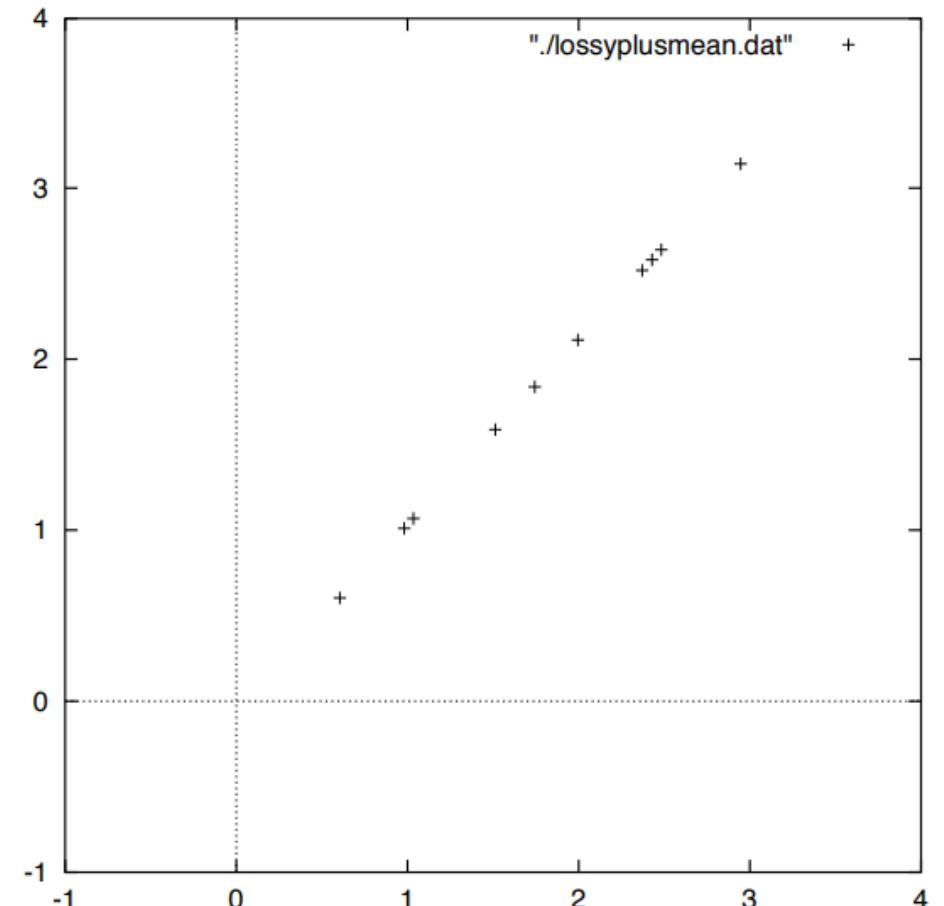
Step 5: Deriving the new data set

Getting the old data back

$$\text{RowOriginalData} = (\text{RowFeatureVector}^T \times \text{FinalData}) + \text{OriginalMean}$$

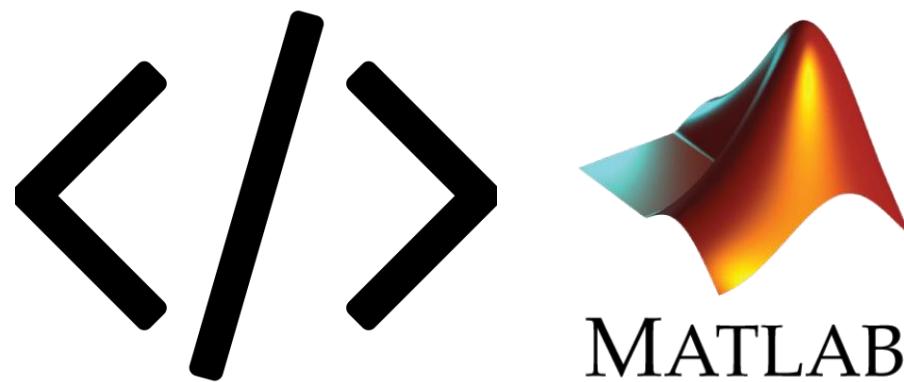


Original data restored using only a single eigenvector



A tutorial on Principal Components Analysis by Lindsay I. Smith

Feature Extraction | Principal Components Analysis



Feature Extraction

- Accuracy improvements.
- Overfitting risk reduction.
- Speed up in training.
- Improved Data Visualization.
- Increase in explainability of our model.

Many other
techniques...

FEATURE SELECTION

Linear Discriminant Analysis

Riduzione di dimensionalità lineare e supervisionata il cui obiettivo è massimizzare la separazione tra le classi.

Per formulare il criterio di ottimizzazione di massima separazione tra le classi sono definite le seguenti matrici di scattering (sparpagliamento):

within-class

indica come i vettori sono scattered rispetto al centro delle classi (ciascuno rispetto alla propria classe).

between-class

indica come i centri delle classi sono scattered rispetto al centro generale della distribuzione (ovvero quanto le classi sono scattered).

Linear Discriminant Analysis

Dato un training set contenente n pattern $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, y_n)$,

dove $\mathbf{x}_i \in \Re^d$ sono i pattern multidimensionali e $y_i \in [1 \dots s]$ le etichette delle s classi.

Siano n_i e $\bar{\mathbf{x}}$ il numero di pattern e il vettore medio della classe i -esima.

Allora le matrici di scattering sono definite come:

within-class

$$\mathbf{S}_w = \sum_{i=1 \dots s} \mathbf{S}_i, \quad \mathbf{S}_i = \sum_{\mathbf{x}_j | y_j=i} (\mathbf{x}_j - \bar{\mathbf{x}}_i)(\mathbf{x}_j - \bar{\mathbf{x}}_i)^t$$

pattern della classe i – esima

matrice di covarianza senza
normalizzare per il numero di pattern

between-class

$$\mathbf{S}_b = \sum_{i=1 \dots s} n_i \cdot (\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_0)(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_0)^t, \quad \bar{\mathbf{x}}_0 = \frac{1}{n} \sum_{i=1 \dots s} n_i \cdot \bar{\mathbf{x}}_i$$

media globale

Linear Discriminant Analysis

Il criterio per la soluzione ottimale e' intuitivo in quanto cerca di massimizzare lo scattering tra le classi (between class S_b) minimizzando al contempo quello all'interno di ogni classe (within class, S_w)

Il che equivale a massimizzare la quantità:

$$J_1 = \text{tr}(\mathbf{S}_w^{-1} \mathbf{S}_b) = \sum_{i=1 \dots d} \lambda_i$$

ove tr è la traccia (somma degli autovalori) della matrice.

Si dimostra che per massimizzare J_1 lo spazio LDA e' definito dagli autovettori relativi ai primi k autovalori della matrice $\mathbf{S}_w^{-1} \mathbf{S}_b$ ($k < n$, $k < s$, $k < d$) (analogia con PCA)

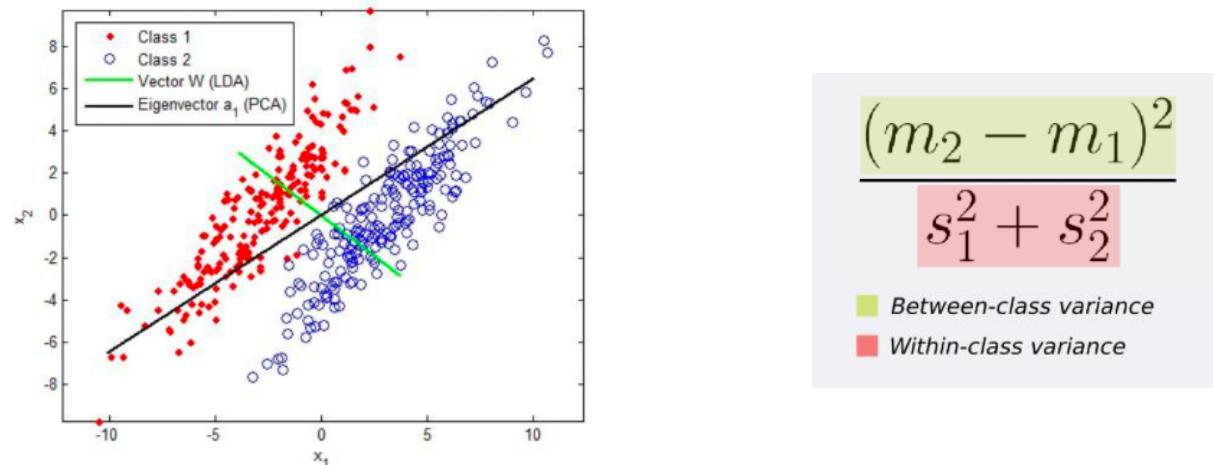


Valore massimo di $k = s - 1$

Linear Discriminant Analysis vs. Principal Components Analysis

Riduzione di dimensionalità da $d = 2$ a $k = 1$

Sono eseguiti mapping lineari $\mathbb{R}^2 \rightarrow \mathbb{R}^1$ ma la soluzione (retta) è profondamente diversa.



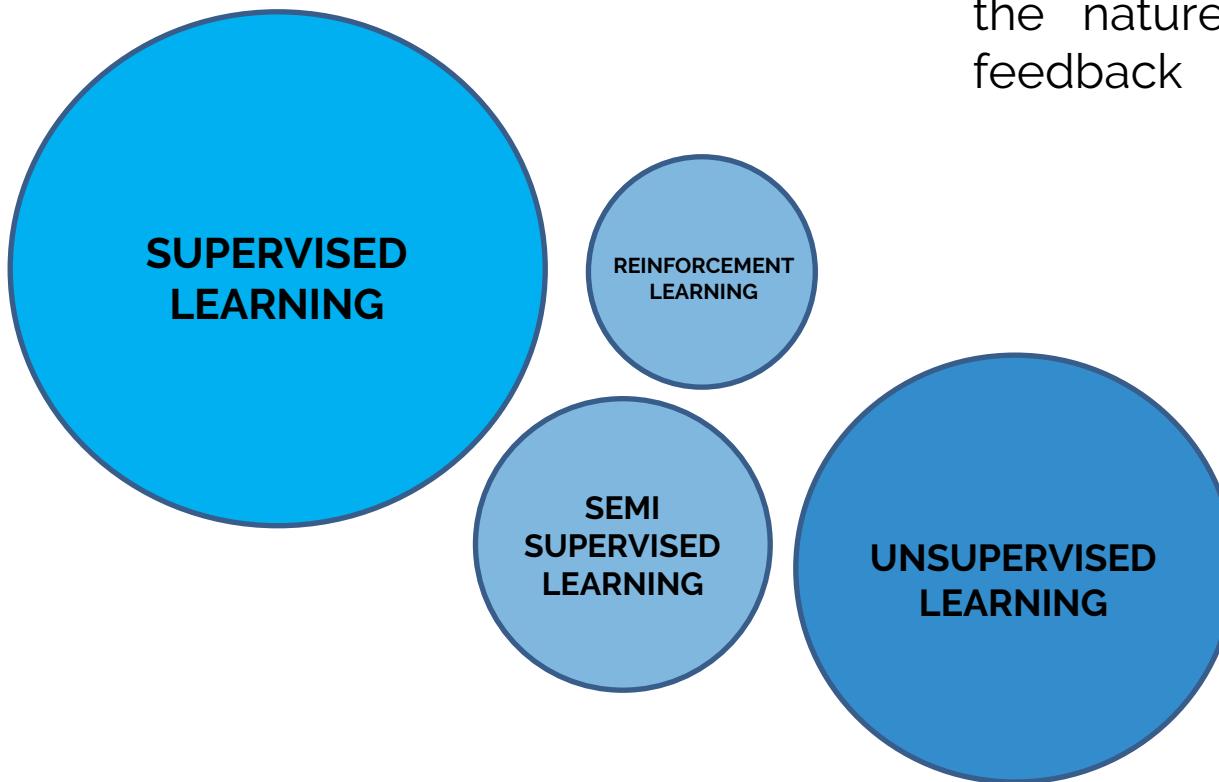
Il segmento nero che identifica la soluzione PCA è l'iperpiano sul quale proiettando i pattern (indipendentemente dalla loro classe) conserviamo al massimo l'informazione.

Il segmento verde che identifica la soluzione LDA è l'iperpiano sul quale proiettando i pattern siamo in grado di discriminare al meglio le due classi

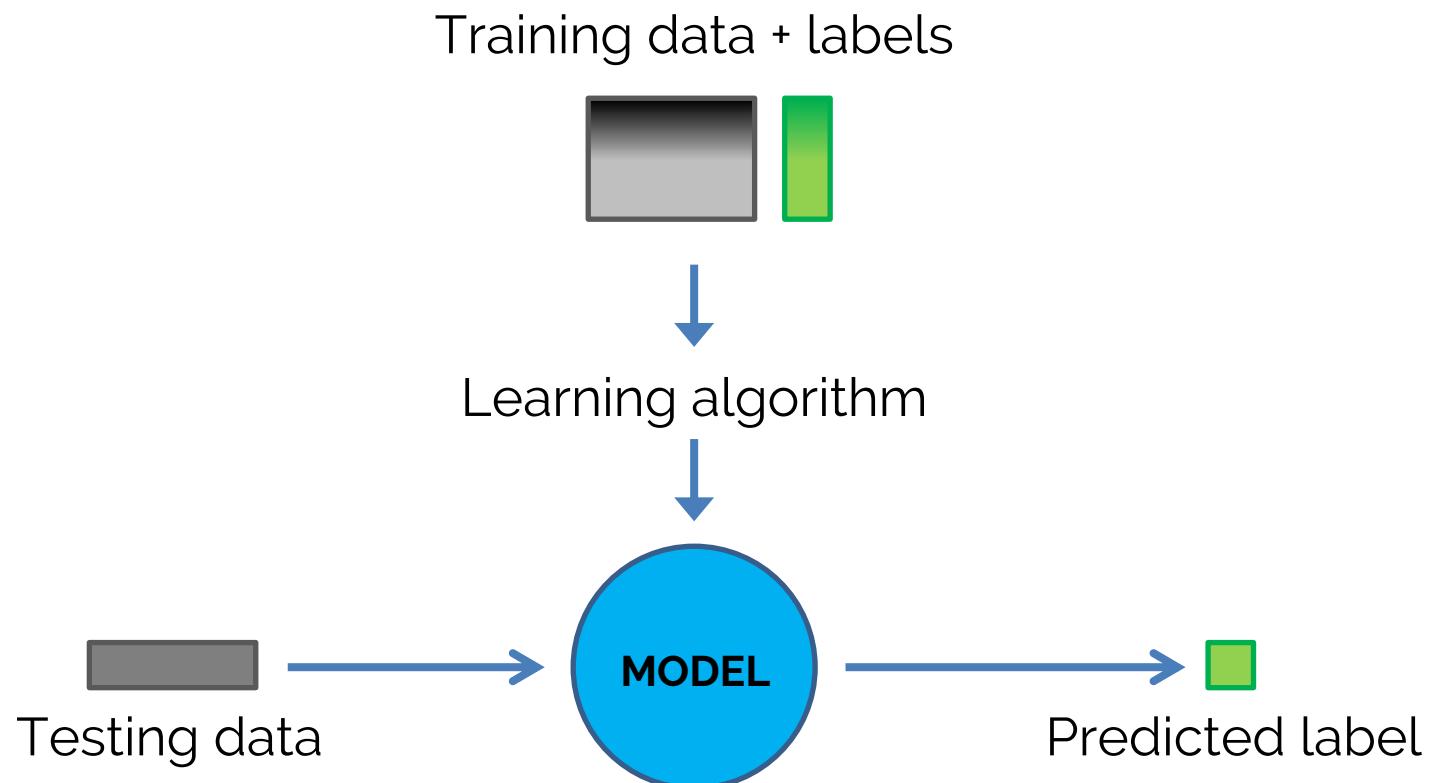
Mentre PCA privilegia le dimensioni che rappresentano al meglio i pattern, LDA privilegia le dimensioni che discriminano al meglio i pattern del TS.

CLASSIFICATION

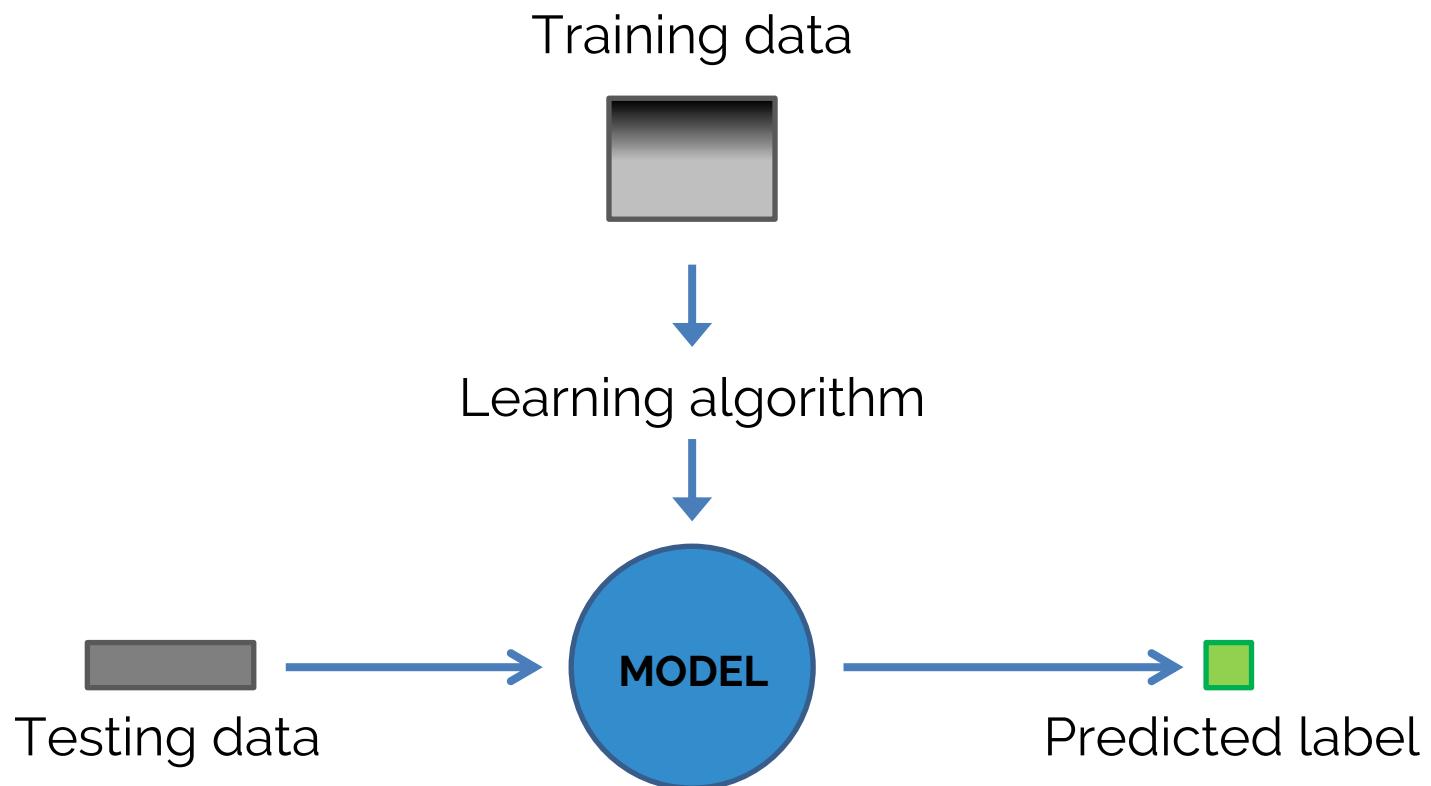
Machine learning



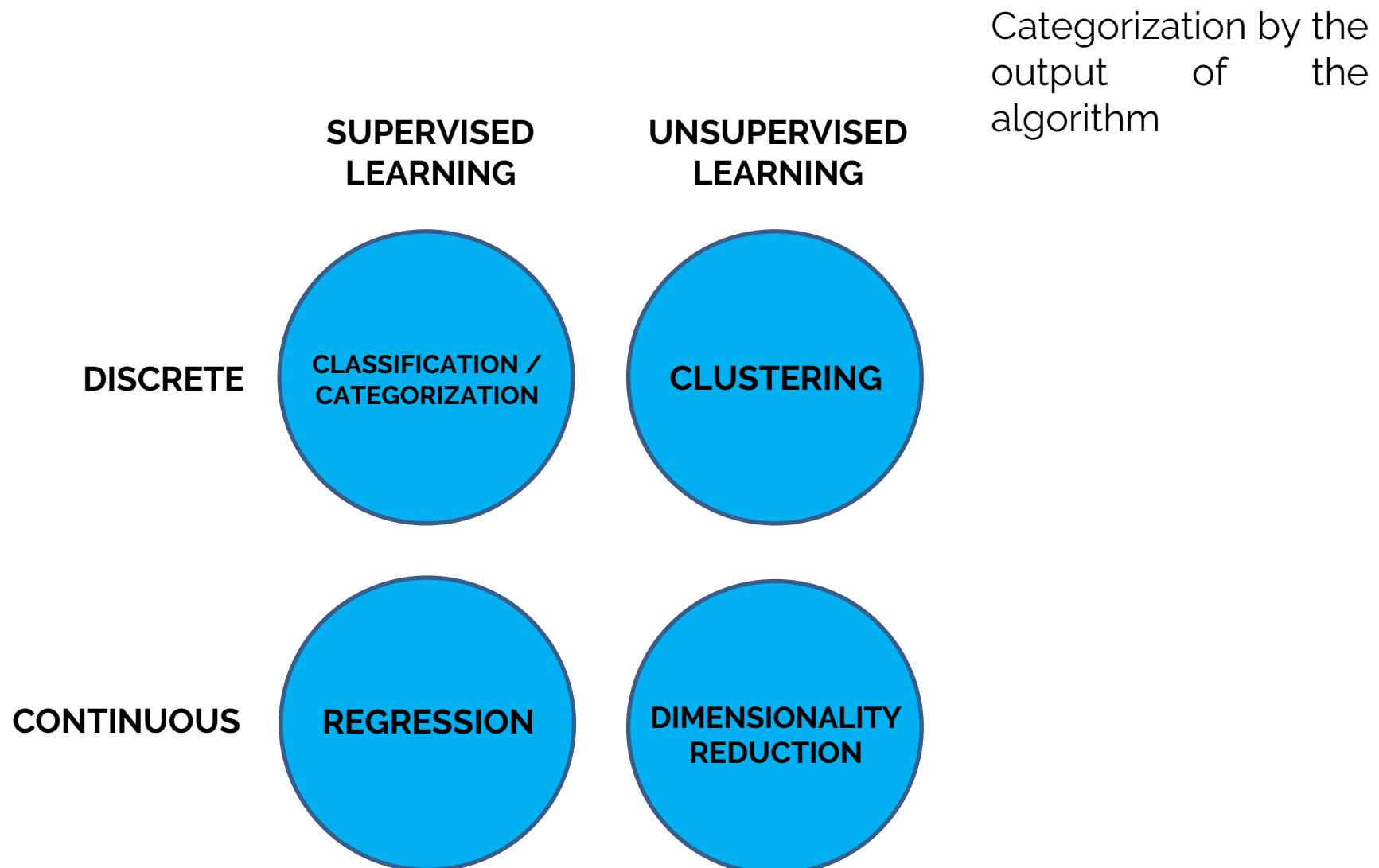
Supervised learning



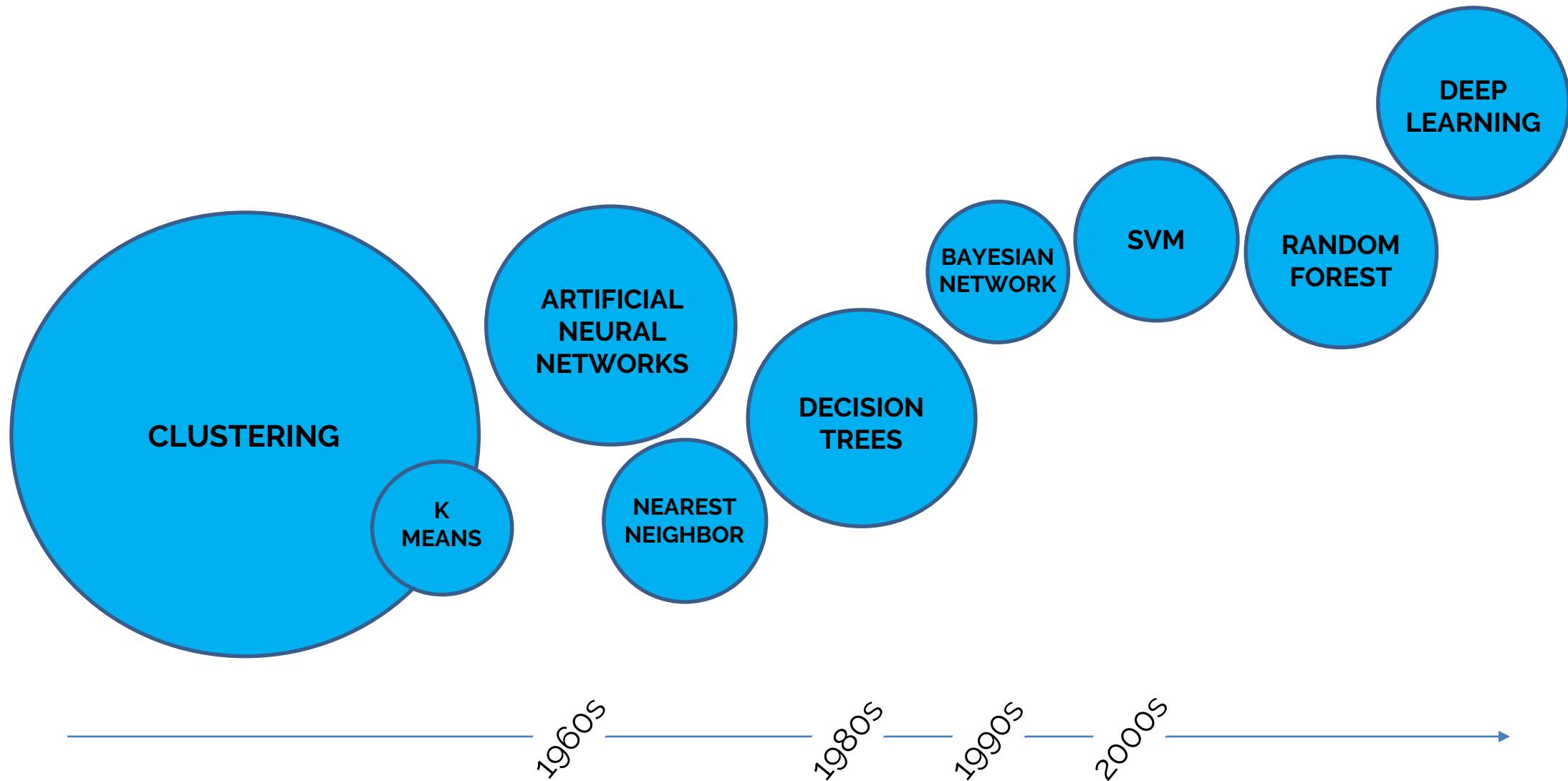
Unsupervised learning



Machine learning



Machine learning



UNSUPERVISED CLASSIFICATION

CLUSTERING

Definizioni

- Definizioni, Criteri, Algoritmi
- Centroid-based Clustering
 - K-means
 - Fuzzy k-means
 - Expectation-Maximization (Gaussian Mixture)

Clustering (raggruppamento), Robert Tyron 1939

Famiglia di metodi non supervisionati in grado di individuare raggruppamenti intrinseci (cluster) di pattern nello spazio multidimensionale, e (opzionalmente) di definire in corrispondenza di tali raggruppamenti le classi (incognite).

Il problema è molto complesso: determinare la soluzione ottimale con ricerca esaustiva è possibile solo nei casi di pattern limitati

Dimensionalità

Nel caso $c = 2$, dati 4 pattern {A, B, C, D} e $d = 2$, il numero di soluzioni possibili è 7

- (A) (B,C,D)
- (B) (A,C,D)
- (C) (A,B,D)
- (D) (A,B,C)
- (A,B) (C,D)
- (A,C) (B,D)
- (A,D) (B,C)

Dimensionalità

Dati n pattern, assumendo di conoscere c , il numero di soluzioni è dell'ordine di $c^n/c!$ (approssimazione numero di Stirling di seconda specie*).

Esempio: per $n = 100$, $c = 5$, il numero di soluzioni è 1067.

Se c non è fissato a priori, il numero di soluzioni è dato dal numero di Bell, ottenibile sommando tutti i casi per tutti i valori di c da 1 a n .

*il numero di Stirling di seconda specie è il numero di modi in cui n oggetti distinguibili possono essere suddivisi tra k sottoinsiemi disgiunti e non vuoti.

La maggior parte dei criteri di clustering sono definiti sulla base delle due osservazioni seguenti:

- 1) i cluster sono costituiti da nuvole di punti a densità relativamente elevata separate da zone dove la densità è più bassa;
- 2) i pattern all'interno dello stesso cluster devono essere tra loro più simili rispetto a pattern appartenenti a cluster diversi

- Minimizzazione delle distanze dai centroidi

Minimizza la somma dei quadrati delle distanze dei pattern x dai centroidi (i.e. baricentri) delle classi

$$J_e = \sum_{i=1..s} \sum_{x \in C_i} \|x - \bar{x}_i\|^2, \quad \bar{x}_i = \frac{1}{n_i} \sum_{x \in C_i} x$$

dove C_i è l' i -esimo cluster, n_i il numero di pattern che il cluster contiene e \bar{x}_i il suo centroide (media).

È un buon criterio per cluster a simmetria radiale (i.e., circolari), ma penalizza forme allungate o cluster innestati (i.e. un cluster a forma di anello con all'interno un altro cluster)

- Minimizzazione distanze intra-classe

$$J_e = \sum_{i=1..s} n_i \cdot \bar{s}_i, \quad \bar{s}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} f_s(\mathbf{x}, C_i)$$

dove f_s è una misura di distanza tra \mathbf{x} e il cluster a cui appartiene. Ad esempio:

$$f_s(\mathbf{x}, C_i) = \frac{1}{n_i} \sum_{\mathbf{x}' \in C_i} \|\mathbf{x} - \mathbf{x}'\|^2 \quad \text{criterio simile alla minimizzazione distanze dai centroidi}$$

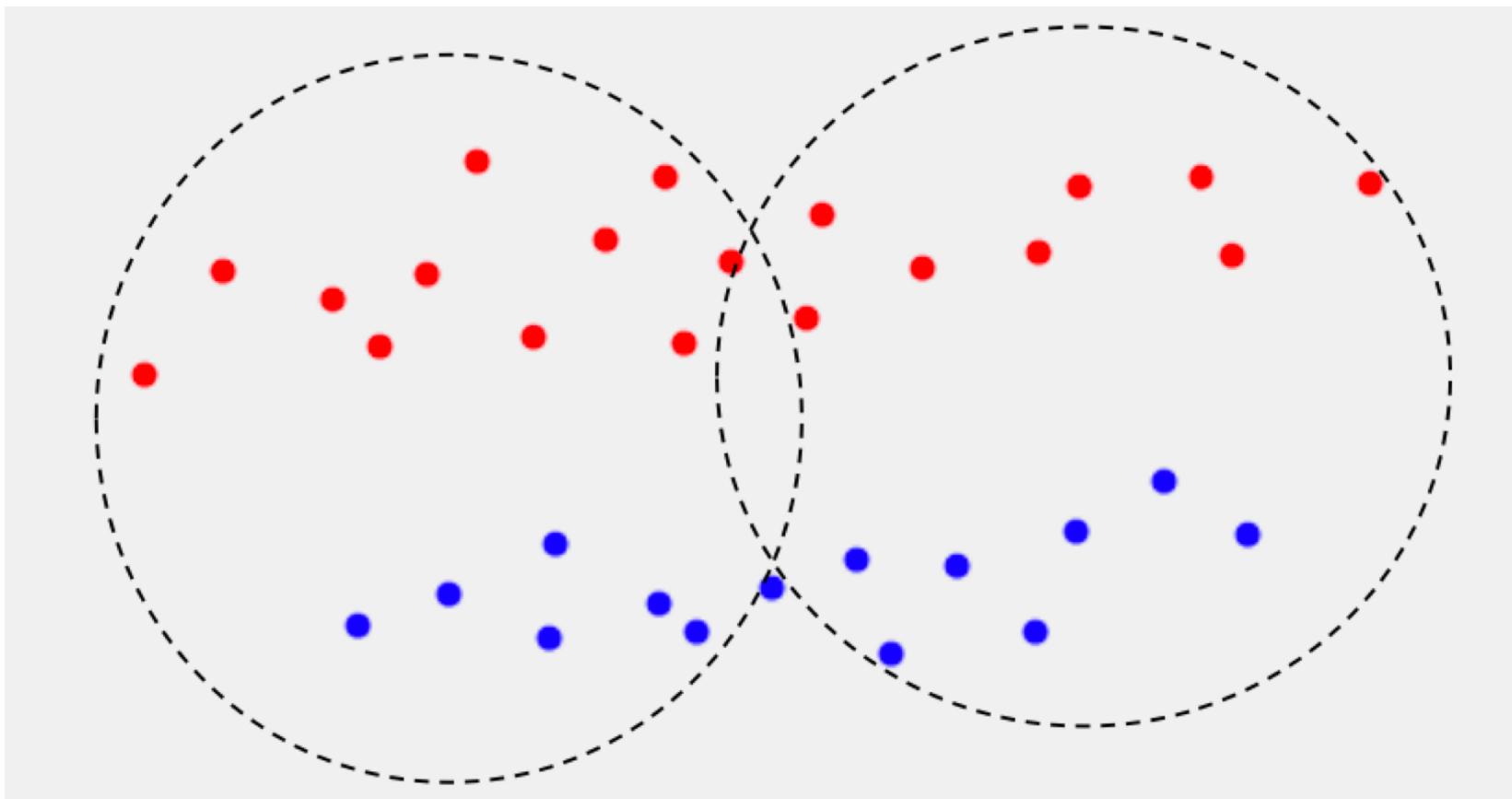
$$f_s(\mathbf{x}, C_i) = \min_{\substack{\mathbf{x}' \in C_i \\ \mathbf{x}' \neq \mathbf{x}}} \|\mathbf{x} - \mathbf{x}'\|^2$$

non penalizza cluster allungati affinché f_s assuma valore ridotto, non è necessario che tutti (o gran parte de) i pattern di C siano vicini a x , ma è sufficiente un vicino.

Criteri

$C=2$

- 1) Minimizzazione distanze tra i centroidi: aggregazione come da cerchi tratteggiati
- 2) Minimizzazione della distanza dei vicini: come da colori rosso e blu



Algoritmi

- Clustering basato su centroidi

Attraverso successive iterazioni, si individuano i cluster cercando di minimizzare la distanza dei pattern dai centroidi dei cluster cui appartengono:

- K-means
- Fuzzy K-means
- Expectation – Maximization (Gaussian Mixture)

- Clustering gerarchico

Attraverso operazioni che aggregano pattern in base a una misura di distanza, si organizzano i dati in struttura ad albero (dendogramma)

- Clustering basato sulla densità:

I cluster individuati sono regioni connesse in aree ad elevata densità

Appartenenza

- Clustering hard (esclusivo)

Un pattern è assegnato (in modo esclusivo) a un solo cluster

- Clustering soft (fuzzy)

I pattern appartengono ai diversi cluster con un certo grado di appartenenza (es. tra 0 e 1)

È più efficace nel gestire pattern vicino al bordo di due o più cluster e outlier.

L'assegnazione può diventare esclusiva scegliendo, per ogni pattern, il cluster verso cui il grado di appartenenza è massimo

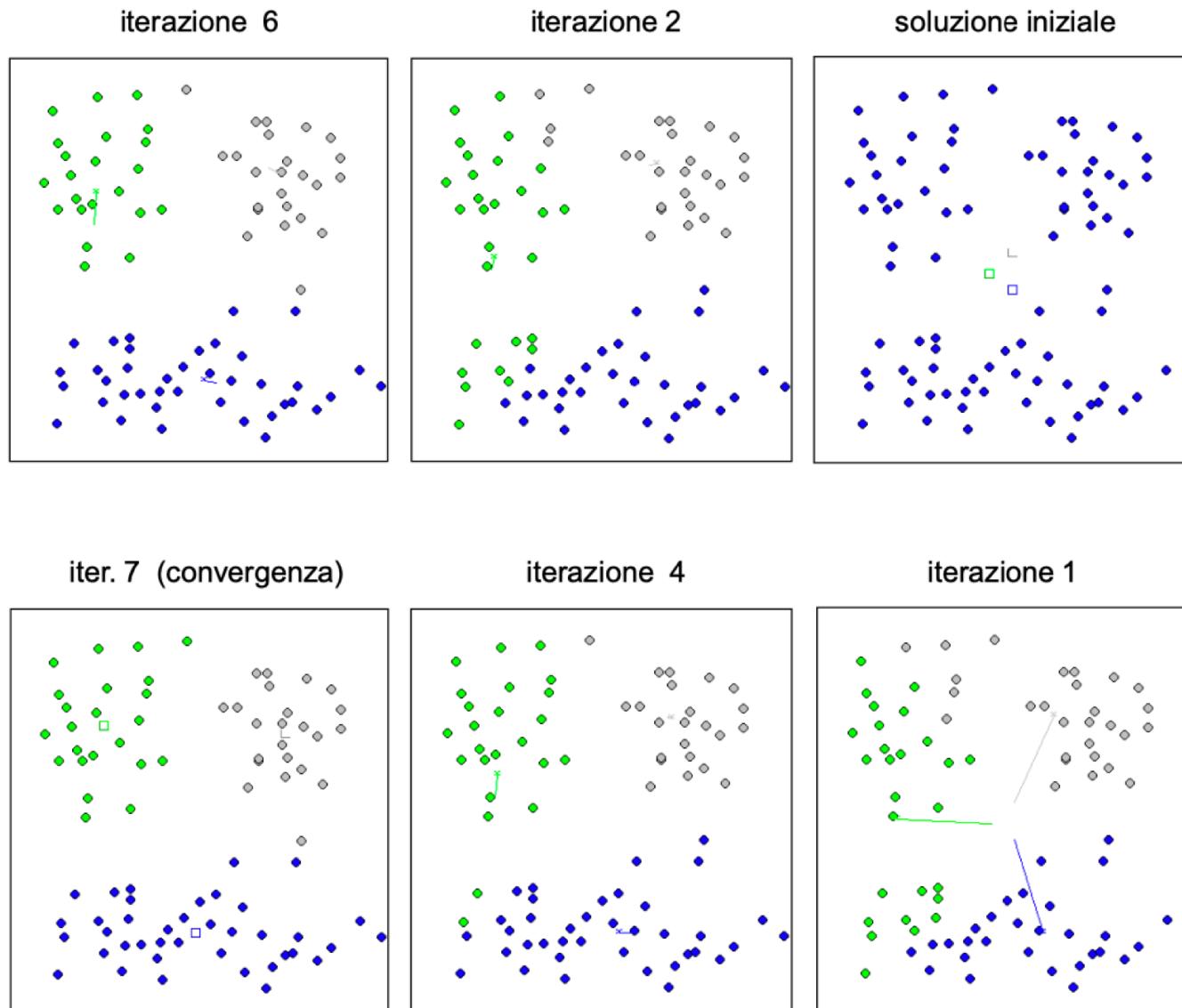
K-means (hard)

Clustering basato sui centroidi (hard)

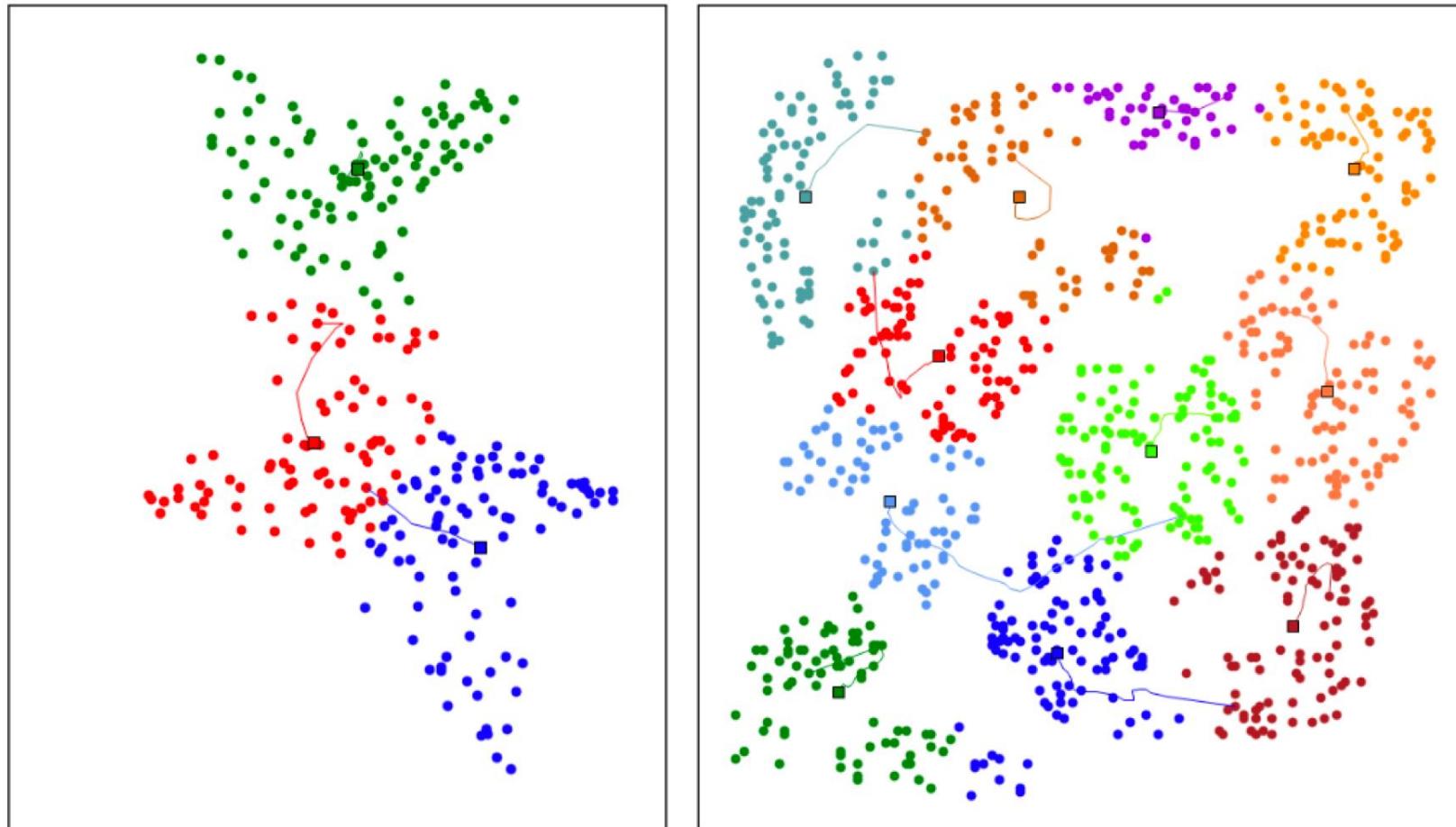
- metodo computazionalmente molto semplice e altrettanto semplice da implementare (spesso la prima scelta per risolvere problemi di clustering)
- minimizza le distanze dai centroidi
- richiede in input il numero di cluster c e una soluzione iniziale
- assegna ogni pattern al cluster per cui è minima la distanza dal centroide
- i cluster sono modificati iterativamente a seguito del ricalcolo del loro centroide.
- l'algoritmo termina (converge) quando i centroidi sono stabili e quindi le partizioni non cambiano.
- la convergenza si ottiene solitamente in pochi passi (< 10).
- Il tipo di ottimizzazione è iterativa e locale, pertanto il metodo può convergere a massimi locali della soluzione.

Identifica cluster iper-sferici nel caso in cui venga utilizzata la distanza euclidea come misura di distanza tra i pattern o cluster iper-ellisoidali nel caso di distanza di Mahalanobis.

K-means (hard)



K-means (hard)



Minimizzando le distanze dai centroidi, K-means non è in grado di identificare cluster dalla forma non sferica

K-means (hard) | Soluzione iniziale e validazione

Diverse varianti per generare buone soluzioni iniziali e determinare il numero di classi (clustering validation).

Per minimizzare il rischio di convergenza verso minimi locali, l'algoritmo può essere eseguito più volte a partire da soluzioni iniziali diverse:

Es:

- random
- prodotte da un (diverso) euristico

Le tecniche di validazione tendono a valutare a posteriori la bontà delle soluzioni prodotte per diversi valori di c , e a sceglierne una sulla base di un criterio di validazione che tenga conto sia della bontà della soluzione sia della sua complessità

Problema: considerando il criterio di minimizzazione delle distanze dai centroidi e lanciando K-means con diversi valori di c , è molto più facile ottenere valori ridotti di pattern per valori elevati di S .

Può essere utile provare a penalizzare le soluzioni con molti cluster, ad esempio introducendo penalità sul numero di pattern.

Fuzzy K-means (soft)

La variante Fuzzy del K-means consente a un pattern di appartenere con un certo grado di probabilità a diverse classi. Il criterio di ottimizzazione in questo caso è:

$$J_{fuz} = \sum_{i=1..s} \sum_{j=1..n} [P(C_i | \mathbf{x}_j, \Theta)]^m \cdot \|\mathbf{x}_j - \boldsymbol{\theta}_i\|^2$$

dove P è la probabilità che dato il pattern x_j e l'insieme Θ di centroidi che definiscono la soluzione attuale, il cluster di appartenenza sia C_i

Fuzzy K-means (soft)

I centrodi invece che come semplice media dei pattern si calcolano come media pesata rispetto alle probabilità di appartenenza:

$$\Theta_i = \frac{\sum_{j=1..n} P(C_i | \mathbf{x}_j, \Theta) \cdot \mathbf{x}_j}{\sum_{j=1..n} P(C_i | \mathbf{x}_j, \Theta)}$$

Fuzzy K-means (soft)

Le probabilità di appartenenza si calcolano come:

$$P(C_i | \mathbf{x}_j, \Theta) = \frac{1}{\sum_{k=1..s} \left(\frac{\|\mathbf{x}_j - \boldsymbol{\theta}_i\|}{\|\mathbf{x}_j - \boldsymbol{\theta}_k\|} \right)^{\frac{2}{m-1}}}$$

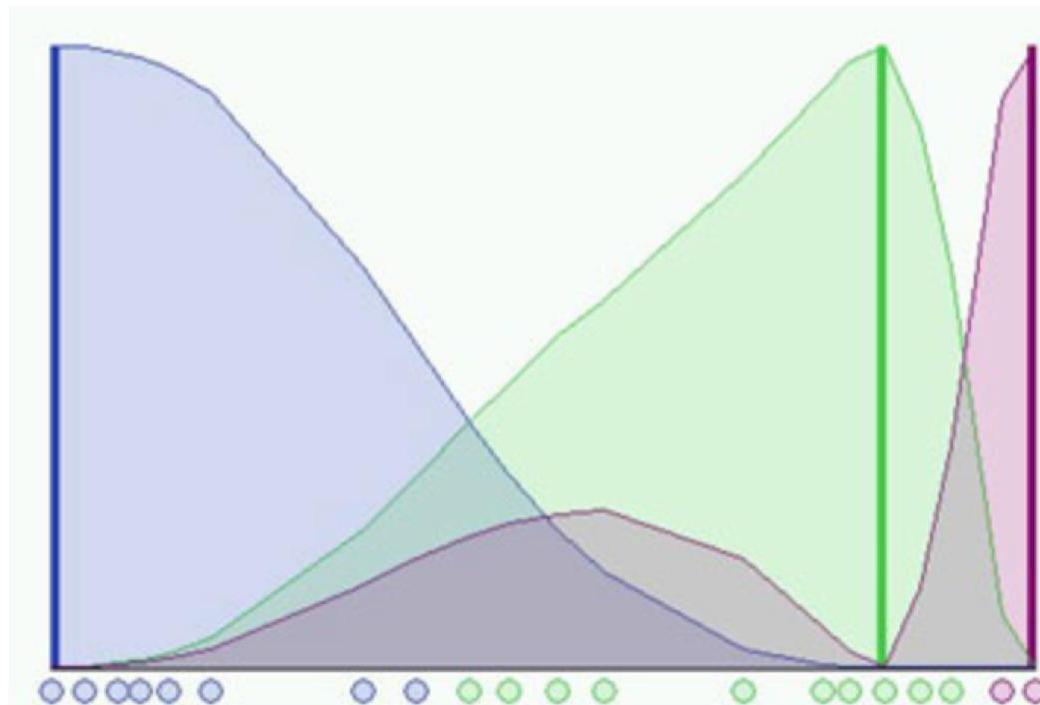
$m > 1$ è un parametro che definisce quanto l'appartenenza ai diversi cluster debba essere sfumata. Valore tipico $m = 2$.

Fuzzy K-means (soft)

Caso di un'applicazione monodimensionale

$N = 20$, $c = 3$ per inizializzare l'algoritmo

Le figure seguenti mostrano il valore dell'appartenenza per ciascun dato e per ciascun cluster. Il colore dei dati è quello del cluster più vicino in base alla funzione di appartenenza.



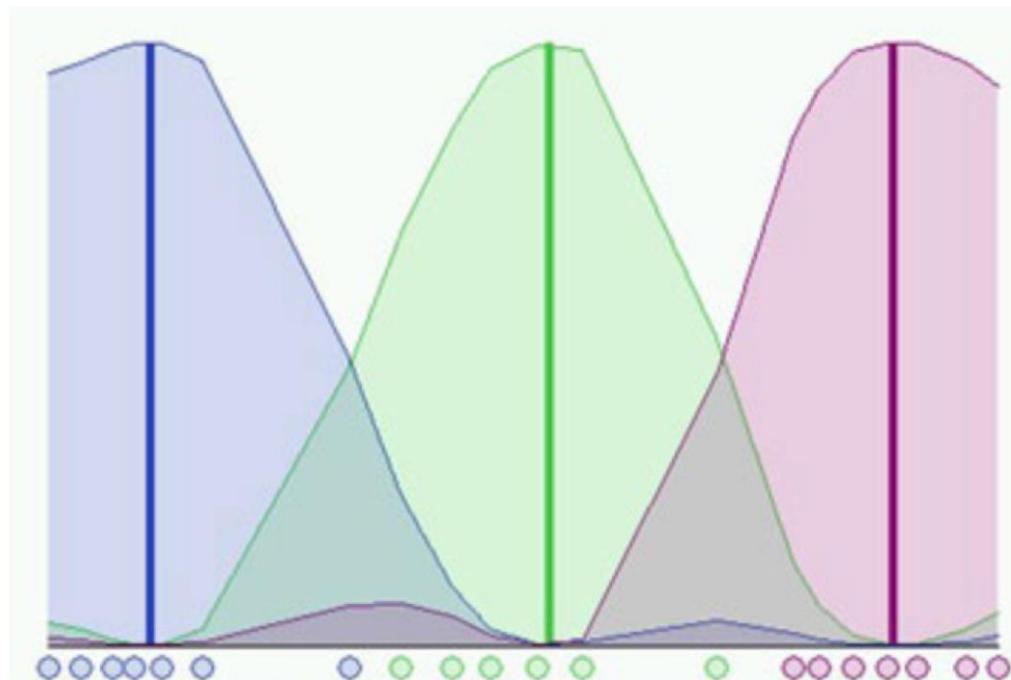
Condizione iniziale in cui la distribuzione fuzzy dipende dalla particolare posizione dei cluster

Fuzzy K-means (soft)

Caso di un'applicazione monodimensionale

$N = 20$, $c = 3$ per inizializzare l'algoritmo

Le figure seguenti mostrano il valore dell'appartenenza per ciascun dato e per ciascun cluster. Il colore dei dati è quello del cluster più vicino in base alla funzione di appartenenza.



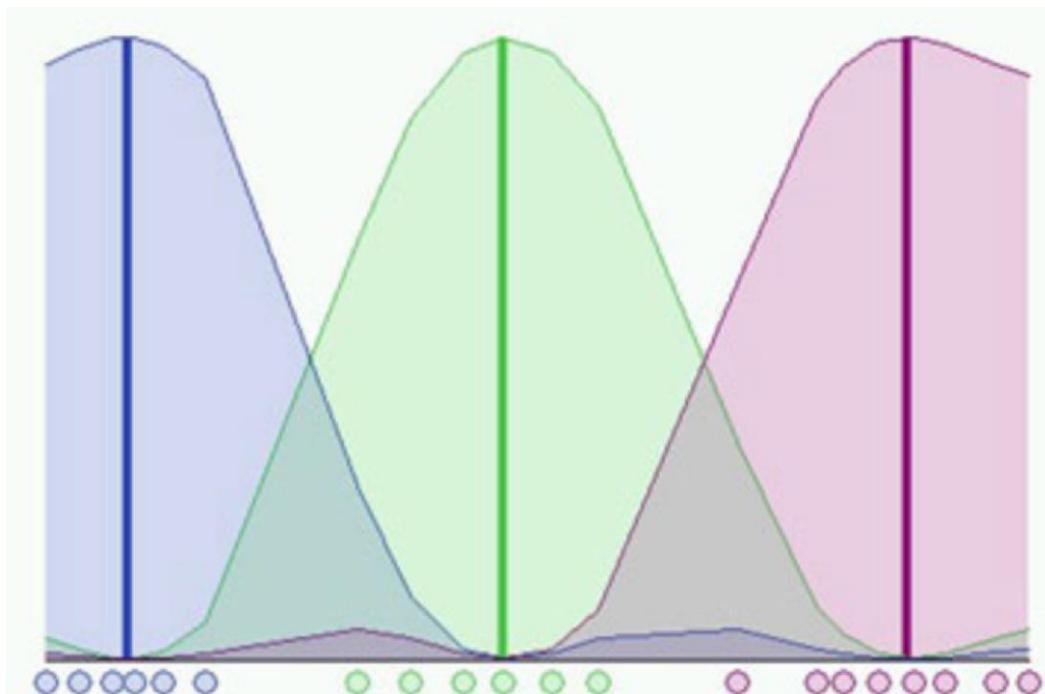
Condizione finale raggiunta
dopo 8 iterazioni ($m = 2$)

Fuzzy K-means (soft)

Caso di un'applicazione monodimensionale

$N = 20$, $c = 3$ per inizializzare l'algoritmo

Le figure seguenti mostrano il valore dell'appartenenza per ciascun dato e per ciascun cluster. Il colore dei dati è quello del cluster più vicino in base alla funzione di appartenenza.



Condizione finale raggiunta
dopo 37 iterazioni ($m = 2$)

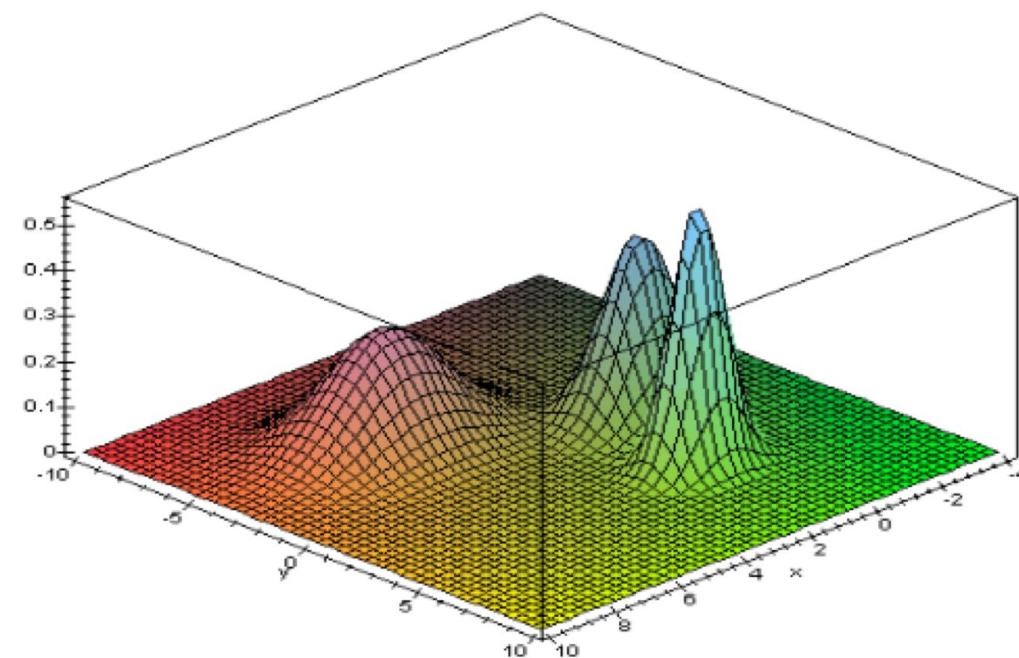
Fuzzy K-means | Vantaggi / Svantaggi

Rispetto al K-means, la variante fuzzy fornisce a volte una convergenza più robusta verso la soluzione finale.

D'altro canto uno svantaggio è legato al fatto che l'appartenenza di un pattern a un cluster dipende implicitamente dal numero di cluster, e se questo è specificato in modo incorretto si possono ottenere soluzioni non ottimali.

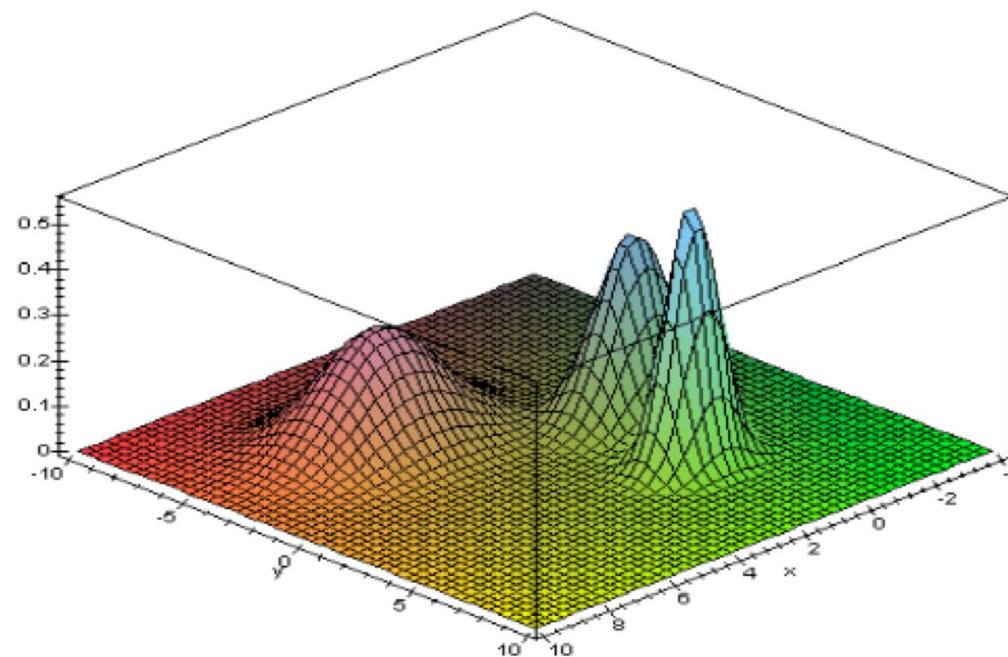
Expectation-Minimization (EM)

Si ipotizza che i pattern siano stati generati da un mix di distribuzioni. Ogni classe ha generato dati in accordo con una specifica distribuzione, ma al termine della generazione i pattern appaiono come prodotti da un'unica distribuzione multimodale.



Expectation-Minimization (EM)

Obiettivo del clustering con EM è risalire (a partire dai pattern del training set) ai parametri delle singole distribuzioni che li hanno generati



Gaussian mixture

A tal fine si ipotizza nota la forma delle distribuzioni e si assume, per semplicità, che esse siano tutte dello stesso tipo.

Il caso più frequente è quello di mix di s distribuzioni multinormali (gaussiane), di cui si vogliono stimare i parametri di definizione (s vettori medi + s matrici di covarianza + s coefficienti alfa)

$$p(\mathbf{x}|\boldsymbol{\Theta}) = \sum_{i=1..s} \alpha_i \cdot p_i(\mathbf{x}|\boldsymbol{\theta}_i) \quad \boldsymbol{\theta}_i = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$$

$\boldsymbol{\Theta} = \alpha_1, \alpha_2 \dots \alpha_s, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \dots \boldsymbol{\theta}_s$ è il vettore di parametri che definisce il mix di distribuzioni,
 $p_i(\mathbf{x}|\boldsymbol{\theta}_i)$ è una multinormale di parametri $\boldsymbol{\theta}_i$

Funzione obiettivo

La stima dei parametri avviene secondo il criterio generale del maximum likelihood (massima verosimiglianza).

Il likelihood L dei parametri Θ , dati i pattern x , corrisponde alla probabilità di aver ottenuto i pattern x , dati i parametri Θ , (inversione):

$$\mathcal{L}(\Theta | \mathcal{X}) = p(\mathcal{X} | \Theta)$$

Funzione obiettivo

Per semplicità, al posto della likelihood, si massimizza il suo logaritmo. Considerando i pattern indipendenti, la probabilità di avere ottenuto i pattern del training set è il prodotto delle probabilità delle singole generazioni:

$$\log p(\mathcal{X}|\boldsymbol{\theta}) = \log \prod_{j=1 \dots n} p(\mathbf{x}_j|\boldsymbol{\theta}) = \sum_{j=1 \dots n} \log \left(\sum_{i=1 \dots s} \alpha_i \cdot p_i(\mathbf{x}_j|\boldsymbol{\theta}_i) \right)$$

Purtroppo questa massimizzazione è molto difficile a causa della sommatoria dentro al logaritmo.

Per eliminare la sommatoria, ci servirebbe sapere, per ogni pattern x_j del training set da quale componente p_i della mixture è stato generato.

EM: soluzione al problema di ottimizzazione della funzione obiettivo

$$\log p(\mathcal{X}|\boldsymbol{\Theta}) = \log \prod_{j=1 \dots n} p(\mathbf{x}_j|\boldsymbol{\Theta}) = \sum_{j=1 \dots n} \log \left(\sum_{i=1 \dots s} \alpha_i \cdot p_i(\mathbf{x}_j|\boldsymbol{\theta}_i) \right)$$

dove $p(\mathbf{x}|\boldsymbol{\Theta}) = \sum_{i=1 \dots s} \alpha_i \cdot p_i(\mathbf{x}|\boldsymbol{\theta}_i)$ $\boldsymbol{\theta}_i = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$

A tal fine si utilizza EM, che è un approccio iterativo nato per il calcolo del maximum likelihood nel caso in cui i dati a disposizione $X = x_1, x_2 \dots x_n$ siano incompleti per la mancanza di alcuni valori y .

Ogni pattern completo $z_j = x_j, y_j$, con $j=1 \dots n$ è costituito da due parti di cui solo la prima è nota.

Nel caso di interesse (derivare i parametri di una gaussian mixture) i dati sono completi ma le potenzialità di EM sono utilizzate per considerare le componenti più ignote che hanno generato i singoli pattern come valori y non noti di dati a disposizione completi.

EM: caso generale

Obiettivo di EM è dunque la massimizzazione del log-likelihood dei dati completi o log-likelihood-completo così definito:

$$\log \mathcal{L}(\Theta | \mathcal{Z}) = \log \mathcal{L}(\Theta | X, y) = p(X, y | \Theta)$$

Per risolvere vengono eseguiti iterativamente (fino a convergenza) due passi, Expectation e Maximization.

Expectation: viene calcolato il valore atteso $E(\cdot)$ del log-likelihood-completo, dato il training set X e una stima dei parametri Θ_g ottenuti all'iterazione precedente:

$$Q(\Theta | \Theta^g) = E(\log p(X, y | \Theta) | X, \Theta^g)$$

EM: caso generale

Il valore atteso (media) è calcolato rispetto alla variabile y , governata dalla distribuzione f :

$$E(\log p(\mathcal{X}, y|\Theta) | \mathcal{X}, \Theta^g) = \int_{\mathbf{y} \in \Psi} \log p(\mathcal{X}, y|\Theta) \cdot f(\mathbf{y}|\mathcal{X}, \Theta^g) d\mathbf{y}$$

Maximization: viene calcolato il valore dei parametri che massimizzano il valore atteso calcolato al passo di Expectation:

$$\Theta^{g+1} = \underset{\Theta}{\operatorname{argmax}} Q(\Theta|\Theta^g)$$

EM: caso specifico

Si assume l'esistenza di una componente nascosta y (per ogni pattern) che indica quale delle s_j distribuzioni gaussiane ha generato il pattern x_j .

Il log-likelihood-completo assume ora la forma:

$$\log \mathcal{L}(\Theta | \mathcal{X}, \mathcal{Y}) = \sum_{j=1 \dots n} \log \left(\alpha_{y_j} \cdot p_{y_j}(\mathbf{x}_j | \Theta_{y_j}) \right)$$

sebbene gli y non siano noti, una loro stima (o meglio una stima j della loro distribuzione) può essere derivata (teorema di Bayes) dai parametri Θ^g disponibili all'iterazione (g) corrente.

EM: caso specifico

Per un singolo y_j (e corrispondente x_j):

$$P(y_j = k | \mathbf{x}_j, \Theta^g) = P(k | \mathbf{x}_j, \Theta^g) = \frac{\alpha_k^g \cdot p_k(\mathbf{x}_j | \Theta_k^g)}{\sum_{i=1 \dots s} \alpha_i^g \cdot p_i(\mathbf{x}_j | \Theta_i^g)}$$

Per un generico vettore \mathbf{y} di osservazioni:

$$P(\mathbf{y} | \mathcal{X}, \Theta^g) = \prod_{j=1 \dots n} P(y_j | \mathbf{x}_j, \Theta^g)$$

EM: caso specifico

Il valore atteso Q delle slide precedenti può essere scritto come:

$$\begin{aligned} Q(\Theta|\Theta^g) &= \sum_{\mathbf{y} \in \Psi} \log \mathcal{L}(\Theta|\mathcal{X}, \mathbf{y}) \cdot P(\mathbf{y}|\mathcal{X}, \Theta^g) = \\ &= \sum_{\mathbf{y} \in \Psi} \left(\sum_{j=1 \dots n} \log \left(\alpha_{y_j} \cdot p_{y_j}(\mathbf{x}_j | \Theta_{y_j}) \right) \cdot \prod_{j=1 \dots n} P(y_j | \mathbf{x}_j, \Theta^g) \right) \end{aligned}$$

EM: formule incrementali

A seguito di alcuni passaggi (non banali) durante i quali l'ottimizzazione è eseguita eguagliando a zero le derivate parziali rispetto ai parametri incogniti, si ottengono le seguenti formule per l'aggiornamento incrementale dei parametri:

$$P(C_k | \mathbf{x}_j, \boldsymbol{\theta}^g) = \frac{\alpha_k^g \cdot p(\mathbf{x}_j | \boldsymbol{\theta}_k^g)}{\sum_{i=1 \dots s} \alpha_i^g \cdot p(\mathbf{x}_j | \boldsymbol{\theta}_i^g)} \quad \text{eq (1)}$$

$$\alpha_k^{g+1} = \frac{1}{n} \sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g) \quad \text{eq (2)}$$

$$\boldsymbol{\mu}_k^{g+1} = \frac{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g) \cdot \mathbf{x}_j}{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g)} \quad \text{eq 3)$$

$$\boldsymbol{\Sigma}_k^{g+1} = \frac{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g) (\mathbf{x}_j - \boldsymbol{\mu}_k^{g+1}) (\mathbf{x}_j - \boldsymbol{\mu}_k^{g+1})^t}{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g)} \quad \text{eq (4)}$$

EM: formule incrementali

Intuitivamente, trattando le componenti della mixture come cluster:

Eq (1), stessa delle slide precedenti: indica la probabilità di appartenenza di x_j al cluster C_k . Calcolata sfruttando l'inversione di Bayes $p(\cdot | \Theta_k)$ è una multinormale con parametri Θ_k

Eq (2). Calcola il peso del cluster C in base alla somma delle k appartenenze a C di tutti i pattern del training set

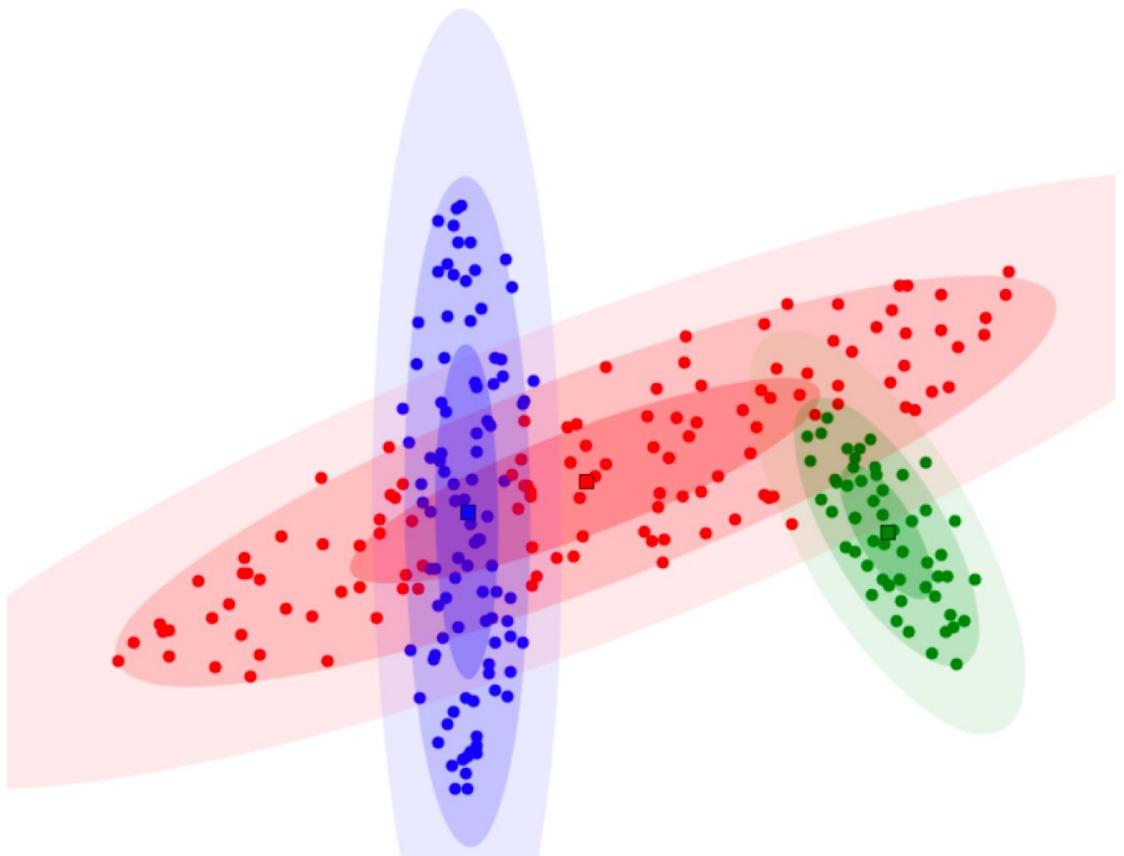
Eq (3) e (4). Calcolano media e matrice di covarianza del cluster C pesando i pattern rispetto al loro grado di appartenenza a C

$$P(C_k | \mathbf{x}_j, \Theta^g) = \frac{\alpha_k^g \cdot p(\mathbf{x}_j | \Theta_k^g)}{\sum_{i=1 \dots n} \alpha_i^g \cdot p(\mathbf{x}_j | \Theta_i^g)} \quad \text{eq (1)}$$

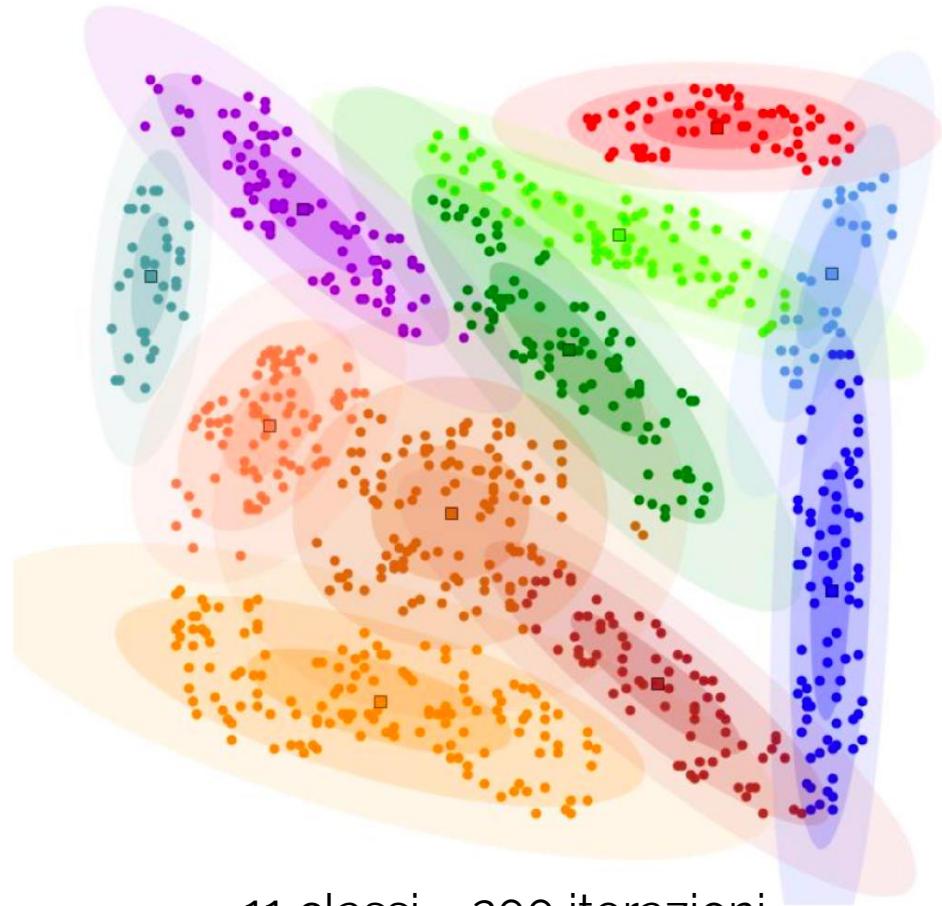
$$\alpha_k^{g+1} = \frac{1}{n} \sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \Theta_k^g) \quad \text{eq (2)}$$

$$\boldsymbol{\mu}_k^{g+1} = \frac{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \Theta_k^g) \cdot \mathbf{x}_j}{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \Theta_k^g)} \quad \text{eq 3)$$

$$\boldsymbol{\Sigma}_k^{g+1} = \frac{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \Theta_k^g) (\mathbf{x}_j - \boldsymbol{\mu}_k^{g+1}) (\mathbf{x}_j - \boldsymbol{\mu}_k^{g+1})^t}{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \Theta_k^g)} \quad \text{eq (4)}$$



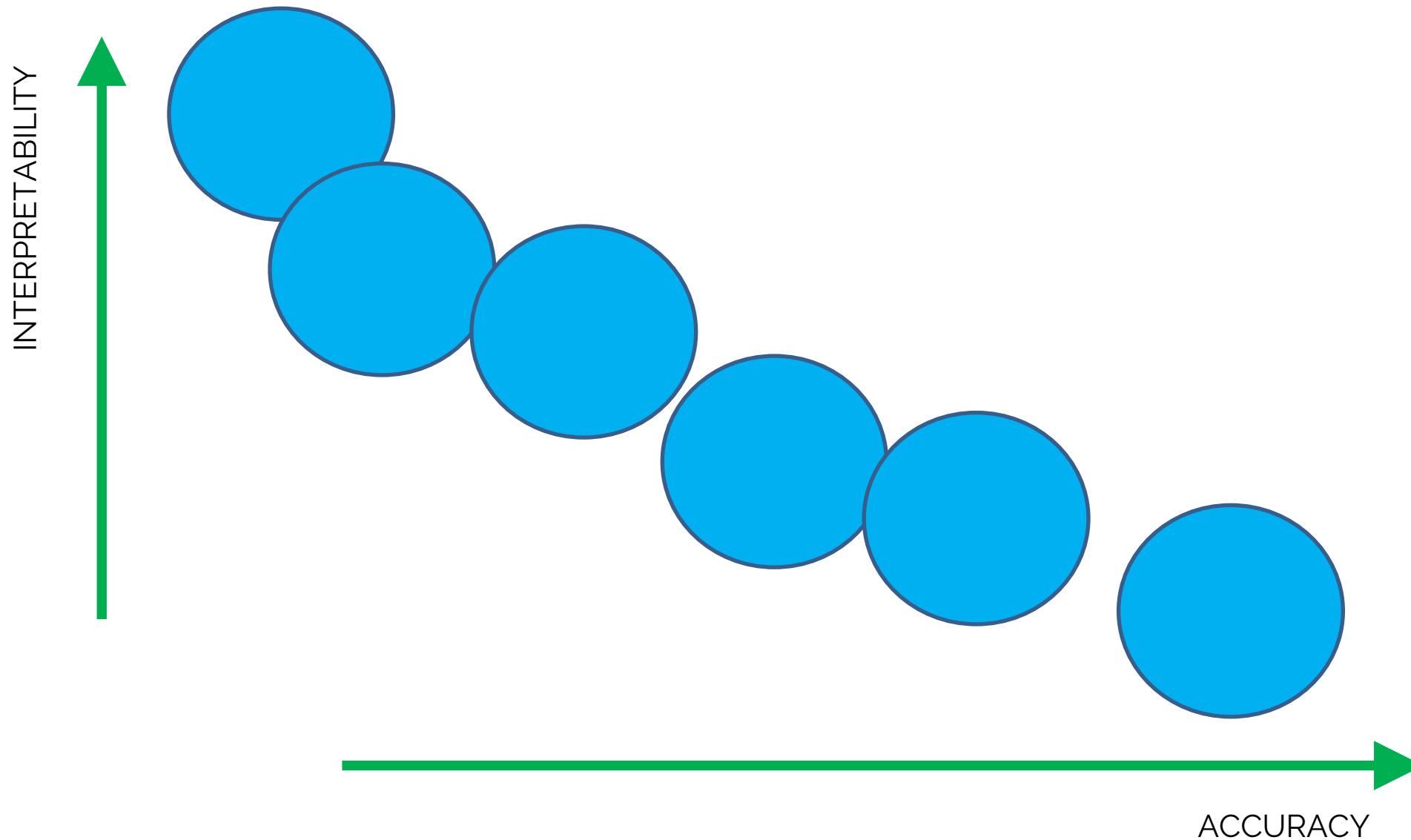
3 classi – 40 iterazioni



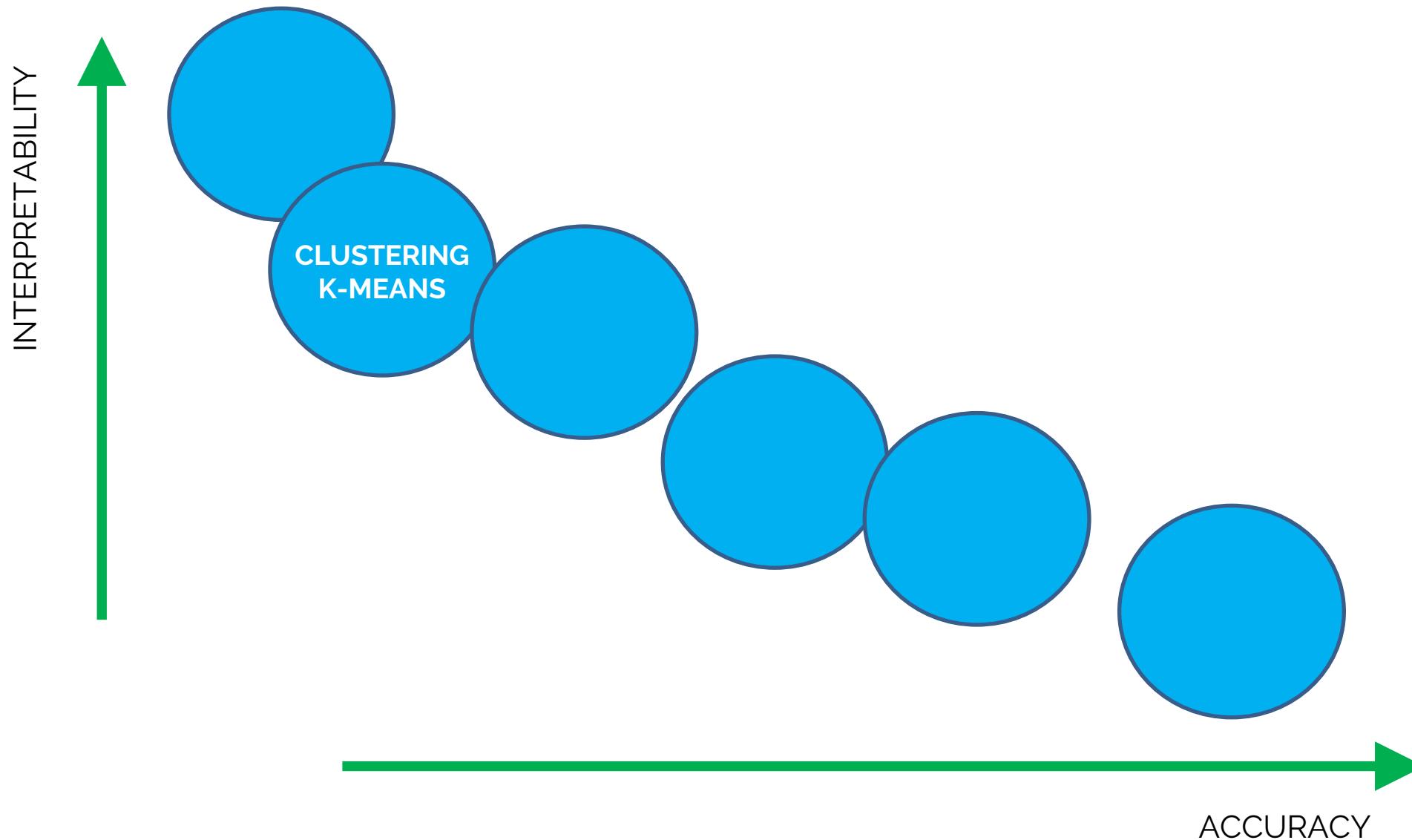
11 classi – 200 iterazioni

EM è in grado di identificare nuvole di punti con forme ellissoidali e capsule, a differenza di K-means e fuzzy K-means

Interpretability-Accuracy TRADEOFF



Interpretability-Accuracy TRADEOFF



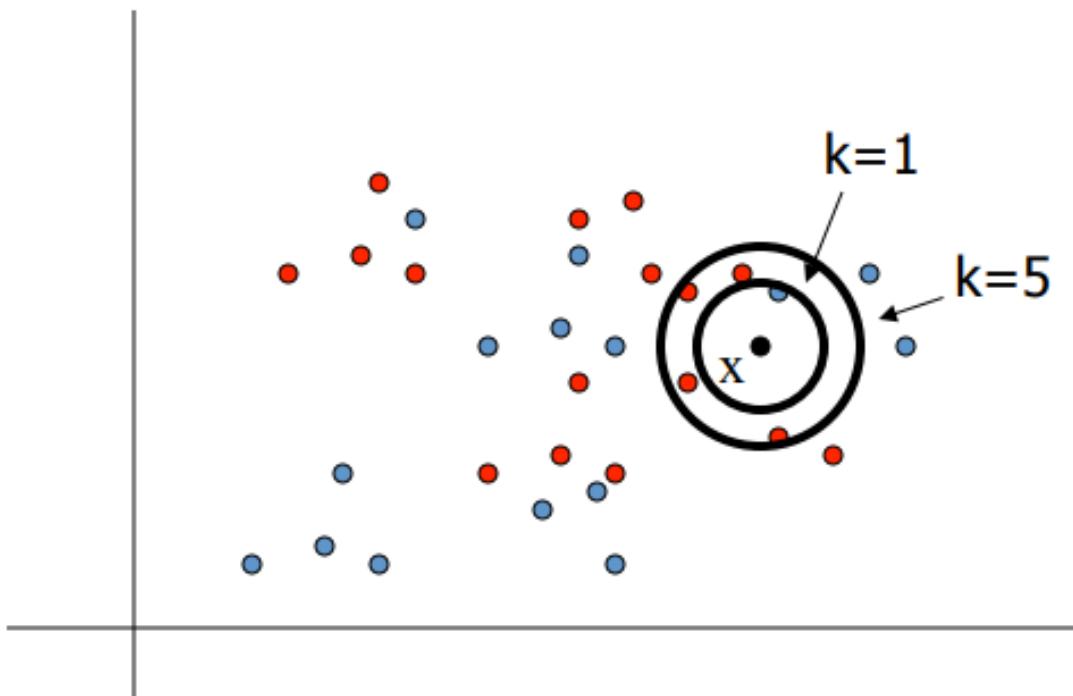
CLASSIFICATION

Nearest Neighbors

WITH SOME MATERIALS FROM DAVID SONTAG, VIBHAV GOGATE, CARLOS
GUESTRIN, MEHRYAR MOHRI, LUKE ZETTLEMOYER, DHILIP SUBRAMANIAN

Nearest Neighbors

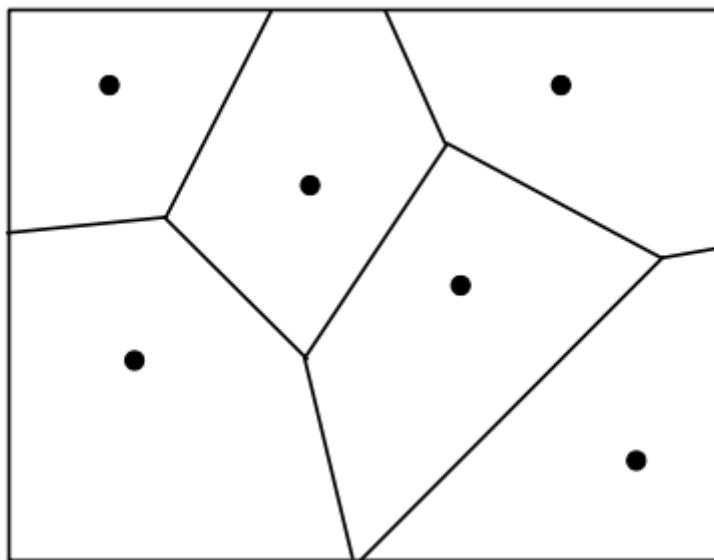
To classify a new input vector x , examine the k -closest training data points to x and assign the object to the most frequently occurring class



Common values for $k = 3, 5$

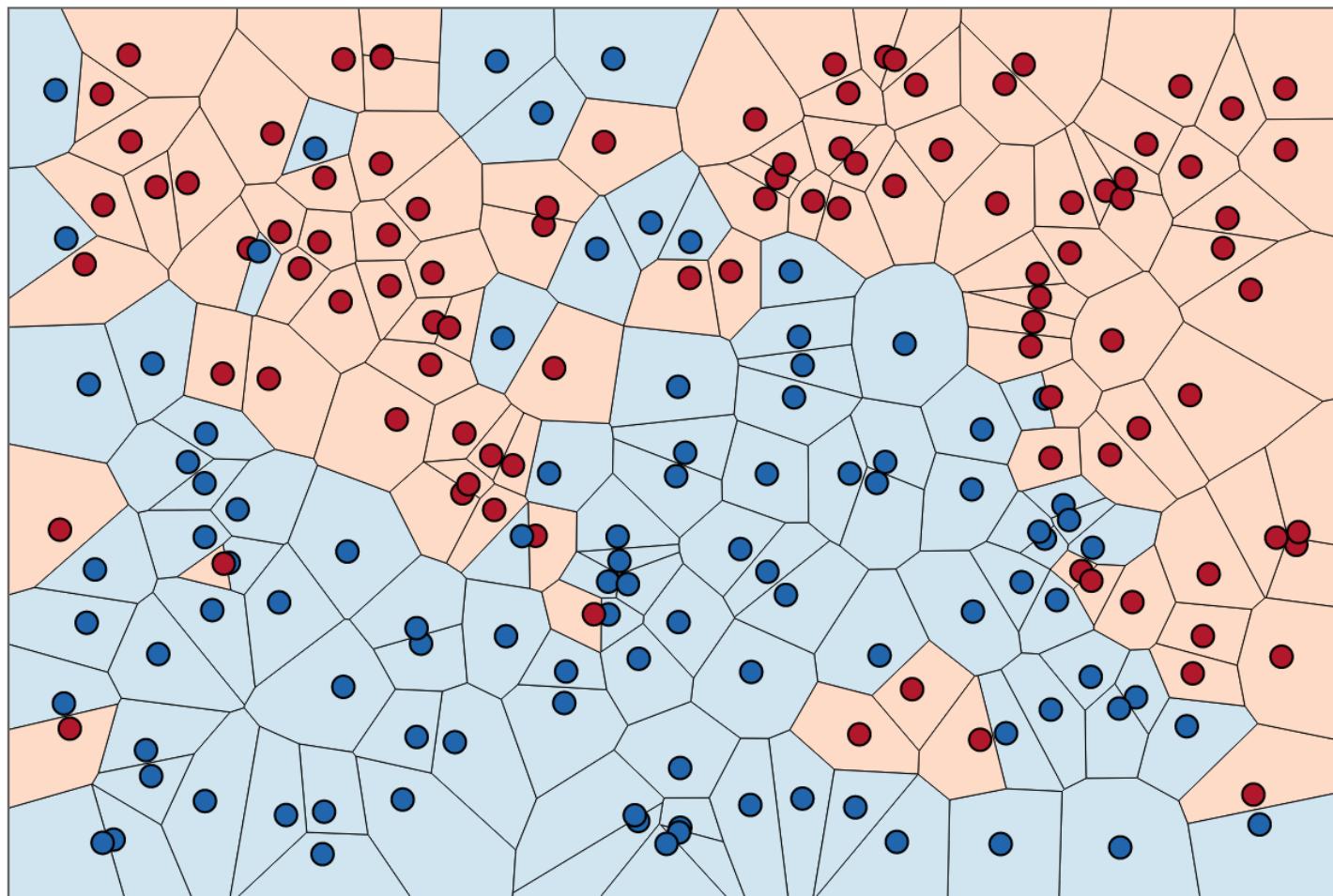
Nearest Neighbors

The nearest neighbor algorithm does not explicitly compute decision boundaries.



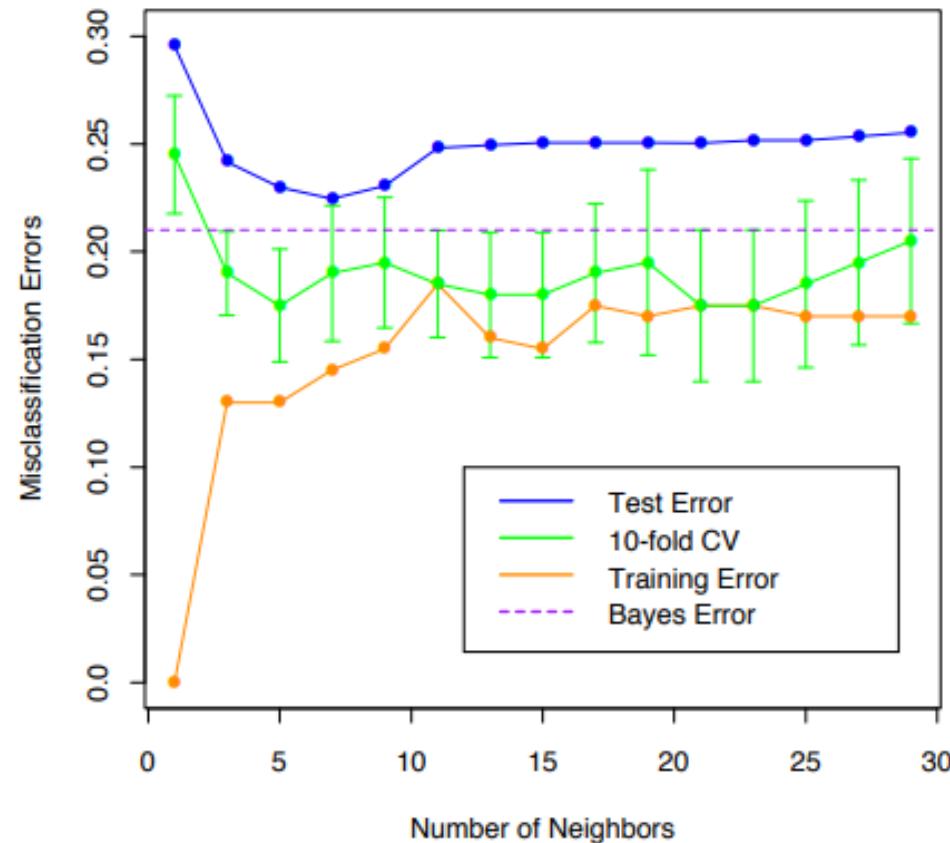
The decision boundaries form a subset of the Voronoi diagram for the training data.

Nearest Neighbors

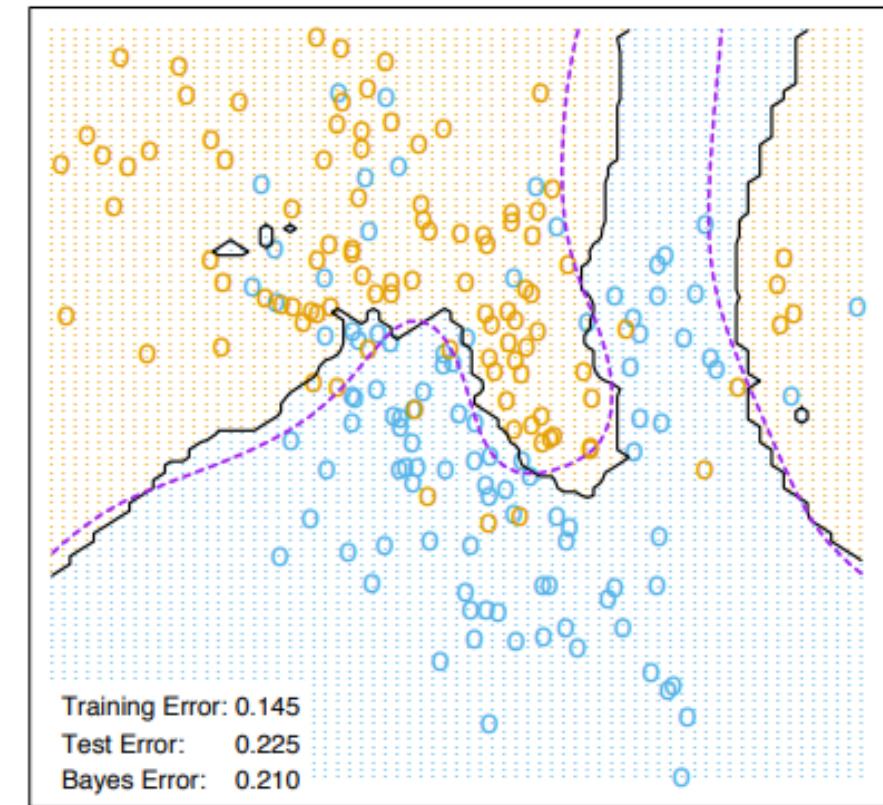


The more examples that are stored,
the more complex the decision boundaries can become

Nearest Neighbors



7-Nearest Neighbors



Nearest Neighbors

CONSIDERATIONS

- Instance map to points in R^n
- Less than 20 attributes per instance
- Lots of training data

Advantages

- Training is very fast
- Learn complex target functions
- Do not lose information

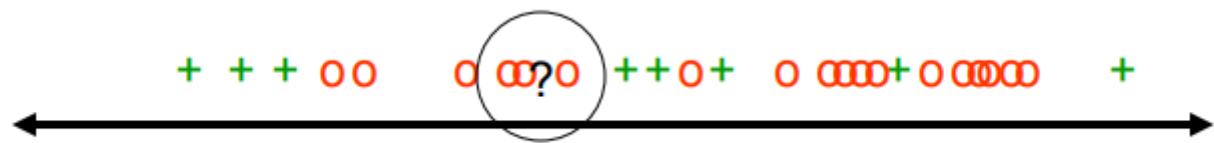
Disadvantages

- Slow at classification
- Easily fooled by irrelevant attributes
(see next slides)

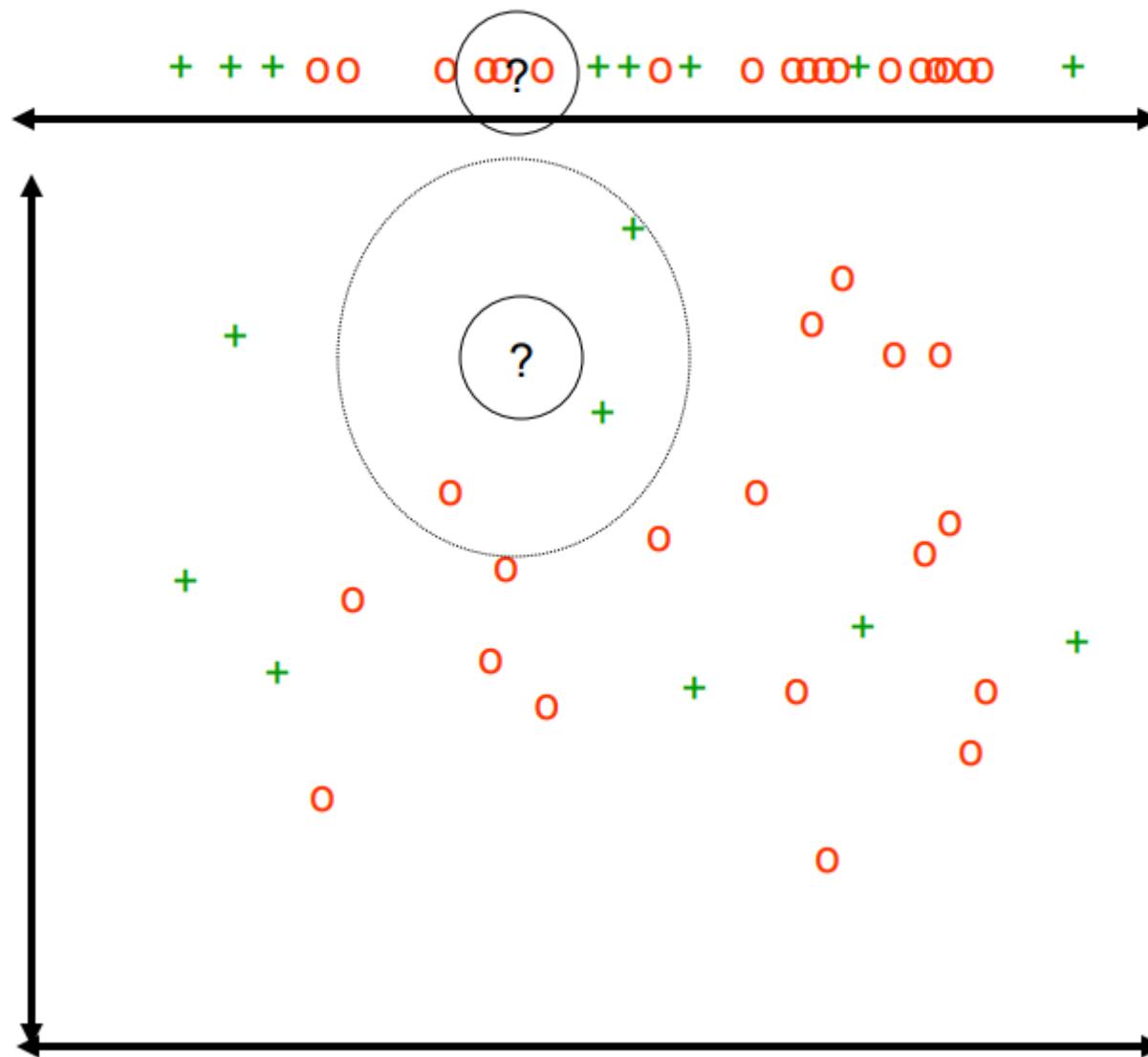
ISSUES

- Distance measure – Most common: Euclidean
- Choosing k – Increasing k reduces variance, increases bias
- For high dimensional space, the nearest neighbor may be very distant
- Memory-based technique. Must make a pass through the data for each classification (think about big data!)

Nearest Neighbors



Nearest Neighbors



Nearest Neighbors

Notation: object with p measurements

$$\mathbf{x}^i = (x_1^i, x_2^i, \dots, x_p^i)$$

Most common distance metric is the Euclidean Distance:

$$d_E(x^i, x^j) = \left(\sum_{k=1}^p (x_k^i - x_k^j)^2 \right)^{\frac{1}{2}}$$

The use of the Euclidean Distance is appropriate when different measurements are commensurate; each is variable measured in the same units. If the measurements are not comparable, this may affect training/classification results.

Nearest Neighbors

STANDARDIZATION

When variables are not commensurate, we can standardize them by dividing by the sample standard deviation.

The estimate for the standard deviation of x_k is

$$\hat{\sigma}_k = \left(\frac{1}{n} \sum_{i=1}^n (x_k^i - \bar{x}_k)^2 \right)^{\frac{1}{2}}$$

where \bar{x}_k is the sample mean

$$\bar{x}_k = \frac{1}{n} \sum_{i=1}^n x_k^i$$

Nearest Neighbors

WEIGHTED EUCLIDEAN DISTANCE

If we have a-priori knowledge of a relative importance of each variable, we can weight them by a factor

$$d_{WE}(i, j) = \left(\sum_{k=1}^p w_k (x_k^i - x_k^j)^2 \right)^{\frac{1}{2}}$$

Nearest Neighbors

Nearest neighbor breaks down in high dimensional spaces because the "neighborhood" becomes very large.

- Suppose we have 5000 points uniformly distributed in the unit hypercube and we want to apply the 5 nearest neighbor algorithm.
- Suppose our query point is at the origin.
 - 1D
 - we must go at a distance of $5/5000 = 0.001$ on average to capture the 5 nearest neighbors
 - 2D
 - we must go $\sqrt{0.001}$ to get a square that contains 0.001 of the volume
 - Generic D
 - we must go $(0.001)^{1/D}$

... CURSE OF DIMENSIONALITY

Nearest Neighbors

No structured method to find the best value for "K".

- Trial and error approach, assuming that training data is unknown.
- Choosing smaller values for K can be noisy and will have a higher influence on the result.
- Larger values of K will have smoother decision boundaries which mean lower variance but increased bias.
- Larger values of K are computationally expensive.
- Through cross-validation (K can be treated as a hyperparameter...)
- GENERAL PRACTICE: choosing the value of k is $k = \sqrt{N}$ where N stands for the number of samples in your training dataset.
- ODD!

Nearest Neighbors

Advantages:

- Simple to implement
- Flexible to feature/distance choices
- Naturally handles multi-class cases
- Can do well in practice with enough representative data (comparable to more complex models)
- No optimization or training required

Drawbacks:

- Need to determine the value of parameter K (number of nearest neighbors)
- Computation cost is quite high because we need to compute the distance of each query instance to all training samples.
- Storage of data
- Choose a meaningful distance function.

christian.salvatore@iusspavia.it

<https://christiansalvatore.github.io/machinelearning-iuss/>