Machine Learning: Basi e Sue Applicazioni | L1

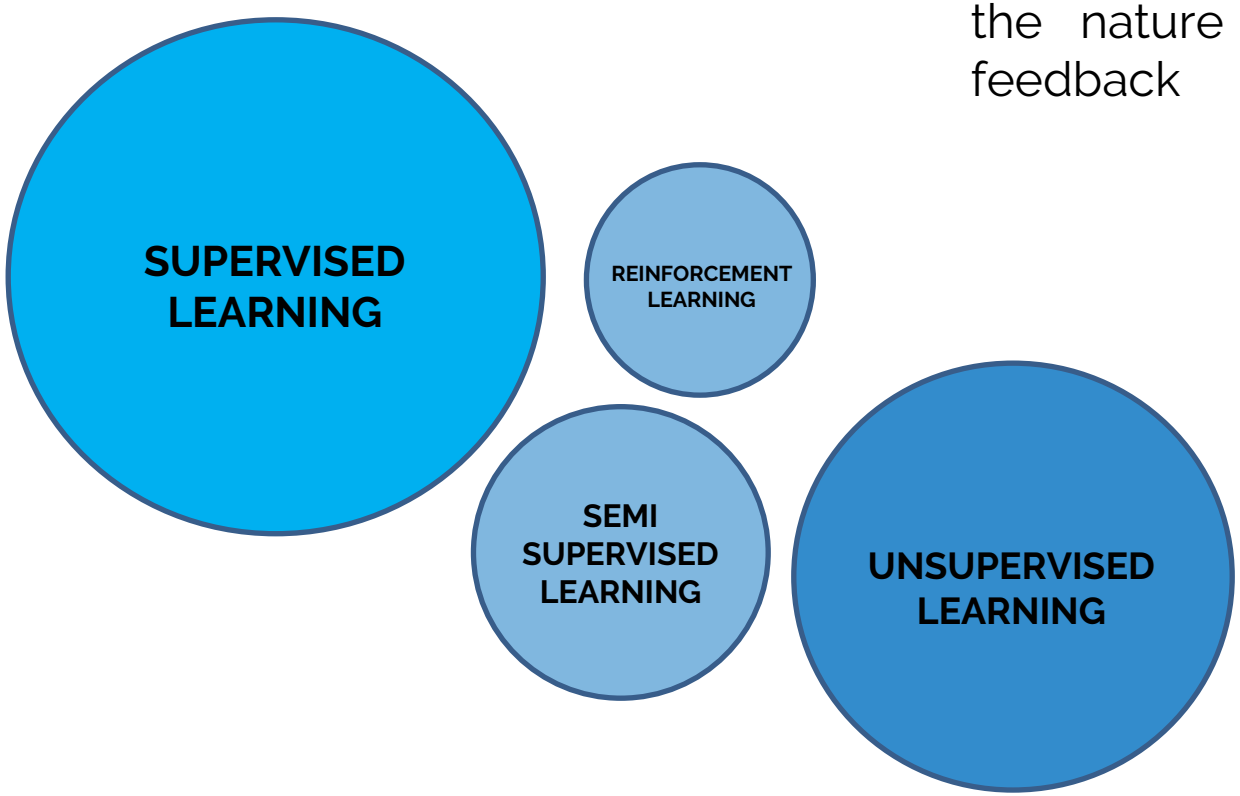
Christian Salvatore
Scuola Universitaria Superiore IUSS Pavia

2021-04-09

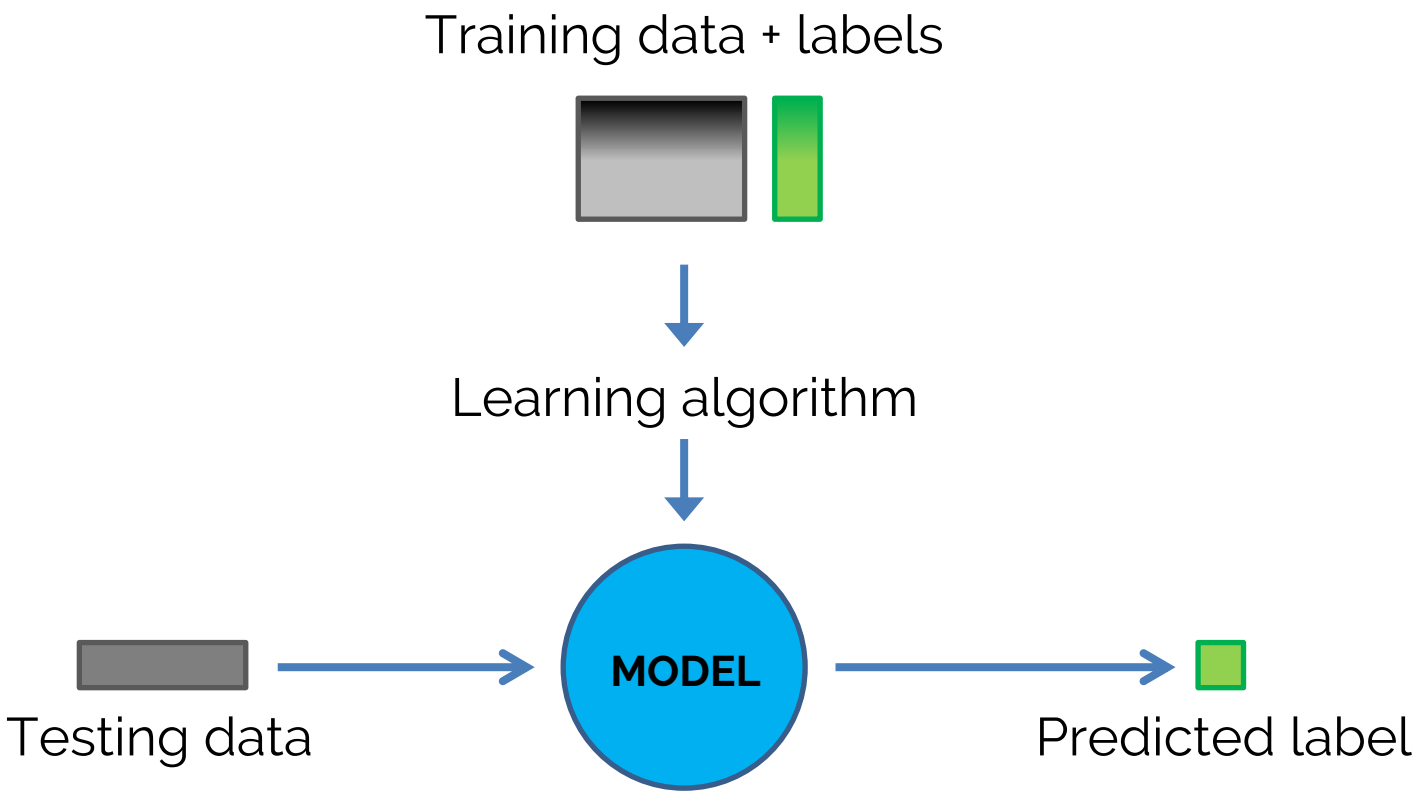
CLASSIFICATION

Machine learning

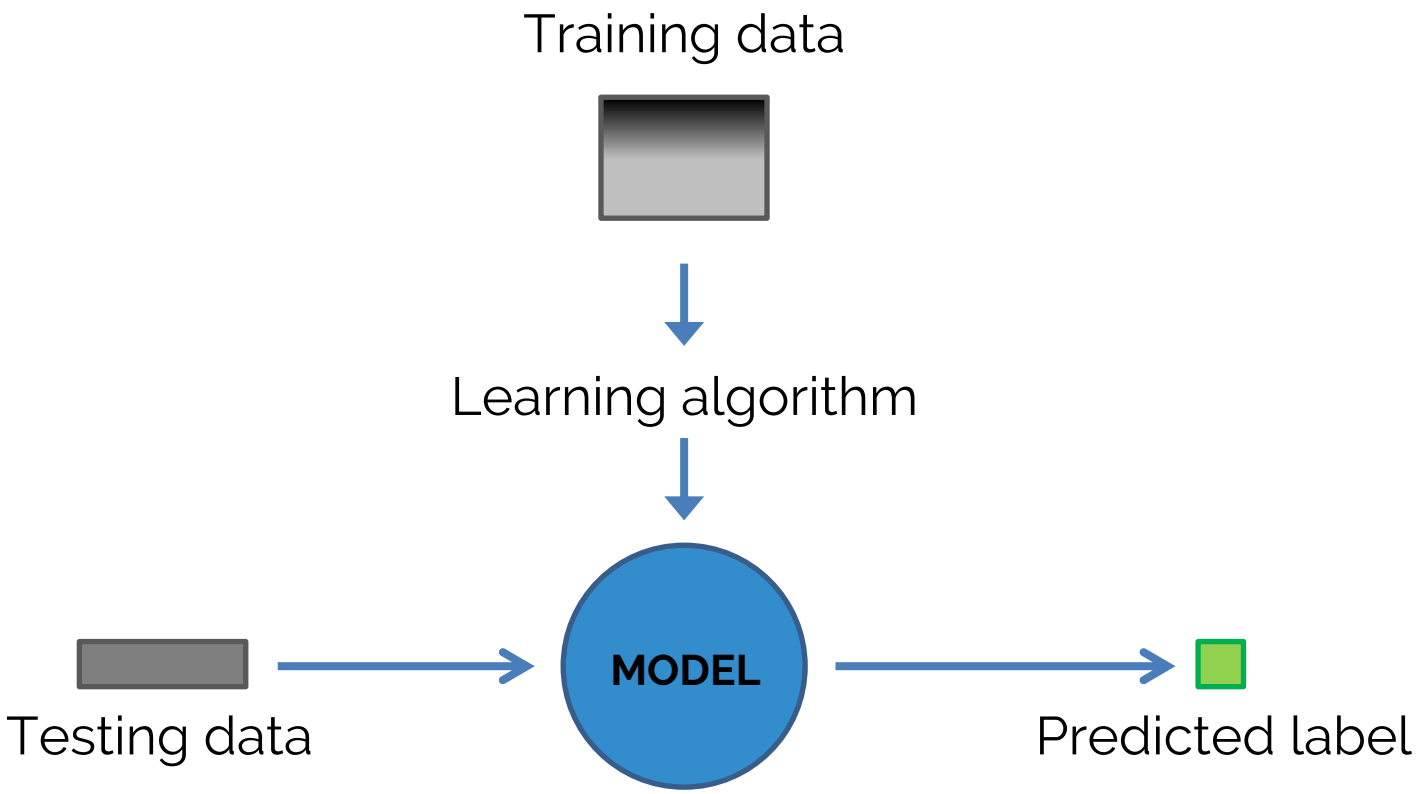
Categorization by the nature of the feedback



Supervised learning

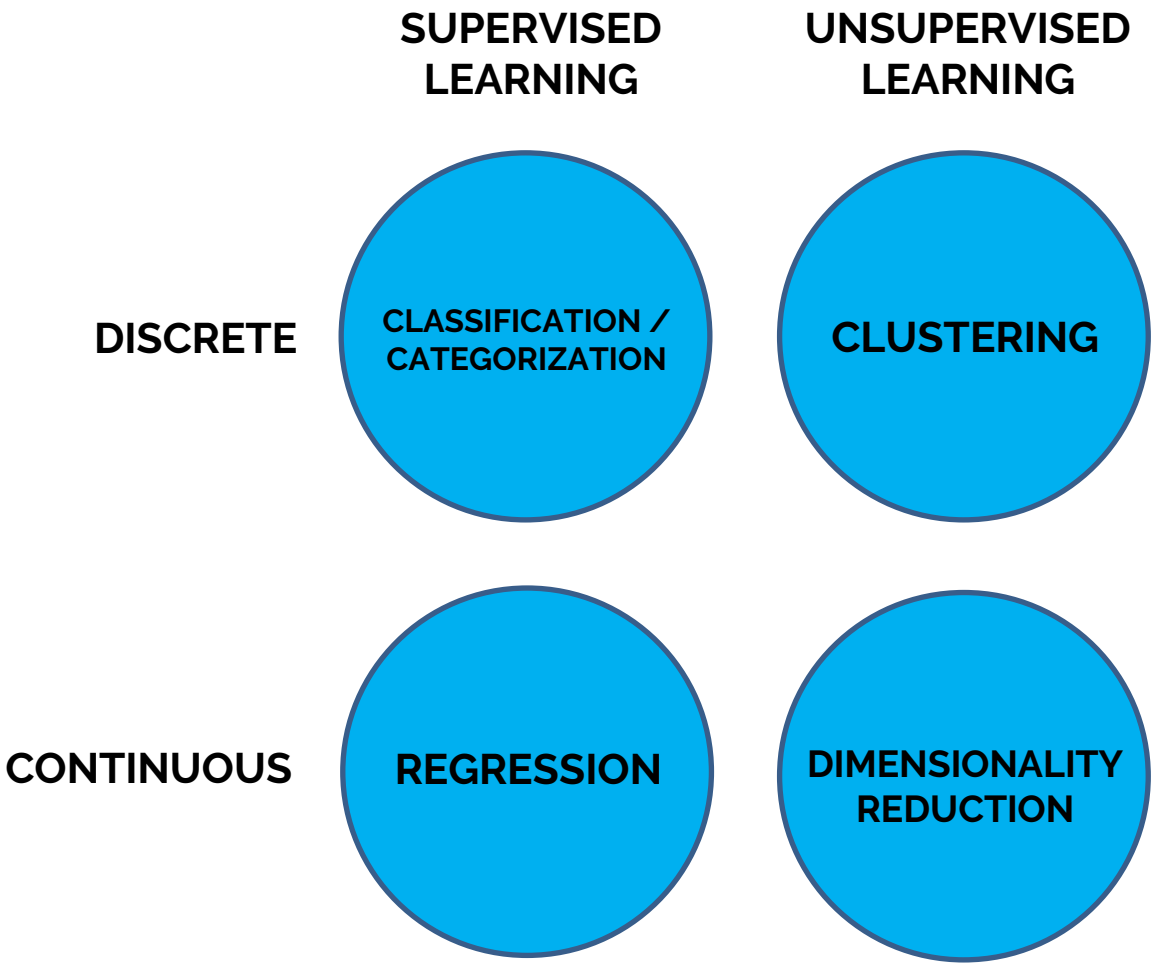


Unsupervised learning

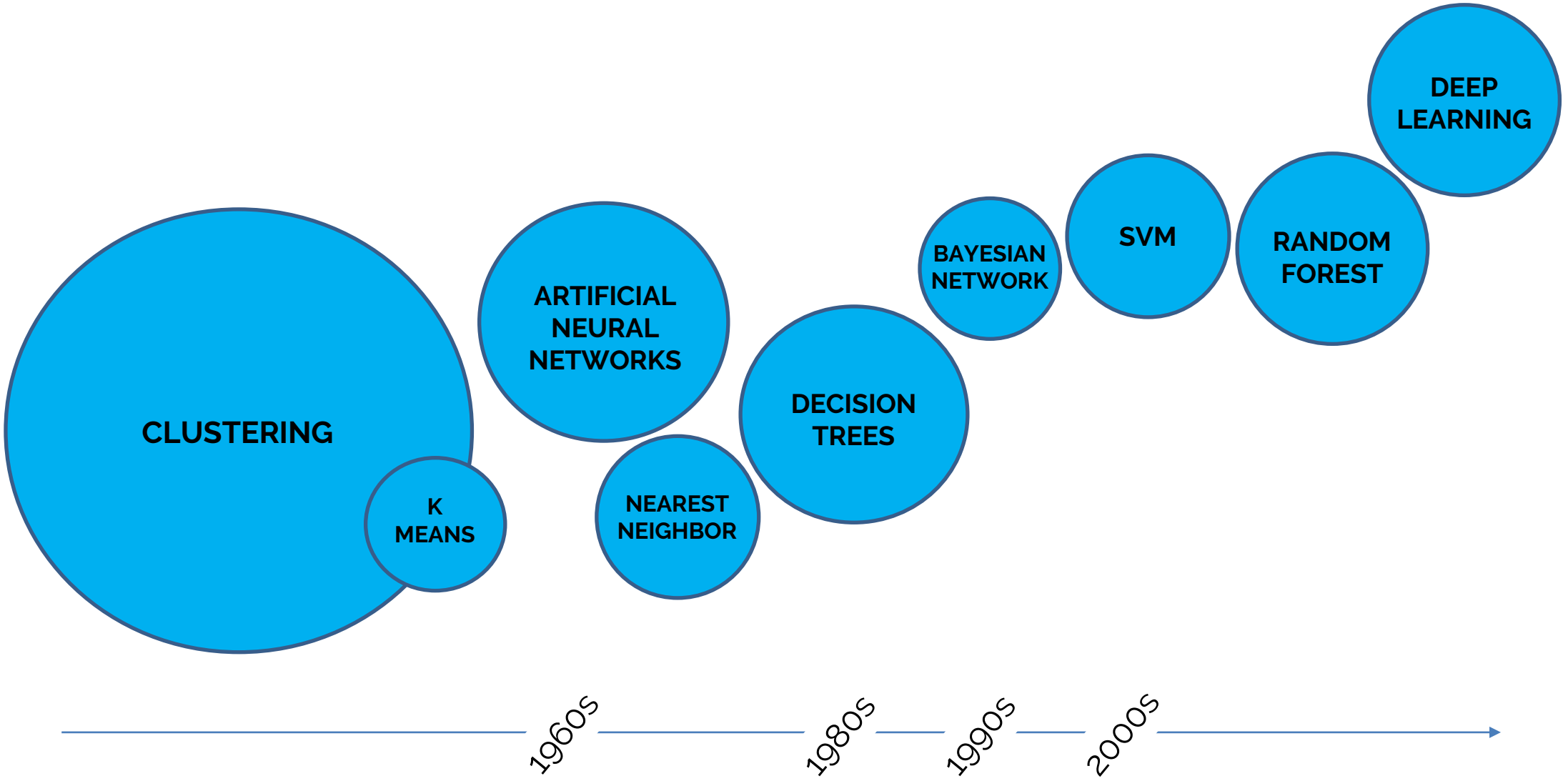


Machine learning

Categorization by the output of the algorithm



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 casi per tutti 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.

Criteri

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 e più bassa;

2) i pattern all'interno dello stesso cluster devono essere tra loro più simili rispetto a pattern appartenenti a cluster diversi

Criteri

- Minimizzazione delle distanze dai centroidi

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

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

dove C_i è l'i-esimo cluster, n_i il numero di pattern che il cluster contiene e 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)

Criteri

- Minimizzazione distanze intra-classe

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

dove f_S è una misura di distanza tra x e il cluster a cui appartiene. Ad esempio:

$$f_s(\mathbf{x}, C_i) = \frac{1}{n_i} \sum_{\mathbf{x}' \in C_i} \|\mathbf{x} - \mathbf{x}'\|^2$$

criterio simile alla minimizzazione distanze dai centroidi

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

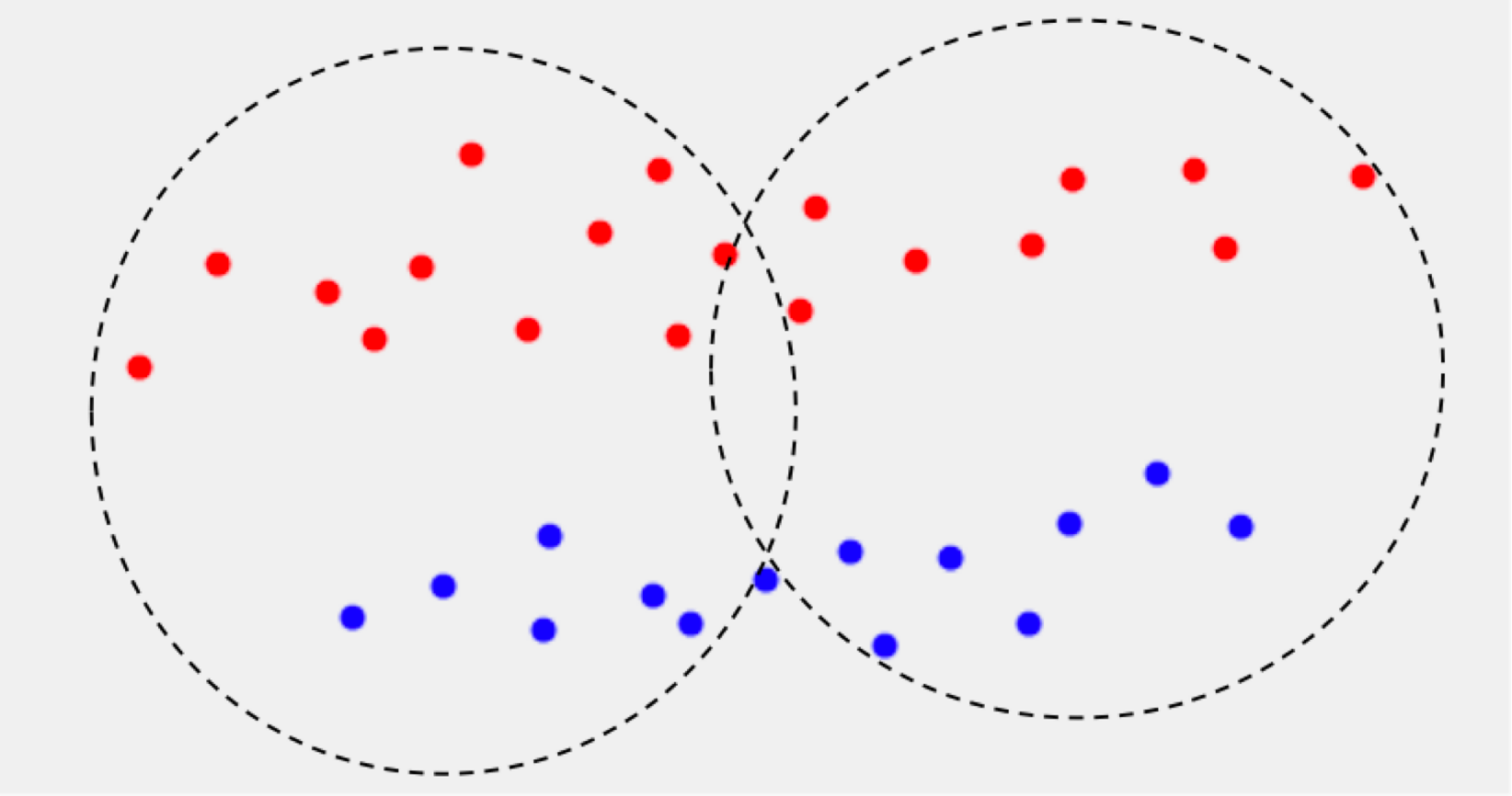
non penalizza cluster allungati affinché f_S assuma valore ridotto, non è necessario che tutti (o gran parte de) i pattern di C siano vicini a x, ma è sufficiente un vicino.

Criteri

C=2

1) Mimizzazione 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.

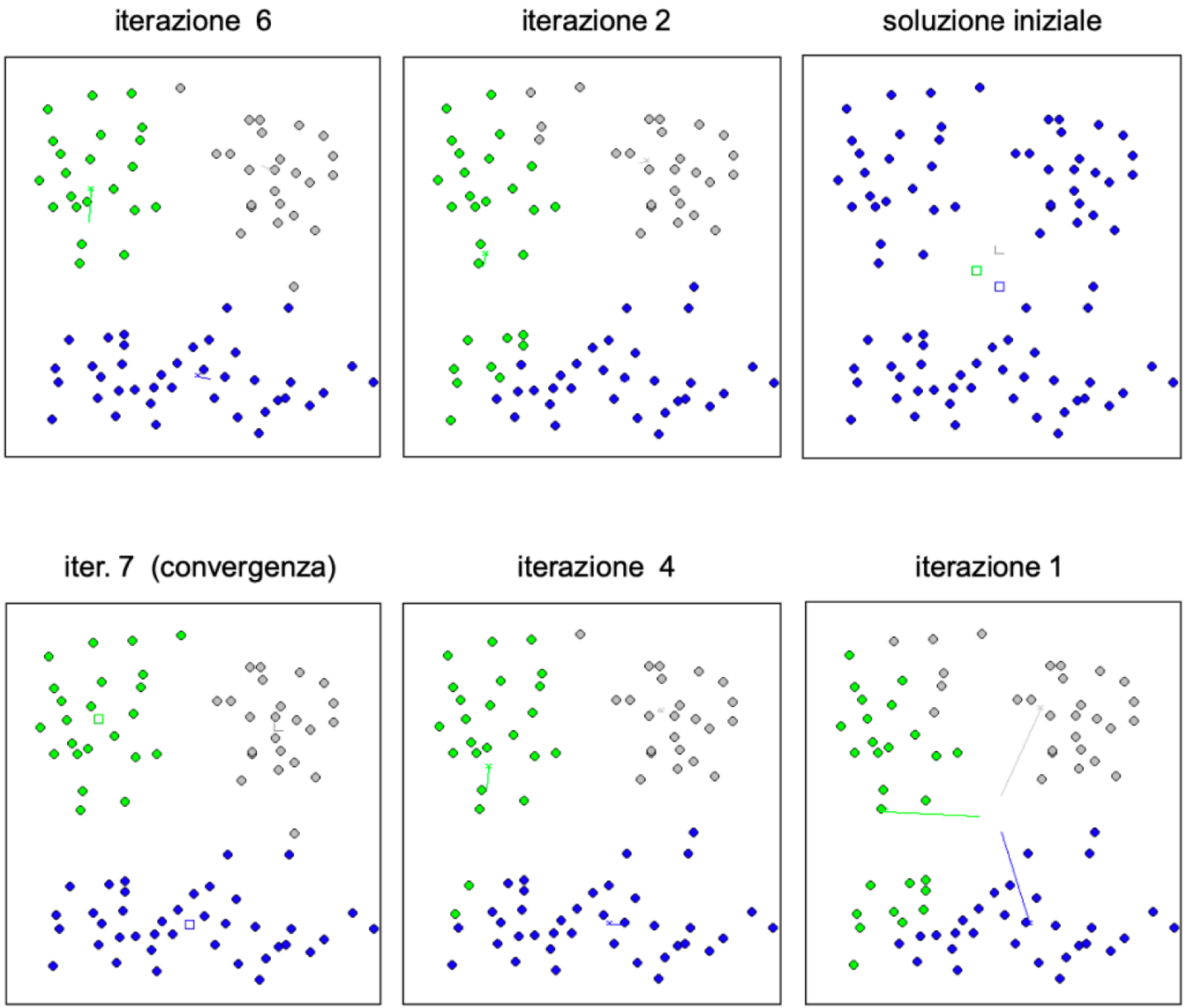
L'assegnazione può diventare esclusiva scegliendo, per ogni pattern, il cluster verso cui il grado di appartenenza è massimo

K-means (hard)

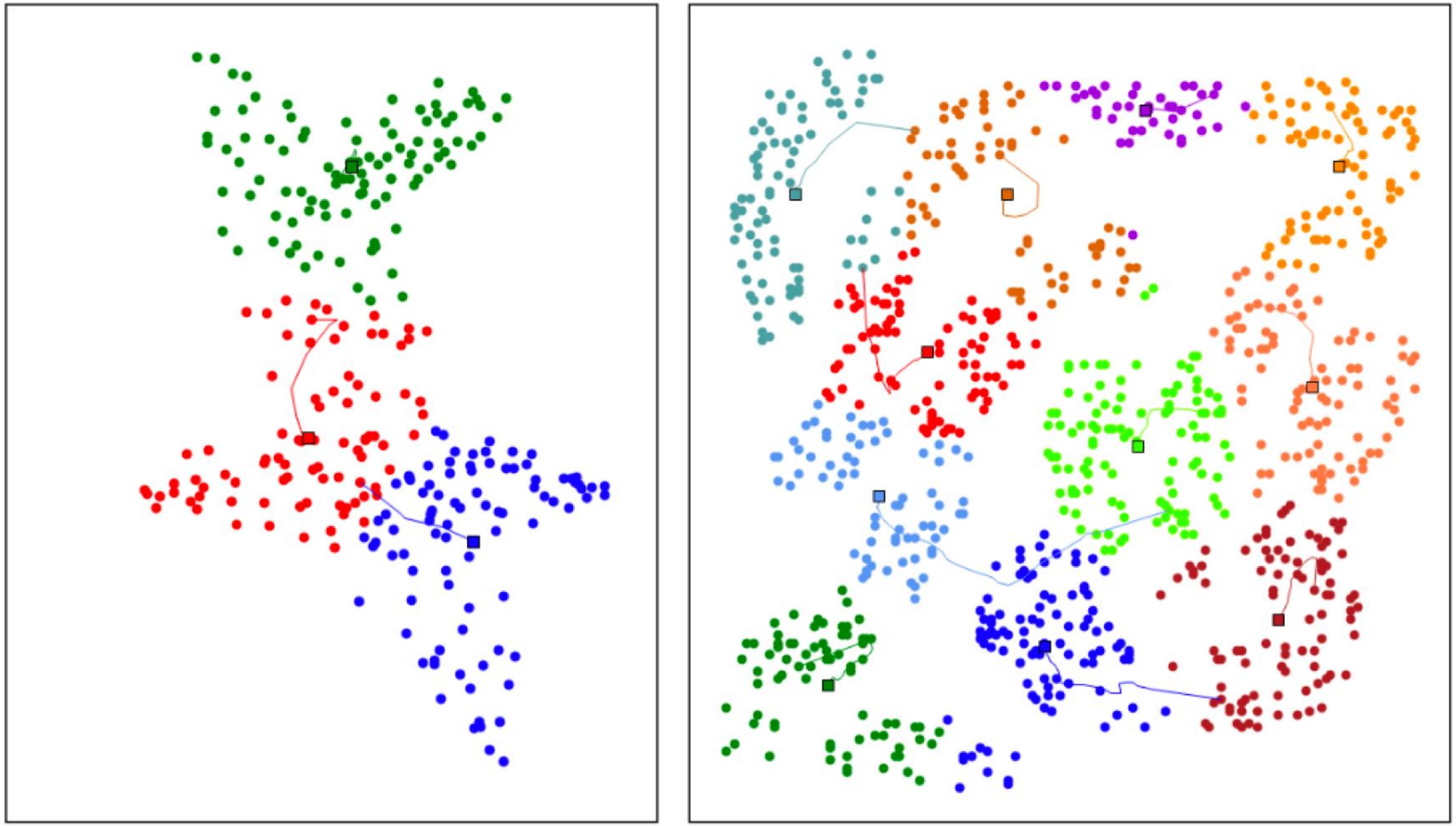
Clustering basato sui centroidi (hard)
- metodo computazionalmente molto semplice e altrettanto semplice da implementare (spesso la prima scelta per risolvere problemi di clustering)
- minimizza le distanze dai centroidi
- richiede in input il numero di cluster c e una soluzione iniziale
- assegna ogni 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-ellissoidali nel caso di distanza di Mahalanobis.

K-means (hard)


iterazione 6
iterazione 2
soluzione iniziale
iter. 7 (convergenza)
iterazione 4
iterazione 1

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 di 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 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:

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

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:

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

Fuzzy K-means (soft)

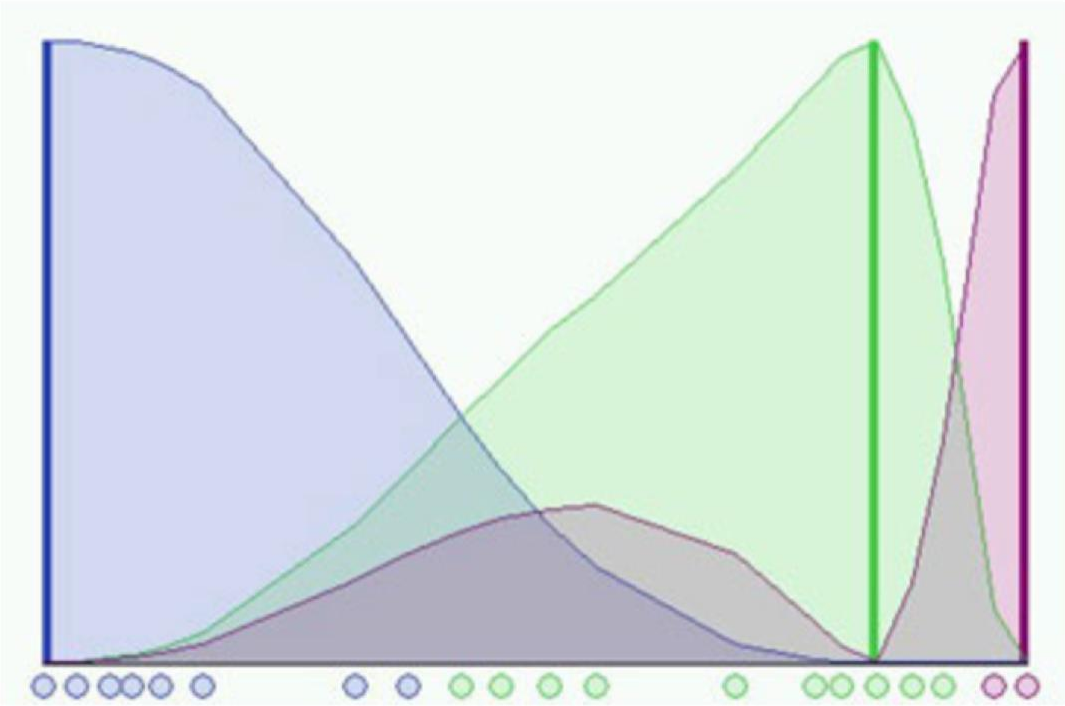
Le probabilità di appartenenza si calcolano come:

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

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

Fuzzy K-means (soft)

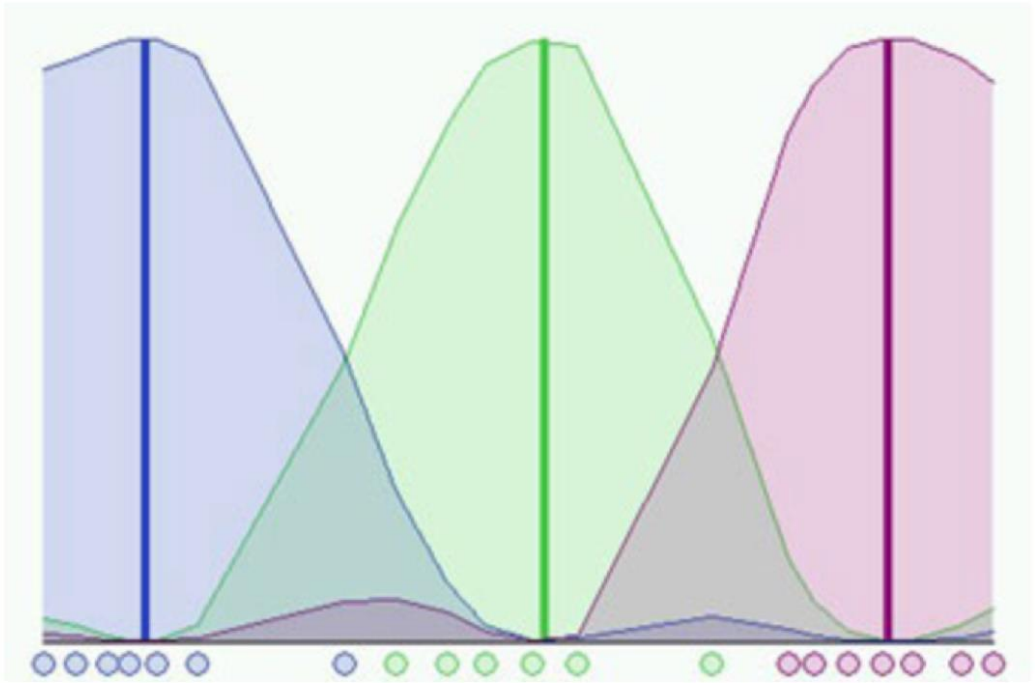
Caso di un'applicazione monodimensionale
N = 20, c = 3 per inizializzare l'algoritmo
Le figure seguenti mostrano il valore dell'appartenenza per ciascun dato e per ciascun cluster. Il colore dei dati è quello del cluster più vicino in base alla funzione di appartenenza.



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

Fuzzy K-means (soft)

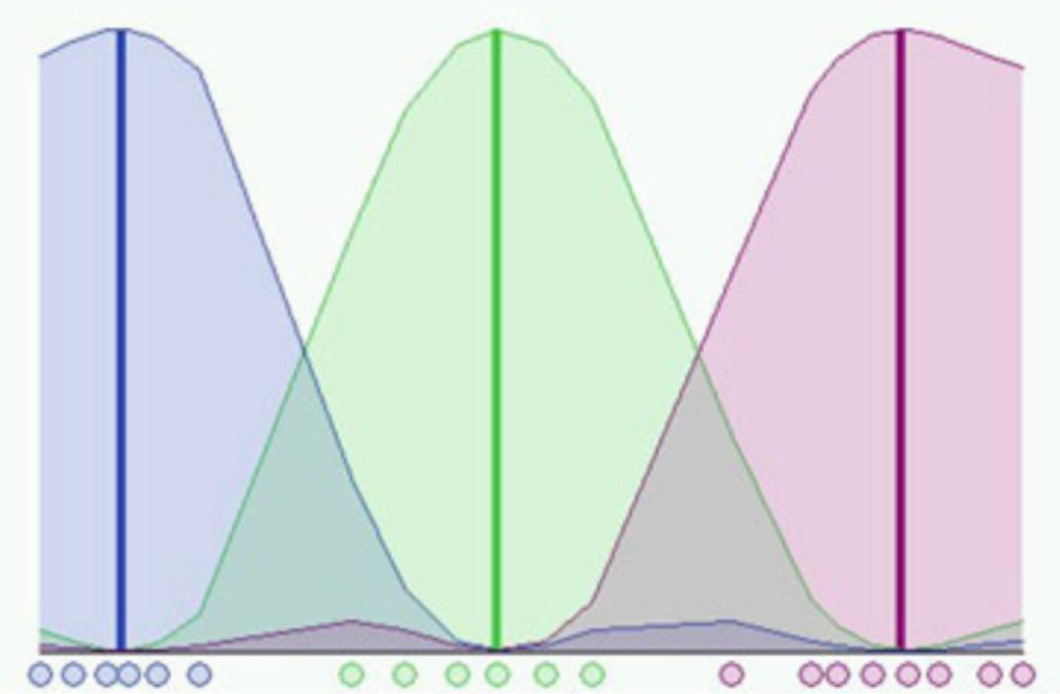
Caso di un'applicazione monodimensionale
N = 20, c = 3 per inizializzare l'algoritmo
Le figure seguenti mostrano il valore dell'appartenenza per ciascun dato e per ciascun cluster. Il colore dei dati è quello del cluster più vicino in base alla funzione di appartenenza.



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)

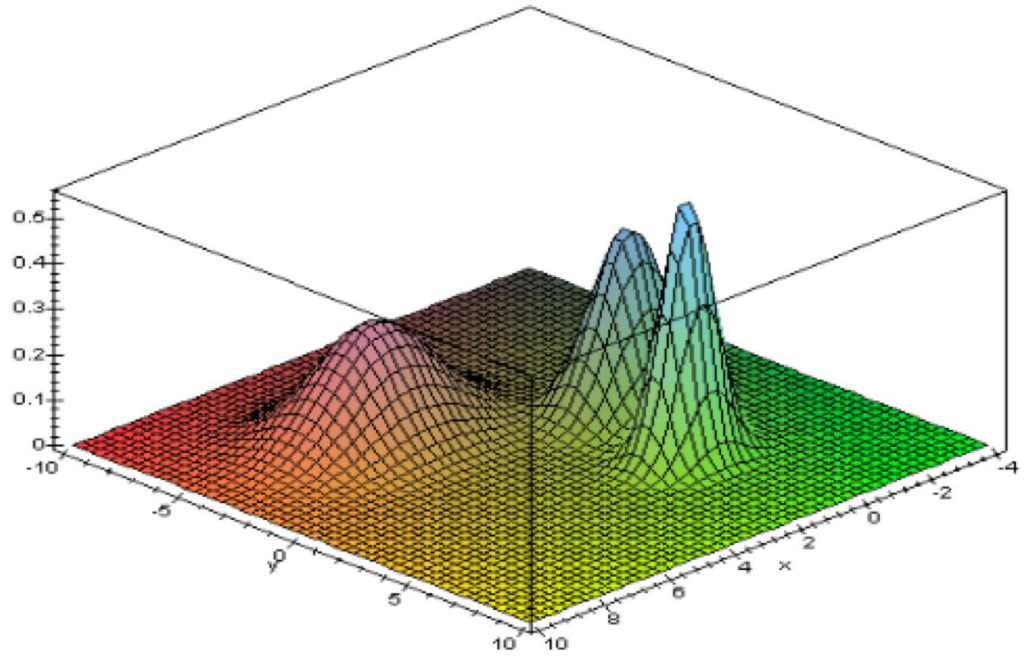
Fuzzy K-means | Vantaggi / Svantaggi

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.

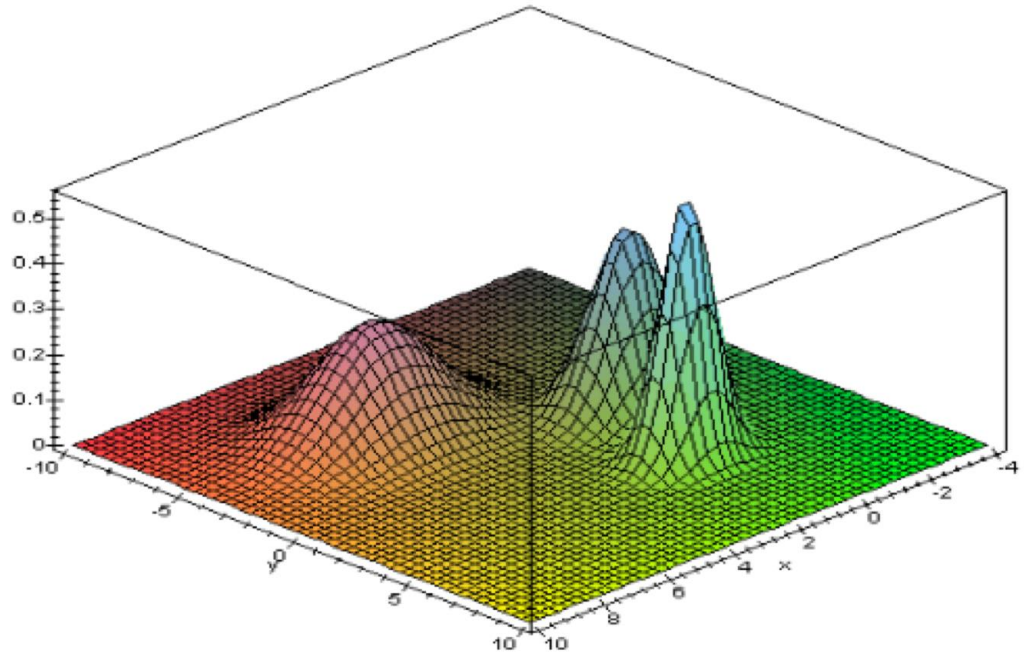
Expectation-Minimization (EM)

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)

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

$\boldsymbol{\Theta} = \alpha 1, \alpha 2 \ldots \alpha s, \boldsymbol{\theta} 1, \boldsymbol{\theta} 2 \ldots \boldsymbol{\theta} s$ è il vettore di parametri che definisce il mix di distribuzioni,

$pi \cdot (\mathrm{x}|\boldsymbol{\theta} i\ )$ e' 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 massimiza 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:

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

Purtroppo questa massimizzazione è molto difficile a causa della sommatoria dentro al logaritmo.
Per eliminare la sommatoria, ci servirebbe sapere, per ogni pattern xj del training set da quale componente pi della mixture è stato generato.

EM: soluzione al problema di ottimizzazione della funzione obiettivo

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

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

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= x1, x2 … xn siano incompleti per la mancanza di alcuni valori y.

Ogni pattern completo zj = xj, yj, con j=1…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 pi 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 cosi definito:

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

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

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

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

EM: caso generale

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

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

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

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

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:

$$log\mathcal{L}(\Theta|\mathcal{X}, y) = \sum_{j=1\ldots n} log\left(\alpha_{y_j} \cdot p_{y_j}\left(\mathbf{x}_j|\boldsymbol{\theta}_{y_j}\right)\right)$$

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):

$$P(y_j = k|\mathbf{x}_j, \Theta^g) = P(k|\mathbf{x}_j, \Theta^g) = \frac{{\alpha_k}^g \cdot p_k\left(\mathbf{x}_j|{\boldsymbol{\theta}_k}^g\right)}{\sum_{i=1\ldots s} {\alpha_i}^g \cdot p_i\left(\mathbf{x}_j|{\boldsymbol{\theta}_i}^g\right)}$$

Per un generico vettore y di osservazioni:

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

EM: caso specifico

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

$$Q(\Theta|\Theta^g) = \sum_{\mathbf{y}\in\Psi} log\mathcal{L}(\Theta|\mathcal{X},\mathcal{Y}) \cdot P(\mathbf{y}|\mathcal{X},\Theta^g) =$$

$$= \sum_{\mathbf{y}\in\Psi} \left(\sum_{j=1\ldots n} log\left(\alpha_{y_j} \cdot p_{y_j}\left(\mathbf{x}_j|\boldsymbol{\theta}_{y_j}\right)\right) \cdot \prod_{j=1\ldots n} P\left(y_j|\mathbf{x}_j,\Theta^g\right) \right)$$

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:

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

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

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

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

EM: formule incrementali

Intuitivamente, trattando le componenti della mixture come cluster:

Eq (1), stessa delle slide precedenti: indica la probabilità di appartenenza di xj al cluster Ck . Calcolata sfruttando l'inversione di Bayes p(. | Θ_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

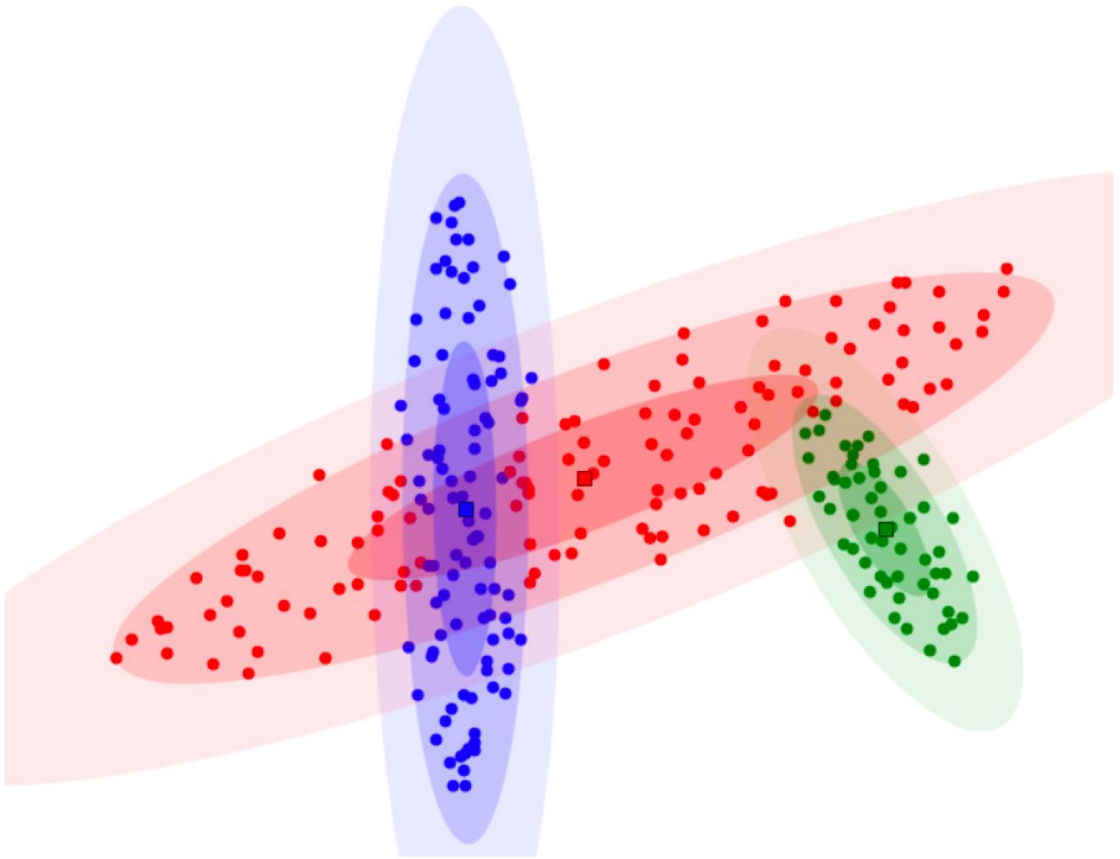
$$P(C_k|\mathbf{x}_j, \Theta^g) = \frac{\alpha_k{}^g \cdot p(\mathbf{x}_j|\boldsymbol{\theta}_k{}^g)}{\sum_{i=1\ldots s} \alpha_i{}^g \cdot p(\mathbf{x}_j|\boldsymbol{\theta}_i{}^g)} \qquad \text{eq (1)}$$

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

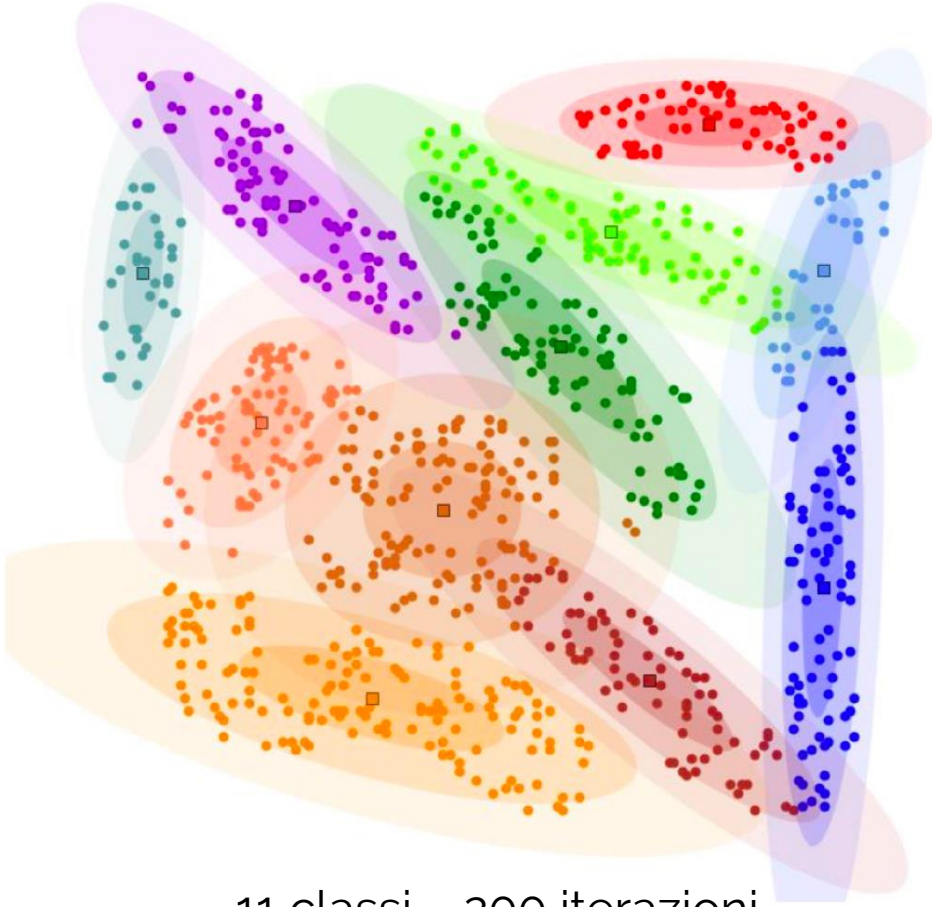
$$\boldsymbol{\mu}_k{}^{g+1} = \frac{\sum_{j=1\ldots n} P(C_k|\mathbf{x}_j, \boldsymbol{\theta}_k{}^g) \cdot \mathbf{x}_j}{\sum_{j=1\ldots n} P(C_k|\mathbf{x}_j, \boldsymbol{\theta}_k{}^g)} \qquad \text{eq 3)}$$

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

EM



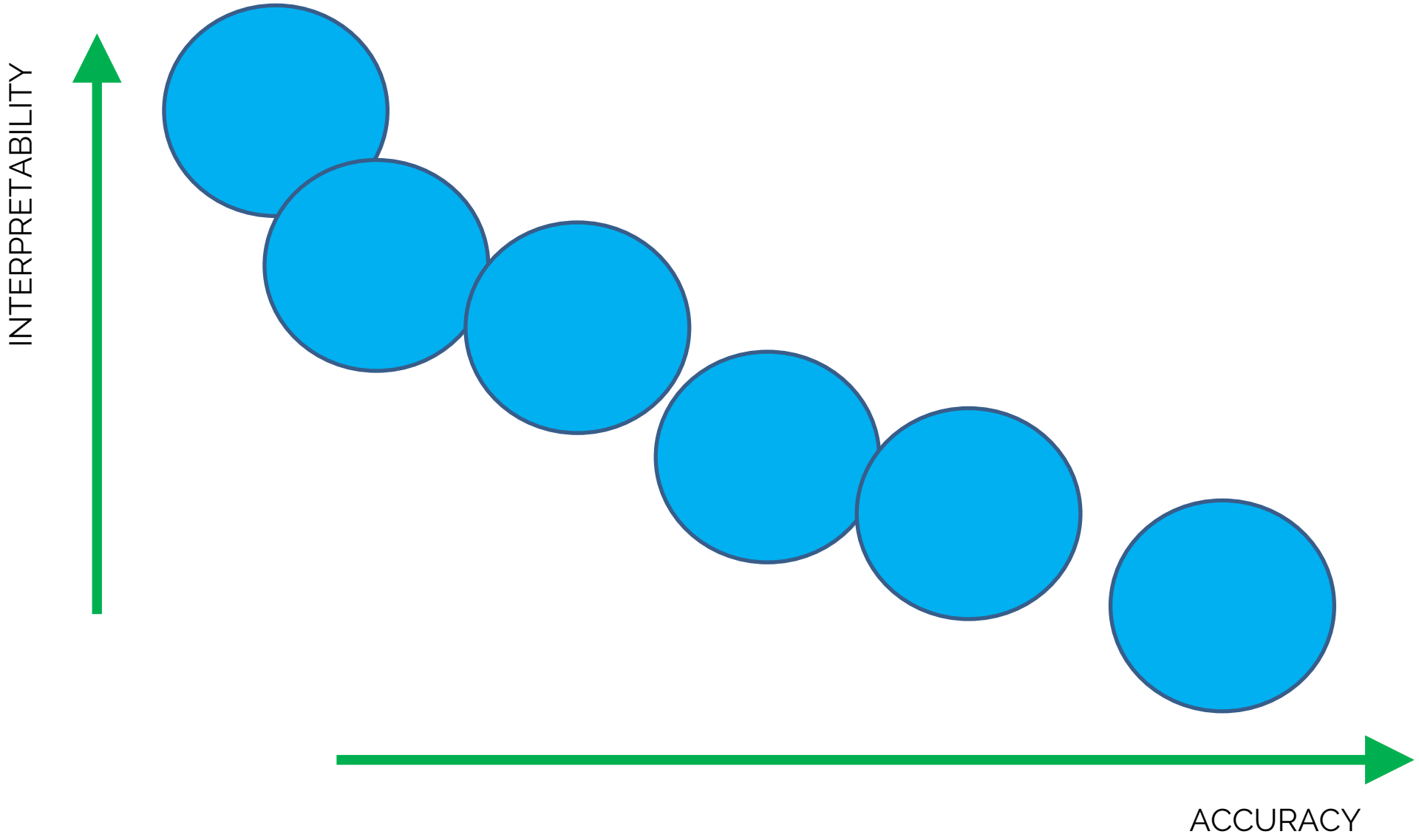
3 classi – 40 iterazioni



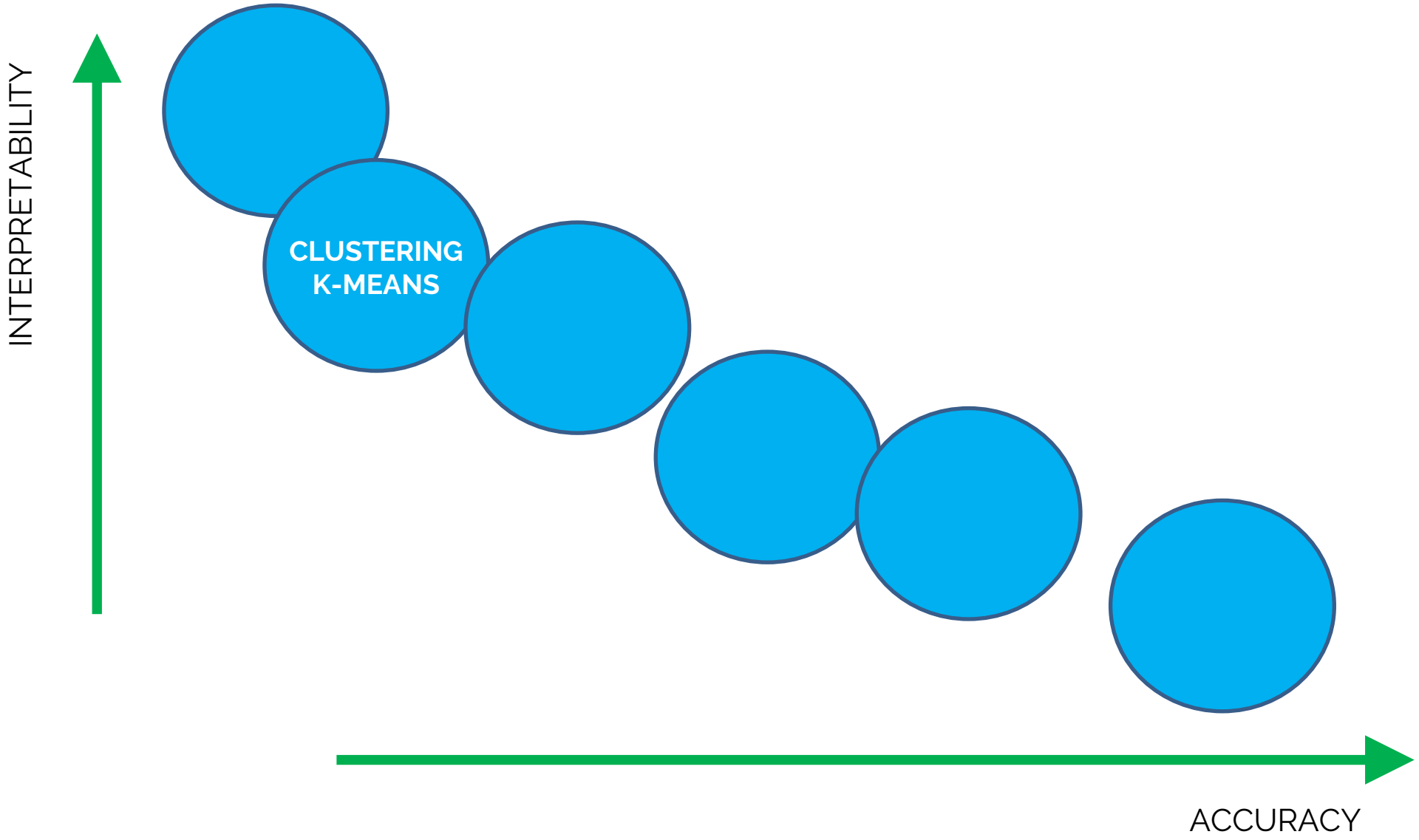
11 classi – 200 iterazioni

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

Interpretability-Accuracy TRADEOFF



Interpretability-Accuracy TRADEOFF



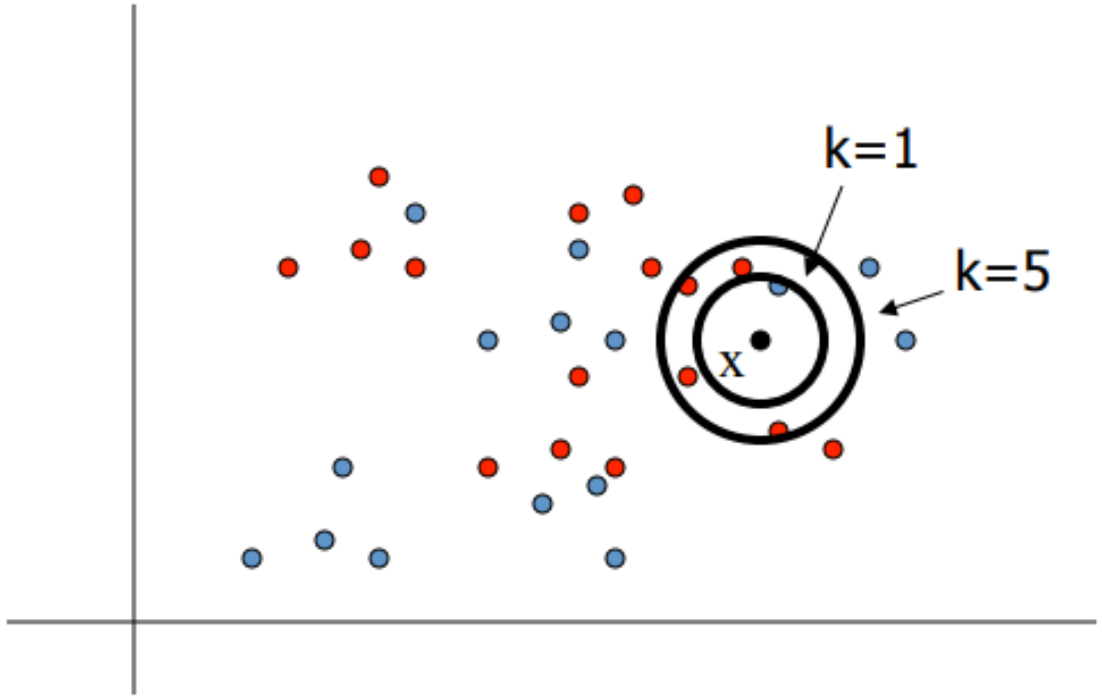
CLASSIFICATION

Nearest Neighbors

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

Nearest Neighbors

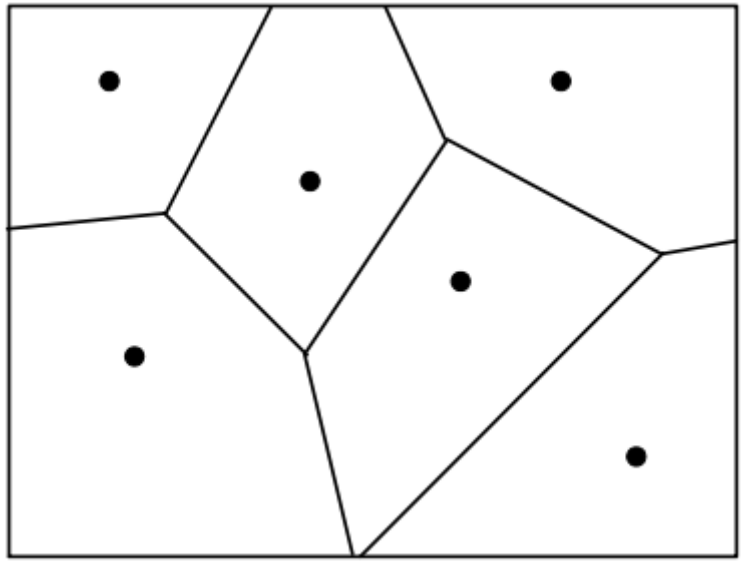
To classify a new input vector x, examine the k-closest training data points to x and assign the object to the most frequently occurring class



Common values for k = 3, 5

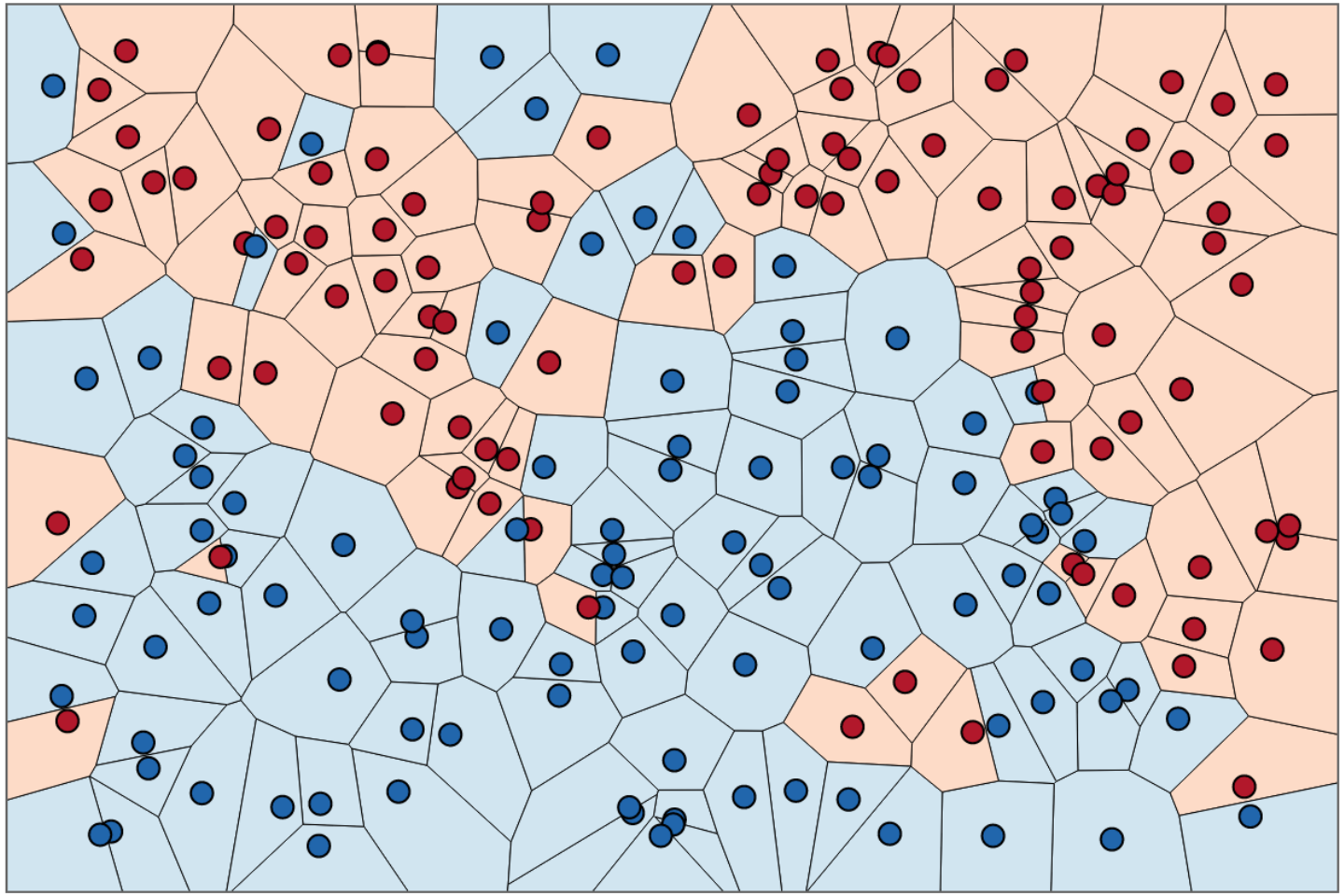
Nearest Neighbors

The nearest neighbor algorithm does not explicitly compute decision boundaries.



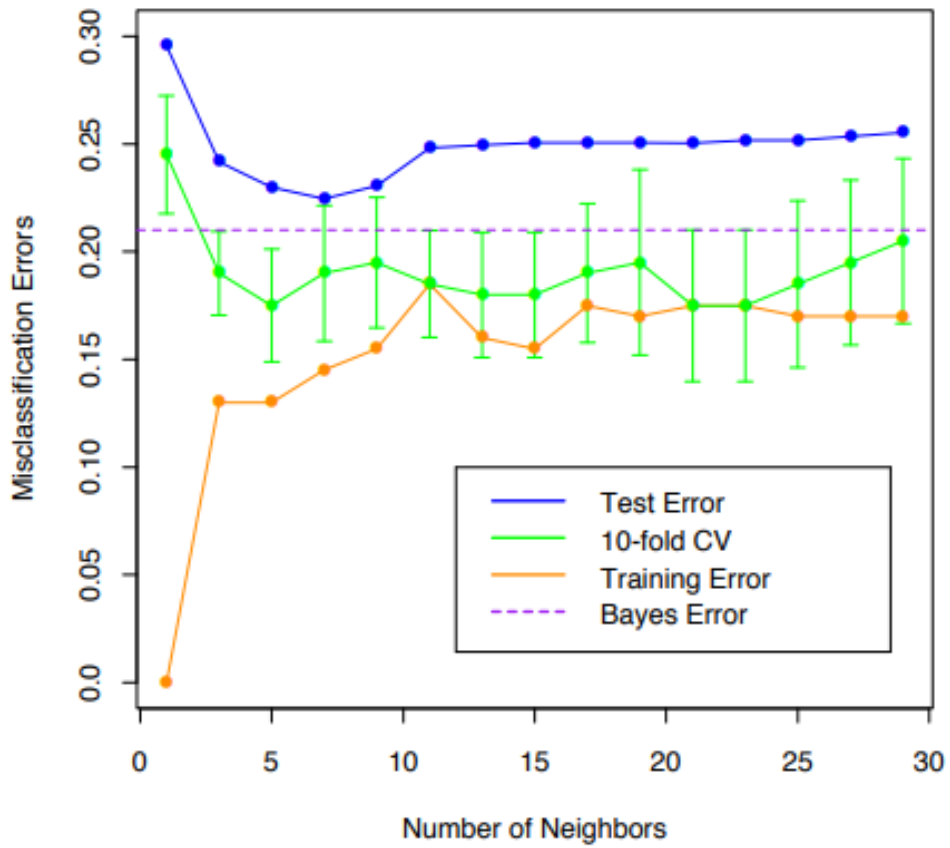
The decision boundaries form a subset of the Voronoi diagram for the training data.

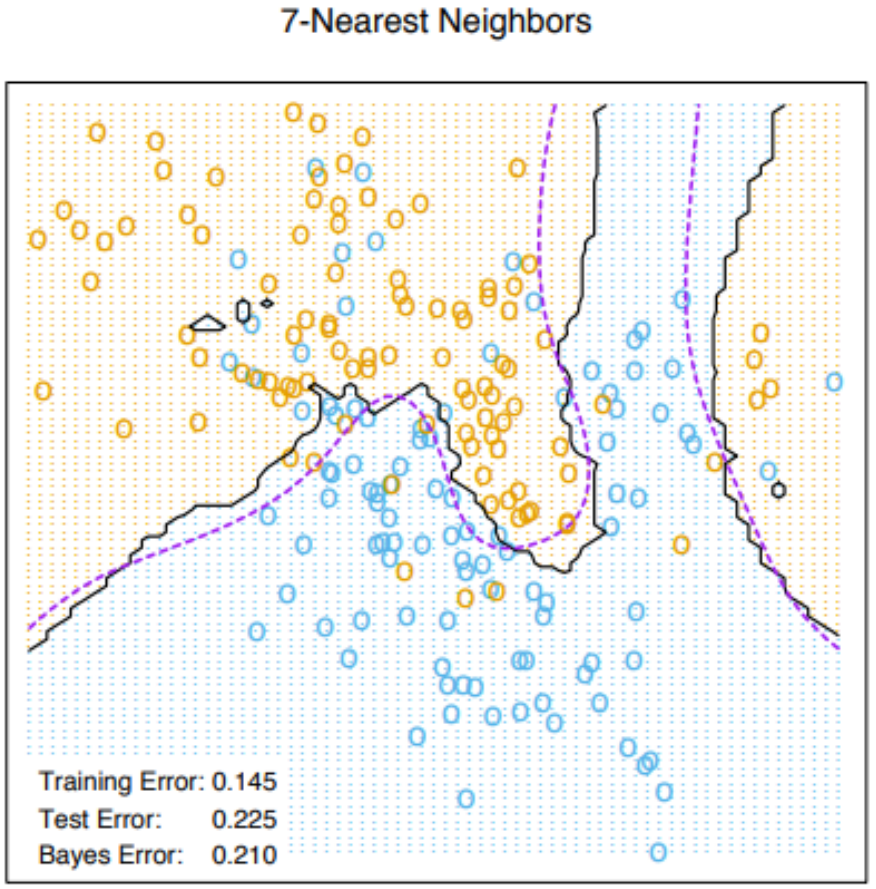
Nearest Neighbors



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

Nearest Neighbors





HASTIE & TIBSHIRANI

Nearest Neighbors

CONSIDERATIONS

– Instance map to points in Rn
– 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

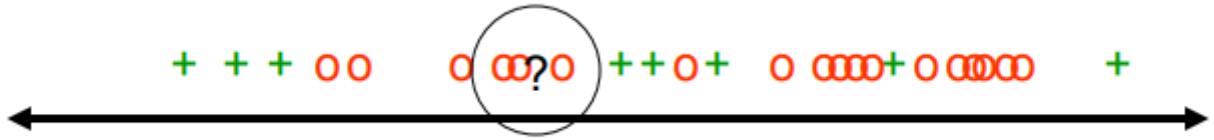
• Distance measure – Most common: Euclidean

• Choosing k – Increasing k reduces variance, increases bias

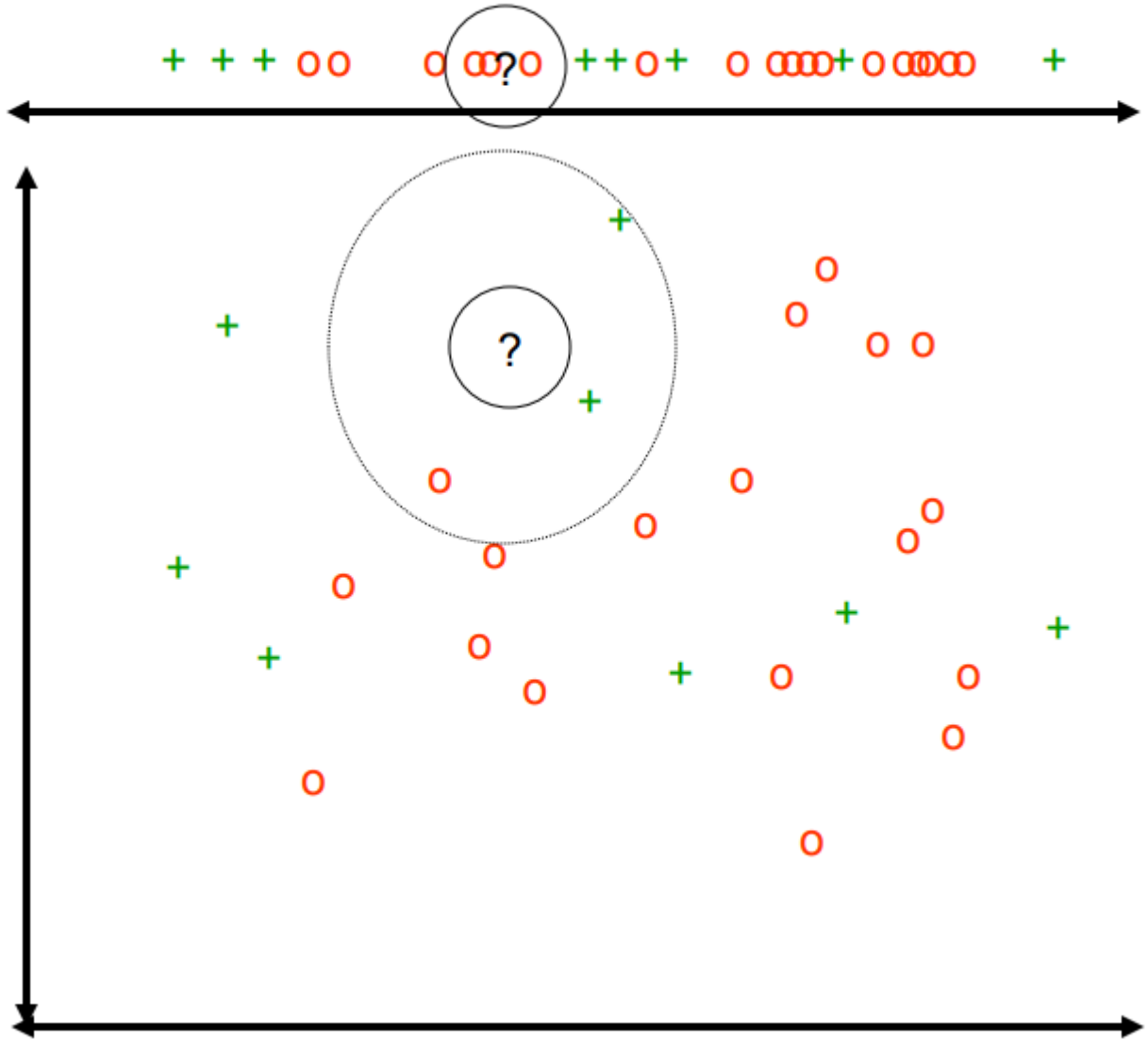
• For high dimensional space, the nearest neighbor may be very distant

• 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

$$x^i = (x^i_1, x^i_2, \ldots, x^i_p)$$

Most common distance metric is the Euclidean Distance:

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

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 xk is

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

where xk 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

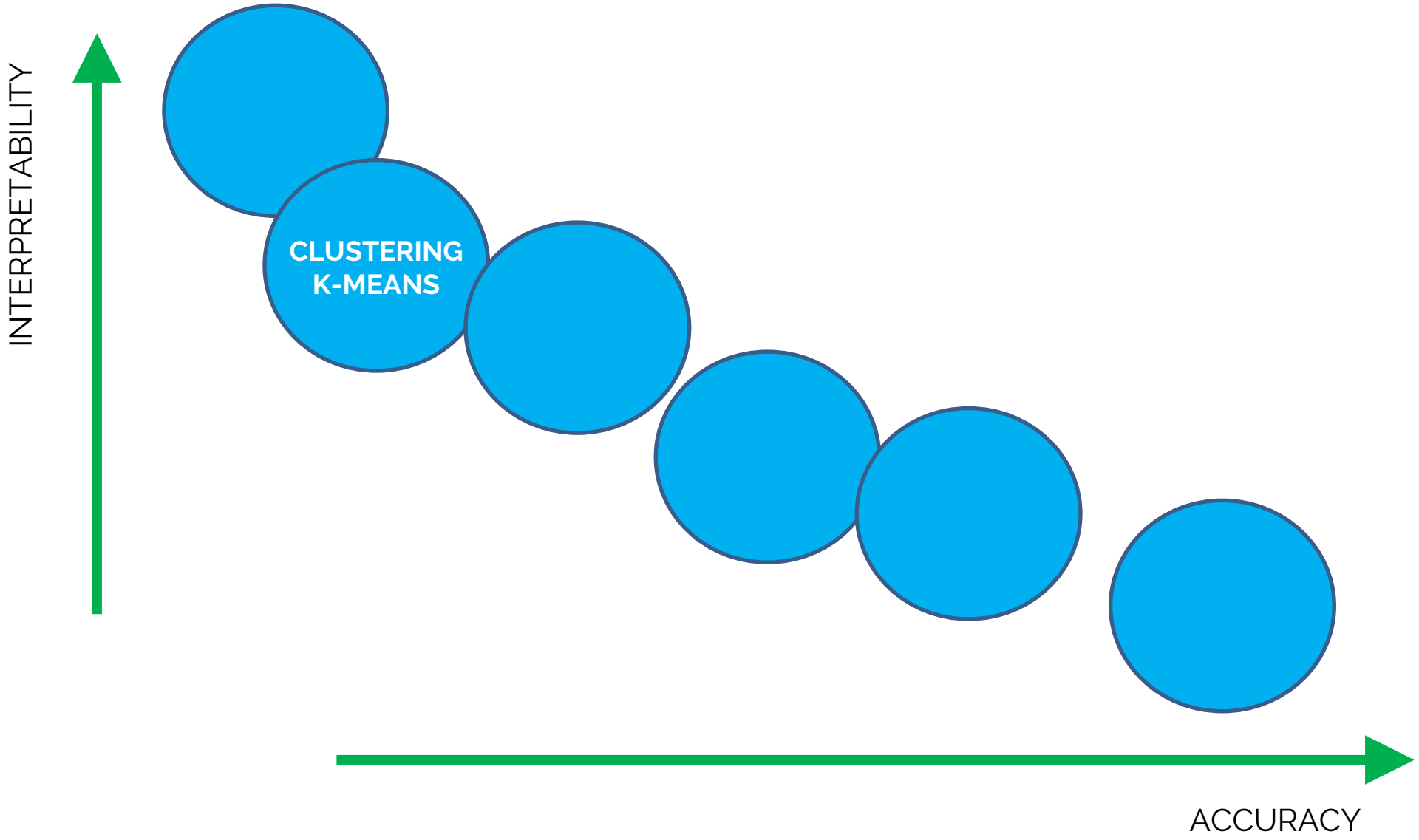
Advantages:

- 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

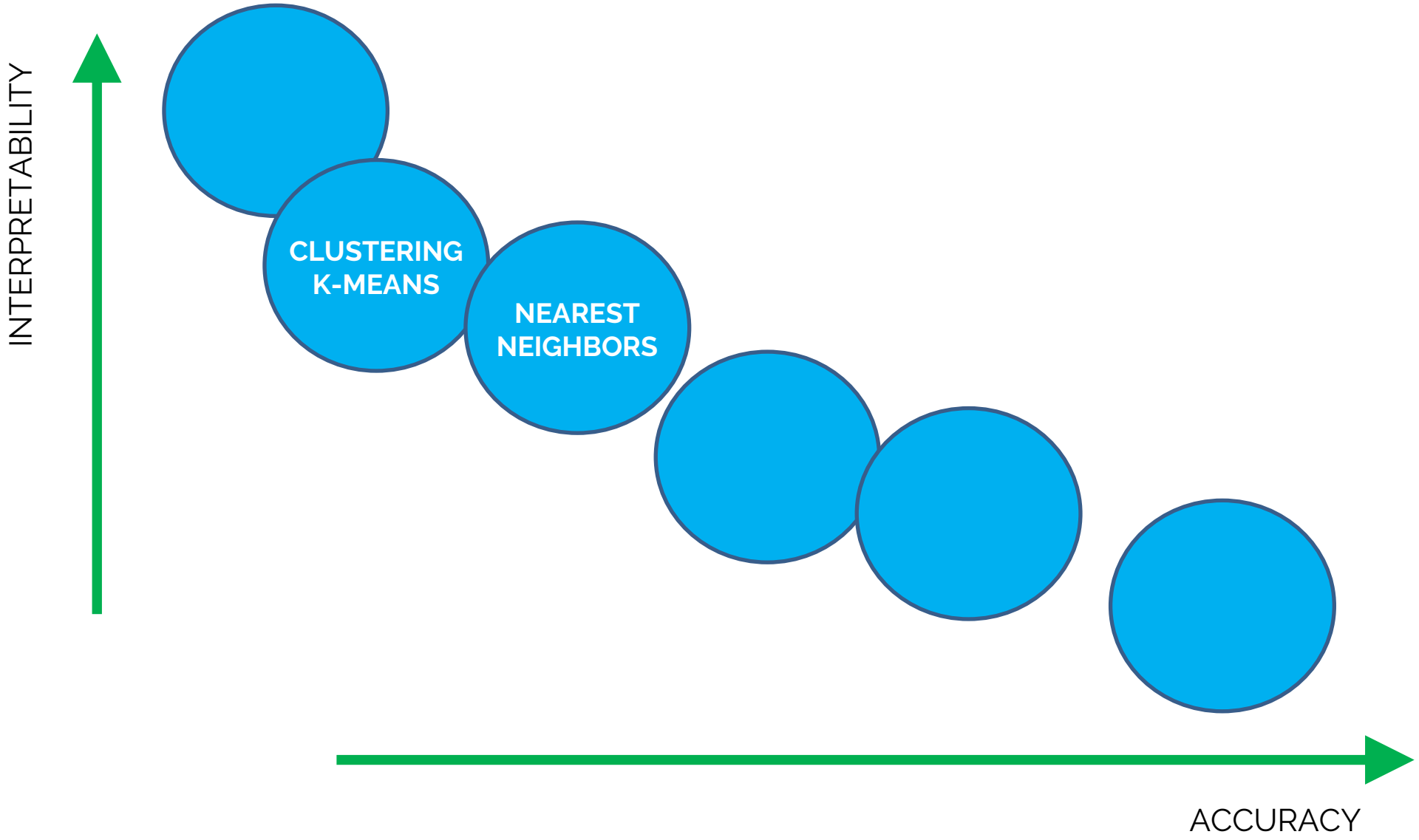
Drawbacks:

- 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.

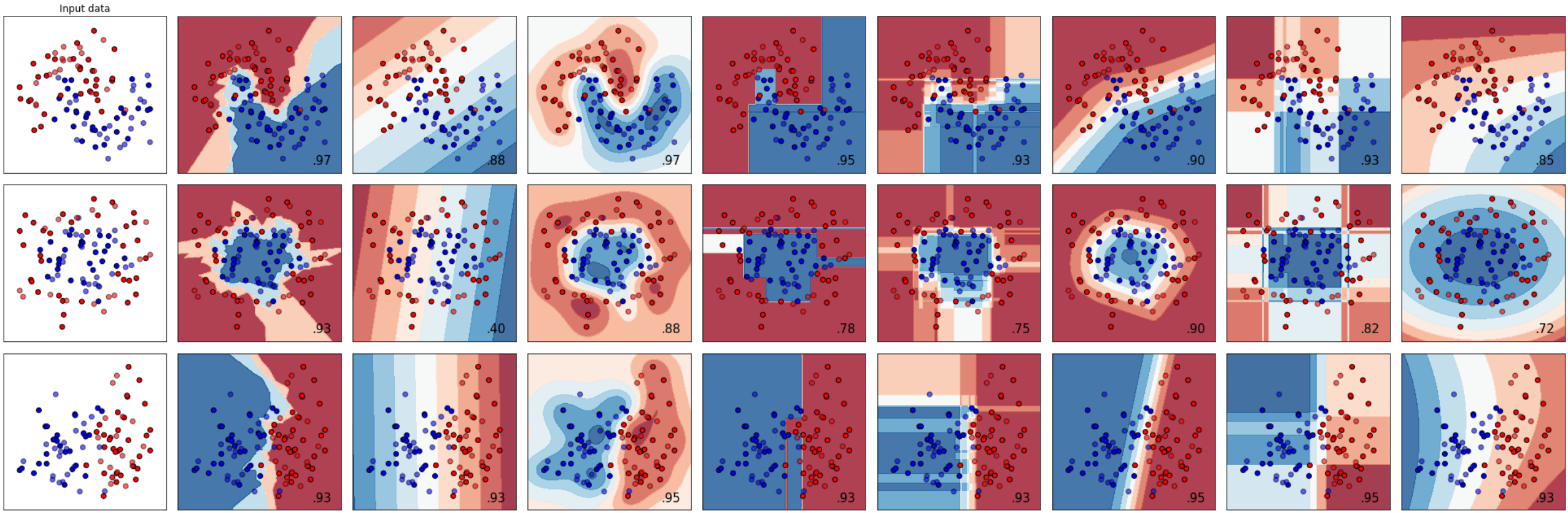
Interpretability-Accuracy TRADEOFF



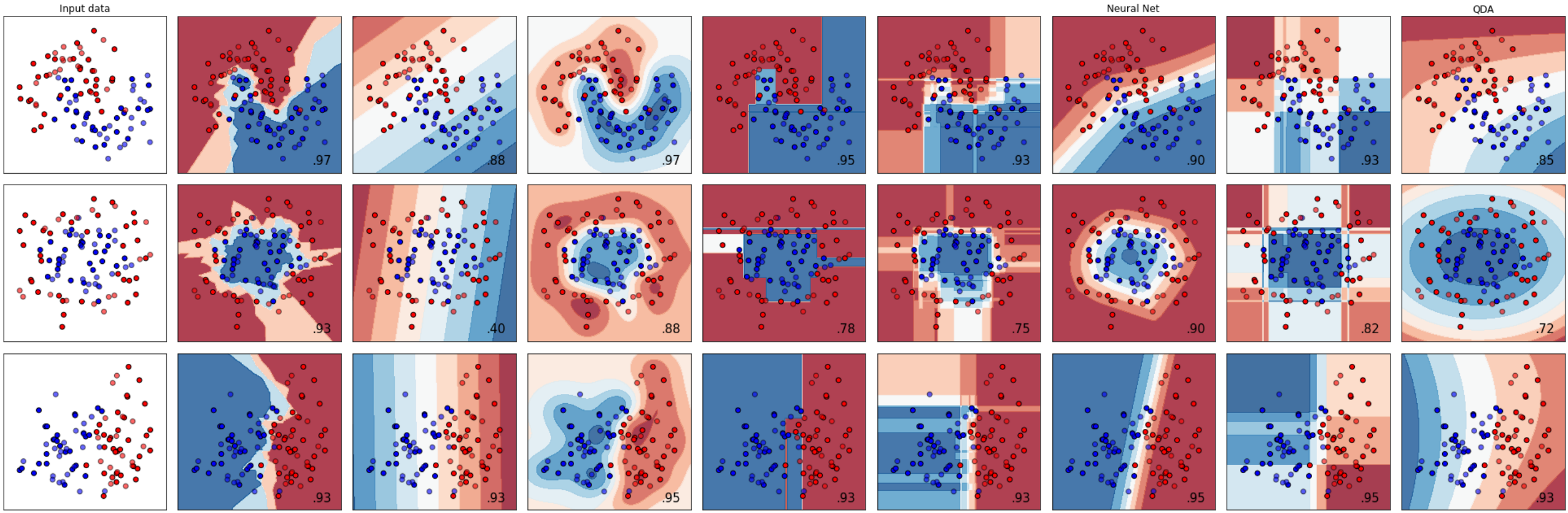
Interpretability-Accuracy TRADEOFF



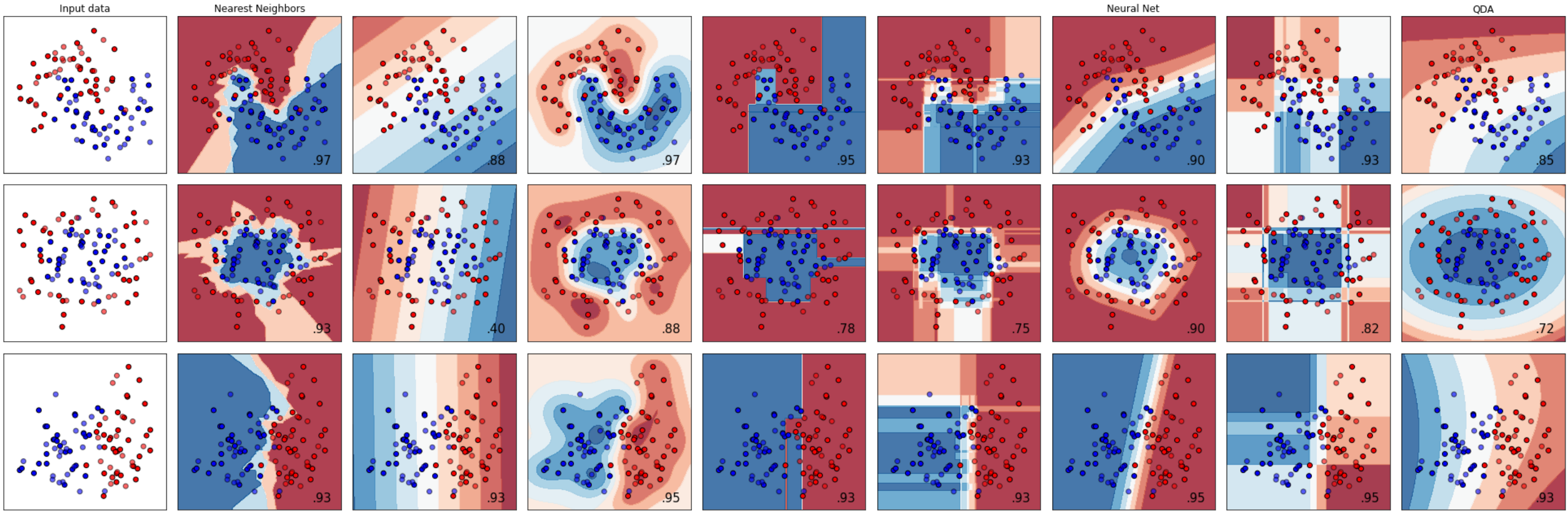
Which is which | Decision Function



Which is which | Decision Function



Which is which | Decision Function



CLASSIFICATION

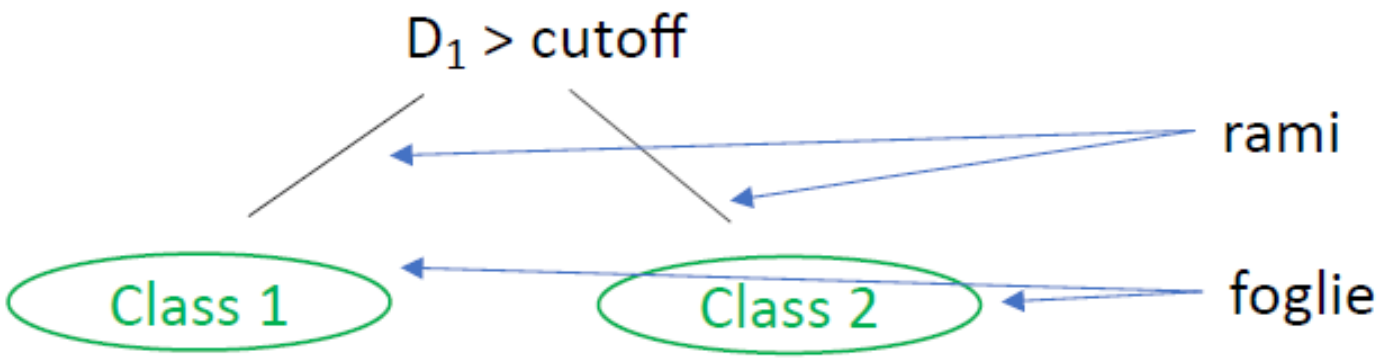
DECISION TREE

Decision Tree

Un decision tree o classification tree è un albero binario in cui ogni nodo divide i pattern sulla base di un criterio su una singola feature (o dimensione).

In queste strutture ad albero, i nodi a foglia rappresentano le etichette delle classi e i rami rappresentano l'insieme delle caratteristiche che portano a quelle classificazioni.

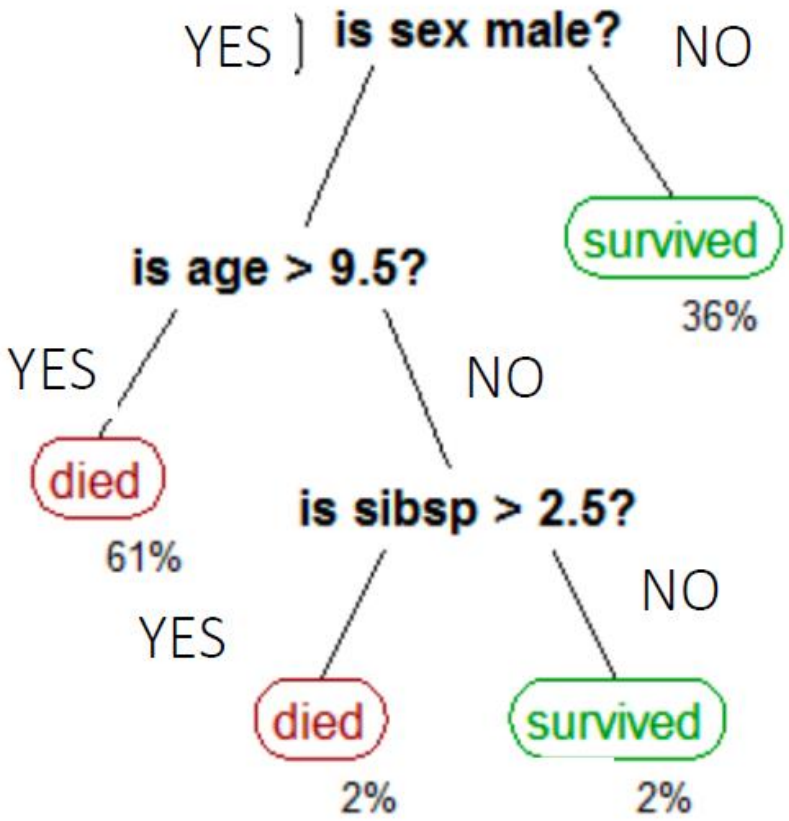
Algoritmi ad albero in cui la variabile target può assumere un insieme discreto di valori sono chiamati alberi di classificazione, quelli in cui la variabile target può assumere valori continui (in genere numeri reali) sono chiamati alberi di regressione.



Decision Tree

Sopravvivenza passeggeri del Titanic

Classi: died e survived
I numeri sotto le foglie indicano la percentuale di campioni nella foglia (classe)



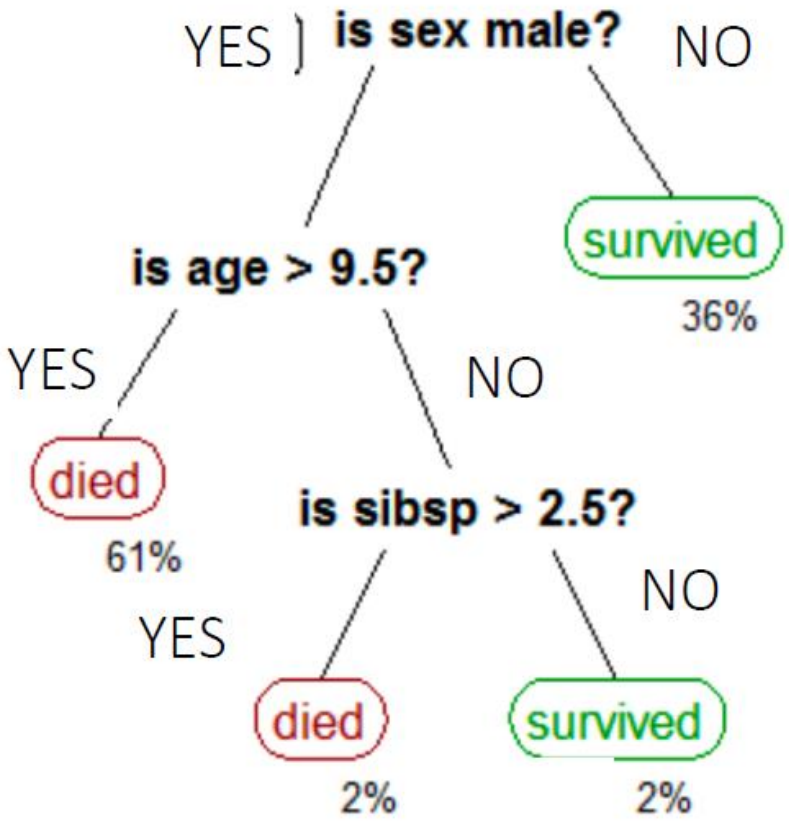
Le possibilità di sopravvivenza erano buone se il soggetto era (i) una femmina o (ii) un maschio di età inferiore ai 9,5 anni con meno di 2,5 fratelli

Decision Tree

Ogni nodo interno risulta essere una macro-classe costituita dall'unione delle classi associate ai suoi nodi figli.

Il predicato che si associa ad ogni nodo interno (sulla base del quale avviene la ripartizione dei dati) è chiamato condizione di split.

E' utile fissare un minimo numero di campioni richiesti per effettuare lo spit.



Le possibilità di sopravvivenza erano buone se il soggetto era (i) una femmina o (ii) un maschio di età inferiore ai 9,5 anni con meno di 2,5 fratelli

Decision Tree | Objective Function

In un buon albero di classificazione, i nodi foglia dovrebbero essere il più possibile puri (ovvero contenere solo dati che appartengono ad una sola classe).

Un parametro che definisce il criterio di impurità è l'indice di Gini* (impurità di Gini), che può quindi essere considerata come una funzione obiettivo.
Si stima il guadagno informativo come la differenza fra l'impurità del nodo genitore e la somma delle impurità dei nodi figli: minore è l'impurità dei nodi figli, maggiore è il guadagno informativo (decrease in Gini index).
L'indice di Gini raggiunge il suo minimo (zero) quando il nodo appartiene ad una singola categoria. Intuitivamente, l'impurità di Gini può essere considerata come un criterio per minimizzare la probabilità di un'errata classificazione.

$$I_G(i) = 1 - \sum_{j=1}^{m} f(i,j)^2$$

f rappresenta la frequenza del valore j nel nodo i.

*misura della diseguaglianza di una distribuzione, valori bassi indicano una distribuzione abbastanza omogenea (0-1)

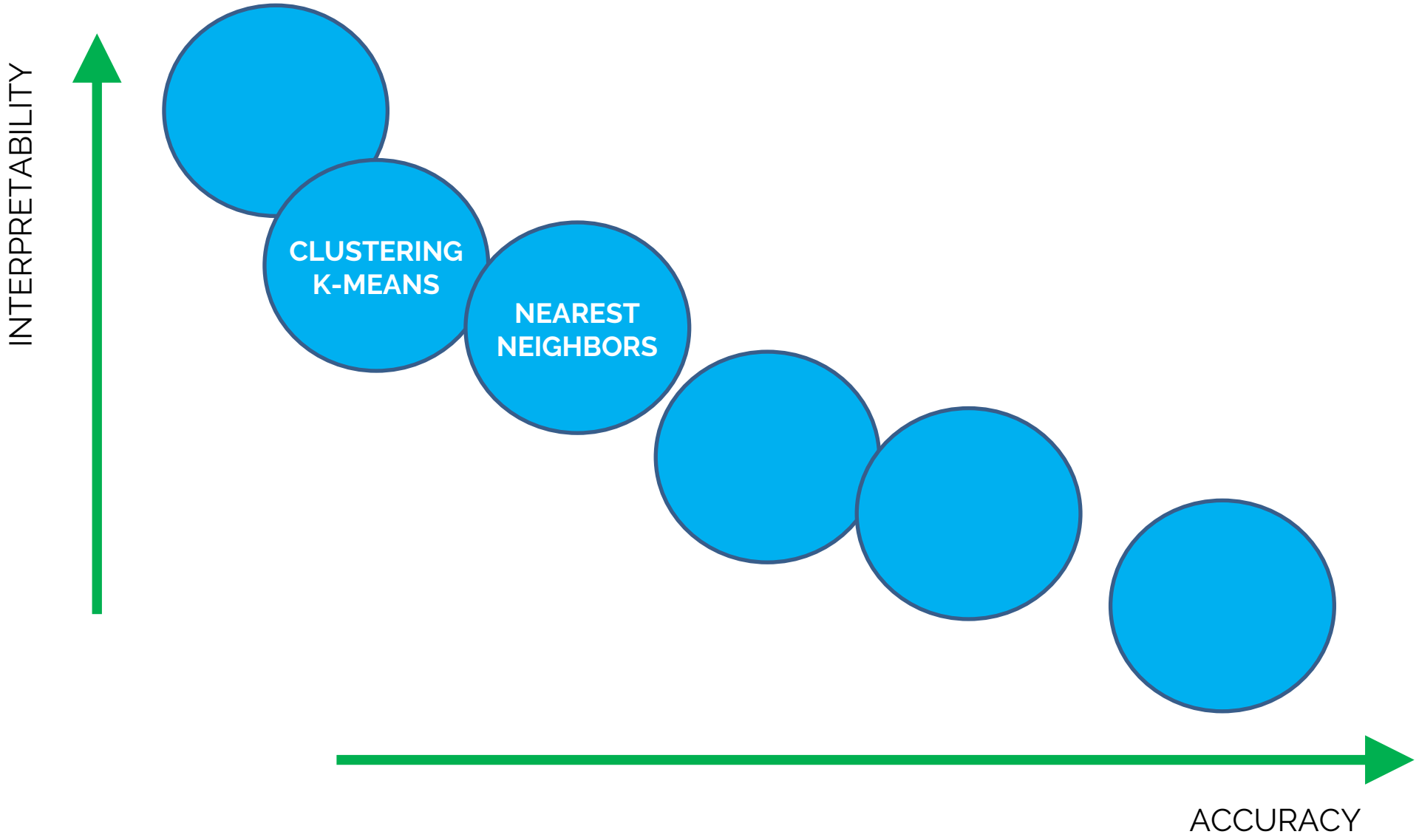
Decision Tree | Objective Function

Un altro parametro parametro che definisce il criterio di impurità è l'indice di entropia

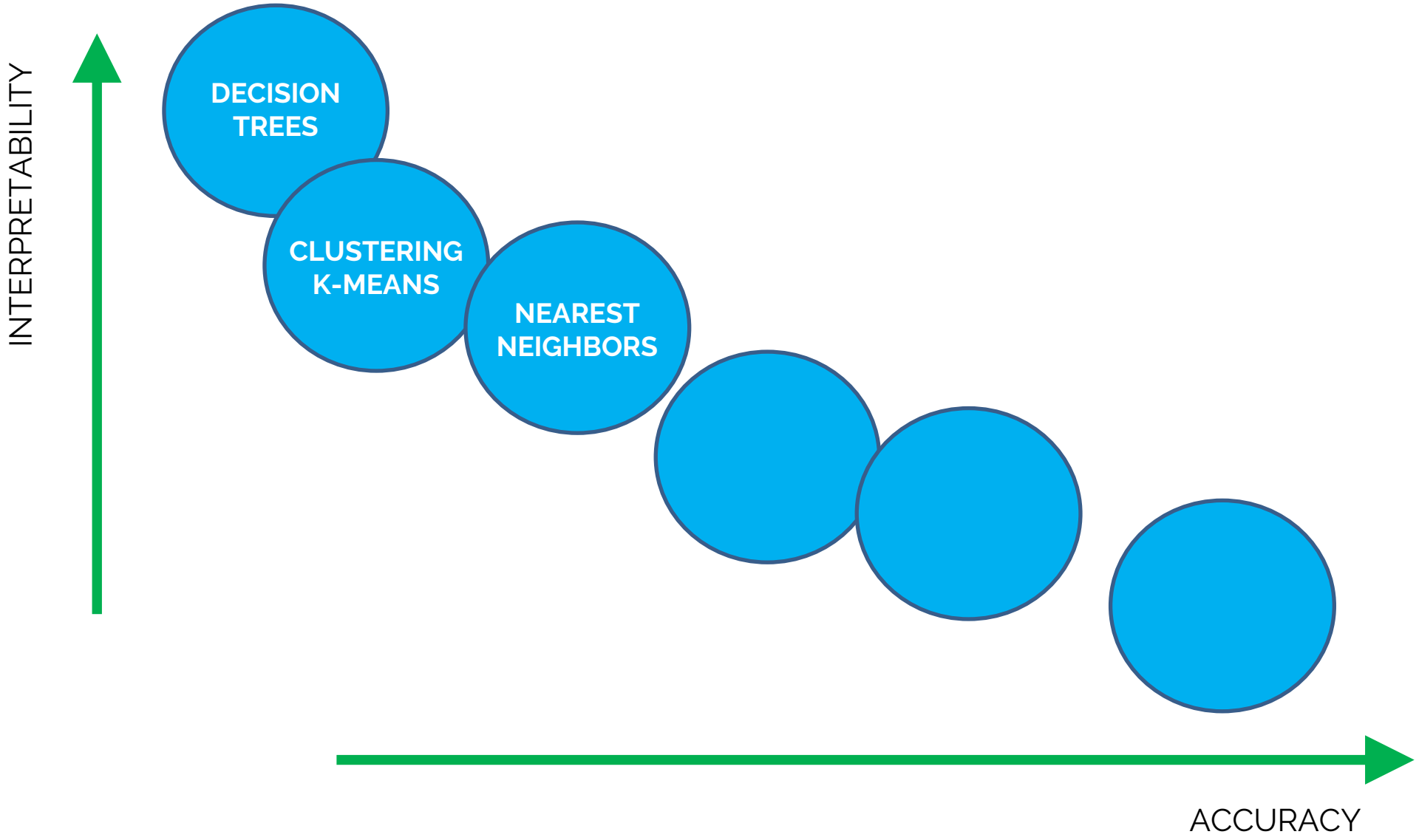
$$I_E(i) = -\sum_{j=1}^{m} f(i,j) \log f(i,j)$$

L'indice di Gini e l'indice di entropia sono i parametri che vengono usualmente utilizzati per guidare la costruzione dell'albero.
Il tasso di errore di classificazione che viene utilizzato per effettuare una ottimizzazione dell'albero nota come processo di arresto (halting) o di pruning («potatura» dei nodi superflui), al fine di determinarne la profondità massima dell'albero (max depth). Il crescere della profondità di un albero (ovvero della sua dimensione) non influisce direttamente sulla bontà del modello. Infatti, una crescita eccessiva della dimensione dell'albero potrebbe portare solo ad aumento sproporzionato della complessità computazionale rispetto ai benefici riguardanti l'accuratezza delle previsioni/classificazioni.

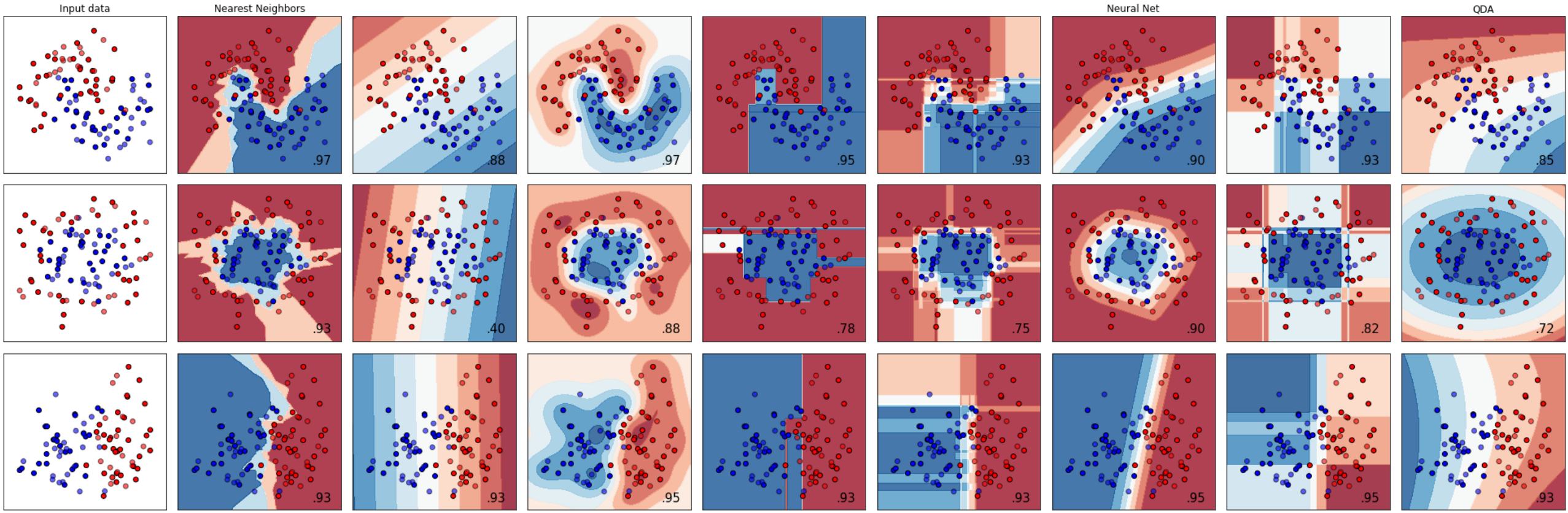
Interpretability-Accuracy TRADEOFF



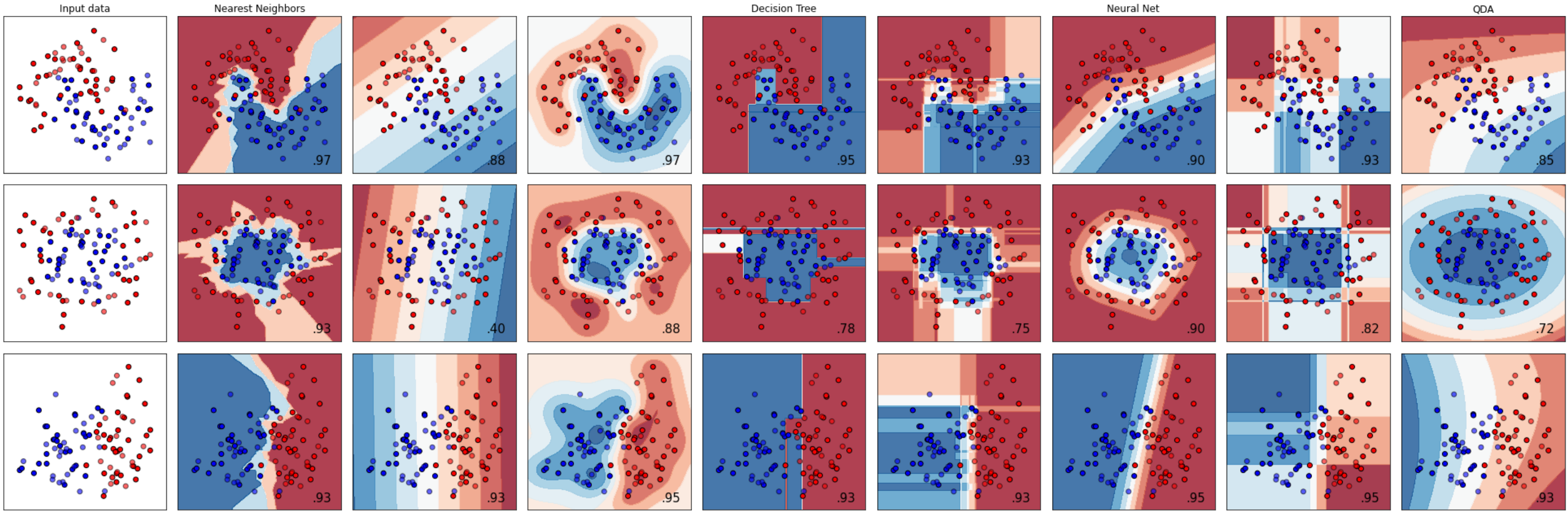
Interpretability-Accuracy TRADEOFF



Which is which | Decision Function



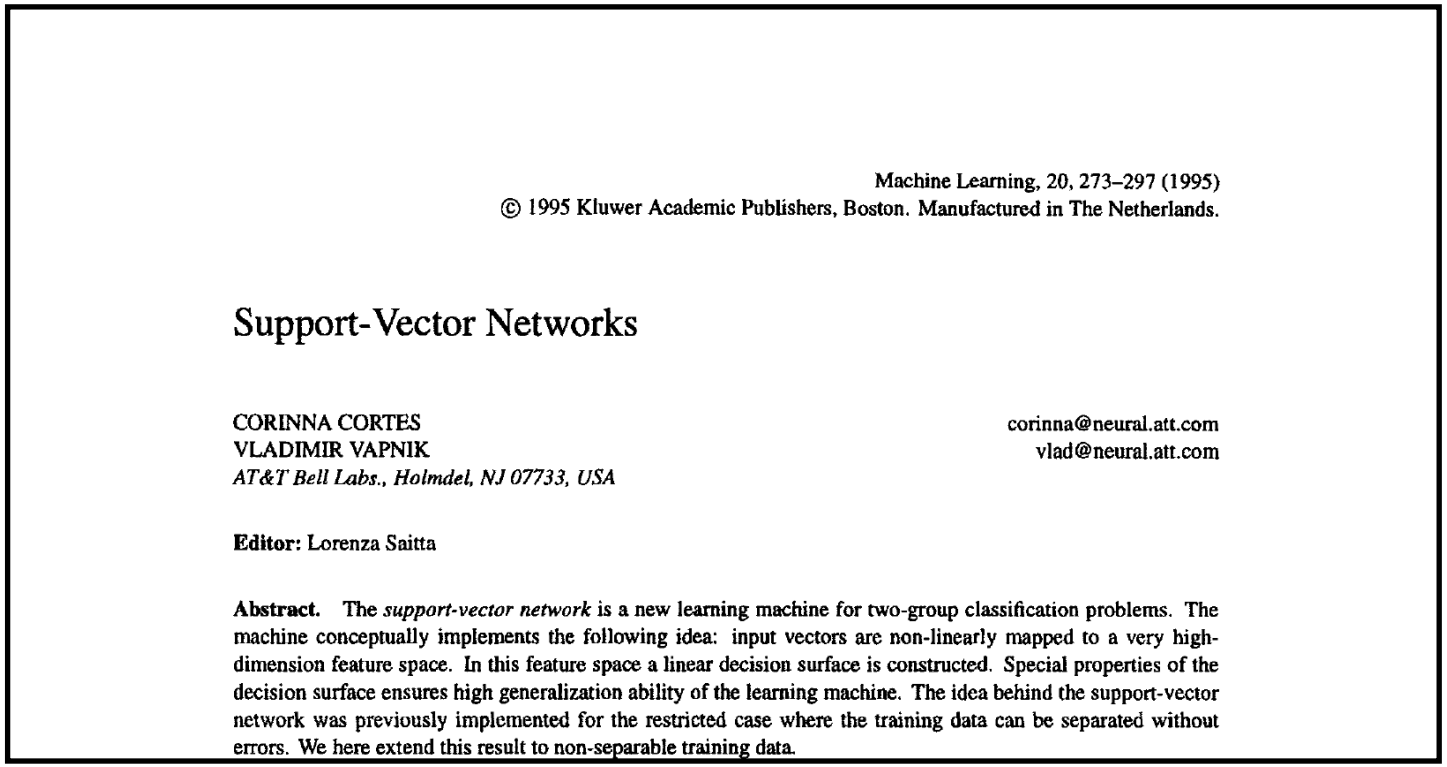
Which is which | Decision Function



CLASSIFICATION

SUPPORT VECTOR MACHINE

Support Vector Machine



Machine Learning, 20, 273–297 (1995)

Support-Vector Networks

CORINNA CORTES corinna@neural.att.com
VLADIMIR VAPNIK vlad@neural.att.com
AT&T Bell Labs., Holmdel, NJ 07733, USA

Editor: Lorenza Saitta

Abstract. The *support-vector network* is a new learning machine for two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space. In this feature space a linear decision surface is constructed. Special properties of the decision surface ensures high generalization ability of the learning machine. The idea behind the support-vector network was previously implemented for the restricted case where the training data can be separated without errors. We here extend this result to non-separable training data.

SVM is a binary classifier which aims at generating a predictive model for the discrimination of new samples.

Support Vector Machine

Teoria introdotta da Vapnik (statistical learning theory) nel 1965.
Perfezionata più tardi dallo stesso Vapnik e altri nel 1995.

Vapnik suggerisce di determinare superfici decisionali tra le classi (classification boundaries) per separarle.

SVM nasce come classificatore binario (2 classi), con gradi diversi di complessità:

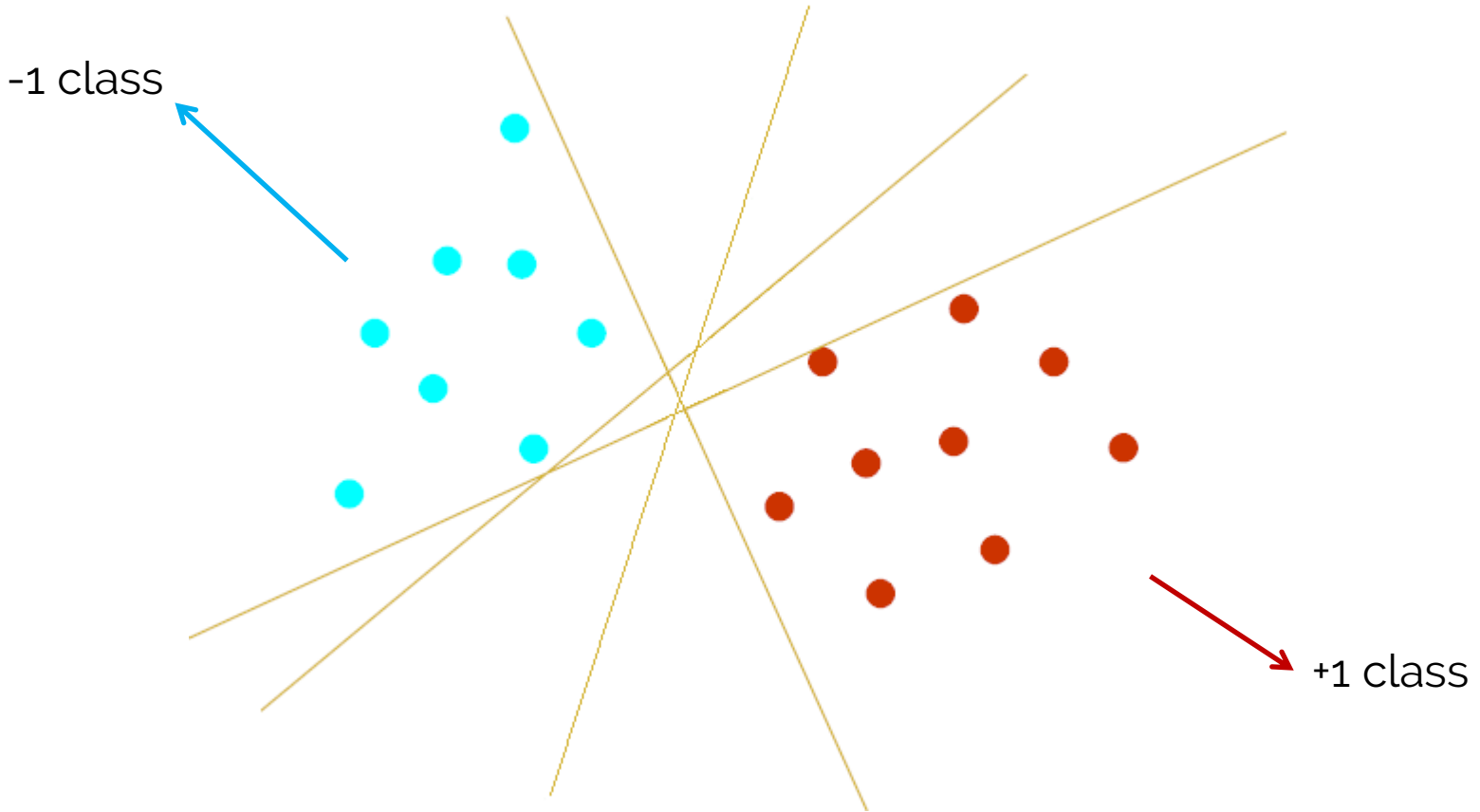
- SVM lineare (i.e., la superficie di separazione è un iperpiano) e i pattern del training set sono linearmente separabili (i.e., esiste per ipotesi almeno un iperpiano in grado di separarli).

- SVM lineare e pattern non linearmente separabili. Errori di classificazione non esistendo alcun iperpiano in grado di separare i pattern.

- SVM non lineare (i.e., superficie di separazione complessa) senza ipotesi sulla separabilità dei pattern.

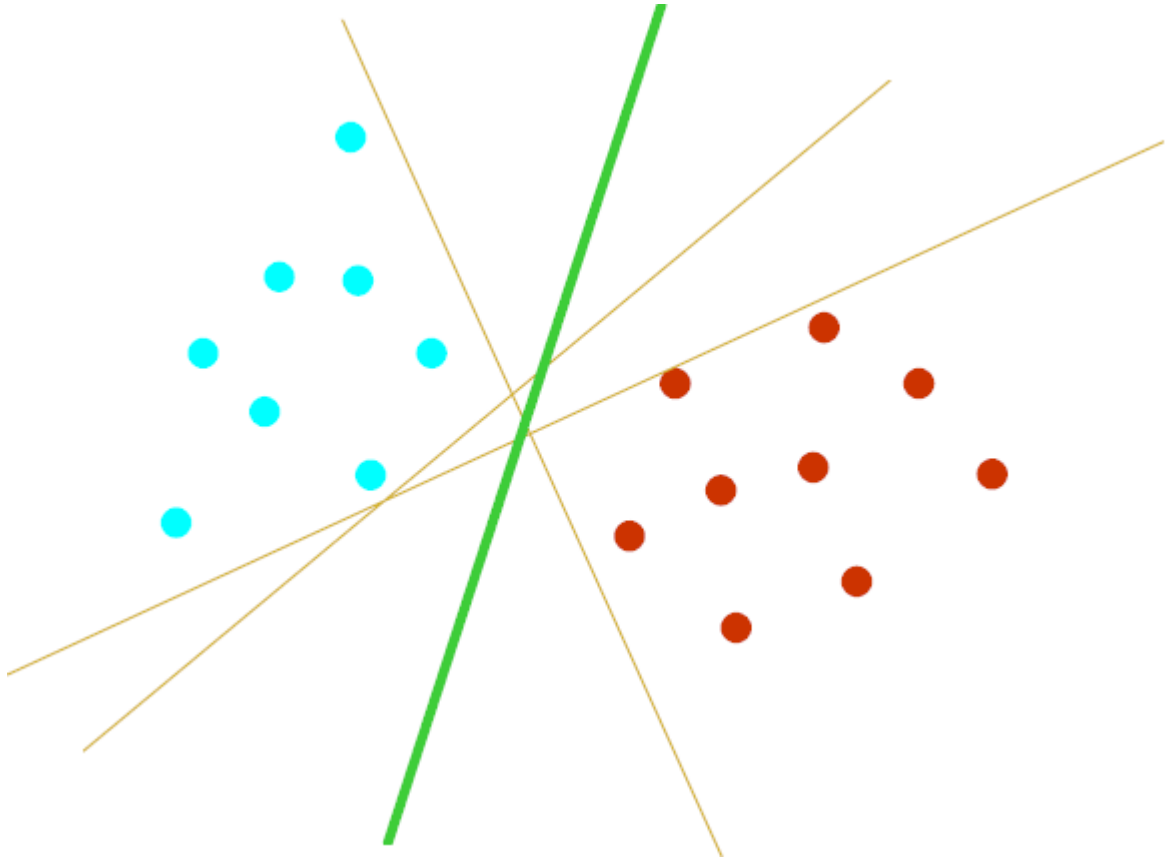
- Estensione SVM a multi-classe.

Support Vector Machine



Let us suppose to have a set of training data consisting of a vector of N samples belonging to two classes and the corresponding vector of class labels (e.g. -1 and +1 for control and patient class, respectively)

Support Vector Machine



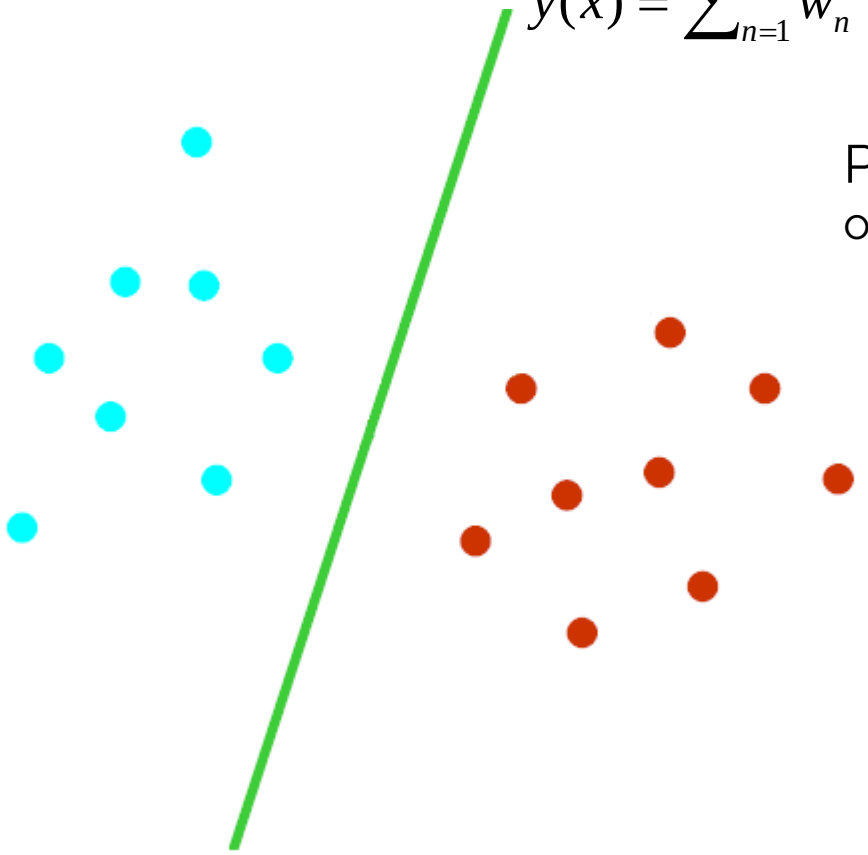
SVM generates a hyper-plane able to discriminate between the two classes of training dataset

Support Vector Machine

$$y(x) = \sum_{n=1}^{N} w_n \cdot t_n \cdot k(x, x_n) + b$$

=

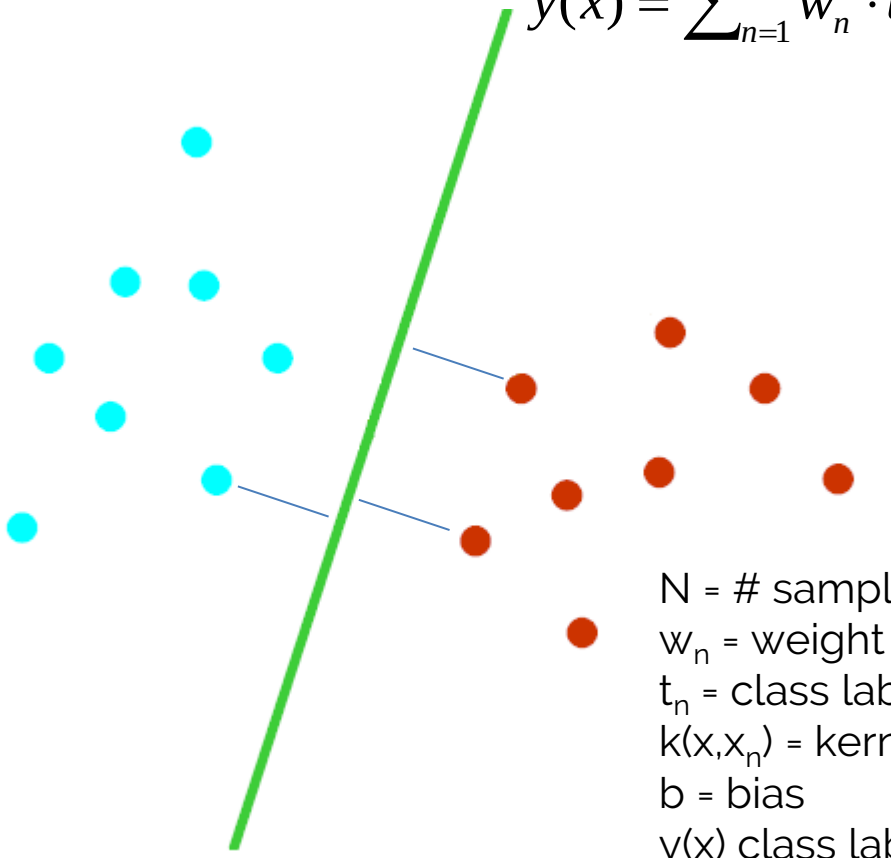
Predictive model
or decision function



SVM generates a hyper-plane able to discriminate between the two classes of training dataset

Support Vector Machine

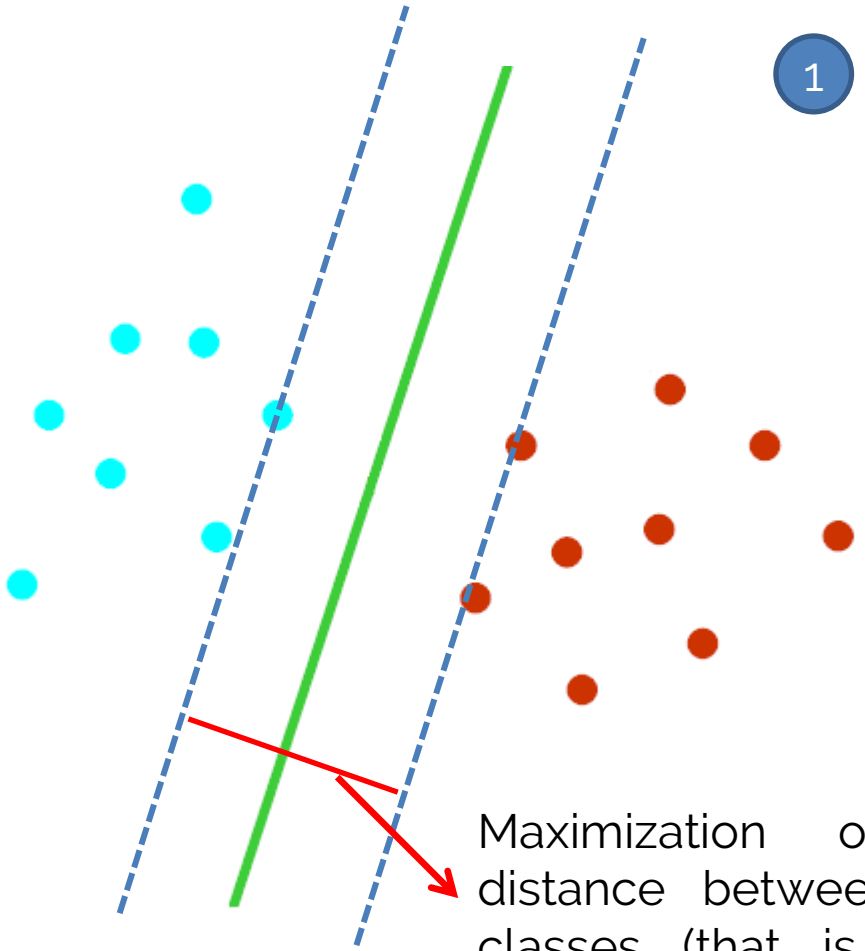
$$y(x) = \sum_{n=1}^{N} w_n \cdot t_n \cdot k(x, x_n) + b$$



N = # samples in the training dataset
w_n = weight assigned by SVM
t_n = class label of the training sample n
$k(x,x_n)$ = kernel function
b = bias
y(x) class label for unseen sample x

The weight assigned by SVM during the training phase to each training sample reflects its importance for classes discrimination

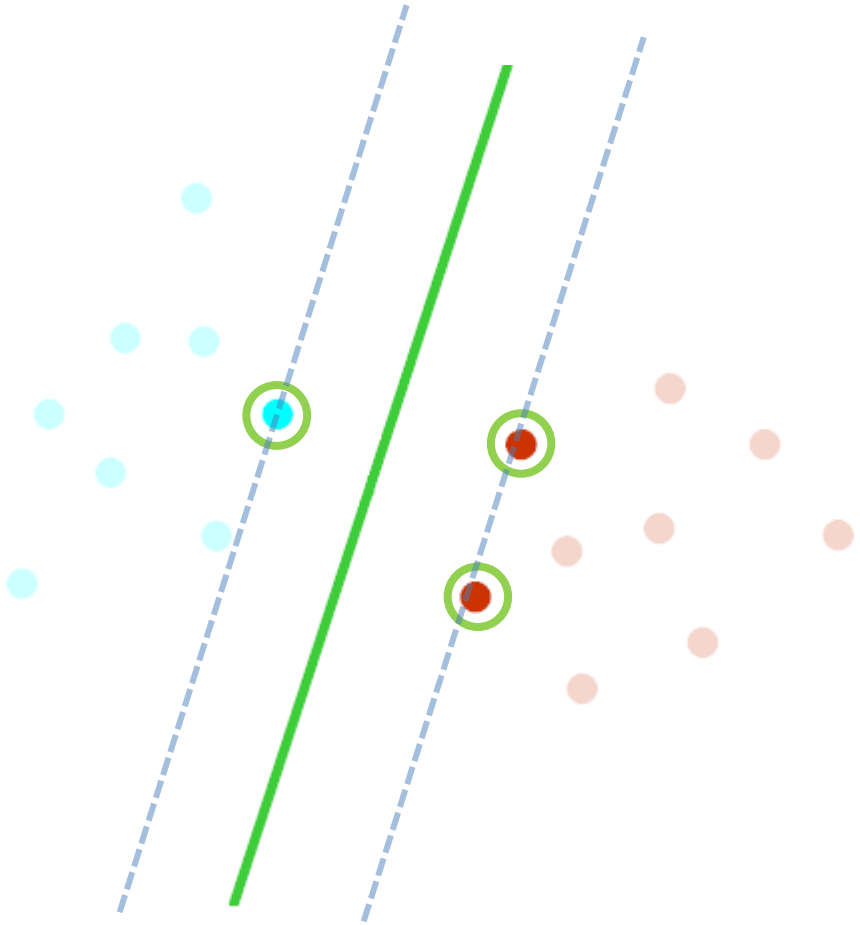
Support Vector Machine



1. Minimization of the empirical classification error

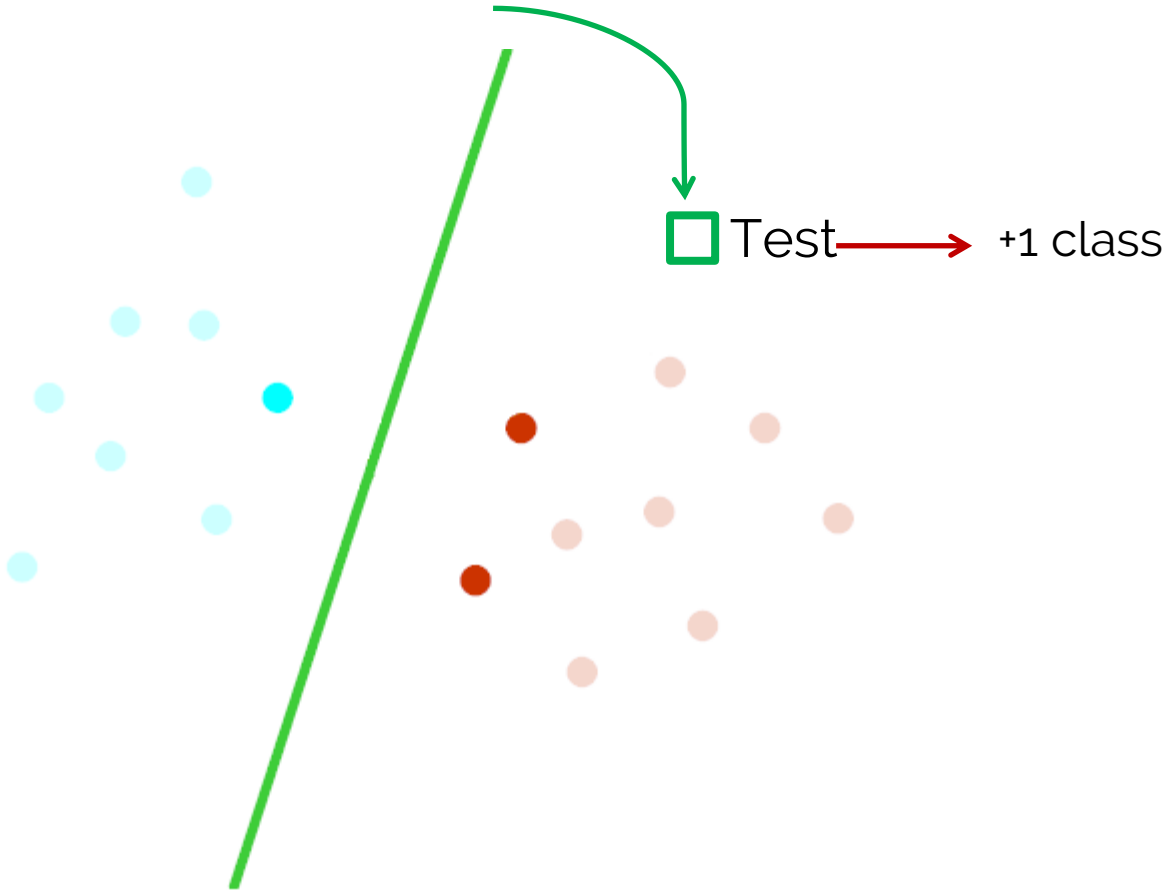
2. Maximization of the separation distance between the two training classes (that is why the decision function is often called maximum margin hyper-plane)

Support Vector Machine



Samples of the two classes which lie on the margin of the hyper-plane are called support vector, and they are the only samples for which the assigned weight w is non-zero.

Support Vector Machine

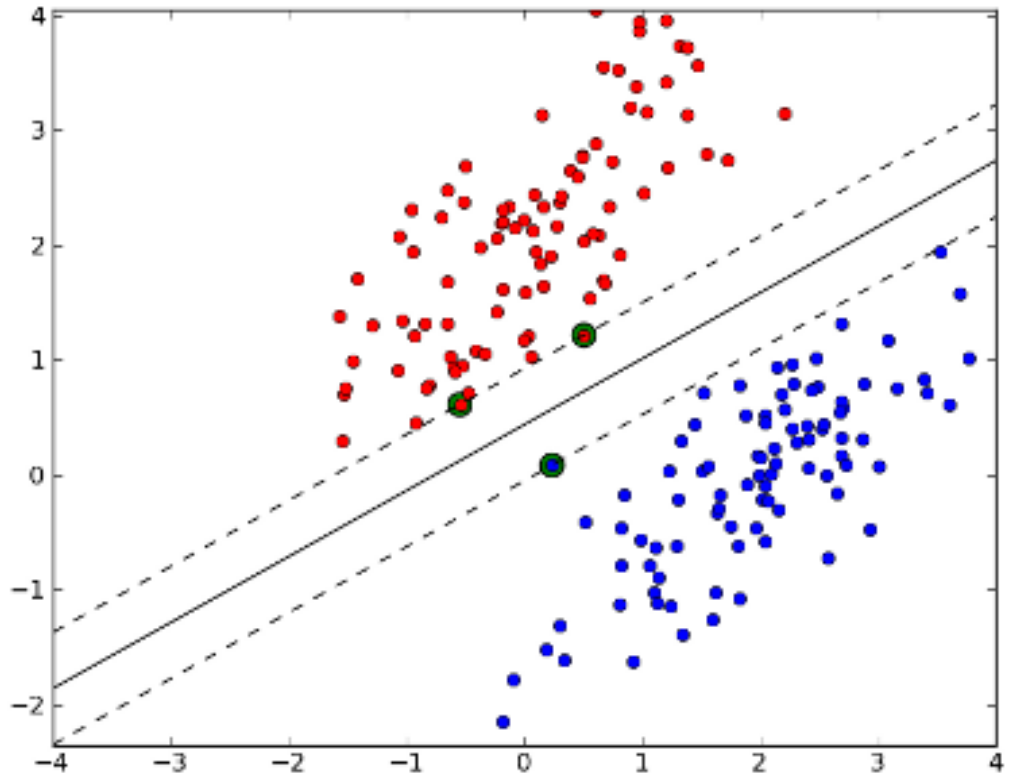


Once the hyper-plane is defined, the model is able to predict the class label for unseen samples.

Support Vector Machine

The main parameter to be set in a SVM classifier is the kernel function, which is mostly set as linear, polynomial or Gaussian Radial Basis Function (RBF). Linear kernels are defined as

$$k(x_i, x_j) = (x_i \cdot x_j)$$



Support Vector Machine | Proprietà degli Iperpiani

$D(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x} + b$

$\mathbf{w}$: vettore normale all'iperpiano

$b / ||\mathbf{w}||$: distanza dall'origine

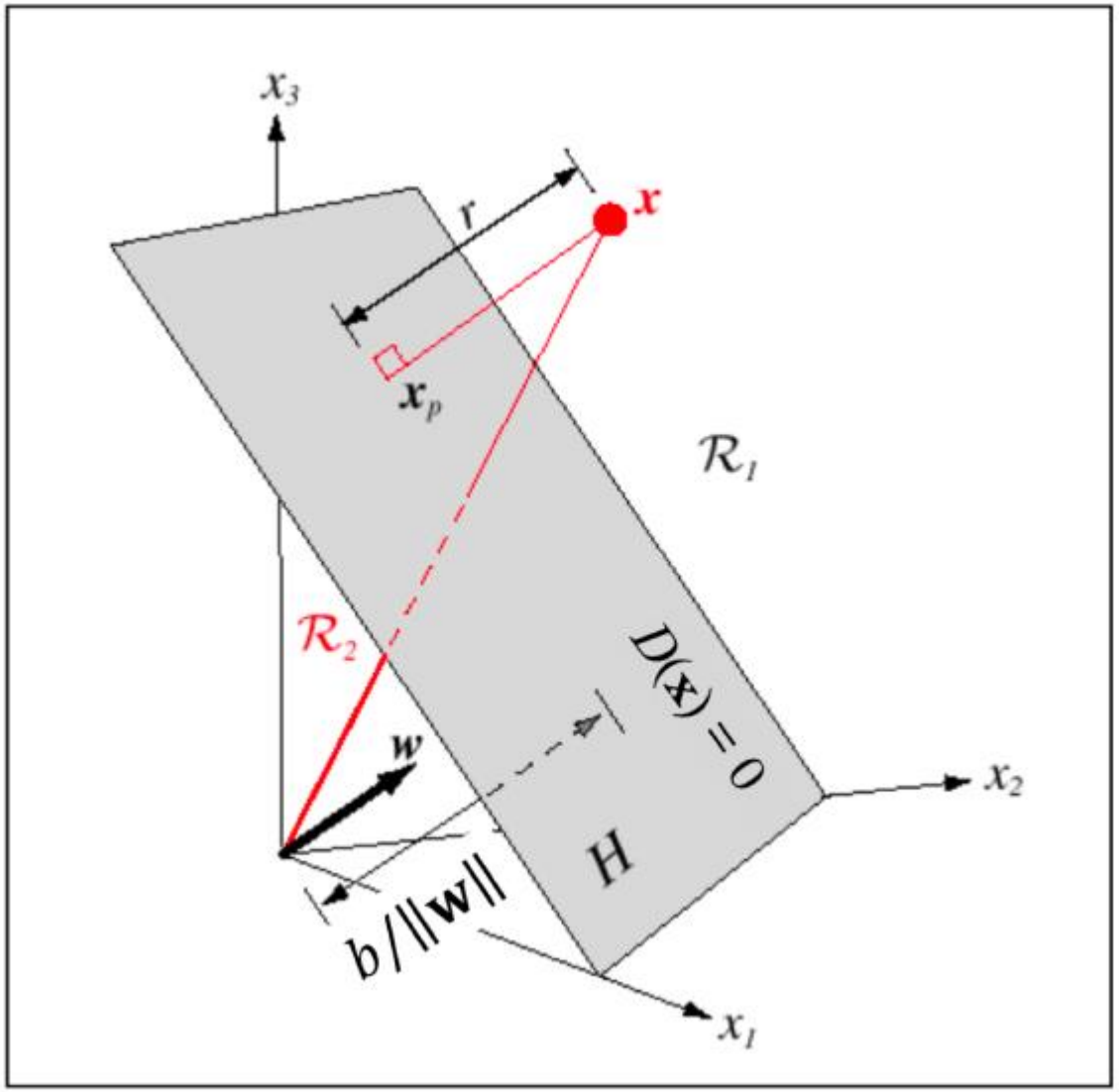
$D(\mathbf{x}) = 0$: luogo dei vettori sull'iperpiano

$D(\mathbf{x}_p) = 0$

$\mathbf{x} = \mathbf{x}_p + r\, \mathbf{w} / ||\mathbf{w}||$

$D(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x} + b = r\, ||\mathbf{w}||$

distanza di un vettore $\mathbf{x}$ dall'iperpiano $r = D(\mathbf{x}) / ||\mathbf{w}||$

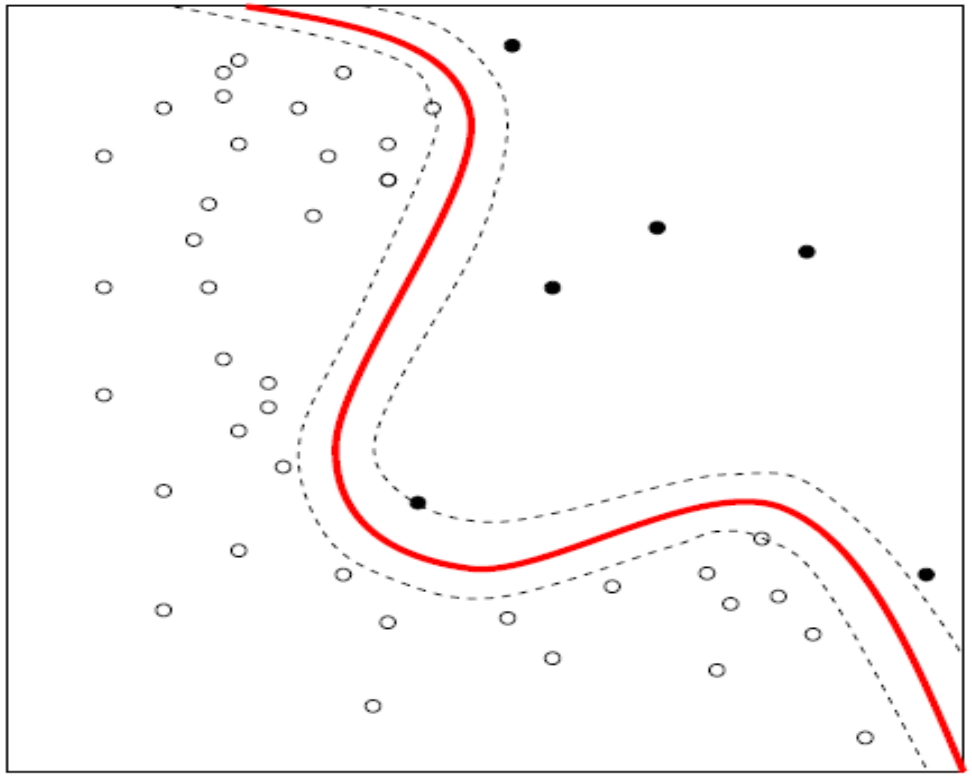


Support Vector Machine

while (homogeneous and inhomogeneous) polynomial kernels are given by

$$k(x_i, x_j) = (x_i \cdot x_j)^d$$

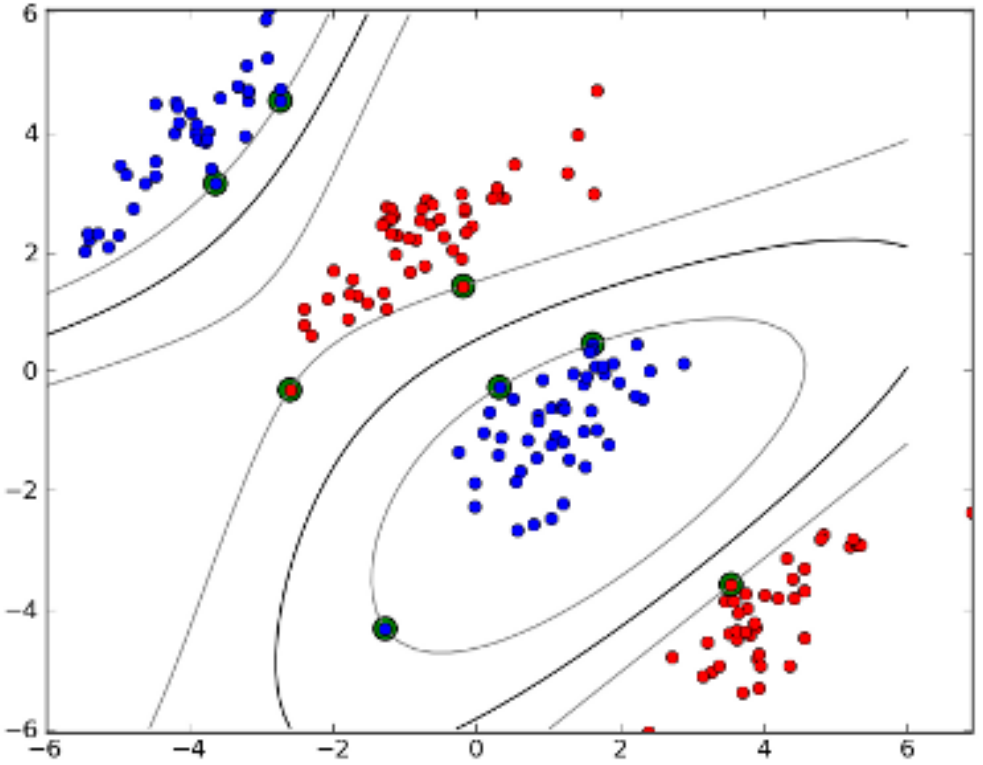
$$k(x_i, x_j) = (x_i \cdot x_j + 1)^d$$



Support Vector Machine

and Gaussian RBF kernels
have the following form

$$k(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right) \qquad \gamma > 0$$



Support Vector Machine

Imagine we are given N observations, each one consisting of a pair: an input vector $x_n \in R^N$, n = 1, ...N and the corresponding target value $t_n \in \{\pm 1\}$, given to us by a trusted source. Data are assumed to be *iid*, i.e., independently drawn from an unknown probability distribution P(x; t) and identically distributed. In our example, $x_n \in R^N$ might be a vector of pixel values (representing images of patients) and $t_n \in \{\pm 1\}$ would be +1 for images belonging to the feature class, -1 for images belonging to the normal class. The aim is to estimate a function that will correctly classify unseen examples (x; t), i.e., we want f(x) = t for samples (x; t) that were also generated from P(x; t). This problem reduces to the goal of separating the two classes by a function which is induced from available examples, in order to produce a classifier, based on input-output data training, which will work well on unseen samples. Let us now make the further assumption that the training data set is linearly separable in feature space and consider the example in Figure 1.3.1: here there are many possible linear classifiers that can separate the data, but there is only one that maximizes the margin (i.e., the distance between it and the nearest data point of each class). This classifier is termed the optimal separating hyper-plane. Intuitively, we would expect this boundary to generalize well as opposed to the other possible boundaries (Figure 1.3.1).

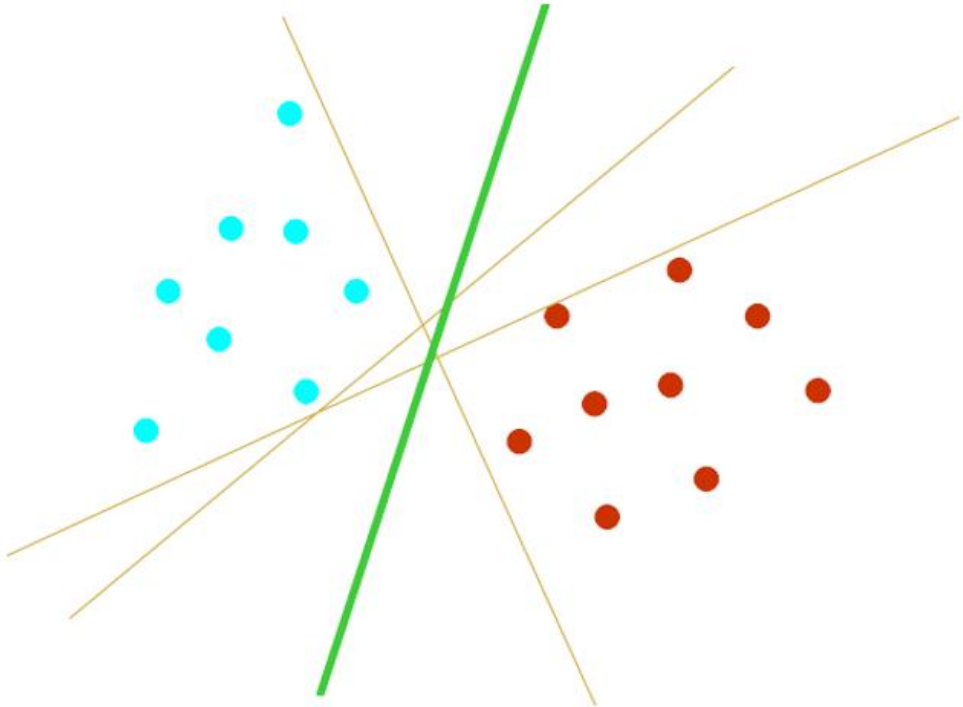


Figure 1.3.1 *Optimal separating hyper-plane*

Support Vector Machine

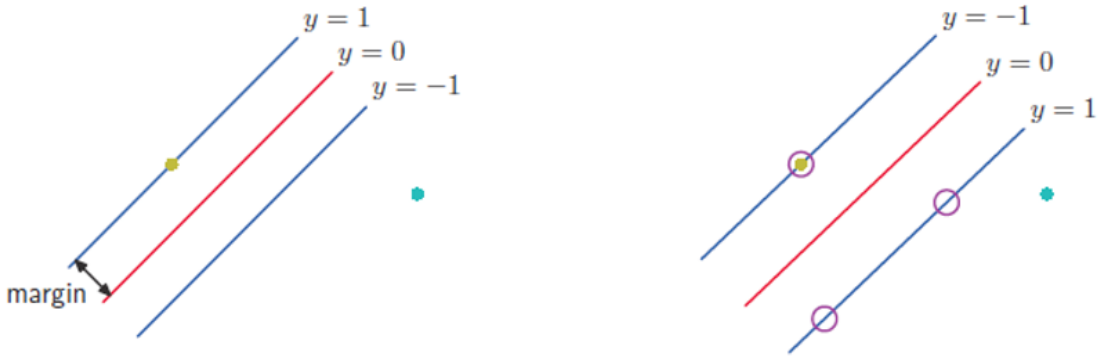


Figure 1.3.2 *Margin and support vectors*

If we consider the following class of hyper-planes (linear model)

$$y(x) = \omega^T \varphi(x) + b \qquad (1.3.1)$$

Where $\varphi(x)$ denotes a fixed feature-space transformation and the explicit parameter b is called bias parameter, then new data points x can be classified according to the sign of $y(x)$. In fact, as we assumed the training data set to be linearly separable in feature space, by definition there exists at least one choice of the parameters w and b such that a function of the form 1.3.1 satisfies $y(x_n) > 0$ for points having $t_n = +1$ and $y(x_n) < 0$ for points having $t_n = -1$. Furthermore, $t_n \cdot y(x_n) > 0$ for all training data points. As we said before, there may exist many such solutions that separate the classes exactly. This problem can be solved by introducing the concept of the margin, which is defined to be the smallest distance between the decision boundary and any of the samples, as illustrated in Figure 1.3.2.

In SVM, the decision boundary is chosen to be the one for which the margin is maximized. The maximum margin solution can be motivated using computational learning theory, also known as statistical learning theory or VC (Vapnik-Chervonenkis) theory.

Support Vector Machine

It must be noted that the margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left part of Figure 1.3.2. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right part of Figure 1.3.2. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles. The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left part of Figure 1.3.2. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right of Figure 1.3.2. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.

In general, the perpendicular distance of a point x from a hyper-plane defined by y(x) = 0, where y(x) takes the form specified in equation 1.3.1, is given by $|y(x)|/\|w\|$. Furthermore, in this case we are only interested in solutions for which all data points are correctly classified, so that $t_n \cdot y(x_n) > 0$ for all n. Thus the distance of a point x_n to the decision surface is given by

$$\frac{t_n \cdot y(x_n)}{\|w\|} = \frac{t_n \cdot (\omega^T \varphi(x) + b)}{\|w\|} \qquad (1.3.2)$$

Support Vector Machine

The margin is given by the perpendicular distance to the closest point x_n from the data set, and we wish to optimize the parameters w and b in order to maximize this distance. Thus the maximum margin solution is found by solving

$$argmax_{W,b}\left\{\frac{1}{\|w\|}min_n[t_n\cdot(\omega^T\varphi(x)+b)]\right\} \quad (1.3.3)$$

where we have taken the factor $\frac{1}{\|w\|}$ outside the optimization over n because w does not depend on n. Solving this optimization problem simply reduces to the requirement of minimizing $\|w\|^2$

$$argmax_{W,b}\left\{\frac{1}{2}\|w\|^2\right\} \quad (1.3.4)$$

(the factor of $\frac{1}{2}$ is included for later convenience), subject to the constraints

$$t_n\cdot(\omega^T\varphi(x)+b\ \geq 1, n=1,\dots N \quad (1.3.5)$$

Support Vector Machine

It appears that the bias parameter b has disappeared from the optimization. However, it is determined implicitly via the constraints, because these require that changes to $\|w\|$ be compensated by changes to b. This constrained optimization problem is dealt with by introducing Lagrange multipliers $a_n > 0$, with one multiplier a_n for each of the constraints in equation 1.3.5, giving the Lagrangian function

$$L(\omega, b, a) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} a_n\{t_n \cdot (\omega^T \varphi(x) + b) - 1\} \qquad (1.3.6)$$

where $a = (a_1, \dots a_N)^T$. The negative sign in front of the Lagrange multiplier term is a consequence of the fact that minimization is performed with respect to w and b, and maximization is performed with respect to a. By setting the derivatives of $L(\omega, b, a)$ with respect to w and b equal to zero, the two following conditions can be obtained:

$$\omega = \sum_{n=1}^{N} a_n \cdot t_n \cdot \varphi(x) \qquad (1.3.7)$$

and

$$\sum_{n=1}^{N} a_n \cdot t_n = 0 \qquad (1.3.8)$$

Support Vector Machine

Eliminating w and b from $\mathrm{L}(\omega, \mathrm{b}, \mathrm{a})$ using these conditions, then, gives the dual representation of the maximum margin problem in which we maximize

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_m a_n t_m t_n k(x_n, x_m) \qquad (1.3.9)$$

with respect to the constraints

$$a_n \geq 0, n = 1, \ldots N \qquad (1.3.10)$$

$$\sum_{n=1}^{N} a_n \cdot t_n = 0 \qquad (1.3.11)$$

Here, the kernel function is defined by $k(x, x') = \varphi(x)^T \cdot \varphi(x)'$: this kernel formulation makes clear the role of the constraint that the kernel function $\mathrm{k}(\mathrm{x}, \mathrm{x}')$ be positive definite, because this ensures that the Lagrangian function $\tilde{L}(a)$ is bounded below, giving rise to a well-defined optimization problem. In order to classify new data points using the trained model, we evaluate the sign of y(x) defined in equation 1.3.1. This can be expressed in terms of the parameters an and the kernel function by substituting for ω using 1.3.7 to give the hyper-plane decision function

$$y(x) = \sum_{n=1}^{N} a_n \cdot t_n \cdot k(x, x_n) + b \qquad (1.3.12)$$

Support Vector Machine

It can be shown that a constrained optimization of this form satisfies the Karush-Kuhn-Tucker (KKT) conditions, which, in this case, require that the following three properties hold:

$$a_n \geq 0 \tag{1.3.13}$$

$$t_n \cdot y(x_n) - 1 \geq 0 \tag{1.3.14}$$

$$a_n\{t_n \cdot y(x_n) - 1\} = 0 \tag{1.3.15}$$

Thus for every data point, either $a_n = 0$ or $t_n \cdot y(x_n) = 1$. Any data point for which $a_n = 0$ will not appear in the sum in 1.3.12 and, hence, they will play no role in making predictions for new data points. The remaining data points are called support vectors, and because they satisfy $t_n \cdot y(x_n) = 1$, they correspond to points that lie on the maximum margin hyper-planes in feature space, as illustrated in Figure 1.3.2. This property is central to the practical applicability of SVM. Once the model is trained, a significant proportion of the data points can be discarded and only the support vectors retained. Having solved the quadratic programming problem and found a value for a, we can then determine the value of the threshold parameter b by noting that any support vector x_n satisfies $t_n \cdot y(x_n) = 1$.

Support Vector Machine

Using 1.3.12, this gives

$$t_n\left(\sum_{m\in S} a_m t_m k(x_n, x_m) + b\right) = 1 \qquad (1.3.16)$$

where S denotes the set of indices of the support vectors. Although we can solve this equation for b using an arbitrarily chosen support vector x_n, a numerically more stable solution is obtained by first multiplying through by t_n, making use of $(t_n)^2 = 1$, and, then, averaging these equations over all support vectors and solving for b to give

$$b = \frac{1}{N_S}\sum_{n\in S}\left(t_n - \sum_{m\in S} a_m t_m k(x_n, x_m)\right) \qquad (1.3.17)$$

where N_S is the total number of support vectors.

Support Vector Machine

In practice...

Support Vector Machine

In practice, a separating hyper-plane may not exist, e.g., if a high noise level causes a large overlap of the class-conditional distribution. In order to overcome this limitation, Cortes and Vapkin (1995) proposed a modified version of SVM introducing the idea of soft margin, which is useful when training classes cannot be sharply discriminated. Specifically, the soft margin approach allows to misclassify a fraction of training samples, while preserving the ability of the hyper-plane to maximizing its distance from the nearest samples of the two classes.

In order to allow for the possibility of violating 1.3.14, the general approach has to be modified so that data points are allowed to be on the *wrong side* of the margin boundary, but with a penalty that increases with the distance from that boundary. For the subsequent optimization problem, it is convenient to make this penalty a linear function of this distance; to do this, we introduce slack variables

$$\xi_n \geq 0, n = 1, \dots N \qquad (1.3.18)$$

with one slack variable for each training data point. These are defined by $\xi_n = 0$ for data points that are on or inside the correct margin boundary and $\xi_n = |t_n - y(x_n)|$ for other points.

Support Vector Machine

Thus, a data point that is on the decision boundary $y(x_n) = 0$ will have $\xi_n = 1$ and points with $\xi_n > 1$ will be misclassified. In this case, classification constraints relax to

$$t_n \cdot y(x_n) \geq 1 - \xi_n, n = 1, \dots N \qquad (1.3.19)$$

in which the slack variables are constrained to satisfy $\xi_n \geq 0$. Data points for which $\xi_n = 0$ are correctly classified and are either on the margin or on the correct side of the margin. Points for which $0 < \xi_n \leq 1$ lie inside the margin, but on the correct side of the decision boundary, and those data points for which $\xi_n \geq 1$ lie on the wrong side of the decision boundary and are misclassified, as illustrated in Figure 1.3.3. This is sometimes described as relaxing the hard margin constraint to give a soft margin and it allows some of the training set data points to be misclassified. Note that while slack variables allow for overlapping class distributions, this framework is still sensitive to outliers, because the penalty for misclassification increases linearly with ξ. In order to realize a soft margin classifier, we now have to minimize

$$\frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n \qquad (1.3.20)$$

subject to the constraints 1.3.18 and 1.3.19; here, the parameter C > 0 controls the trade-off between the slack variable penalty and the margin.

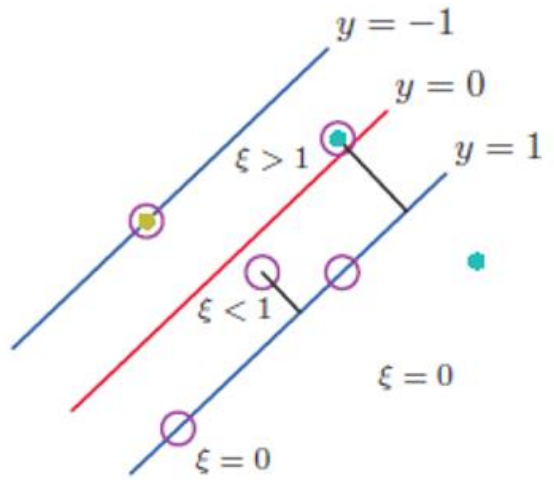


Figure 1.3.3 *Slack variables in support vector classifiers objective function. Data points with circles around them are support vectors.*

Support Vector Machine

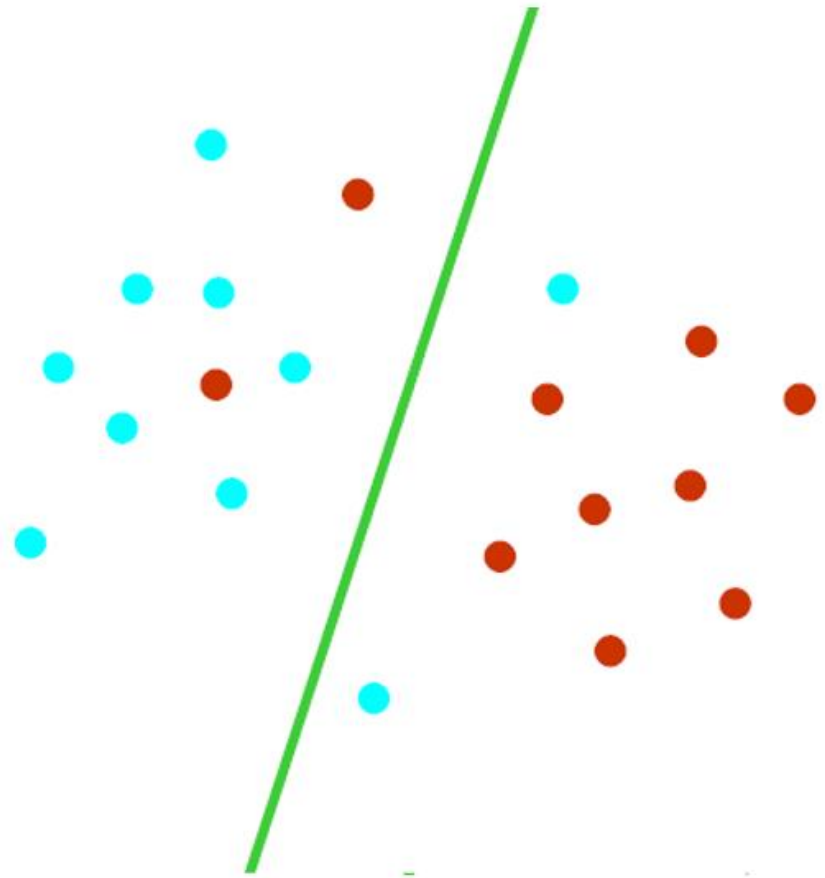


Figure 1.3.4 *Generalized optimal separating hyper-plane*

Incorporating kernels and rewriting it in terms of Lagrange multipliers, this again leads to the problem of maximizing 1.3.9, subject to the constraints

$$0 \leq a_n \leq C \tag{1.3.21}$$

$$\sum_{n=1}^{N} a_n t_n = 0 \tag{1.3.22}$$

for $n = 1, \ldots N$ (constraints in 1.3.21 are known as box constraints).

The only difference from the separable case is the upper bound C on the Lagrange multipliers a_n. In this way, the influence of the individual patterns (which could be outliers) gets limited. As above, the solution shows that predictions for new data points are again made by using 1.3.12. The threshold b can be computed by exploiting the fact that for all support vectors x_n, with $0 \leq a_n \leq C$, the slack variable $\xi_n = 0$ and, hence, will satisfy 1.3.16. As before, a subset of the data points may have $a_n = 0$, in which case it does not contribute to the predictive model 1.3.12. The remaining data points constitute the support vectors.

Support Vector Machine

In practice, when dealing with a SVM-based classifier, the main parameter to be set for a study is the kernel function. The most popular kernels in literature are the linear, the polynomial and the Gaussian Radial Basis Function (RBF) kernels:

$$k(x_i, x_j) = (x_i \cdot x_j) \tag{1.3.23}$$

$$k(x_i, x_j) = (x_i \cdot x_j)^d \tag{1.3.24}$$

$$k(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{1.3.25}$$

$$k(x_i, x_j) = exp\left(-\gamma \|x_i - x_j\|^2\right) \qquad \gamma > 0 \tag{1.3.26}$$

where relation 1.3.23 is associated to linear kernels, relations 1.3.24-25 are associated to (homogeneous and inhomogeneous) polynomial kernels and relation 1.3.26 is associated to Gaussian RBF kernels, respectively.

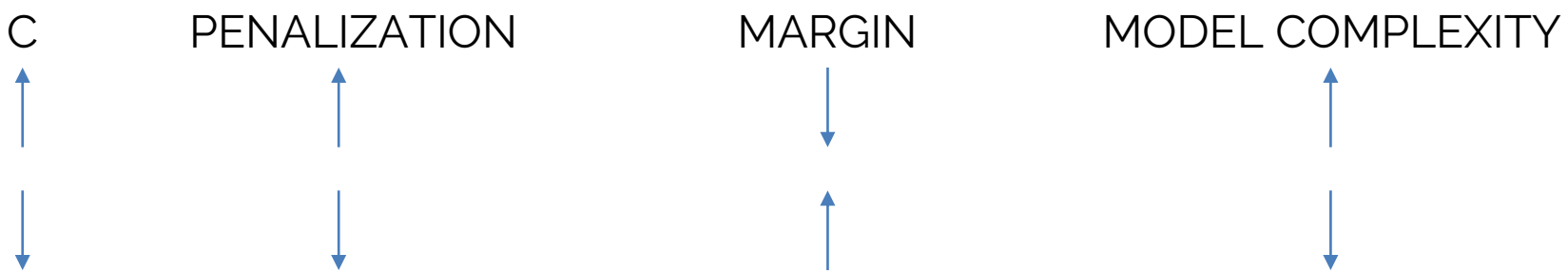
Support Vector Machine | C HYPERPARAMETER

The C parameter is related to "how much you want to avoid misclassification" of each training example

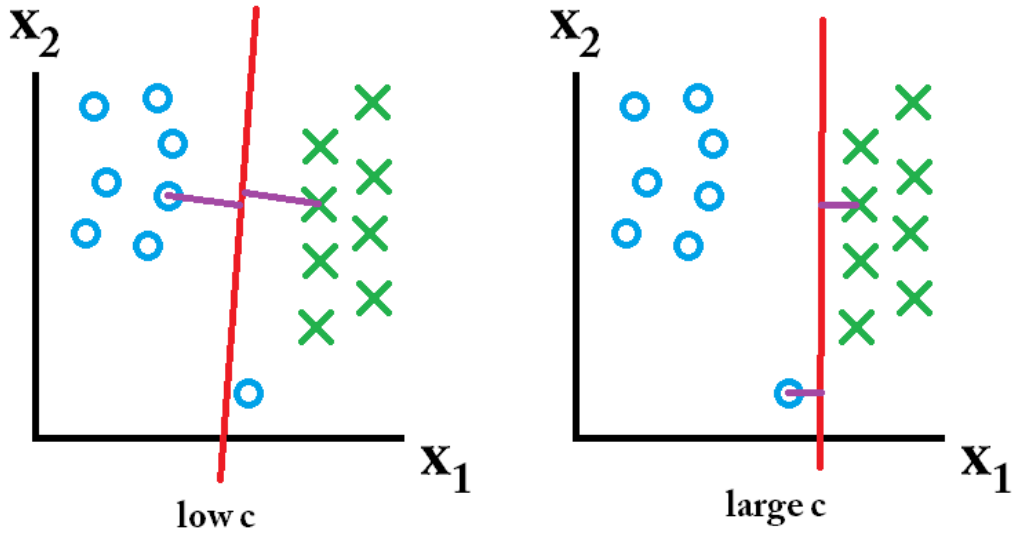
LARGE VALUES of C correspond to a smaller-margin separating hyperplane if that hyperplane correctly classifies all training data

SMALL VALUES of C correspond to a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

CONSIDER THAT an optimal C parameter should be larger when you scale down your data, smaller when data are not scaled



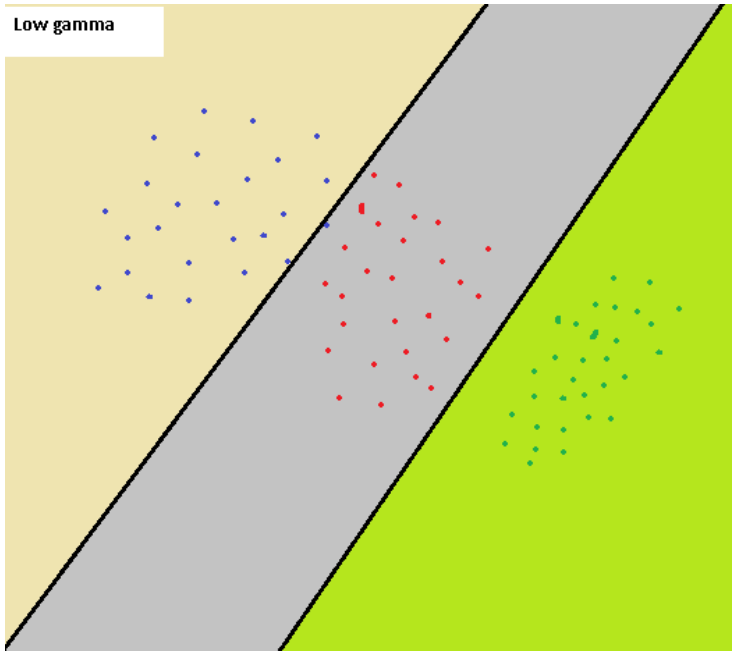
Support Vector Machine | C HYPERPARAMETER

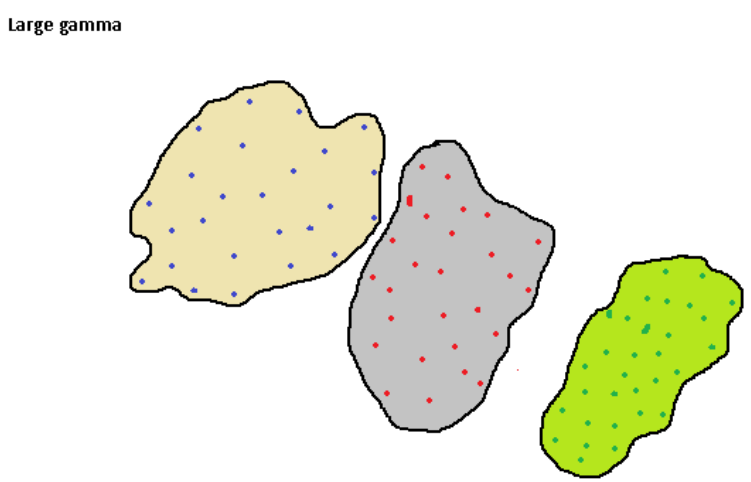


C ↕ PENALIZATION ↕ MARGIN ↕ MODEL COMPLEXITY

Support Vector Machine | GAMMA HYPERPARAMETER

"Gamma parameter of RBF controls the distance of influence of a single training point. Low values of gamma indicates a large similarity radius which results in more points being grouped together. For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class). Therefore, models with very large gamma values tend to overfit."





If gamma is large, the effect of c becomes negligible.
If gamma is small, c affects the model just like how it affects a linear model.

$$0.0001 < \text{gamma} < 10$$

$$0.1 < c < 100$$

Support Vector Machine

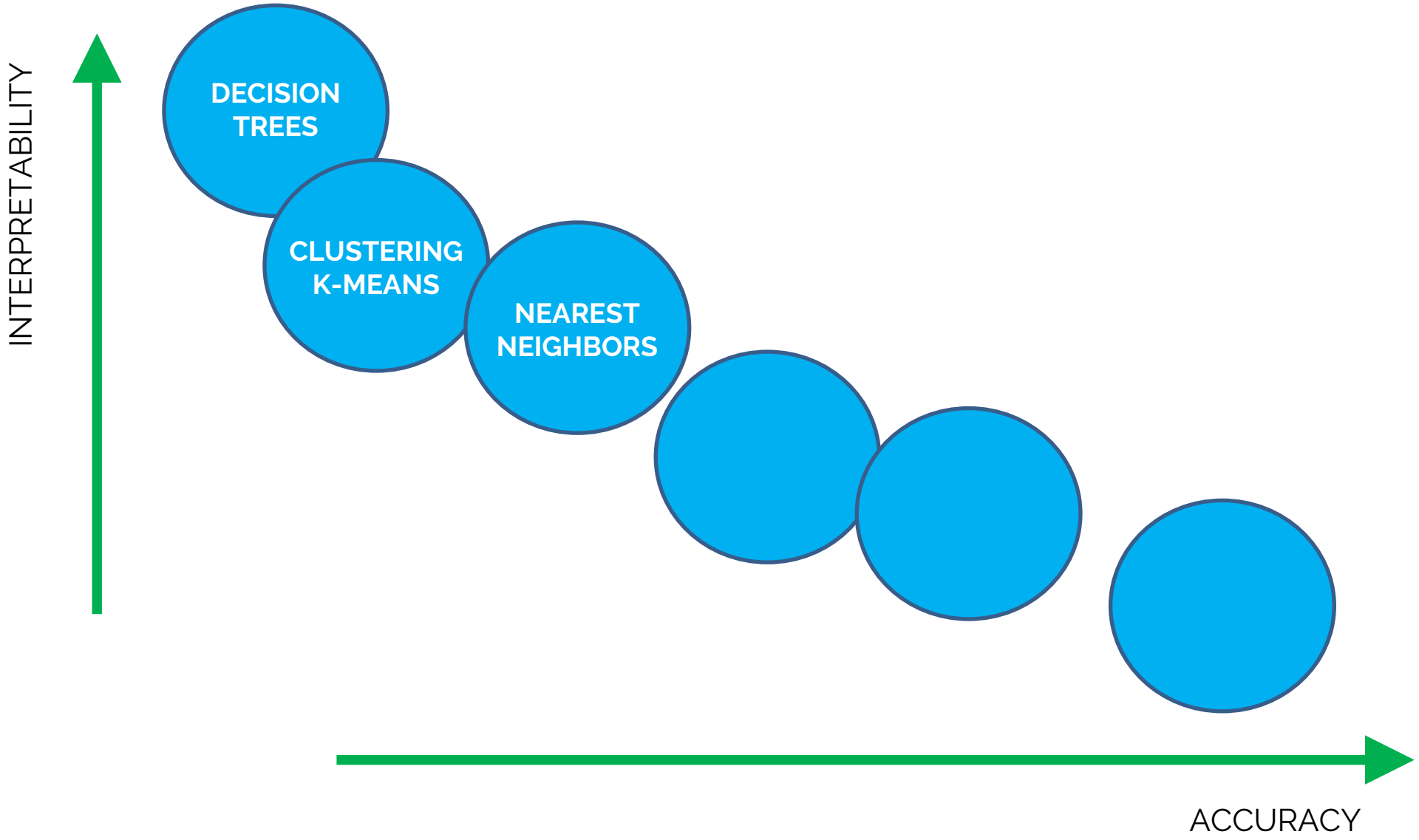
Advantages:

- Very powerful framework
- SVMs work well in practice (even with small datasets)

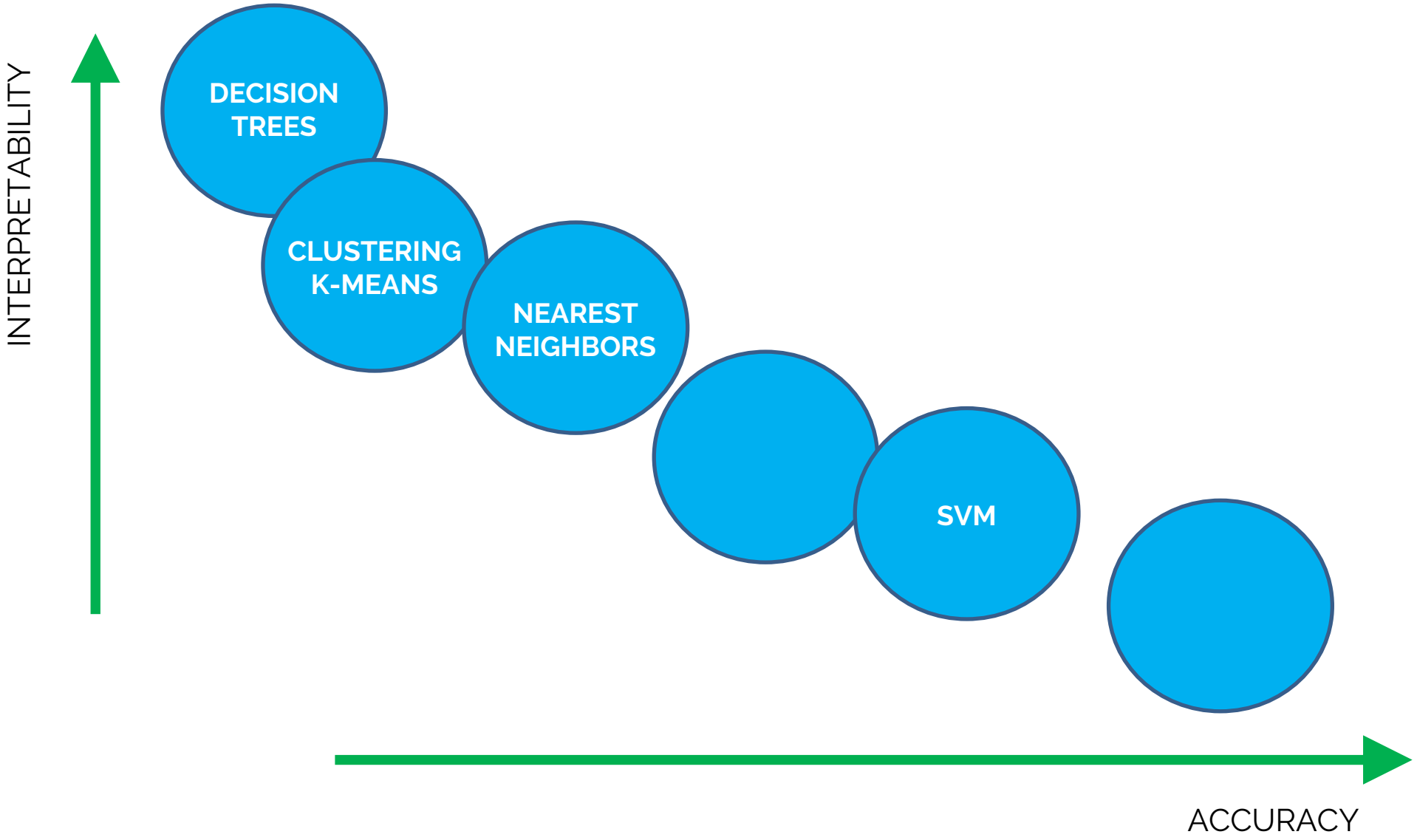
Drawbacks:

- High computational costs for training
- SVMs only allow direct binary classification
- Classification [+1/-1] is the only output (no probability-of-classification is given)

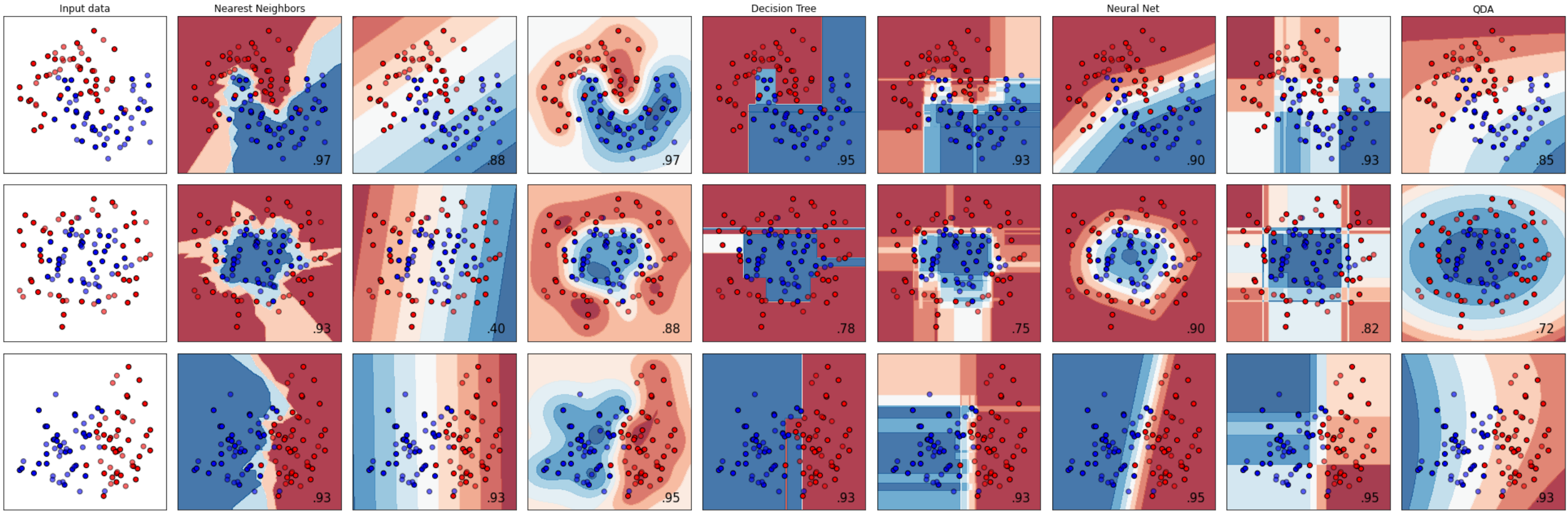
Interpretability-Accuracy TRADEOFF



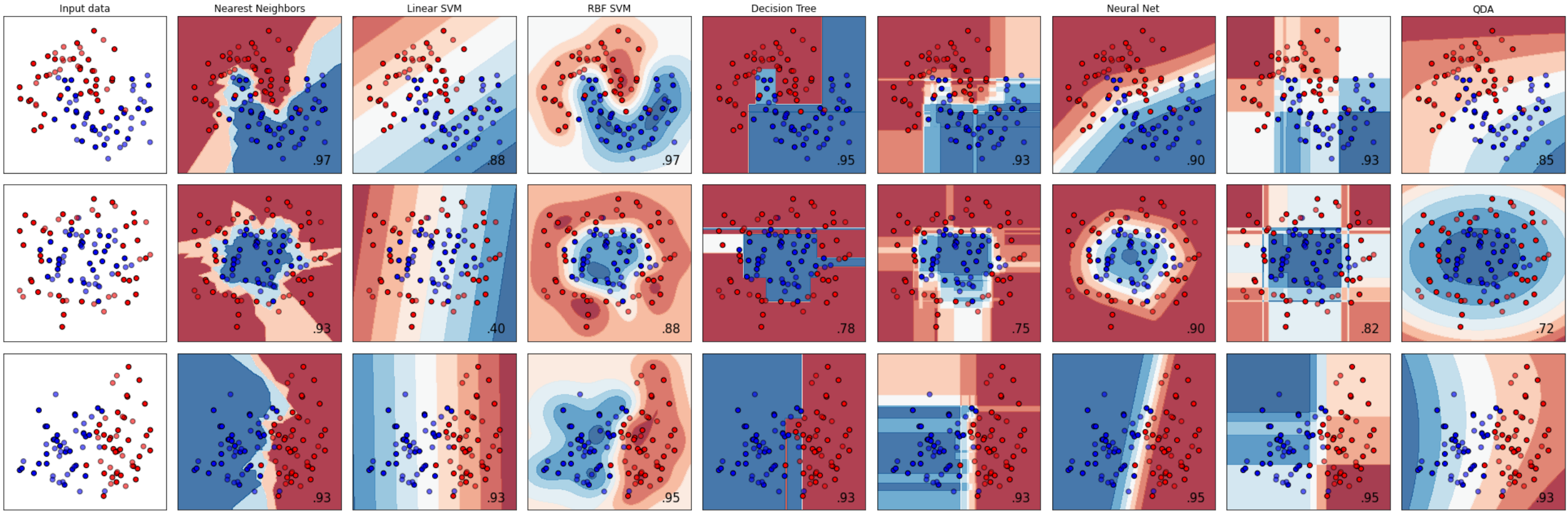
Interpretability-Accuracy TRADEOFF



Which is which | Decision Function



Which is which | Decision Function



Support Vector Machine

- Consider the following generalization of the *k*-NN algorithm (specialized to binary classification):

$$\hat{y}(\vec{x}) \leftarrow \text{sign}\left(\sum_{i=1}^{N} y_i d(\vec{x}_i, \vec{x})\right) \quad \text{with } d(\vec{x}_i, \vec{x}) = \frac{1}{||\vec{x}_i - \vec{x}||_2^2} \text{ or... } d(\vec{x}_i, \vec{x}) = \exp\left(-\frac{||\vec{x}_i - \vec{x}||_2^2}{2\sigma^2}\right)$$

- Looks at *all* training points (i.e., *k*=N), but weights the *i*'th training point's label by how far $\mathbf{x}_i$ is from $\mathbf{x}$

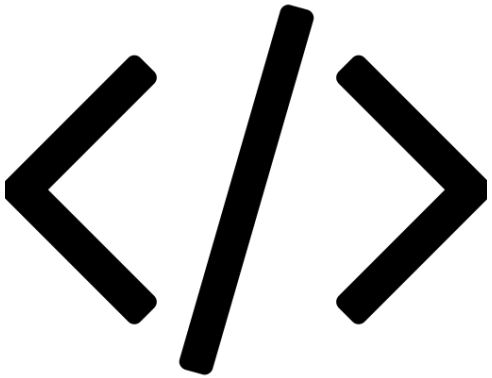
- Now compare this to classification with SVM and a Gaussian kernel:

$$\hat{y}(\vec{x}) \leftarrow \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\vec{x}_i, \vec{x})\right) \qquad K(\vec{u}, \vec{v}) = \exp\left(-\frac{||\vec{u} - \vec{v}||_2^2}{2\sigma^2}\right) \qquad 0 \leq \alpha_i \leq C$$

- The discriminant functions are nearly identical! The SVM has parameters α_i that can be learned to maximize predictive accuracy

FROM DAVID SONTAG

Support Vector Machine



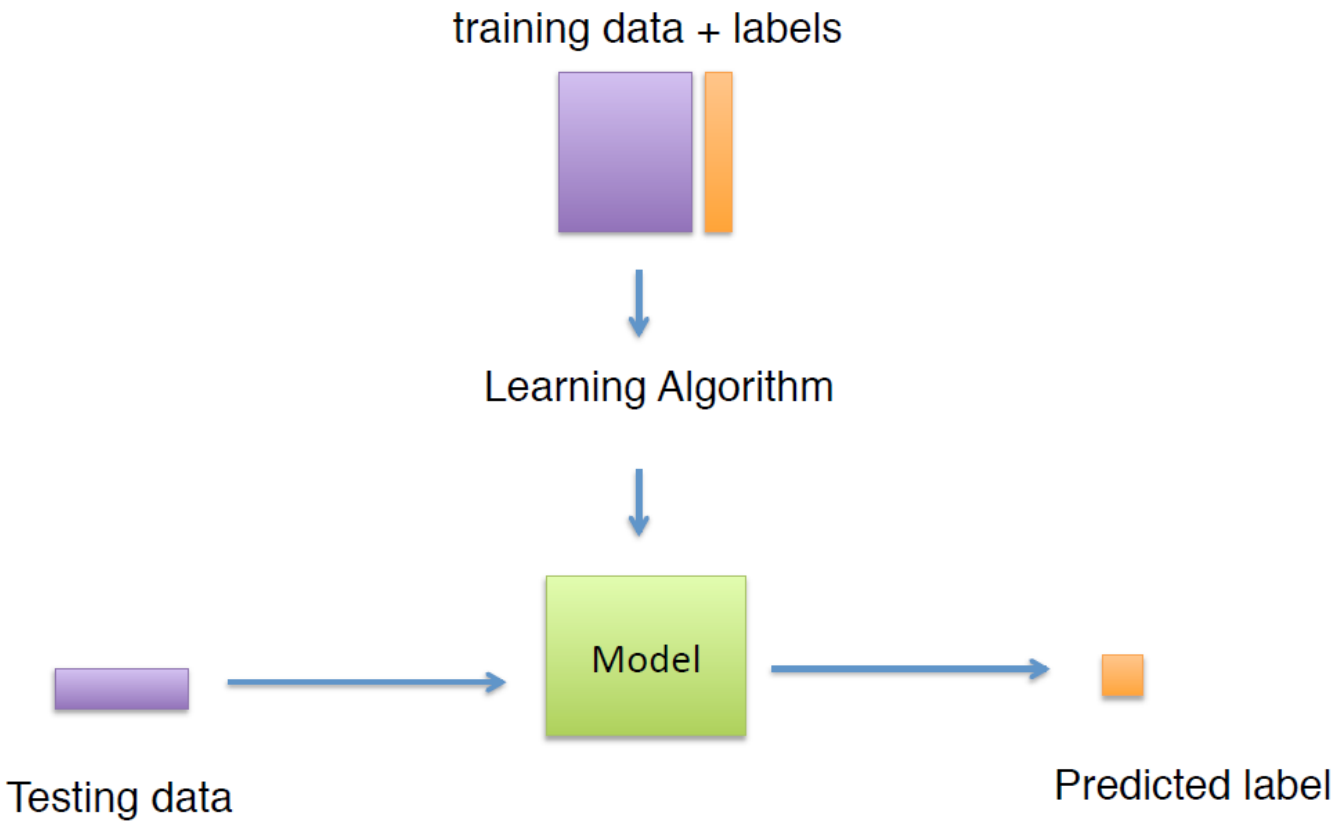
CLASSIFICATION

ENSEMBLES OF CLASSIFIERS

WITH SOME SLIDES FROM PROF. GAVIN BROWN

Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN



Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 12, NO. 10, OCTOBER 1990

Neural Network Ensembles

LARS KAI HANSEN AND PETER SALAMON

***Abstract*—We propose several means for improving the performance and training of neural networks for classification. We use crossvalidation as a tool for optimizing network parameters and architecture. We show further that the remaining residual "generalization" error can be reduced by invoking ensembles of similar networks.**

***Index Terms*—Crossvalidation, fault tolerant computing, neural networks, *N*-version programming.**

I. Introduction

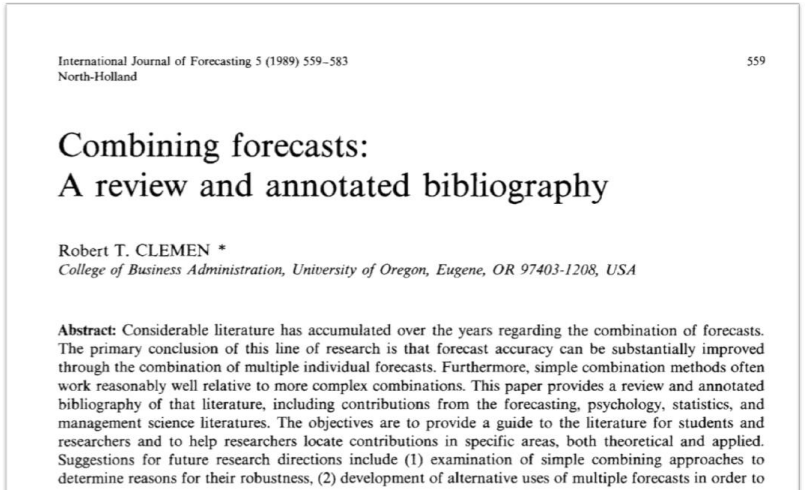
RECENT schemes for training neural networks involving hidden neurons have caused a resurgence of interest in nonalgorithmic supervised learning. A supervised learning scheme is implemented using a *database* which consists of a set of input patterns (a sample from the set of possible inputs) together with the corresponding targets (classifications). The objective of the training is to

performance of the network, we can optimize such performance by varying network characteristics and architecture.

A residual error will typically remain even after optimizing all available network characteristics [6]. To further reduce this error we propose to use a device from fault tolerant computing [7]. We run not a single network but an *ensemble* of networks, each of which have
trained on the same database. The basic idea is to cl
a given input pattern by obtaining a classification
each copy of the network and then using a cons
scheme to decide the collective classification by vo

II. Crossvalidation for Network Optimizati

For supervised learning we employ a database
scribed above including a representative sample of t



International Journal of Forecasting 5 (1989) 559–583
North-Holland

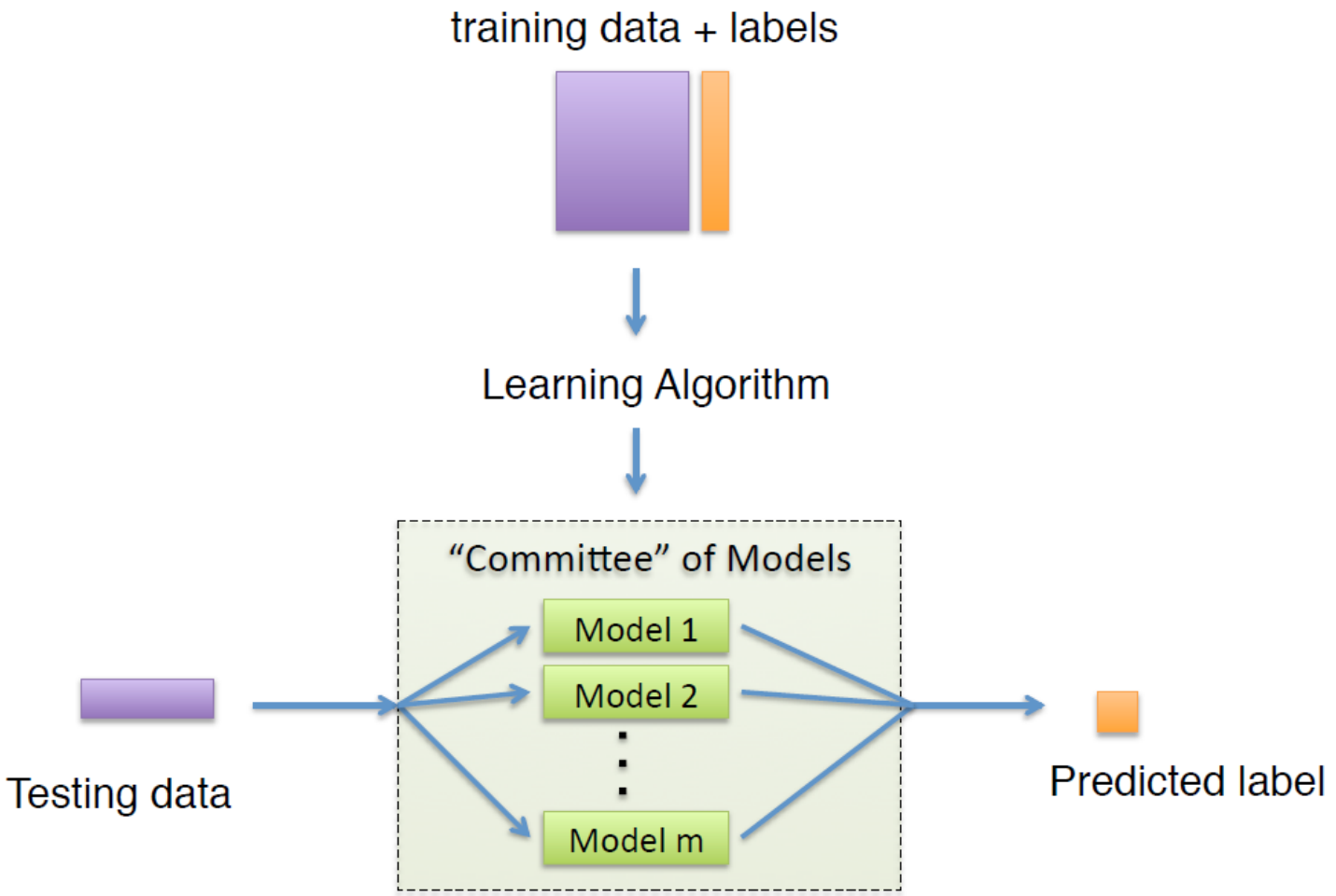
Combining forecasts: A review and annotated bibliography

Robert T. CLEMEN *
College of Business Administration, University of Oregon, Eugene, OR 97403-1208, USA

Abstract: Considerable literature has accumulated over the years regarding the combination of forecasts. The primary conclusion of this line of research is that forecast accuracy can be substantially improved through the combination of multiple individual forecasts. Furthermore, simple combination methods often work reasonably well relative to more complex combinations. This paper provides a review and annotated bibliography of that literature, including contributions from the forecasting, psychology, statistics, and management science literatures. The objectives are to provide a guide to the literature for students and researchers and to help researchers locate contributions in specific areas, both theoretical and applied. Suggestions for future research directions include (1) examination of simple combining approaches to determine reasons for their robustness, (2) development of alternative uses of multiple forecasts in order to

Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN



Ensembles of Classifiers

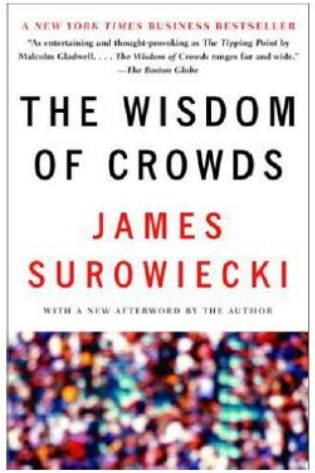
WITH SOME SLIDES FROM PROF. GAVIN BROWN

Combining general predictions?

1906 … county fair in Cornwall, England.

Competition: guess the weight of the cow!

Francis Galton recorded some statistics on the day….



Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN

NATURE [MARCH 7, 1907

mean of the ire for month

l year- Both years. Bulletin ontains station ults of ions in 51 and y in a ation is and is l appli- o with mpera- ximum bsolute bsolute il rain- nt was Most tember. it being J. D.

results leutsche second

Distribution of the estimates of the dressed weight of a particular living ox, made by 787 different persons.

Degrees of the length of Array 0°—100°	Estimates in lbs.	Centiles: Observed deviates from 1207 lbs.	Centiles: Normal p.e =37	Excess of Observed over Normal
5	1074	−133	−90	+43
10	1109	− 98	−70	+28
15	1126	− 81	−57	+24
20	1148	− 59	−46	+13
q_1 25	1162	− 45	−37	+ 8
30	1174	− 33	−29	+ 4
35	1181	− 26	−21	+ 5
40	1188	− 19	−14	+ 5
45	1197	− 10	− 7	+ 3
m 50	1207	0	0	0
55	1214	+ 7	+ 7	0
60	1219	+ 12	+14	− 2
65	1225	+ 18	+21	− 3
70	1230	+ 23	+29	− 6
q_3 75	1236	+ 29	+37	− 8
80	1243	+ 36	+46	−10
85	1254	+ 47	+57	−10
90	1267	+ 52	+70	−18
95	1293	+ 86	+90	− 4

q_1, q_3, the first and third quartiles, stand at 25° and 75° respectively.
m, the median or middlemost value, stands at 50°.
The dressed weight proved to be 1198 lbs.

787 guesses.

Truth	1198 lb (~543kg)
Median	1207 lb
Mean	1197 lb

Combining Votes

In 1786 Nicolas de Condorcet (political theorist) asked how do parliaments behave when voting?

…assuming M independent voters…

Marquis de Condorcet

If a single voter has a probability ϵ of making an error, then

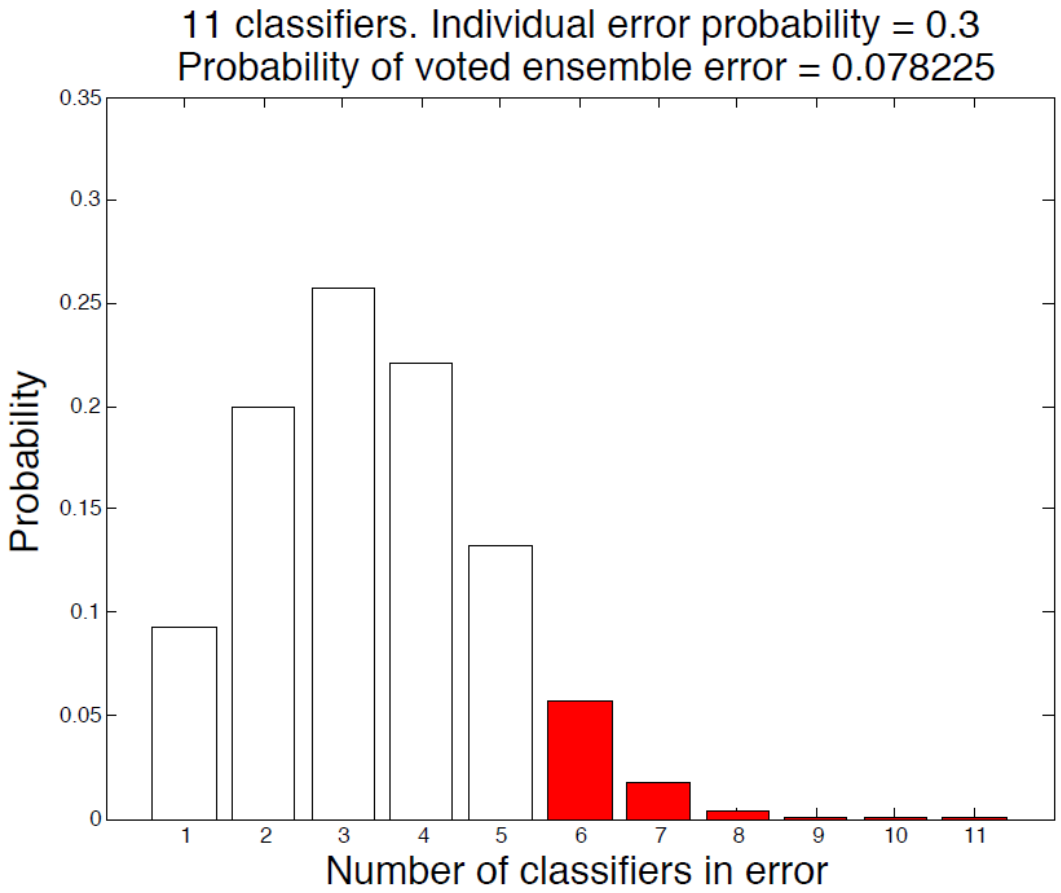
$$p(\text{exactly } k \text{ errors}) = \binom{M}{k} \epsilon^k (1-\epsilon)^{(M-k)}$$

and…

$$p(majority\ vote\ error) = \sum_{k \geq \lceil \frac{M+1}{2} \rceil} \binom{M}{k} \epsilon^k (1-\epsilon)^{(M-k)}$$

Ensembles of Classifiers

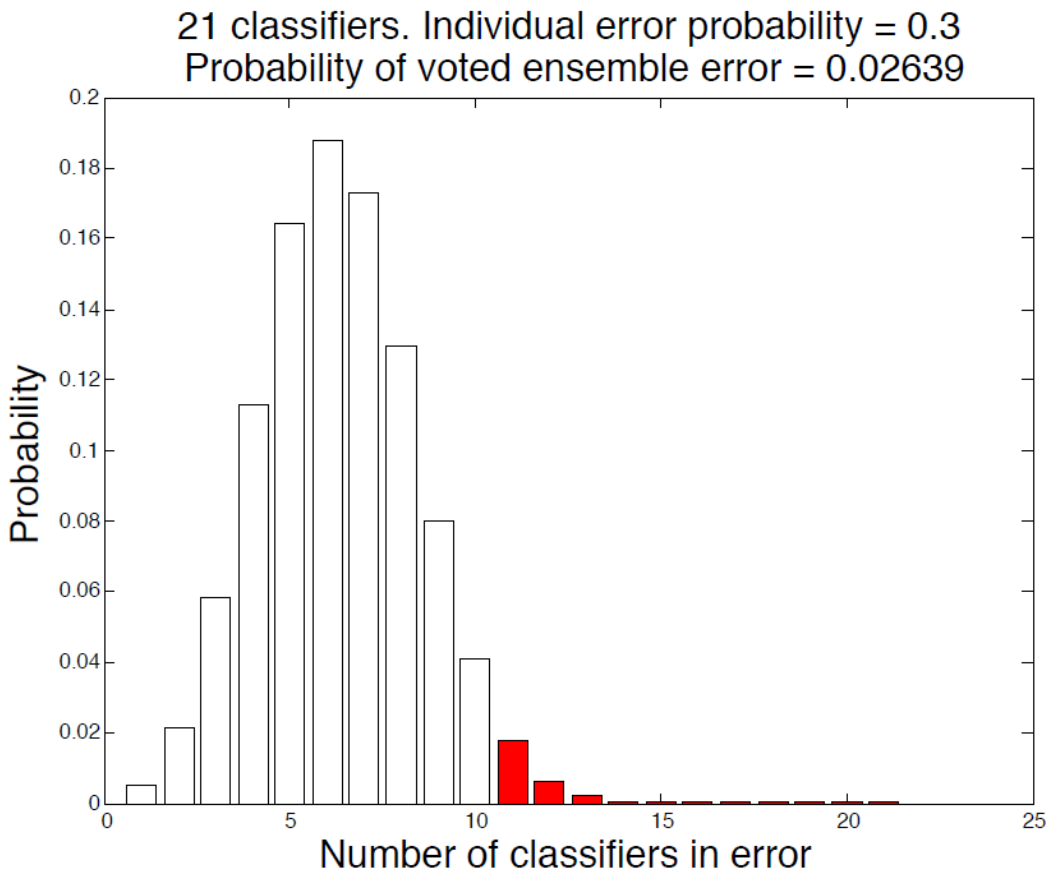
$$p(majority\ vote\ error) = \sum_{k \geq \lceil \frac{M+1}{2} \rceil} \binom{M}{k} \epsilon^k (1-\epsilon)^{(M-k)}$$



Ensembles of Classifiers

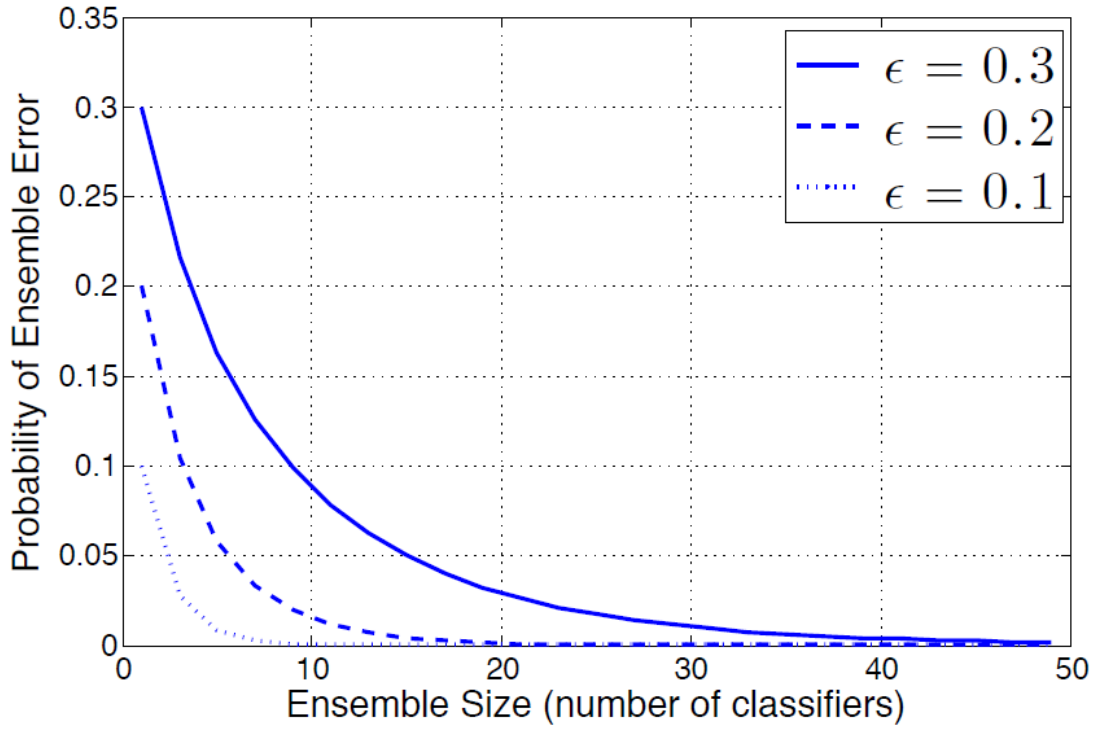
WITH SOME SLIDES FROM PROF. GAVIN BROWN

$$p(majority\ vote\ error) = \sum_{k \geq \lceil \frac{M+1}{2} \rceil} \binom{M}{k} \epsilon^k (1-\epsilon)^{(M-k)}$$



Ensembles of Classifiers

$$p(majority\ vote\ error) = \sum_{k \geq \lceil \frac{M+1}{2} \rceil} \binom{M}{k} \epsilon^k (1-\epsilon)^{(M-k)}$$



Virtually ZERO error by M = 50 !!

$$\varphi\left(\mathbb{E}[X]\right) \leq \mathbb{E}[\varphi(X)]$$

$\phi\left(\sum_{i=1}^{M} \lambda_i x_i\right) \leq \sum_{i=1}^{M} \lambda_i \phi(x_i) \quad \sum_i \lambda_i = 1$

se... $\phi(z) = (z-t)^2$

e $\lambda_i = \frac{1}{M}$ per tutti i valori di i

$\rightarrow$

$\left(\sum_i \lambda_i x_i - t\right)^2 \leq \sum_i \lambda_i (x_i - t)^2$

$\left(\sum_i \frac{x_i}{M} - t\right)^2 \leq \sum_i \frac{(x_i - t)^2}{M}$

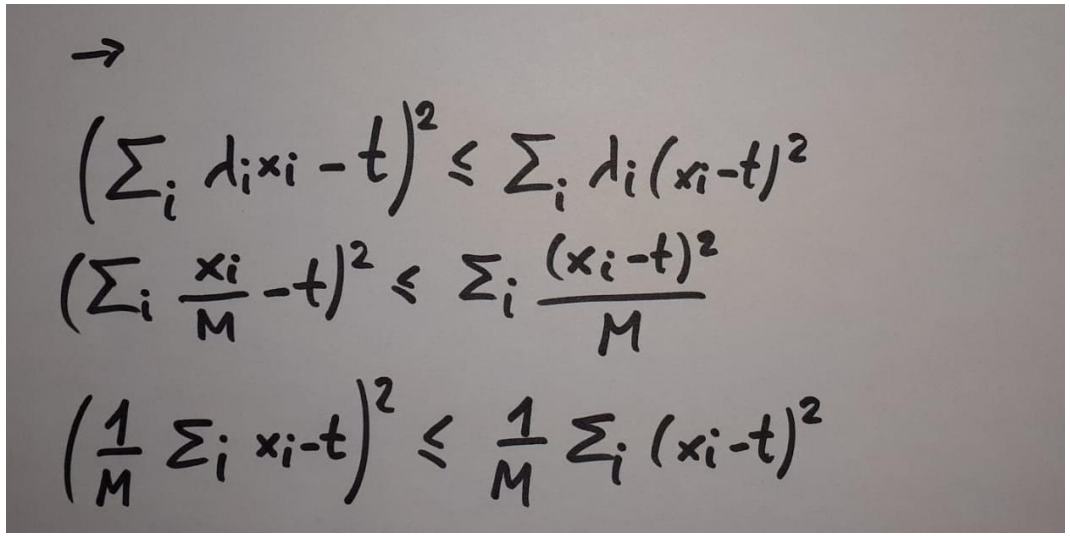
$\left(\frac{1}{M} \sum_i x_i - t\right)^2 \leq \frac{1}{M} \sum_i (x_i - t)^2$

Ensembles of Classifiers

The squared error of a linear combination of predictions...

....is guaranteed to be less than or equal to...

...the average squared error of the individual predictions.



Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN

Training *different* classifiers . . .

but how “different” the single classifiers should be?

Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN

Training *different* classifiers . . .

but how “different” the single classifiers should be?

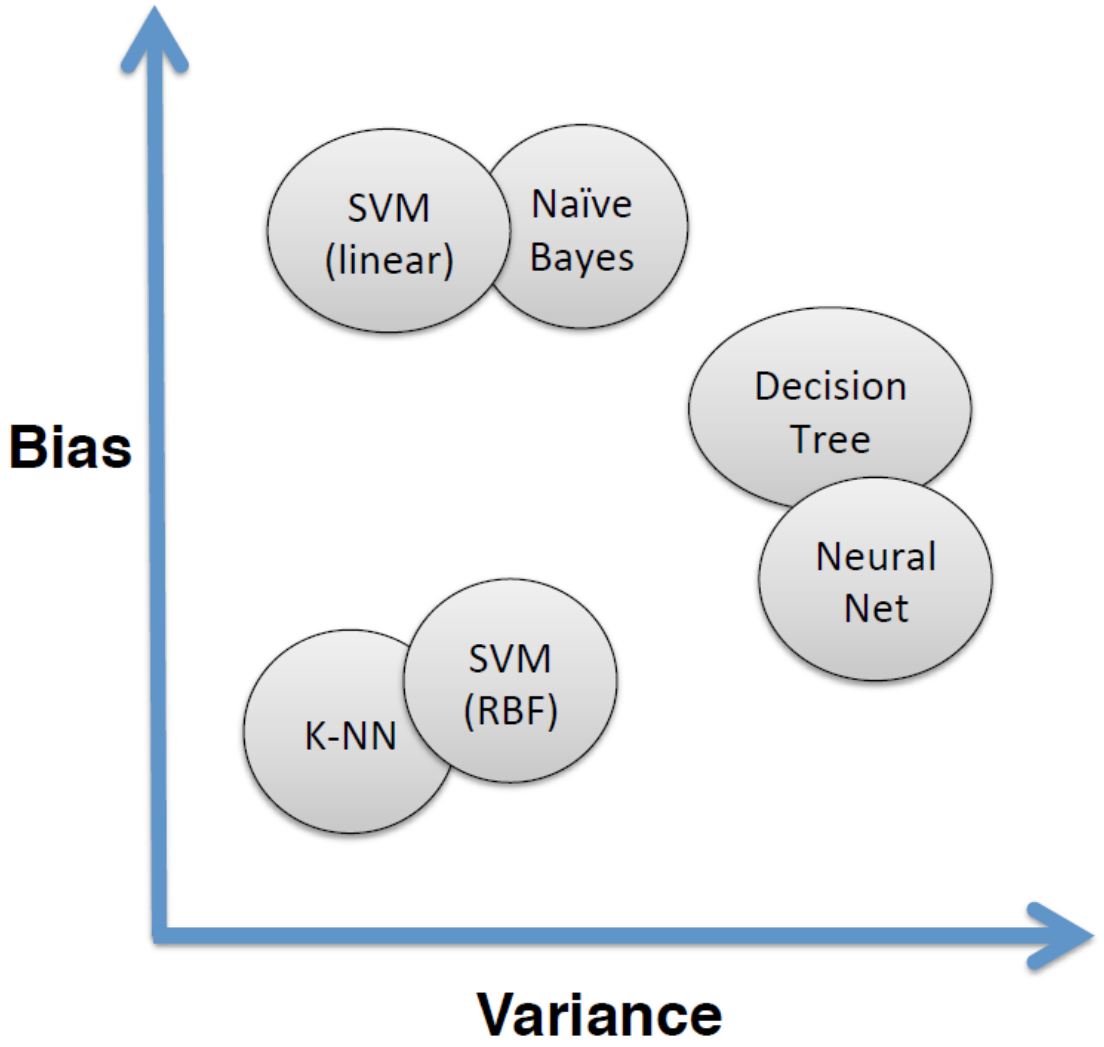
DIFFERENT
(?)

ACCURATE
(?)

INDEPENDENT
(?)

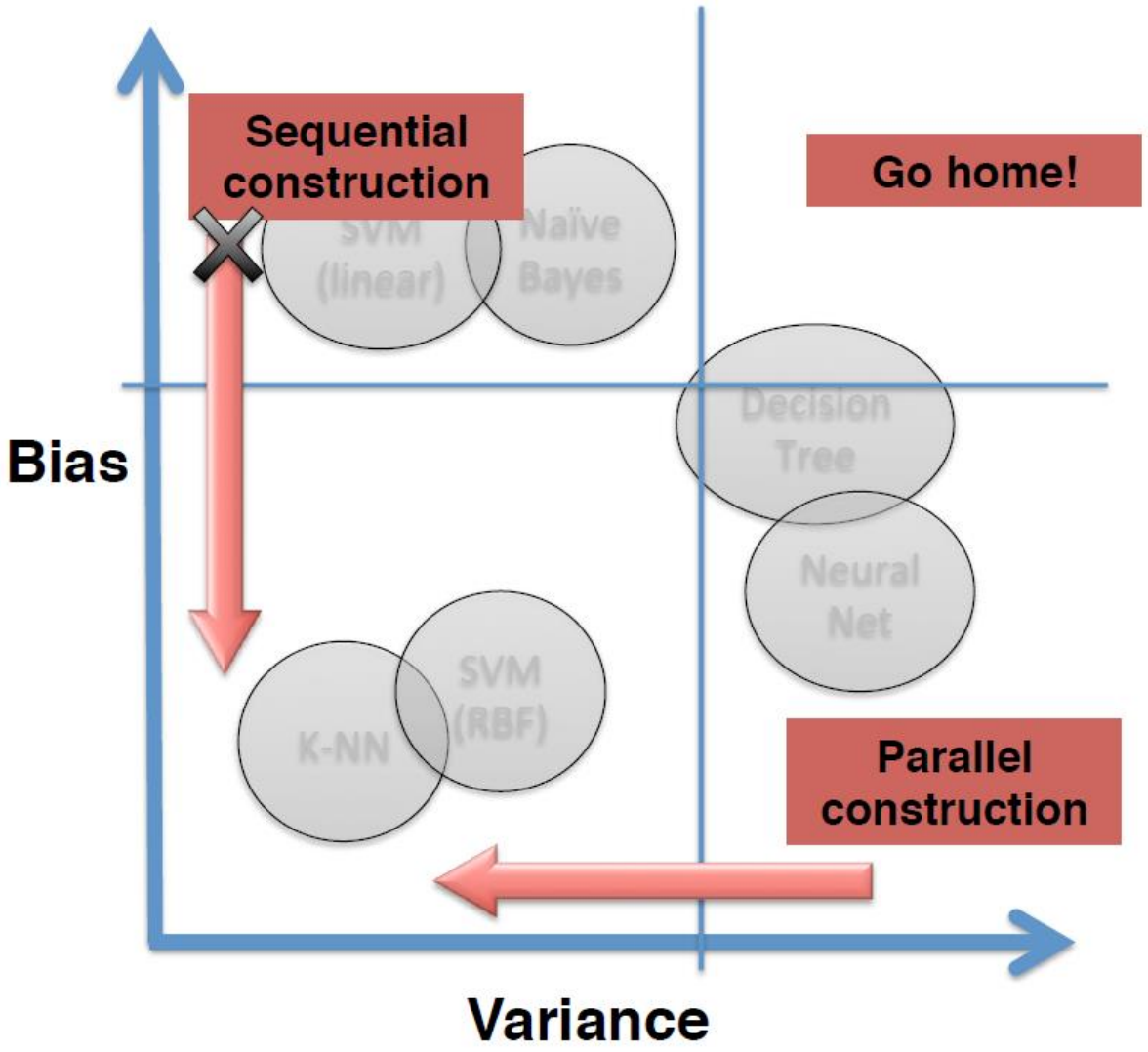
Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN

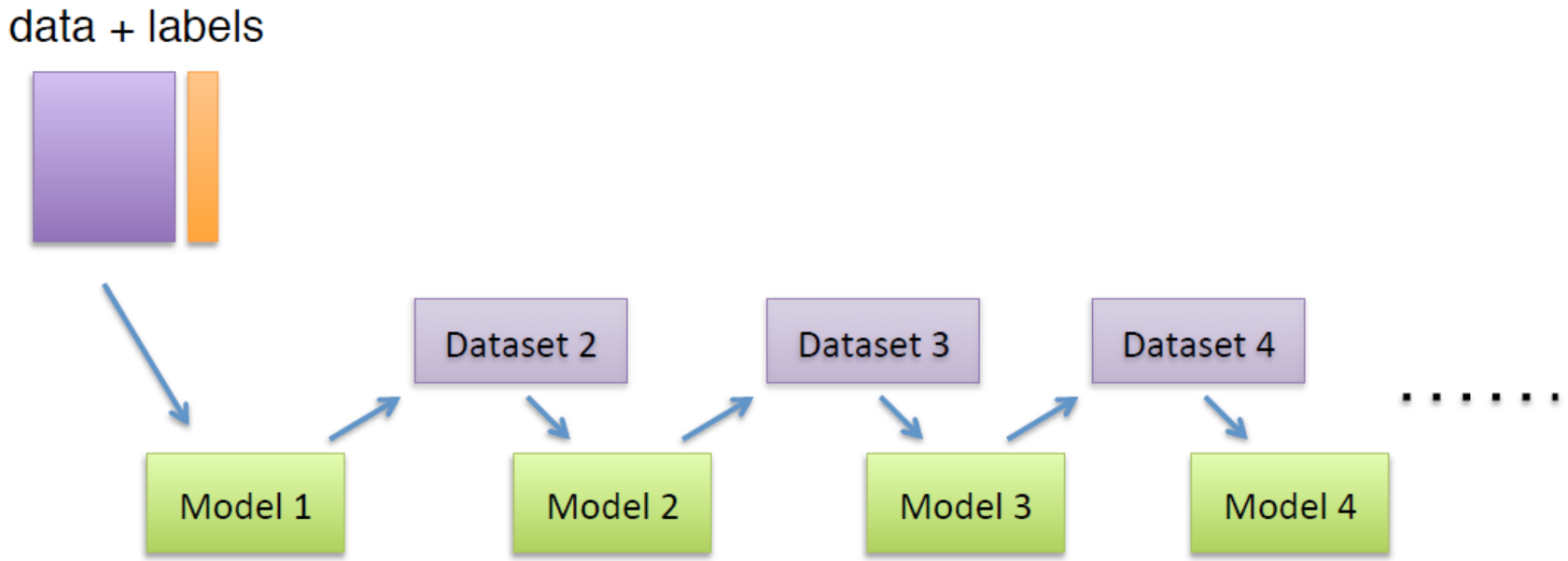


Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN



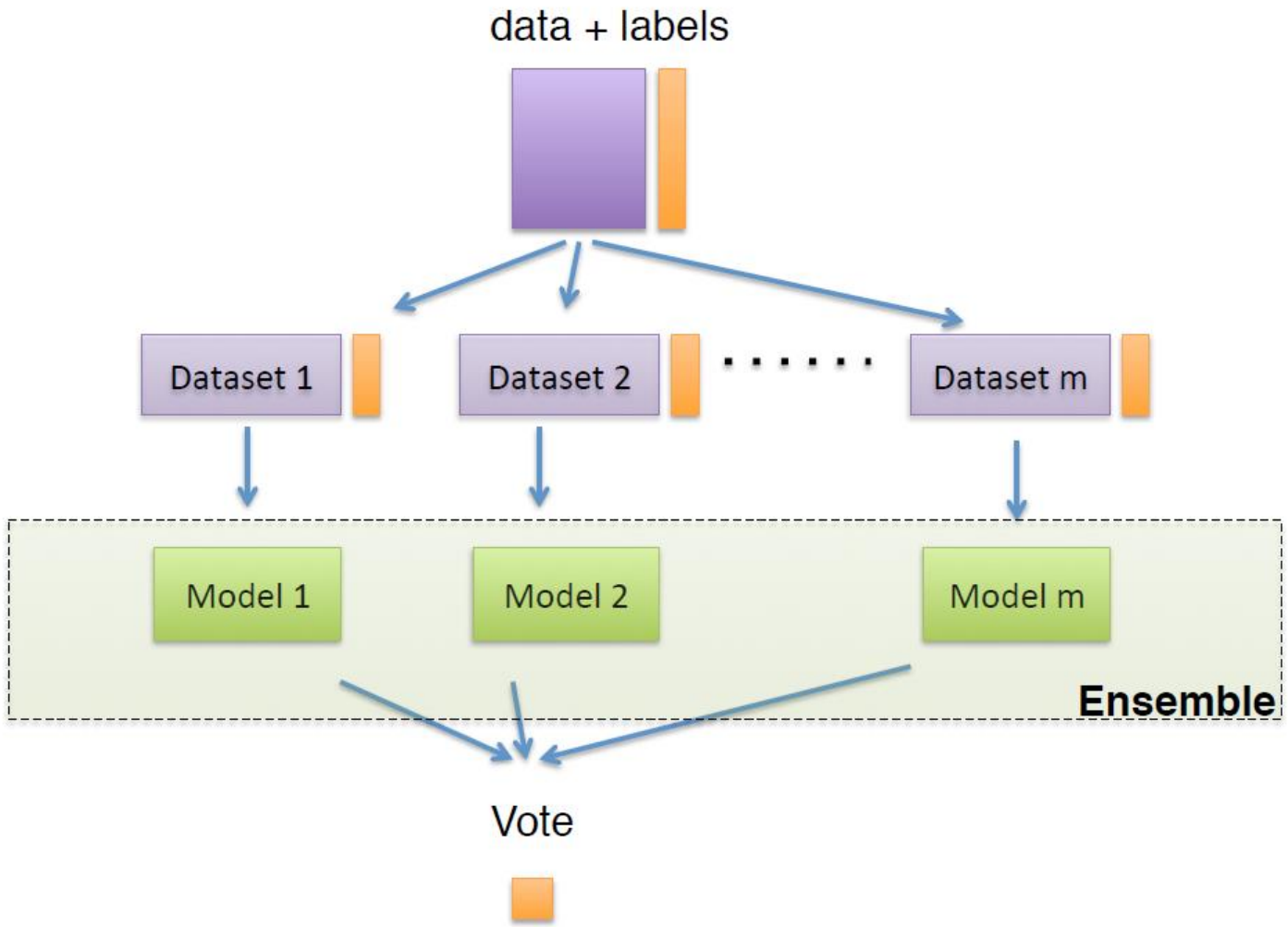
Ensembles of Classifiers



Each model corrects the mistakes of its predecessor.

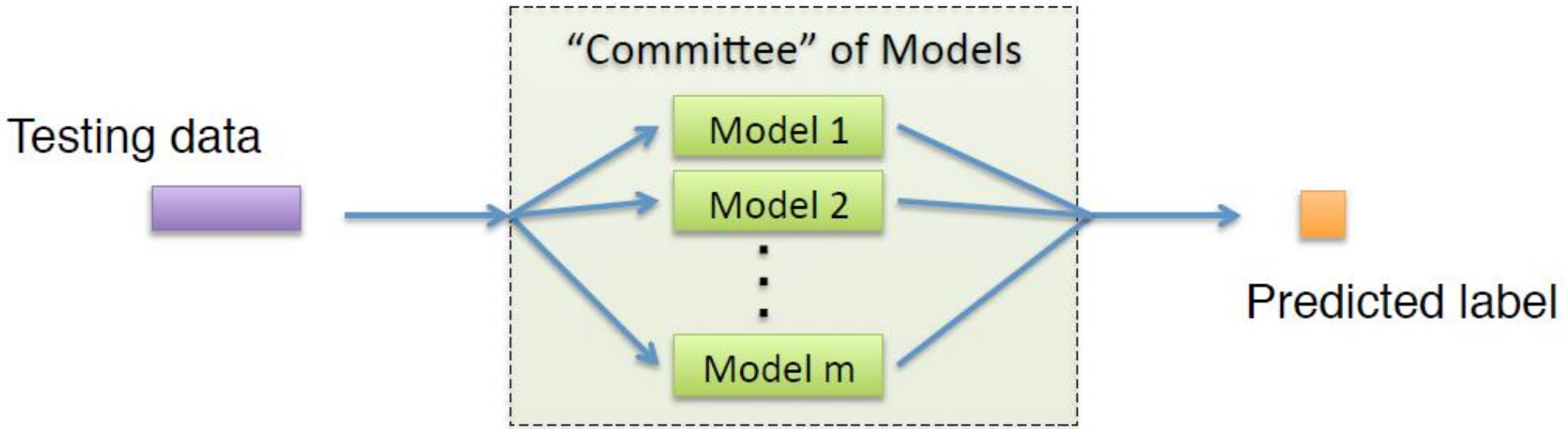
Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN



Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN



Decisions of individuals combined.

At testing phase is the same…

Ensembles of Classifiers

WITH SOME SLIDES FROM PROF. GAVIN BROWN

Which are the most popular methods to build "ensemble" classifiers?

BAGGING

BOOSTING

STACKING

BLENDING

christian.salvatore@iusspavia.it

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