

Classificatori

CORSO DI BIG DATA – MODULO ANALISI PER I BIG DATA
a.a. 2022/2023

Prof. Roberto Pirrone

Sommario

- Generalità
- Selezione delle feature
- Decision Tree
- Classificatori probabilistici
 - Regressione logistica
 - Naive Bayes
- Instance based learning
- Support Vector Machine
- Classificazione multclasse
- Rare class learning
- Regressione per dati numerici
- Metodi di ensemble
 - Random Forests
 - AdaBoost
 - Gradient boosting
 - XGBoost
- Valutazione della bontà della classificazione

Generalità

- Dato un insieme di dati di addestramento, ciascuno associato ad una «etichetta» che individua l'appartenenza del dato ad una classe, *la classificazione consiste nel predire il valore dell'etichetta per dati di test mai visti dall'algoritmo*
- È una classe di algoritmi di apprendimento supervisionati
 - Le etichette provengono da una associazione artificiale (dipendente dall'applicazione) ai dati e non dalla naturale tendenza di questi ultimi a formare cluster
 - È, forse, la tipologia di algoritmo di ML più comune

Generalità

- In generale, dato un insieme di n punti in \mathbb{R}^d appartenenti ad un dataset \mathcal{D} , questi vengono associati ad un insieme di etichette in $\{1, \dots, k\}$
 - Spesso la classificazione è binaria: $\{0, 1\}$ ovvero $\{-1, 1\}$
- Due modalità di funzionamento
 - Predizione esplicita dell'etichetta
 - Score numerico (probabilità) di appartenenza del punto ad una certa classe

Selezione delle feature

- Filtri: indicatori numerici della rilevanza delle feature
- Modelli «wrapped»: un algoritmo di classificazione viene usato per valutare la performance su un sotto-insieme di feature e quindi «avvolge» il vero e proprio algoritmo di classificazione in uno schema di ricerca delle feature rilevanti
- Modelli «embedded»: l'algoritmo stesso fornisce indicazioni sulle feature rilevanti e, dopo averle individuate, viene riaddestrato solo su di esse

Selezione delle feature

- Filtri

- Usati per dati categorici

$$G(v_i) = 1 - \sum_{j=1}^k p_j^2$$

Valore i-esimo dell'attributo categorico a r valori

Frazione dei punti che hanno il valore v_i nella classe j

- Gini index

$$G = \sum_{i=1}^r n_i G(v_i) / n$$

Frazione dei punti che hanno il valore v_i

$G(v_i) = 0$ se tutti i valori v_i appartengono alla stessa classe

Selezione delle feature

- Filtri
 - Usati per dati categorici

- Entropia

$$E(v_i) = - \sum_{j=1}^k p_j \log_2(p_j), \quad E = \sum_{i=1}^r n_i E(v_i) / n$$

$E(v_i) = \log_2(k)$ se i valori v_i sono uniformemente distribuiti tra le classi

Selezione delle feature

- Filtri

- Fisher score

$$F = \frac{\sum_{j=1}^k p_j (\mu_j - \mu)^2}{\sum_{j=1}^k p_j \sigma_j^2}$$

- Usato per attributi numerici

- Per ogni feature:

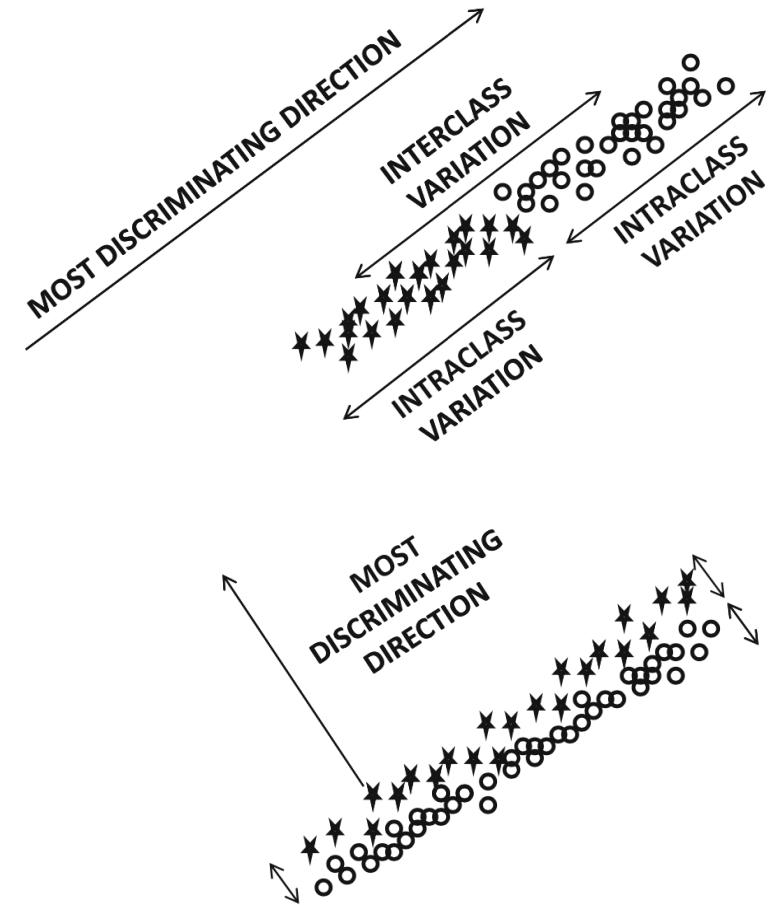
- p_j → frazione dei campioni appartenenti alla classe j
 - μ_j → media dell'attributo nella classe j
 - σ_j → dev. standard dell'attributo nella classe j

Separazione media «inter-classe»

Separazione media «intra-classe»

Selezione delle feature

- Filtri
 - Fisher Linear Discriminant
 - Generalizzazione del Fisher Score per combinazioni lineari di feature numeriche
 - Tende a trovare, in forma supervisionata, la direzione di massima variazione delle feature e, per contro, l'iperpiano perpendicolare che separa meglio le classi rispetto alle feature stesse in modo tale che il rapporto *Inter-class/Intra-class* risulti massimizzato



Selezione delle feature

- Filtri
 - Fisher Linear Discriminant

- Caso binario:

- W direzione di discriminante tra le due classi in \mathbb{R}^d
- Si massimizza $FS(W)$

$$FS(\bar{W}) = \frac{\text{Between Class Scatter along } \bar{W}}{\text{Within Class Scatter along } \bar{W}} \propto \frac{(\bar{W} \cdot \bar{\mu}_1 - \bar{W} \cdot \bar{\mu}_0)^2}{p_0[\text{Variance (Class 0)}] + p_1[\text{Variance (Class 1)}]}$$
$$= \frac{\bar{W} [(\bar{\mu}_1 - \bar{\mu}_0)^T (\bar{\mu}_1 - \bar{\mu}_0)] \bar{W}^T}{p_0 [\bar{W} \Sigma_0 \bar{W}^T] + p_1 [\bar{W} \Sigma_1 \bar{W}^T]} = \frac{[\bar{W} \cdot (\bar{\mu}_1 - \bar{\mu}_0)]^2}{\bar{W} (p_0 \Sigma_0 + p_1 \Sigma_1) \bar{W}^T}.$$

Proiezioni delle matrici di covarianza su W

$$\bar{W}^* \propto (\bar{\mu}_1 - \bar{\mu}_0) (p_0 \Sigma_0 + p_1 \Sigma_1)^{-1}$$

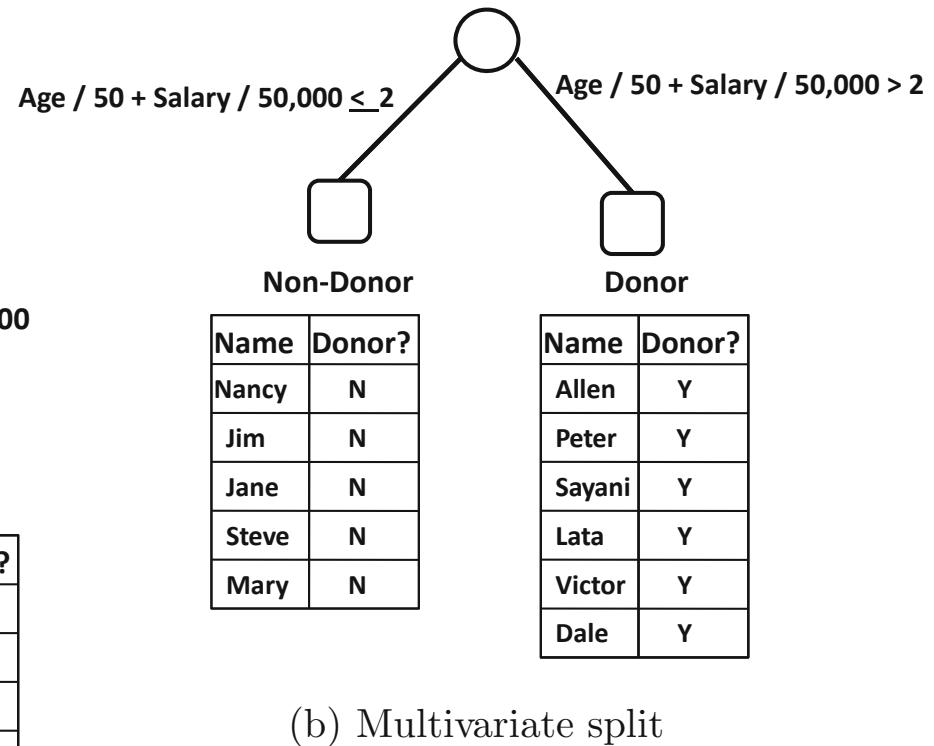
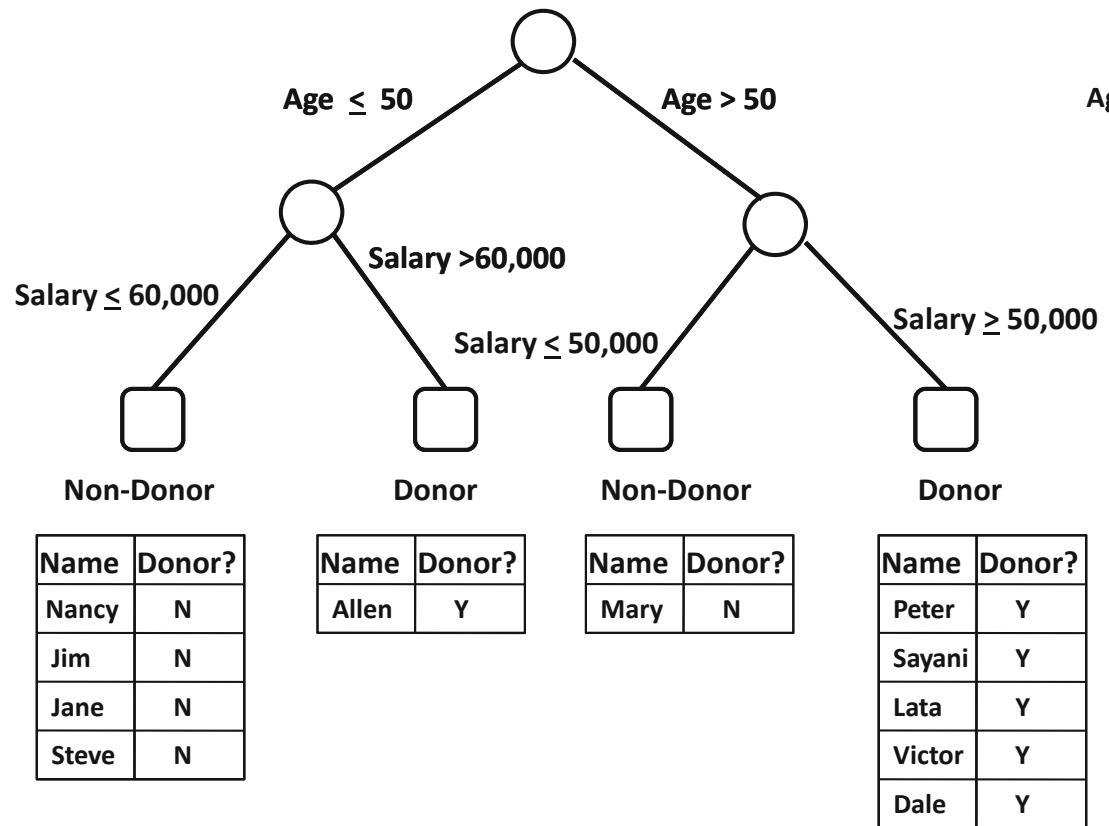
Selezione delle feature

- Modelli «wrapped»
 - Si parte da un insieme di feature $F = \{\}$
 - Si aggiungono feature a F e si testa l'accuratezza di un algoritmo di classificazione \mathcal{A} per accettare l'aggiunta delle nuova feature a F
 - L'incremento di F si può fare secondo diverse strategie
 - Random
 - Aggiunta della feature con maggior potere discriminativo rispetto ad un criterio di filtro

Decision Tree

- Il data set viene suddiviso ricorsivamente in parti più piccole sulla base del discriminare sui valori degli attributi
- I nodi foglia dell'albero vengono attribuiti alla classe dominante
 - Gerarchico similmente al clustering
 - Split univariato → decisione su un solo attributo
 - Split multivariato → decisione su un insieme di attributi

Decision tree



Decision tree

Algorithm *GenericDecisionTree*(Data Set: \mathcal{D})

begin

 Create root node containing \mathcal{D} ;

repeat

 Select an eligible node in the tree;

 Split the selected node into two or more nodes
 based on a pre-defined split criterion;

until no more eligible nodes for split;

 Prune overfitting nodes from tree;

 Label each leaf node with its dominant class;

end

- Split binario
- Split a r vie per attributi categorici o per attributi numerici con piccolo range di variazione
- Split binario per attributi categorici binarizzati
- Split a soglia per attributi numerici

Decision tree

Algorithm *GenericDecisionTree*(Data Set: \mathcal{D})
begin

 Create root node containing \mathcal{D} ;

repeat

 Select an eligible node in the tree;

 Split the selected node into two or more nodes
 based on a pre-defined split criterion;

until no more eligible nodes for split;

Prune overfitting nodes from tree;

Label each leaf node with its dominant class;

end

$$E(S) = - \sum_{j=1}^k p_j \log_2(p_j),$$

- Error rate: la frazione di elementi che *non* appartengono alla classe dominante
- Gini index
- Entropia

$$G(S) = 1 - \sum_{j=1}^k p_j^2,$$

$$\text{Gini-Split } (S \Rightarrow S_1 \dots S_r) = \sum_{i=1}^r \frac{|S_i|}{|S|} G(S_i)$$

Suddivisione del data set in r sottoinsiemi
per uno split a r vie → *si sceglie lo split con
Error rate, Gini index o Entropia più bassa*

$$\text{Entropy-Split } (S \Rightarrow S_1 \dots S_r) = \sum_{i=1}^r \frac{|S_i|}{|S|} E(S_i)$$

Decision Tree

- In genere i nodi vicini ai nodi foglia lavorano su pochi dati e sono proni al rumore nell'applicare lo split
 - Tendenza ad esibire overfitting
 - Si utilizzano tecniche di stop che prediligono alberi poco profondi
- Si utilizzano tecniche di pruning dei nodi foglia che vanno in overfit
 - Si valida l'eventuale incremento dell'accuracy su un validation set espunto dai dati di addestramento

Classificatori probabilistici

- L'appartenenza di un campione ad una classe è espressa in termini di una probabilità
- (Naive) Bayes Classifier
 - Si usa il Teorema di Bayes per stimare la probabilità di classe dall'evidenza delle feature *che si suppongono categoriche*
 - Il campione \mathbf{x} è espresso come d variabili statistiche: $\{x_1 = a_1, x_2 = a_2, \dots, x_d = a_d\}$
 - Il modello è una mistura di distribuzioni di natura generativa analogamente al clustering
 - L'ipotesi «naive» è quella dell'indipendenza statistica delle feature

Classificatori probabilistici

- L'appartenenza di un campione ad una classe è espressa in termini di una probabilità
- Regressione logistica
 - E' un modello discriminativo tra due classi
 - La probabilità della variabile di classe è una distribuzione di Bernoulli parametrizzata in termini delle feature
 - Si cercherà così l'iperpiano di separazione tra le due classi

Naive Bayes

$$P(C = c|x_1 = a_1, \dots, x_d = a_d) = \frac{P(C = c)P(x_1 = a_1, \dots, x_d = a_d|C = c)}{P(x_1 = a_1, \dots, x_d = a_d)}$$
$$\propto P(C = c)P(x_1 = a_1, \dots, x_d = a_d|C = c)$$

$$P(x_1 = a_1, \dots, x_d = a_d|C = c) = \prod_{j=1}^d P(x_j = a_j|C = c)$$

La probabilità di classe si ottiene dal Teorema di Bayes massimizzando la likelihood degli attributi condizionati alla classe

Distribuzione di Bernoulli o Multinomiale *su ogni singolo attributo*

Assunzione «naive» ovvero di indipendenza statistica degli attributi

Naive Bayes

$$P(C = c|x_1 = a_1, \dots, x_d = a_d) \propto P(C = c) \prod_{j=1}^d P(x_j = a_j|C = c)$$

Il Teorema di Bayes fornisce questa forma

$$P(x_j = a_j|C = c) = \frac{q(a_j, c)}{r(c)}$$

Stima maximum likelihood delle probabilità dei singoli attributi

$$P(x_j = a_j|C = c) = \frac{q(a_j, c) + \alpha}{r(c) + \alpha \cdot m_j}$$

Frazione dei campioni che hanno attributo a_j e classe c

Frazione dei campioni in classe c

Laplacian smoothing: m_j è il numero di valori distinti di a_j

La probabilità tende a $1/m_j$ se $r(c)=0$

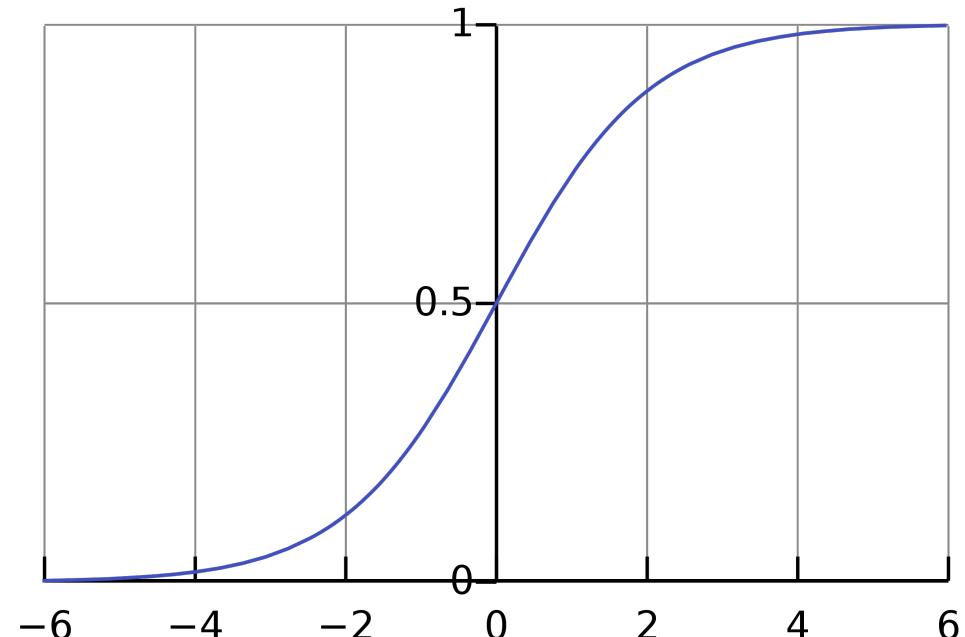
Per attributi numerici si può discretizzare ovvero pensare a $P(C = c|x_1 = a_1, \dots, x_n = a_n)$ come a una mistura di componenti in cui le componenti delle feature numeriche sono gaussiane

Regressione logistica

- La regressione logistica prende il nome dalla funzione di discriminazione della probabilità di appartenenza del campione ad una di due classi individuate come $\{+1, -1\}$.
- Il modello dipende da $d + 1$ parametri che individuano la separazione tra le classi lungo ogni dimensione del campione

$$P(C = +1 | \bar{X}) = \frac{1}{1 + e^{-(\theta_0 + \sum_{i=1}^d \theta_i x_i)}}$$

$$P(C = -1 | \bar{X}) = \frac{1}{1 + e^{(\theta_0 + \sum_{i=1}^d \theta_i x_i)}}$$



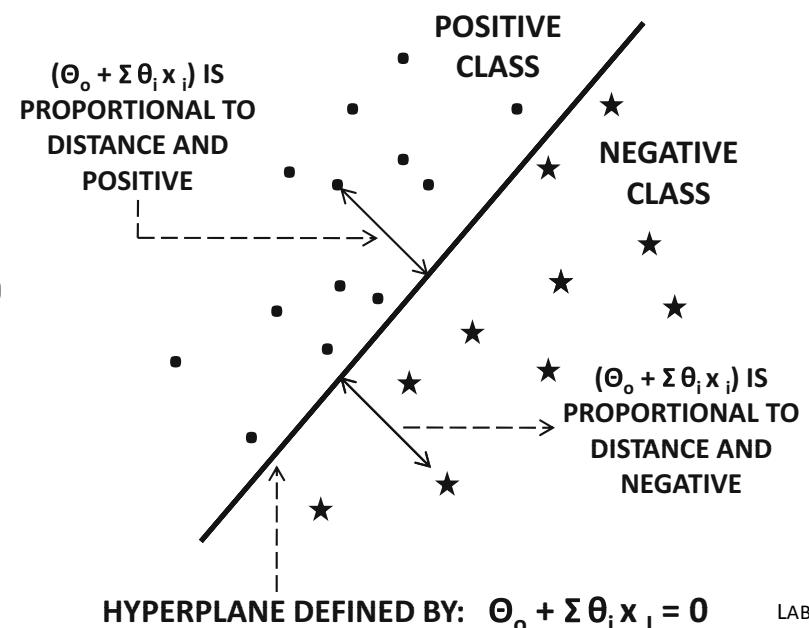
$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x}$$

Funzione logistica o sigmoide

Regressione logistica

- La regressione logistica è un «modello lineare» perché individua l'iperpiano di separazione tra le due classi in termini del luogo dei punti equiprobabili rispetto all'appartenenza ed una delle due classi

$$P(C = +1) \equiv P(C = -1) = 0.5 \rightarrow \theta_0 + \sum_{i=1}^d \theta_i x_i = 0$$



Regressione logistica

Addestramento tramite
massimizzazione della likelihood
del data set

$$\mathcal{L}(\bar{\Theta}) = \prod_{\overline{X}_k \in \mathcal{D}_+} \frac{1}{1 + e^{-(\theta_0 + \sum_{i=1}^d \theta_i x_k^i)}} \prod_{\overline{X}_k \in \mathcal{D}_-} \frac{1}{1 + e^{(\theta_0 + \sum_{i=1}^d \theta_i x_k^i)}}$$

Si passa alla log-likelihood che
si può mostrare essere
concava.

$$\mathcal{LL}(\bar{\Theta}) = \log(\mathcal{L}(\bar{\Theta})) = - \sum_{\overline{X}_k \in \mathcal{D}_+} \log \left(1 + e^{-(\theta_0 + \sum_{i=1}^d \theta_i x_k^i)} \right) - \sum_{\overline{X}_k \in \mathcal{D}_-} \log \left(1 + e^{(\theta_0 + \sum_{i=1}^d \theta_i x_k^i)} \right)$$

Regressione logistica

$$\begin{aligned}\frac{\partial \mathcal{LL}(\bar{\Theta})}{\partial \theta_i} &= \sum_{\bar{X}_k \in \mathcal{D}_+} \frac{x_k^i}{1 + e^{(\theta_0 + \sum_{i=1}^d \theta_i x_i)}} - \sum_{\bar{X}_k \in \mathcal{D}_-} \frac{x_k^i}{1 + e^{-(\theta_0 + \sum_{i=1}^d \theta_i x_i)}} \\ &= \sum_{\bar{X}_k \in \mathcal{D}_+} P(\bar{X}_k \in \mathcal{D}_-) x_k^i - \sum_{\bar{X}_k \in \mathcal{D}_-} P(\bar{X}_k \in \mathcal{D}_+) x_k^i \\ \theta_i &\leftarrow \theta_i + \alpha \left(\sum_{\bar{X}_k \in \mathcal{D}_+} P(\bar{X}_k \in \mathcal{D}_-) x_k^i - \sum_{\bar{X}_k \in \mathcal{D}_-} P(\bar{X}_k \in \mathcal{D}_+) x_k^i \right)\end{aligned}$$

Si massimizza tramite gradient ascent e, in genere, anche termine di regolarizzazione:

$$-\lambda \sum_{i=1}^d \theta_i^2 / 2$$

Instance based learning

- L'idea su cui si basano questi classificatori è
«Campioni simili hanno etichetta di classe simile»
- lazy learners:
 - Il modello addestrato viene costruito solo *alla fine* dell'osservazione di tutti i dati

Instance based learning

- Classificatore k-NN
 - Il classificatore osserva i k elementi più vicini al campione ed attribuisce la classe dominante tra questi
 - Si può pensare di pesare il contributo dei vicini rispetto alla loro distanza δ dal campione, per esempio $f(\delta) = e^{-\frac{\delta^2}{t^2}}$
 - k e t sono iperparametri e la loro scelta è cruciale

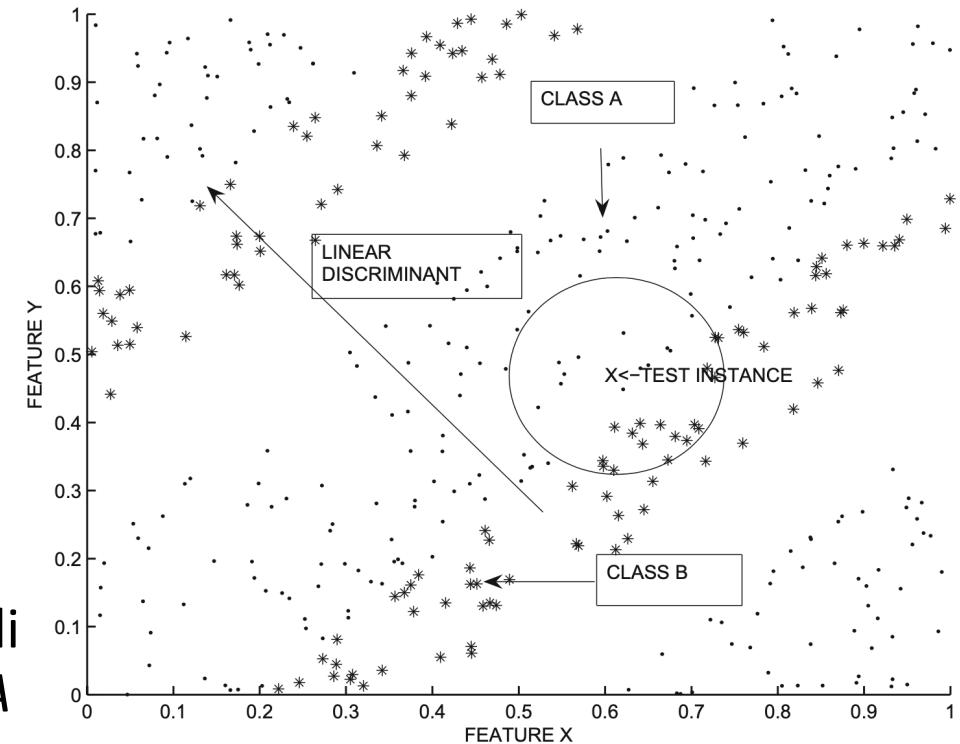
Instance based learning

- Classificatore Classificatore k-NN

- In generale si userà una distanza nella forma:

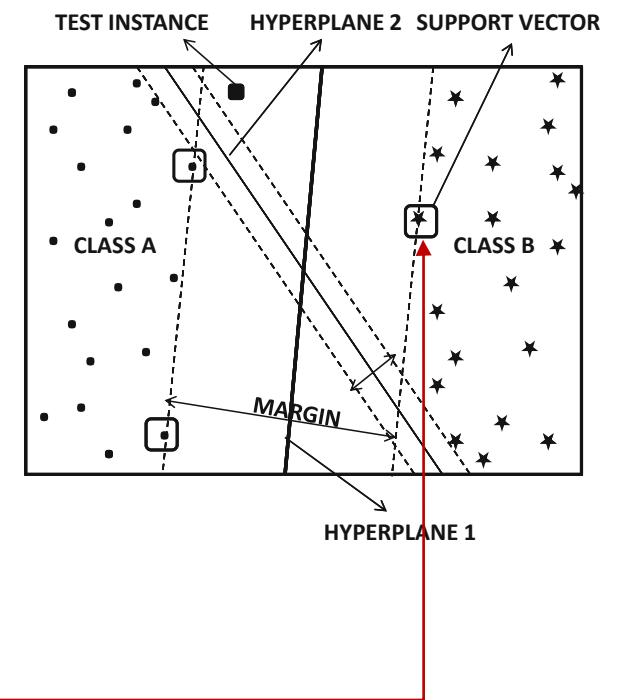
$$\text{Dist}(\bar{X}, \bar{Y}) = \sqrt{(\bar{X} - \bar{Y})A(\bar{X} - \bar{Y})^T}$$

- $A = \Sigma^{-1}$ implica la distanza di Mahalanobis
 - Si possono proiettare le distanze lungo le direzioni di massima variazione tra le diverse classi e costruire A come rapporto delle matrici di *inter class scatter* e *intra class scatter* determinate con la Fisher Linear Discriminant Analysis



Support Vector Machines

- Classificatore *lineare* per problemi di *classificazione binaria*
- Il modello ricerca l'*iperpiano a massimo margine* che separa le due classi
 - *Massimo margine*: la massima distanza tra due iperpiani paralleli che contengono alcune istanze di entrambe le classi dette **support vector**



Support Vector Machines

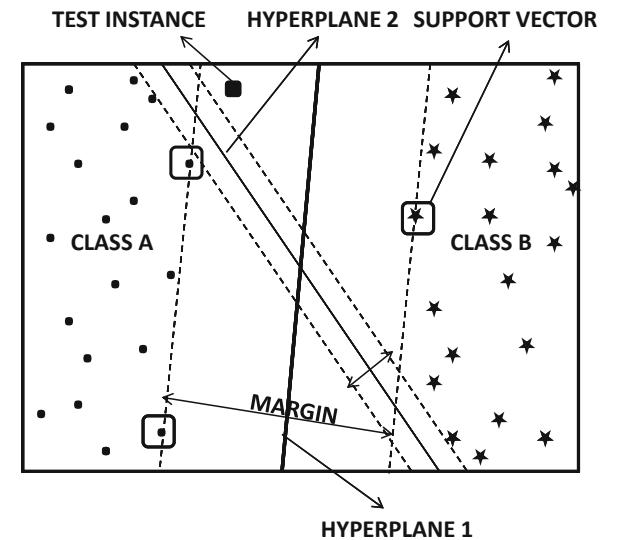
Le equazioni dei due iperpiani che limitano il margine si possono scrivere in termini dei coefficienti W e dell'intercetta b :

$$\overline{W} \cdot \overline{X_i} + b \geq 0 \quad \forall i : y_i = +1$$

$$\overline{W} \cdot \overline{X_i} + b \leq 0 \quad \forall i : y_i = -1$$

Con un apposito scaling di \overline{W} e b :

$$\begin{cases} \overline{W} \cdot \overline{X_i} + b \geq +1 \forall i : y_i = +1 \\ \overline{W} \cdot \overline{X_i} + b \leq -1 \forall i : y_i = -1 \end{cases} \Rightarrow y_i(\overline{W} \cdot \overline{X_i} + b) \geq +1, \forall i$$



Support Vector Machines

Ricordiamo che la distanza tra due piani paralleli $\pi_1 = ax + by + cz + h$ e $\pi_2 = ax + by + cz + k$, con coefficienti direttori siano dati da $\mathbf{n} = \{a, b, c\}$, sarà $d(\pi_1, \pi_2) \equiv d(P, \pi_2)$, $P \in \pi_1$ e che quest'ultima si ottiene come: $d(P, \pi_2) = |ax + by + cz + k| / ||\mathbf{n}||$.

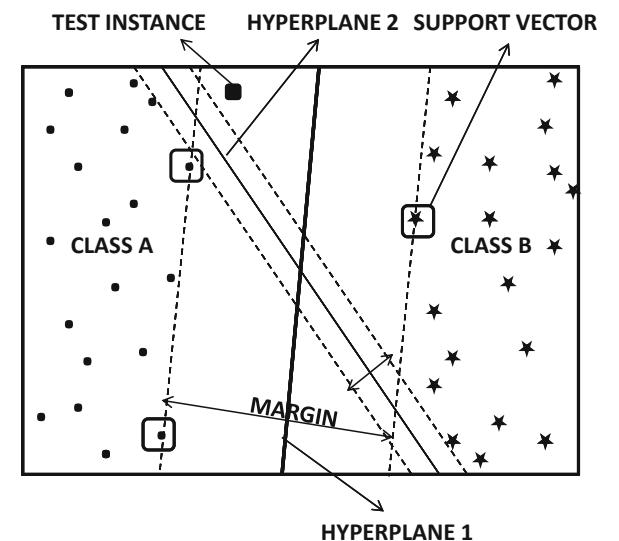
In relazione al margine si deduce che vogliamo massimizzare la

distanza iperpiani : $2/\|\bar{W}\|$

funzione obiettivo : $O = \|\bar{W}\|^2/2$

$$L_P = \frac{\|\bar{W}\|^2}{2} - \sum_{i=1}^n \lambda_i [y_i (\bar{W} \cdot \bar{X}_i + b) - 1], \quad \lambda_i \geq 0$$

support vectors : $\lambda_i [y_i (\bar{W} \cdot \bar{X}_i + b) - 1] = 0$



Moltiplicatori di
Lagrange

Support Vector Machines

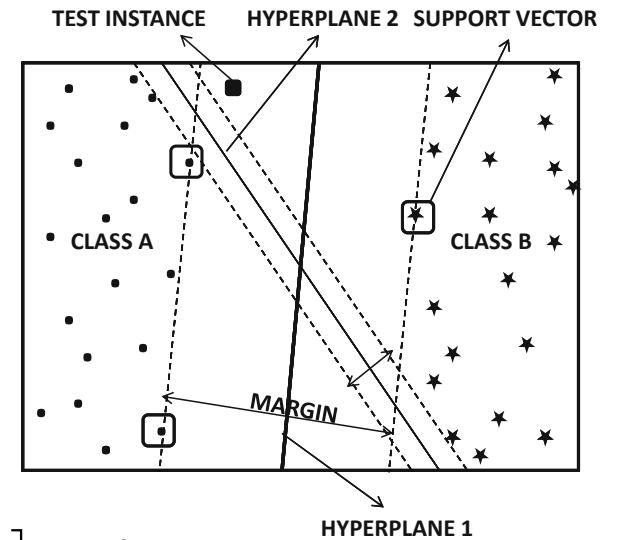
$$L_P = \frac{\|\bar{W}\|^2}{2} - \sum_{i=1}^n \lambda_i [y_i(\bar{W} \cdot \bar{X}_i + b) - 1], \quad \lambda_i \geq 0$$

La minimizzazione di L_P può essere formulata come problema

$$\text{duale di massimizzazione : } O^* = \frac{\|\bar{W}^*\|^2}{2} \geq L_D^* = \max_{\lambda \geq 0} \min_{\bar{W}, b} L_P$$

e si mostra che la condizione sui support vectors: $\lambda_i [y_i(\bar{W} \cdot \bar{X}_i + b) - 1] = 0$

corrisponde alla classica condizione di Kuhn-Tucker per l'ottimizzazione vincolata



Support Vector Machines

Formulando il problema come pura massimizzazione, eliminiamo la dipendenza da \bar{W} e b ponendo a zero il gradiente di L_P

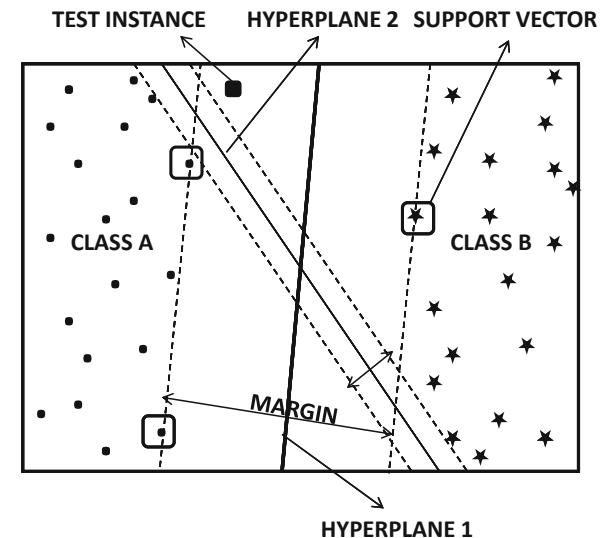
$$\nabla L_P = \nabla \frac{\|\bar{W}\|^2}{2} - \nabla \sum_{i=1}^n \lambda_i [y_i(\bar{W} \cdot \bar{X}_i + b) - 1] = 0$$

$$\bar{W} = \sum_{i=1}^n \lambda_i y_i \bar{X}_i, \quad \sum_{i=1}^n \lambda_i y_i = 0$$

$$\partial L_P / \partial b = 0$$

Il problema duale di massimizzazione si ottiene sostituendo l'espressione di \bar{W} in L_P :

$$L_D = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j \bar{X}_i \cdot \bar{X}_j$$



L'addestramento lungo il gradiente ci fornisce la seguente formulazione:

$$\frac{\partial L_D}{\partial \lambda_i} = 1 - y_i \sum_{j=1}^n y_j \lambda_j \bar{X}_i \cdot \bar{X}_j$$

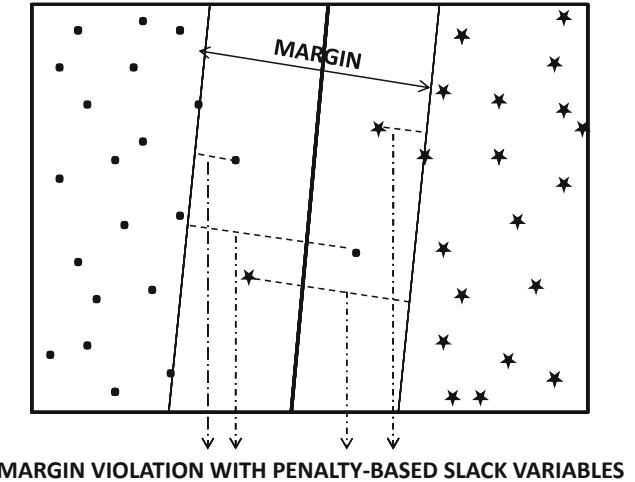
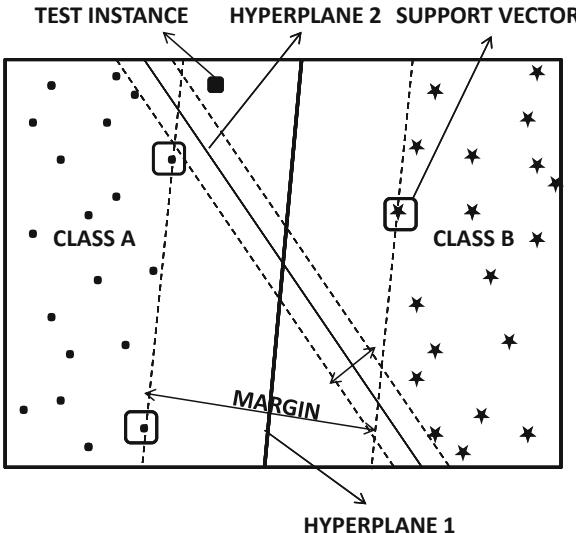
$$(\lambda_1 \dots \lambda_n) \leftarrow (\lambda_1 \dots \lambda_n) + \alpha \left(\frac{\partial L_D}{\partial \lambda_1} \dots \frac{\partial L_D}{\partial \lambda_n} \right)$$

Support Vector Machines

I dati non sono separabili: ammetto delle *violazioni con penalità* sulla funzione obiettivo

$$\begin{cases} \overline{W} \cdot \overline{X_i} + b \geq +1 - \xi_i \quad \forall i : y_i = +1 \\ \overline{W} \cdot \overline{X_i} + b \leq -1 + \xi_i \quad \forall i : y_i = -1 \end{cases}, \quad \xi_i \geq 0 \quad \forall i$$

Slack variables



Si mostra che il problema duale di questa SVM massimizza L_D sotto certe condizioni rispetto alle slack variables:

$$\begin{aligned} \partial L_P / \partial \xi_i &= 0 \rightarrow \xi_i (C - \lambda_i - \beta_i) = 0 \rightarrow (C - \lambda_i - \beta_i) = 0, \quad (C - \lambda_i) = \beta_i \geq 0 \\ &\rightarrow 0 < \lambda_i < C \text{ per i support vectors } (\xi_i = 0) \end{aligned}$$

funzione obiettivo :

$$O = \frac{\|\overline{W}\|^2}{2} + C \sum_{i=1}^n \xi_i$$

Lagrange penalty :

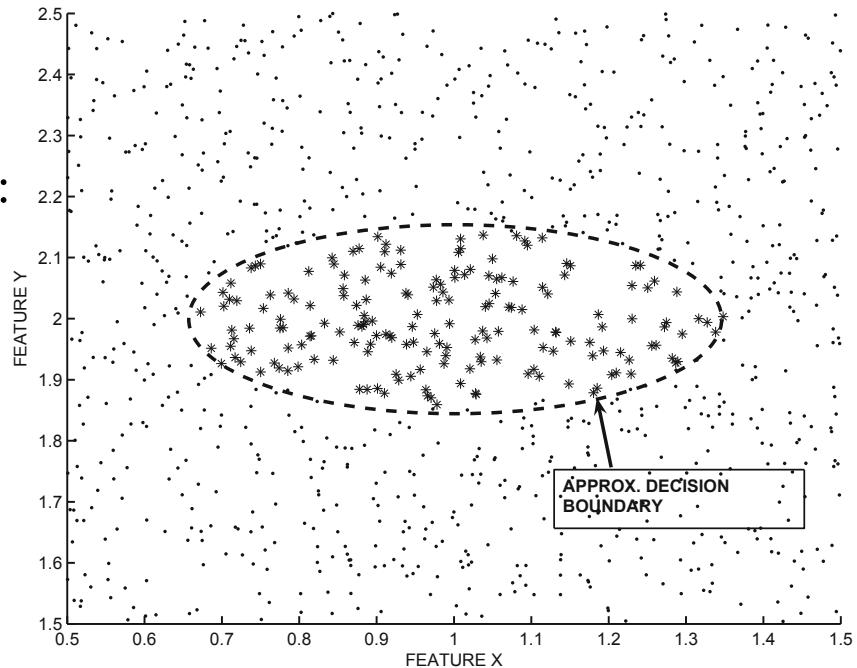
$$L_P = \frac{\|\overline{W}\|^2}{2} + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \lambda_i [y_i(\overline{W} \cdot \overline{X_i} + b) - 1 + \xi_i] - \sum_{i=1}^n \beta_i \xi_i$$

Support Vector Machines

kernel trick per superfici di separazione non lineari :

$$L_D = \sum_{i=1}^n \lambda_i - \frac{1}{2} \cdot \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j K(\overline{X}_i, \overline{X}_j)$$

Function	Form
Gaussian radial basis kernel	$K(\overline{X}_i, \overline{X}_j) = e^{- \overline{X}_i - \overline{X}_j ^2 / 2\sigma^2}$
Polynomial kernel	$K(\overline{X}_i, \overline{X}_j) = (\overline{X}_i \cdot \overline{X}_j + c)^h$
Sigmoid kernel	$K(\overline{X}_i, \overline{X}_j) = \tanh(\kappa \overline{X}_i \cdot \overline{X}_j - \delta)$



Classificazione multiclasse

- Molti algoritmi di classificazione possono lavorare nativamente per più classi
 - Decision tree
 - Classificatori (Naive) Bayes
- La regressione logistica e la SVM sono concepite come classificatori binari
- Si possono definire strategie per estendere il loro comportamento alla gestione di problemi multiclasse

Classificazione multiclasse

- Strategia *one-against-rest*:
 - Si addestrano k classificatori per cui l' i -esimo modello binario predice come classe positiva l'appartenenza alla classe C_i ,
 - Nel test, se il modello i -esimo predice correttamente la classe C_i questa ottiene un voto
 - Altrimenti ottengono un voto tutte le altre classi
 - Il classificatore fornirà come etichetta di classe quella che ha ottenuto la maggioranza dei voti

Classificazione multiclasse

- Strategia *one-against-one*:
 - Si addestrano $\binom{k}{2}$ classificatori per tutte le coppie possibili di classi
 - Nel test, il generico modello C_i -*against*- C_j attribuisce un voto alla classe correttamente predetta
 - Il classificatore fornirà come etichetta di classe quella che ha ottenuto la maggioranza dei voti
 - Molti più classificatori, ma training set più piccoli per cui risulta più conveniente dal punto di vista computazionale

Rare class learning

- La classificazione di un data set molto *sbilanciato* ovvero con dei campioni appartenenti ad una *classe rara* è interessante per diversi aspetti:
 - Nel dominio del problema, la classe rara è quella *interessante* ed un errore di classificazione nella classe rara è *molto dannoso*
 - I classificatori necessitano di tecniche di pesatura e/o campionamento dei dati ad hoc per gestire lo sbilanciamento

Rare class learning

- Pesatura dei campioni
 - Ogni campione ha un peso legato al costo della sua errata classificazione
→ privilegia le classi rare
 - Decision tree: il peso si incorpora nel criterio di split come Gini index o Entropia
 - Naive Bayes: si pesa la stima dei prior e delle probabilità delle feature

Rare class learning

- Pesatura dei campioni
 - Ogni campione ha un peso legato al costo della sua errata classificazione
→ privilegia le classi rare
 - Support Vector Machines (con slack variables): le penalità di violazione del margine sono pesate per le diverse classi
 - k-NN: i voti rispetto alle singole classi dei k vicini vengono pesati

Rare class learning

- Campionamento
 - Si *sovracampiona* la classe rara ovvero si *campionano* le classi abbondanti
 - Campionamento con rimpiazzo o senza rimpiazzo
 - Il sovracampionamento può essere dannoso perché avviene con rimpiazzo e quindi genera più copie dello stesso campione
→ porta ad overfitting

Rare class learning

- Campionamento
 - Sovracampionamento sintetico – SMOTE:
 - Si selezionano i k vicini di un campione raro, appartenenti alla classe minoritaria
 - Si sceglie una frazione casuale di questi, a seconda del livello richiesto di sovraccampionamento
 - Si genera un campione sintetico in un punto casuale lungo la linea congiungente il campione con ognuno dei vicini selezionati

Regressione

- Regressione lineare

- Si consideri un data set D di n vettori in \mathbb{R}^d $D = \begin{bmatrix} \bar{X}_1 \\ \vdots \\ \bar{X}_n \end{bmatrix}$, $(n \times d)$ e siano definite n corrispondenze $X_i \rightarrow y_i$, $i = 1, \dots, n$

- Determinare un vettore di coefficienti $W = \{w_1, \dots, w_d\}$ che consenta di approssimare l'insieme delle corrispondenze $X_i \rightarrow y_i$, $i = 1, \dots, n$, come:

$$y_i \approx \bar{W} \cdot \bar{X}_i \quad \forall i \in \{1 \dots n\}$$

Regressione

- Regressione lineare
 - Si minimizza una funzione obiettivo definita come la somma degli errori quadratici (Sum of Squared Errors – SSE):

$$O = \sum_{i=1}^n (\bar{W} \cdot \bar{X}_i - y_i)^2 = \|D\bar{W}^T - \bar{y}\|^2$$

- Si parlerà di Least Squares Regression

Regressione

- Regressione lineare

- Ponendo a 0 il gradiente di O rispetto a W , si ottiene:

$$2D^T(D\bar{W}^T - \bar{y}) \Rightarrow D^T D \bar{W}^T = D^T \bar{y}, \quad \bar{W}^T = (D^T D)^{-1} D^T \bar{y}$$

- Questa è la matrice pseudoinversa di Moore-Penrose D^+ della matrice D :

$$\bar{W}^T = D^+ \bar{y}$$

- Quando $D^T D$ non è invertibile (per esempio $n \ll d$) si usa la definizione esplicita della pseudoinversa:

$$D^+ = \lim_{\delta \rightarrow 0} (D^T D + \delta^2 I)^{-1} D^T$$

Regressione

- Regressione lineare
 - Si può mostrare che il Fisher linear discriminant è un caso particolare di least squares regression sotto alcune condizioni:
 - D è a media nulla
 - $y_i = -1/p_0$ per la classe negativa e $1/p_1$ per la classe positiva
 - Il risultato della regressione $\overline{W}^T = D^+ \overline{y}$ corrisponde a $\overline{W}^* \propto (\overline{\mu_1} - \overline{\mu_0}) (p_0 \Sigma_0 + p_1 \Sigma_1)^{-1}$

Regressione

- Regressione lineare:

- Ridge regression: si utilizza una regolarizzazione L_2

$$O = \|D\bar{W}^T - \bar{y}\|^2 + \lambda \|\bar{W}\|^2$$

$$\bar{W}^T = (D^T D + \lambda I)^{-1} D^T \bar{y}$$

- La matrice è semidefinita positiva e quindi sempre invertibile
 - la ridge regression riduce la varianza della stima rispetto alla least squares

Regressione

- Regressione lineare

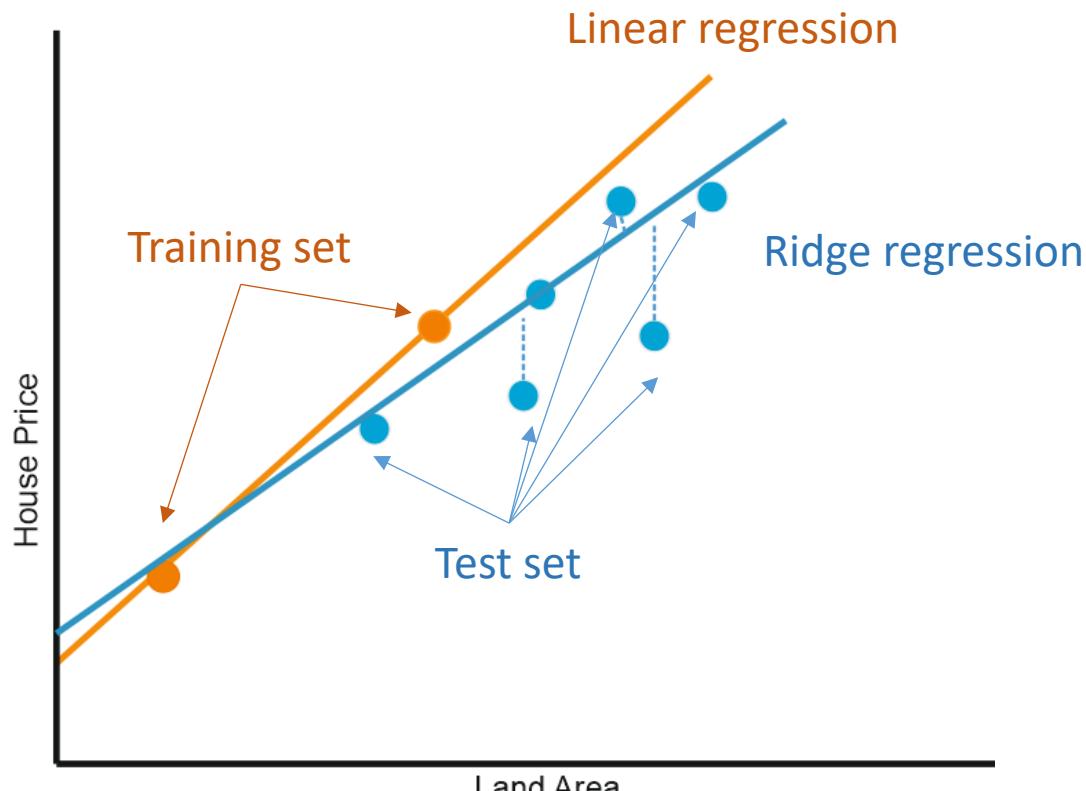
- Lasso regression: si utilizza una regolarizzazione L_1

$$O = \|D\bar{W}^T - \bar{y}\|^2 + \lambda \|\bar{W}\|$$

- La soluzione è iterativa e consente e ammette, tendenzialmente, il caso $W = 0$
 - La regressione Lasso, in genere, predilige un vettore dei coefficienti sparso
 - Indicata nel caso di presenza di molte feature irrilevanti per il problema, le quali vedranno azzerarsi i rispettivi coefficienti

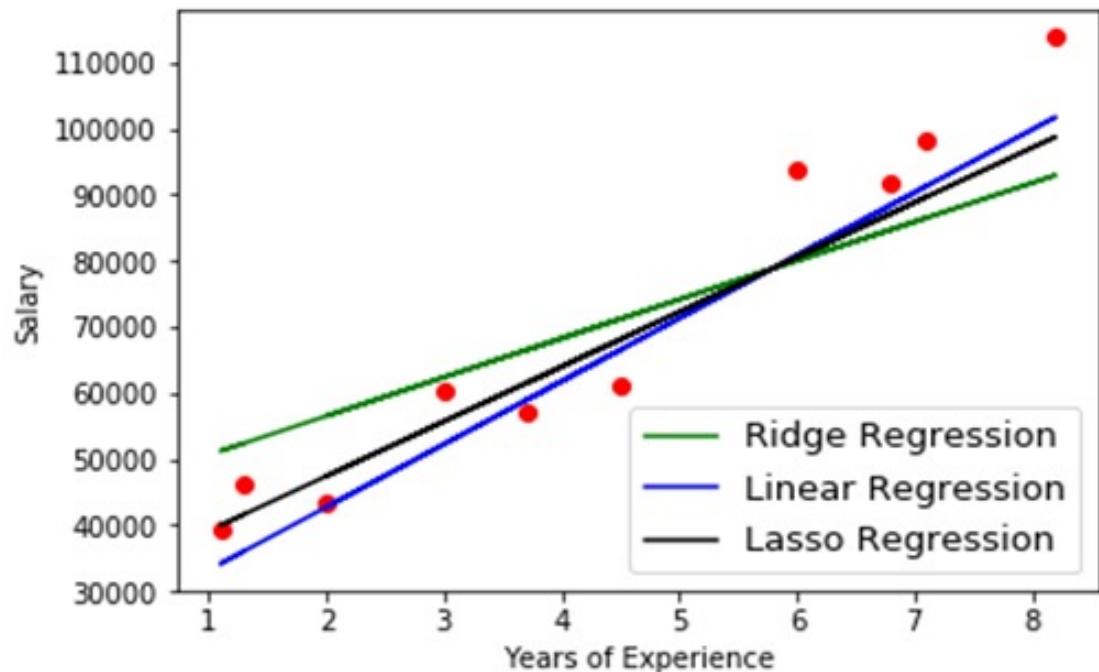
Regressione

- Effetto della regolarizzazione

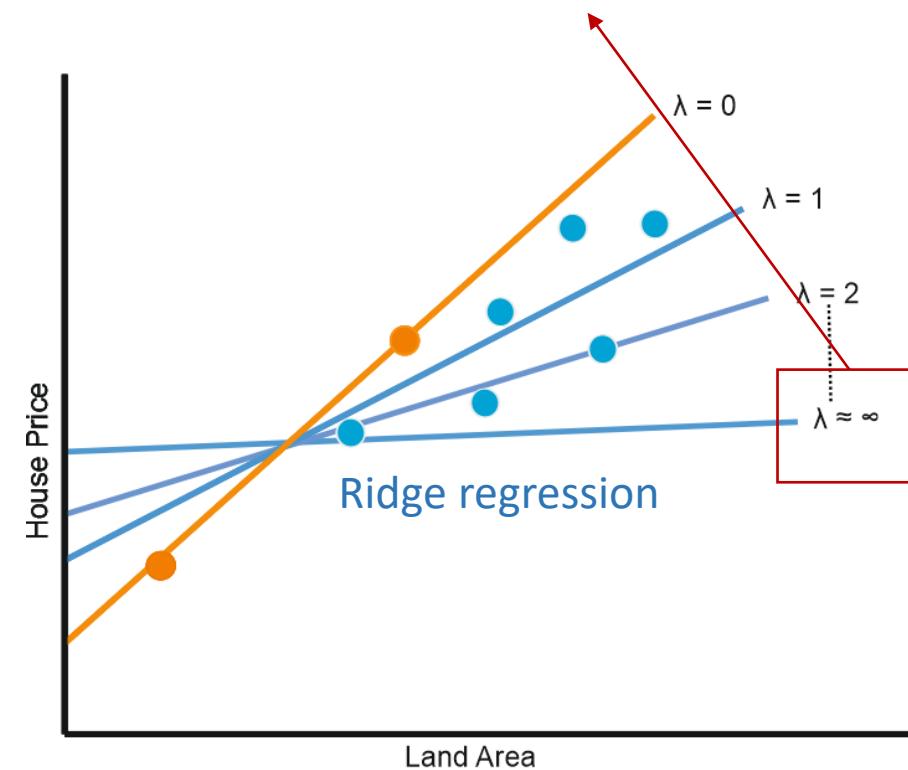


Regressione

- Ridge vs Lasso



La Ridge regression azzerà il contributo di una feature solo asintoticamente ($\lambda \rightarrow \infty$) mentre la Lasso regression può eliminare esattamente la feature (pendenza nulla per λ molto grande, ma non ∞)



Regressione

- Principal Component Regression
 - Utilizzato per ridurre la dimensionalità del data set rispetto alle feature poco interessanti o rumorose
 - Si applica la PCA a D ottenendo una nuova matrice dei dati $R = \begin{bmatrix} \bar{Z}_1 \\ \vdots \\ \bar{Z}_n \end{bmatrix}, (n \times k)$ proiettando D sulle k direzioni principali
 - Si risolve, quindi il problema

$$y_i \approx \bar{W} \cdot \bar{Z}_i \quad \forall i \in \{1 \dots n\}, \quad \bar{W} = (R^T R)^{-1} R^T \bar{y}$$

Regressione

- Generalized Linear Models
 - Modello generalizzato della regressione lineare in cui si stima una *funzione di media* $f(\bar{W} \cdot \bar{X}_i)$, $i = 1, \dots, n$ della distribuzione di probabilità delle etichette numeriche y_i
 - Mantiene le caratteristiche della regressione lineare, ma si estende a diverse tipologie di dati
 - Il problema da risolvere è quindi:

$$y_i \sim \text{Probability distribution with mean } f(\bar{W} \cdot \bar{X}_i) \quad \forall i \in \{1 \dots n\}$$

Regressione

- Generalized Linear Models

- La scelta della funzione di media (o della sua inversa $f^{-1}(\cdot)$ detta *link function*) sottende una particolare distribuzione di probabilità e quindi una determinata tipologia di dato

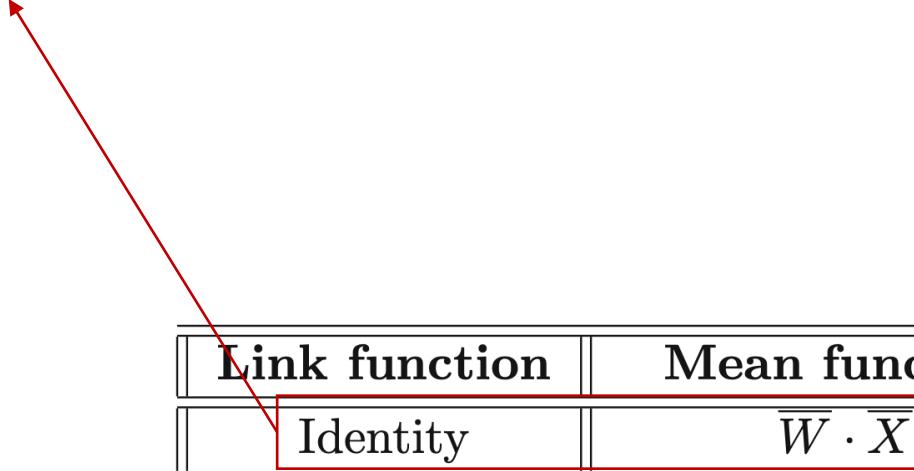
Link function	Mean function	Distribution assumption
Identity	$\bar{W} \cdot \bar{X}$	Normal
Inverse	$-1/(\bar{W} \cdot \bar{X})$	Exponential, Gamma
Log	$\exp(\bar{W} \cdot \bar{X})$	Poisson
Logit	$1/[1 + \exp(-\bar{W} \cdot \bar{X})]$	Bernoulli, Categorical
Probit	$\Phi(\bar{W} \cdot \bar{X})$	Bernoulli, Categorical

Regressione

- Generalized Linear Models

Least squares
regression:

$$\begin{aligned}\text{Likelihood}(\{y_1 \dots y_n\}) &= \prod_{i=1}^n \text{Probability}(y_i) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - f(\bar{W} \cdot \bar{X}_i))^2}{2\sigma^2}\right) \\ &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \bar{W} \cdot \bar{X}_i)^2}{2\sigma^2}\right) \\ &\propto \exp\left(-\frac{\sum_{i=1}^n (y_i - \bar{W} \cdot \bar{X}_i)^2}{2\sigma^2}\right)\end{aligned}$$



Link function	Mean function	Distribution assumption
Identity	$\bar{W} \cdot \bar{X}$	Normal
Inverse	$-1/(\bar{W} \cdot \bar{X})$	Exponential, Gamma
Log	$\exp(\bar{W} \cdot \bar{X})$	Poisson
Logit	$1/[1 + \exp(-\bar{W} \cdot \bar{X})]$	Bernoulli, Categorical
Probit	$\Phi(\bar{W} \cdot \bar{X})$	Bernoulli, Categorical

Regressione

- Generalized Linear Models

Logistic regression:

$$y_i = \begin{cases} 1 & \text{with probability } 1 / [1 + \exp(-\bar{W} \cdot \bar{X}_i)] \\ 0 & \text{with probability } 1 / [1 + \exp(\bar{W} \cdot \bar{X}_i)] \end{cases}$$

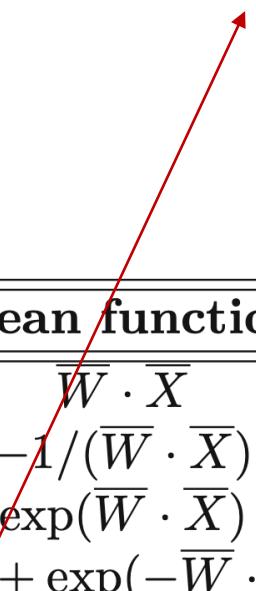
Link function	Mean function	Distribution assumption
Identity	$\bar{W} \cdot \bar{X}$	Normal
Inverse	$-1 / (\bar{W} \cdot \bar{X})$	Exponential, Gamma
Log	$\exp(\bar{W} \cdot \bar{X})$	Poisson
Logit	$1 / [1 + \exp(-\bar{W} \cdot \bar{X})]$	Bernoulli, Categorical
Probit	$\Phi(\bar{W} \cdot \bar{X})$	Bernoulli, Categorical

Regressione

- Generalized Linear Models

Variante della regressione logistica che usa la CDF della distribuzione normale

Utile per fare regressione su dati ordinati come i valori di rating



Link function	Mean function	Distribution assumption
Identity	$\bar{W} \cdot \bar{X}$	Normal
Inverse	$-1/(\bar{W} \cdot \bar{X})$	Exponential, Gamma
Log	$\exp(\bar{W} \cdot \bar{X})$	Poisson
Logit	$1/[1 + \exp(-\bar{W} \cdot \bar{X})]$	Bernoulli, Categorical
Probit	$\Phi(\bar{W} \cdot \bar{X})$	Bernoulli, Categorical

Regressione non lineare

- Estensione della regressione ad una funzione non lineare del campione:

$$y = \sum_{i=1}^m w_i h_i(\bar{X})$$

- Il caso tipico è quello della regressione polinomiale in cui $h_i(\bar{X}) = \bar{X}^i$
- Cattura relazioni funzionali complesse tra i punti del data set

Regressione non lineare

- Kernel ridge regression
 - Si usa il kernel trick per modellare la relazione tra i dati e le etichette numeriche
 - È necessario mostrare che la ridge regression può essere espressa in termini del *prodotto scalare tra i dati di addestramento \bar{X}_i e quelli di test \bar{Z}_i*
 - Si può mostrare, usando la SVD della matrice D , che:

$$(D^T D + \lambda I_d)^{-1} D^T = D^T (D D^T + \lambda I_n)^{-1}$$

Regressione non lineare

- Kernel ridge regression

- La predizione della ridge regression $F(\bar{Z}) = \bar{Z}\bar{W}^T$ si può scrivere:

$$F(\bar{Z}) = \bar{Z} (D^T D + \lambda I_d)^{-1} D^T \bar{y} = \bar{Z} D^T (DD^T + \lambda I_n)^{-1} \bar{y}$$

- La kernel ridge regression si può quindi formulare come:

$$F(\bar{Z}) = \bar{\kappa} (K + \lambda I_n)^{-1} \bar{y}$$

- In cui $\bar{\kappa} = [k(\bar{Z}, \bar{X}_1), \dots, k(\bar{Z}, \bar{X}_n)]$, $K = [k(\bar{X}_i, \bar{X}_j)]$

Regressione

- Alberi di regressione
 - Modella le relazioni non lineari tra i dati come un insieme di *approssimazioni lineari locali* nei nodi foglia dell'albero
 - Lo split per variabili categoriche è quello tipico di un decision tree (Gini index, entropia) e così pure le tecniche di pruning, mentre per dati numerici si minimizza il valore di SSE su tutti i nodi risultanti dallo split
 - Il modello richiederebbe una regressione ad ogni nodo. In alternativa si può controllare la varianza dei dati ad ogni split e poi fare la regressione nei nodi foglia *dopo* aver costruito l'albero

Regressione

- Valutazione della regressione
 - Si usa la statistica R^2 o la sua variante per d elevato:

$$R^2 = 1 - \frac{SSE}{SST}$$

$$R^2 = 1 - \frac{(n - d)}{(n - 1)} \frac{SSE}{SST}$$

$$SSE = \sum_{i=1}^n \left(y_i - g(\bar{X}_i) \right)^2 \quad SST = \sum_{i=1}^n \left(y_i - \sum_{j=1}^n \frac{y_j}{n} \right)^2$$

Metodi di ensemble

- I metodi di ensemble combinano le predizioni di un insieme di algoritmi (base learners) sul data set
- Sfruttano le caratteristiche peculiari di ciascun algoritmo rispetto ai dati per incrementare l'accuratezza

```
Algorithm EnsembleClassify(Training Data Set:  $\mathcal{D}$ 
                           Base Algorithms:  $\mathcal{A}_1 \dots \mathcal{A}_r$ , Test Instances:  $\mathcal{T}$ )
begin
     $j = 1$ ;
repeat
    Select an algorithm  $\mathcal{Q}_j$  from  $\mathcal{A}_1 \dots \mathcal{A}_r$ ;
    Create a new training data set  $f_j(\mathcal{D})$  from  $\mathcal{D}$ ;
    Apply  $\mathcal{Q}_j$  to  $f_j(\mathcal{D})$  to learn model  $\mathcal{M}_j$ ;
     $j = j + 1$ ;
until(termination);
report labels of each  $T \in \mathcal{T}$  based on combination of
predictions from all learned models  $\mathcal{M}_j$ ;
end
```

Metodi di ensemble

- I metodi di ensemble orientati ai dati
 - $\mathcal{A}_1, \dots, \mathcal{A}_r$ sono istanze dello stesso modello
 - $f_j(\mathcal{D})$ è differente per ciascun modello
 - Campionamento
 - Manipolazione delle feature o delle etichette
 - Focalizzazione su porzioni di dati in cui la classificazione non è corretta

Algorithm *EnsembleClassify*(Training Data Set: \mathcal{D}
Base Algorithms: $\mathcal{A}_1 \dots \mathcal{A}_r$, Test Instances: \mathcal{T})
begin
 $j = 1$;
 repeat
 Select an algorithm \mathcal{Q}_j from $\mathcal{A}_1 \dots \mathcal{A}_r$;
 Create a new training data set $f_j(\mathcal{D})$ from \mathcal{D} ;
 Apply \mathcal{Q}_j to $f_j(\mathcal{D})$ to learn model \mathcal{M}_j ;
 $j = j + 1$;
 until(termination);
 report labels of each $T \in \mathcal{T}$ based on combination of
 predictions from all learned models \mathcal{M}_j ;
end

Metodi di ensemble

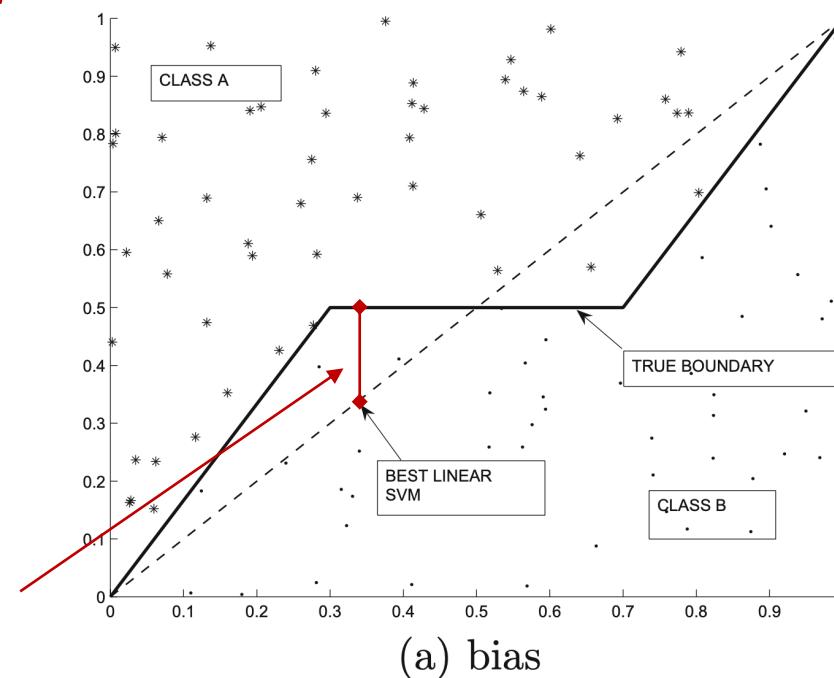
- I metodi di ensemble orientati ai modelli
 - $\mathcal{A}_1, \dots, \mathcal{A}_r$ sono istanze di modelli differenti
 - $f_j(\mathcal{D}) \equiv \mathcal{D}$
 - Differenti modelli esibiscono performance migliori in diverse porzioni del data set

Algorithm *EnsembleClassify*(Training Data Set: \mathcal{D}
Base Algorithms: $\mathcal{A}_1 \dots \mathcal{A}_r$, Test Instances: \mathcal{T})
begin
 $j = 1$;
 repeat
 Select an algorithm \mathcal{Q}_j from $\mathcal{A}_1 \dots \mathcal{A}_r$;
 Create a new training data set $f_j(\mathcal{D})$ from \mathcal{D} ;
 Apply \mathcal{Q}_j to $f_j(\mathcal{D})$ to learn model \mathcal{M}_j ;
 $j = j + 1$;
 until(termination);
 report labels of each $T \in \mathcal{T}$ based on combination of
 predictions from all learned models \mathcal{M}_j ;
end

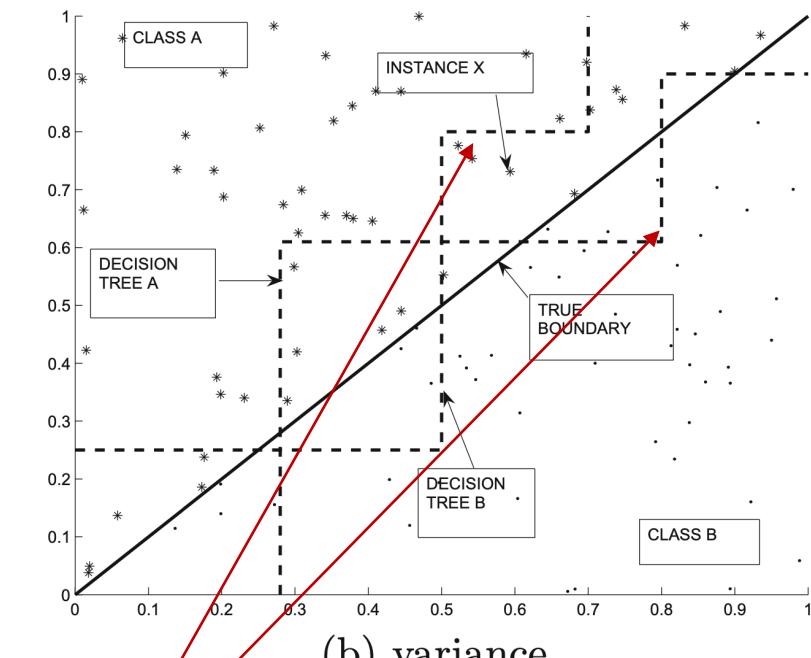
Metodi di ensemble

- I metodi di ensemble migliorano *sia il bias sia la varianza della stima* del classificatore

Un singolo classificatore esibisce un proprio *bias* rispetto alla reale distribuzione delle classi *anche per un numero elevato di dati di addestramento*



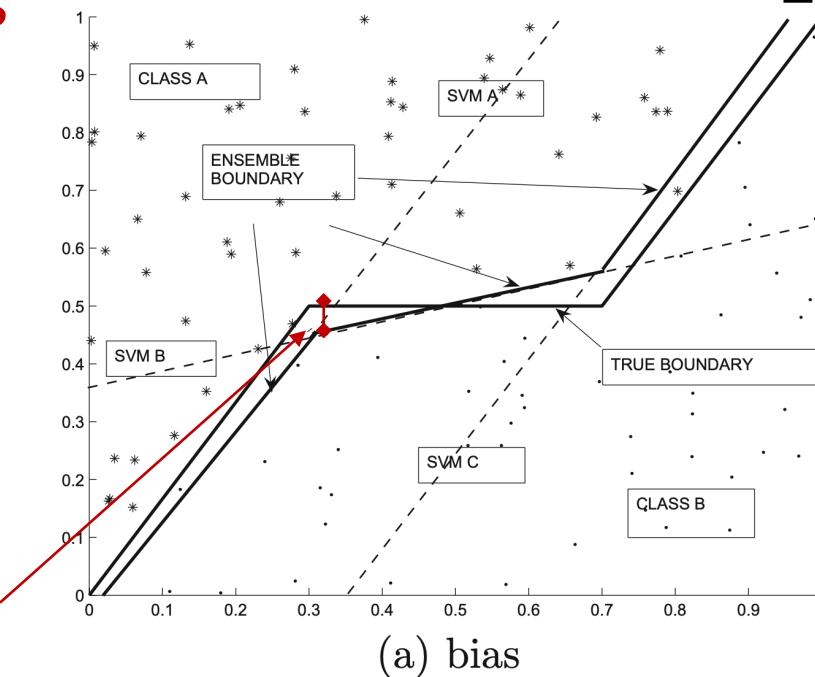
Un singolo classificatore ha *varianza* insita nella stima perché al variare dell'istanza del training set la predizione del medesimo campione può cambiare



Metodi di ensemble

- I metodi di ensemble migliorano *sia il bias sia la varianza della stima* del classificatore

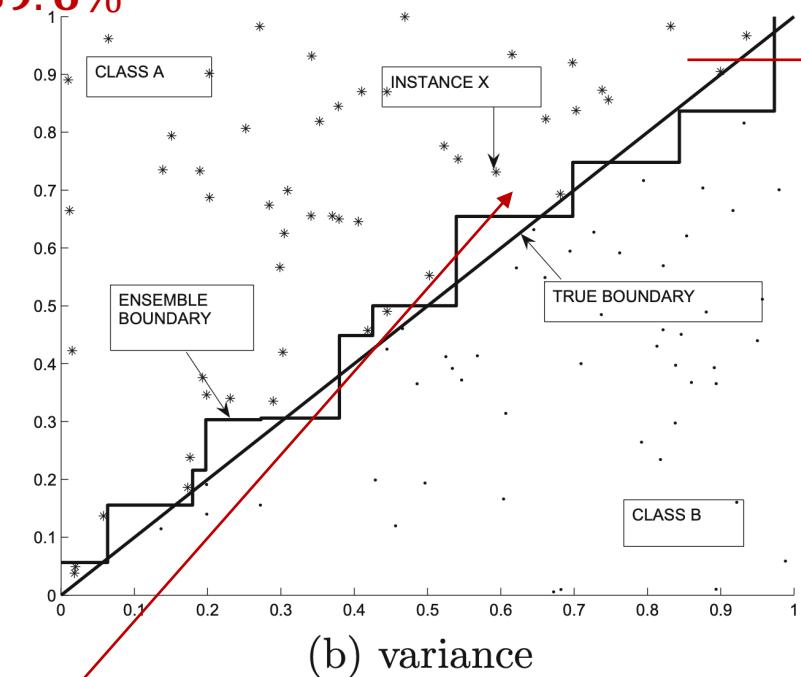
Un ensemble di classificatori approssima meglio un boundary complesso riducendo il bias



Tre decision tree:

$$p_{tree1} = p_{tree2} = p_{tree3} = 80\%$$

$$p_{ensemble} = p_{tree1} \times p_{tree2} \times p_{tree3} + \sum_{c \in C_{1,2,3}} p_{tree,c1} \times p_{tree,c2} \times (1 - p_{tree,c3}) = 89.6\%$$



All'aumentare del numero dei componenti dell'ensemble la stima riduce la propria varianza

Metodi di ensemble

- Bagging (Bootstrapped aggregating)
 - Tecnica per creare un ensemble di classificatori i.i.d. in modo tale da ridurre la varianza della stima da σ^2 a σ^2/k
 - Si generano k data set tramite campionamento con rimpiazzo: i nuovi data set contengono duplicati
 - Si può mostrare che la frazione di dati distinti in ogni data set è $1 - (1 - 1/n)^n \approx 1 - 1/e$ per n elevato
 - Questa tecnica è detta Bootstrap: usata anche per calcolare l'Accuracy di un modello pesata su più istanze del data set «lievemente» diverse

Metodi di ensemble

- Bagging (Bootstrapped aggregating)
 - Si addestrano i k classificatori e la predizione è data dalla maggioranza o dalla media dei voti espressi da ciascuno su ogni campione
 - Riduce la varianza della stima, *ma non il bias*
 - Conviene scegliere dei modelli che tendano di per sé a ridurre il bias

Random Forests

- È un ensemble di decision tree addestrati con il bagging
- Il decision tree, soprattutto se fatto accrescere molto in profondità, riduce molto il bias anche se a forte scapito della varianza della classificazione
- È un modello ideale per il bagging

Random Forests

- I nodi più elevati della gerarchia rimangono sostanzialmente invarianti nei diversi alberi
 - I singoli alberi sono molto correlati e ciò limita la riduzione dell'errore di stima
 - È necessario diversificare molto gli alberi durante la loro crescita per aumentare la capacità di predizione dell'ensemble
- Le Random Forests usano un *criterio random per lo split nei diversi alberi* dell'ensemble

Random Forests

- Random split selection
 - Forest-RI (Random Input): Un sottoinsieme casuale di q feature viene estratto per lo split
 - Forest-RC (Random Combination): un sottoinsieme L di feature viene estratto e si creano q combinazioni lineari che vengono usate come feature multivariate per lo split
 - Preferibile bassa dimensionalità dei dato per cui q diventa molto piccolo

Boosting

- Tecnica di ensemble iterativa orientata a ridurre il bias
- Lo stesso learner viene riaddestrato più volte sul data set in cui ogni campione è pesato *con un peso maggiore se è stato mis-classificato all'iterata precedente*
 - Il classificatore all'iterata t viene guidato dagli errori commessi dalle iterate precedenti

Boosting

- AdaBoost

Predizione:

somma pesata delle predizioni ad ogni iterata t tramite il coefficiente α_t

Si può ridurre la varianza se si ricampiona il data set ad ogni iterazione

Poco indicato per dati rumorosi

Algorithm AdaBoost(Data Set: \mathcal{D} , Base Classifier: \mathcal{A} , Maximum Rounds: T)
begin
 $t = 0$;
 for each i initialize $W_1(i) = 1/n$;
 repeat
 $t = t + 1$;
 Determine weighted error rate ϵ_t on \mathcal{D} when base algorithm \mathcal{A} is applied to weighted data set with weights $W_t(\cdot)$;
 $\alpha_t = \frac{1}{2} \log_e((1 - \epsilon_t)/\epsilon_t)$;
 for each misclassified $\bar{X}_i \in \mathcal{D}$ **do** $W_{t+1}(i) = W_t(i) e^{\alpha_t}$;
 else (correctly classified instance) **do** $W_{t+1}(i) = W_t(i) e^{-\alpha_t}$;
 for each instance \bar{X}_i **do** normalize $W_{t+1}(i) = W_{t+1}(i) / [\sum_{j=1}^n W_{t+1}(j)]$;
 until $((t \geq T) \text{ OR } (\epsilon_t = 0) \text{ OR } (\epsilon_t \geq 0.5))$;
 Use ensemble components with weights α_t for test instance classification;
end

Classificazione perfetta

Performance peggiore del classificatore casuale

Frazione dei campioni mis-classificati

Peso crescente

Peso decrescente

Boosting

- XGBoost (eXtreme Gradient Boosting)
 - Algoritmo molto recente di ensemble per compiti di classificazione e regressione, basato sul concetto di *gradient boosting*
 - L'idea di base è che:
«il boosting può essere interpretato come la minimizzazione di un'apposita funzione di costo nello spazio funzionale dei modelli»

Boosting

- XGBoost (eXtreme Gradient Boosting)

- Gradient boosting:

$$\hat{y} = F(x)$$

- Si consideri il problema di classificazione:

$$L_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^n (y_i - F(x_i))^2$$

- Lo schema di boosting aggiorna il modello come segue:

$$F_{m+1}(x_i) = F_m(x_i) + h_m(x_i) = y_i$$

$$h_m(x_i) = y_i - F_m(x_i)$$

Boosting

- XGBoost (eXtreme Gradient Boosting)

- Gradient boosting:

- L'algoritmo, ad ogni iterata, fa il *fit dei residui* $h_m(x_i)$
 - È facile mostrare che *i residui sono proporzionali alle derivate della funzione di Loss rispetto alla funzione del modello*

Sono i modelli di base dell'ensemble



$$L_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^n (y_i - F_m(x_i))^2$$
$$-\frac{\partial L_{\text{MSE}}}{\partial F_m(x_i)} = \frac{2}{n} (y_i - F_m(x_i)) = \frac{2}{n} h_m(x_i)$$

Boosting

- XGBoost (eXtreme Gradient Boosting)

Input: training set $\{(x_i, y_i)\}_{i=1}^n$, a differentiable loss function $L(y, F(x))$, number of iterations M .

- Gradient boosting:

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma).$$

2. For $m = 1$ to M :

1. Compute so-called *pseudo-residuals*:

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \dots, n.$$

Addestramento del
 m -esimo learner sui residui

2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set

$$\{(x_i, r_{im})\}_{i=1}^n.$$

3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

È la somma di tutti base learner che
minimizza la loss media dell'ensemble

γ_m è la costante di proporzionalità del
 m -esimo learner.

Viene scelta per minimizzare L .

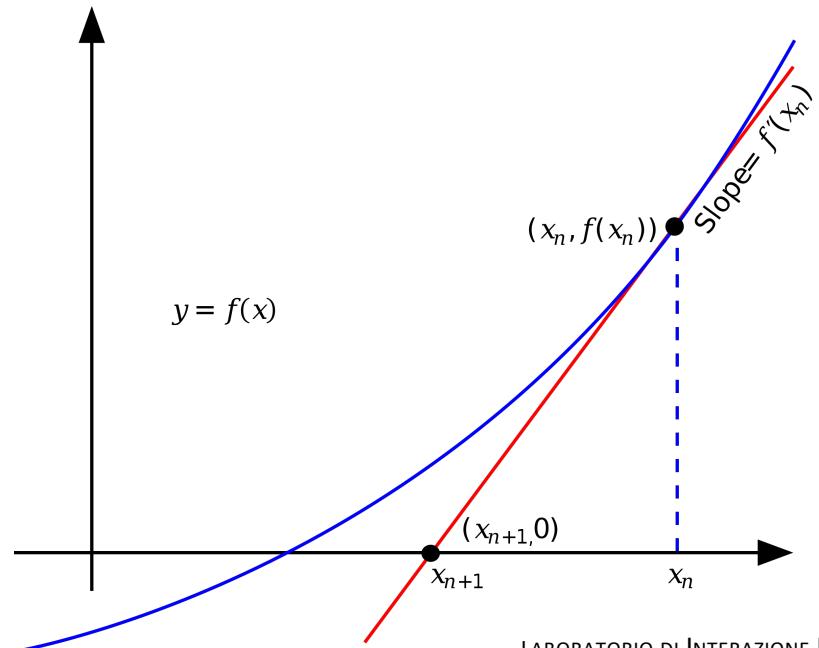
La discesa lungo il gradiente di L viene
approssimata da una ricerca lineare

Boosting

- XGBoost (eXtreme Gradient Boosting)
 - La discesa lungo il gradiente è effettuata usando il metodo di Newton-Raphson

$$f'(x_n) = \frac{f(x_n) - 0}{x_n - x_{n+1}}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



Boosting

- XGBoost (eXtreme Gradient Boosting)

- Nel Newton boosting $f(x)$ è approssimata tramite espansione in serie di Taylor fino al secondo ordine:

$$f(x_k + t) \approx f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2$$

$$0 = \frac{d}{dt} \left(f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2 \right) = f'(x_k) + f''(x_k)t$$

$$t = -\frac{f'(x_k)}{f''(x_k)}, \quad x_{k+1} = x_k + t = x_k - \boxed{-\frac{f'(x_k)}{f''(x_k)}}$$

Questi sono i residui del boosting

Boosting

- XGBoost

Input: training set $\{(x_i, y_i)\}_{i=1}^N$, a differentiable loss function $L(y, F(x))$, a number of weak learners M and a learning rate α .

Algorithm:

1. Initialize model with a constant value:

$$\hat{f}_{(0)}(x) = \arg \min_{\theta} \sum_{i=1}^N L(y_i, \theta).$$

2. For $m = 1$ to M :

1. Compute the 'gradients' and 'hessians':

$$\begin{aligned}\hat{g}_m(x_i) &= \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x)=\hat{f}_{(m-1)}(x)}. \\ \hat{h}_m(x_i) &= \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2} \right]_{f(x)=\hat{f}_{(m-1)}(x)}.\end{aligned}$$

2. Fit a base learner (or weak learner, e.g. tree) using the training set $\left\{x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right\}_{i=1}^N$ by solving the optimization problem below:

$$\begin{aligned}\hat{\phi}_m &= \arg \min_{\phi \in \Phi} \sum_{i=1}^N \frac{1}{2} \hat{h}_m(x_i) \left[-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i) \right]^2. \\ \hat{f}_m(x) &= \alpha \hat{\phi}_m(x).\end{aligned}$$

3. Update the model:

$$\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x).$$

3. Output $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^M \hat{f}_m(x)$.

Il singolo learner minimizza un problema ai minimi quadrati rispetto ai residui

Valutazione della bontà della classificazione

La più semplice misura è l'accuracy ovvero la percentuale di elementi predetti correttamente in tutte le classi.

Fornisce una falsa informazione se le classi sono fortemente sbilanciate.

$$ACC = \frac{1}{N} \sum_{k=1}^C \sum_{x \in C_k} I(C(x) \equiv C_k), \quad I = \begin{cases} 0, & C(x) \neq C_k \\ 1, & C(x) = C_k \end{cases}$$

$$bACC = \frac{1}{C} \sum_{k=1}^C \frac{1}{|C_k|} \sum_{x \in C_k} I(C(x) \equiv C_k)$$

Valutazione della bontà della classificazione

Predizione → Reference ↓	C_i	Altre classi
C_i	TP_i	FN_i
Altre classi	FP_i	TN_i

$$Acc_i = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}$$

$$P_{\text{micro}} = \frac{\sum_{i=1}^C TP_i}{\sum_{i=1}^C TP_i + FP_i}$$

$$R_{\text{micro}} = \frac{\sum_{i=1}^C TP_i}{\sum_{i=1}^C TP_i + FN_i}$$

$$F_{1\text{micro}} = 2 \frac{P_{\text{micro}} \cdot R_{\text{micro}}}{P_{\text{micro}} + R_{\text{micro}}}$$

$$P_i = \frac{TP_i}{TP_i + FP_i}$$

$$R_i = \frac{TP_i}{TP_i + FN_i}$$

$$F_1 = 2 \frac{P_i \cdot R_i}{P_i + R_i}$$

$$P_{\text{macro}} = \frac{1}{C} \sum_{i=1}^C \frac{TP_i}{TP_i + FP_i} = \frac{\sum_{i=1}^C P_i}{C}$$

$$R_{\text{macro}} = \frac{1}{C} \sum_{i=1}^C \frac{TP_i}{TP_i + FN_i} = \frac{\sum_{i=1}^C R_i}{C}$$

$$F_{1\text{macro}} = 2 \frac{P_{\text{macro}} \cdot R_{\text{macro}}}{P_{\text{macro}} + R_{\text{macro}}}$$

Valutazione della bontà della classificazione

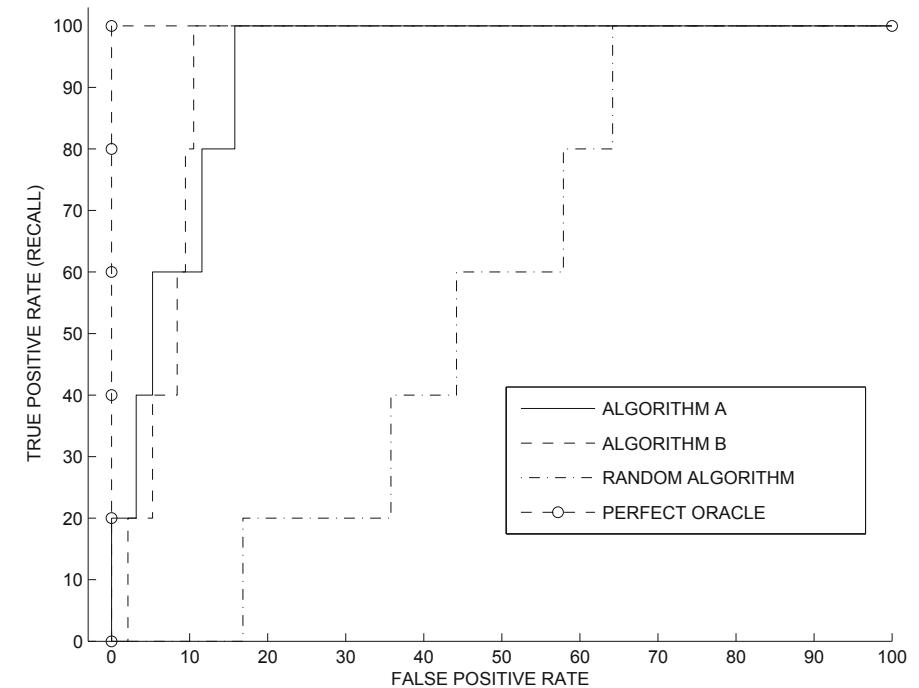
La Receiver Operating Curve (ROC) è tracciata misurando il *False Positive Rate* e il *True Positive Rate* al variare di una soglia definita sullo score della classe positiva.

$$ROC(t) \propto FPR(t) \text{ vs. } TPR(t)$$

$$TPR(t) \equiv Recall(t), FPR(t) = \frac{FP(t)}{FP(t) + TN(t)}$$

AUC, Area Under the Curve, tende a 1 per buona classificazione

Per i problemi multi-classe si possono tracciare più curve, una per singola classe, e calcolare l'AUC media.



(a) ROC