

Introduzione alla Classificazione

Programmazione di Applicazioni Data Intensive

Laurea in Ingegneria e Scienze Informatiche
DISI – Università di Bologna

Gianluca Moro

Dipartimento di Informatica – Scienza e Ingegneria
Università di Bologna
Via dell’Università, 50 – I-47522 Cesena (FC)
Gianluca.Moro@Unibo.it

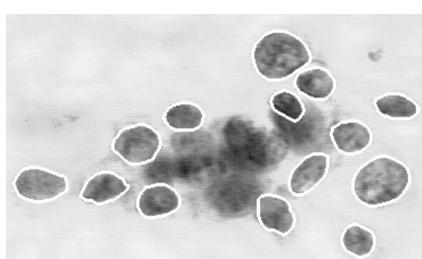


Admire

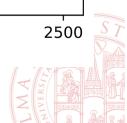
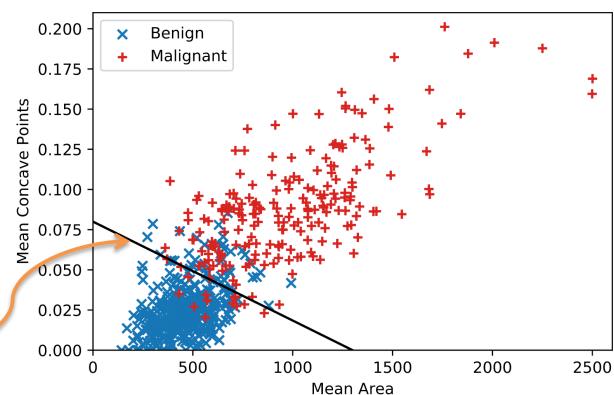
Introduzione alla Classificazione

Cosa Significa Classificare ?

[Street et al., 1993]



- A sinistra cellule cancerogene benigne e maligne descritte da alcune variabili
 - area, perimetro, consistenza, num. concavità, varianza scala di grigi ...
 - e per ognuna anche media, max, varianza
- a dx ogni punto è una cellula
 - rossa maligna, blu benigna: problema a 2 classi (2 insiemi)
 - qui con 2 variabili: num. medio delle concavità (y), area media delle concavità (x)
- **classificare == individuare una funzione che massimizzi la separazione tra le classi**



In uno spazio n-dimensionale, un iperpiano è una superficie (di dimensione n-1) che divide lo spazio in due metà. Serve a separare due classi in un problema di classificazione binaria.

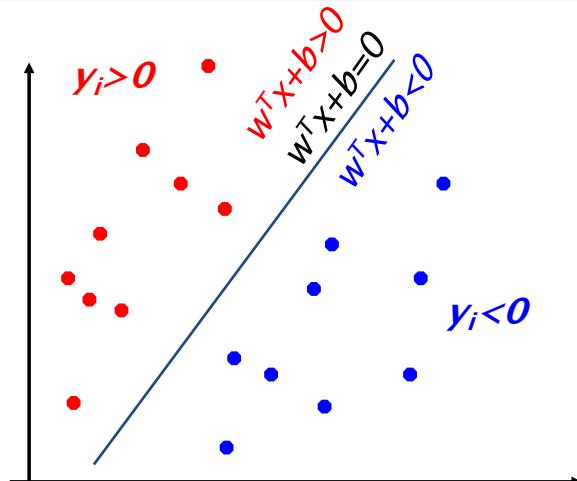
Iperpiano di Separazione e Classificazione

- i punti x_i sull'iperpiano di separazione soddisfano l'uguaglianza $w^T \cdot x_i + b = 0$

- i punti x_i t.c. $w^T \cdot x_i > b$ sono le istanze con $y_i > 0$ e.g. con label 1
- i punti x_i t.c. $w^T \cdot x_i < -b$ sono le istanze con $y_i < 0$ e.g. con label -1
- b è proporzionale alla distanza dell'iperpiano dall'origine i.e. i punti il cui prodotto scalare con l'iperpiano è $> b$ appartengono alla classe $y_i > 0$ e viceversa per l'altra classe
- La distanza dell'iperpiano dall'origine è $\frac{b}{\|w\|}$
- Vediamo perché nella prossima slide

w: vettore dei pesi, perpendicolare all'iperpiano.
x: punto qualsiasi nello spazio.
b: bias o termine di spostamento.

Se $w^T x + b > 0$, appartiene alla classe 1
Se $w^T x + b < 0$, appartiene alla classe -1



Influenza l'equilibrio tra le classi (se è troppo alto o basso, l'iperpiano si sposta verso una classe).



Gianluca Moro - DISI, Università di Bologna

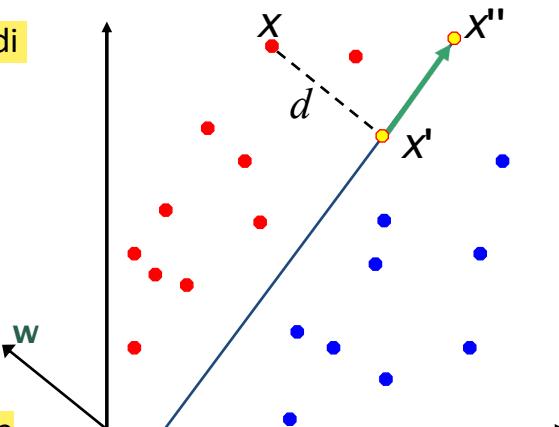
3

Distanza di un Punto da un Iperpiano

- (con la norma 2 calcolo la lunghezza di un vettore)
- La distanza d del punto x dall'iperpiano di separazione è $d = \|x - x'\|$ norma-2 del vettore differenza tra i vettori x e x'
- se 2 vettori sono ortogonali, il loro prodotto scalare è 0 (formano angolo di 90 gradi)
- siano x' e x'' 2 punti sull'iperpiano di separazione $\rightarrow w^T \cdot x' + b = 0$ e $w^T \cdot x'' + b = 0$
- il vettore differenza è $w^T x' - w^T x'' = 0$ ed è parallelo all'iperpiano di separazione ma $w^T \cdot (x' - x'') = 0 \rightarrow$ i 2 vettori sono ortogonali $\rightarrow w$ è ortogonale all'iperpiano
- $w / \|w\|$ è il vettore unitario, perciò $d \cdot w / \|w\| = x - x' \rightarrow x' = x - d \cdot w / \|w\|$ ma x' soddisfa $w^T \cdot x' + b = 0 \rightarrow w^T \cdot (x - d \cdot w / \|w\|) + b = 0 \rightarrow w^T \cdot x - d \cdot w^T w / \|w\| + b = 0$ e poiché $\|w\| = \text{radice_di}(w^T w)$ allora

$$\rightarrow w^T \cdot x - d \cdot \|w\| + b = 0 \rightarrow d = (w^T x + b) / \|w\|$$

perché x' è sull'iperpiano



Gianluca Moro - DISI, Università di Bologna

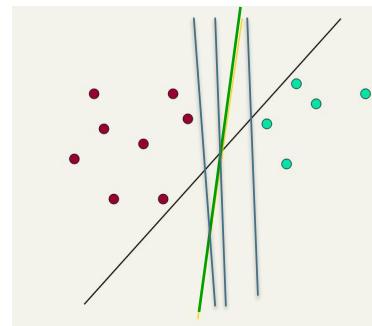
4



In generale, se i dati sono linearmente separabili, esistono infiniti iperpiani che dividono le due classi. Ciò che distingue un classificatore dall'altro è il criterio con cui ne sceglie uno

Alcune Caratteristiche dei Classificatori

- Molte possibili soluzioni con parametri distinti w, b
Nessun criterio di ottimalità esplicito (Qualsiasi iperpiano che separi correttamente i dati va bene)
- Alcuni metodi individuano un iperpiano di separazione non ottimale [in base ad un qualche criterio di ottimalità]
 - E.g., Perceptron, Regressione Logistica, Regressione lineare con soglia
- Criterio di ottimalità esplicito (Si cerca l'iperpiano "migliore" secondo un dato criterio)
 - Altri individuano un iperpiano di separazione ottimale
 - Support Vector Machines
- Quali dati influenzano la ricerca dell'iperpiano ?
 - Tutti i punti
 - Perceptron, Regressione Logistica, Naïve Bayes, Regressione lineare con soglia
 - Solo i "punti difficili", i.e. vicini al decision boundary
 - Support vector machines



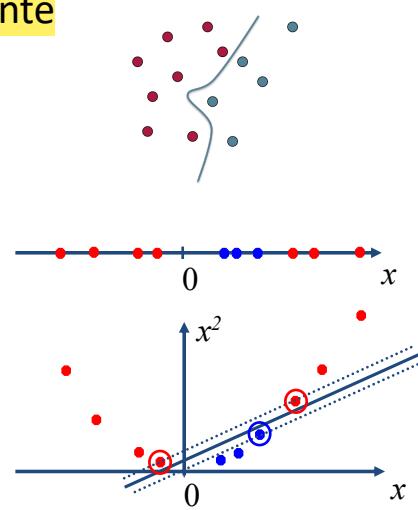
Gianluca Moro - DISI, Università di Bologna

5



Classificatori Non Lineari

- problemi con dati non separabili linearmente
- numerosi approcci
 - Support Vector Machines, Reti Neurali, kNN, Decision Tree, RandomForest
 - Gradient Boosting, XGboost, Logitboost ...
- soluzioni che trasformano lo spazio dei dati in modo che le classi diventino separabili linearmente
 - Support Vector Machines, Reti Neurali ...
- soluzioni intrinsecamente non lineari
 - kNN, Decision Tree, RandomForest, Gradient Boosting, XGboost, lighGBM, Catboost ...
- più variabili hanno i dati, più chance ci sono di separare le classi linearmente



Gianluca Moro - DISI, Università di Bologna

6



cioè più feature hanno le righe del dataset, più la dimensione dello spazio in cui vanno a posizionarsi quelle righe, come punti, aumenta

1

Classificazione Lineare: Perceptron

(Rosenblatt, 1957)

Esempio in 2D: Trovare la retta con parametri w_1, w_2, b tali che

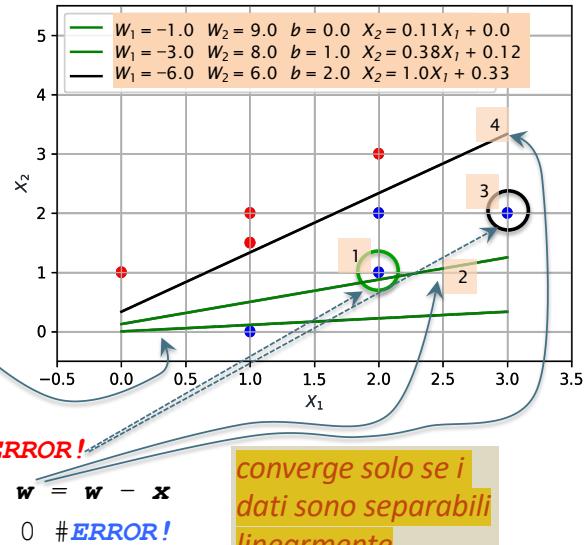
$w_1x_1 + w_2x_2 + b > 0$ per i punti rossi

$w_1x_1 + w_2x_2 + b < 0$ per i punti blu

$\text{class}(x)$: +1 se x è rosso, -1 se blu

```
# Perceptron naive
# b = w_0x_0    x_0=1 for each instance x
w = [0, -1, 9] # initial random plane
repeat
    classification_error = False
    for each instance x
        if class(x) < 0 and w·x ≥ 0 #ERROR!
            classification_error = True; w = w - x
        else if class(x) > 0 and w·x ≤ 0 #ERROR!
            classification_error = True; w = w + x
    until classification_error == True
```

Gianluca Moro - DISI, Università di Bologna



7

Formalizziamo l'Esempio Iniziale a 2 Classi

- variabili di input

$$x_1 = \text{mean_area}$$

$$x_2 = \text{mean_concave_points}$$

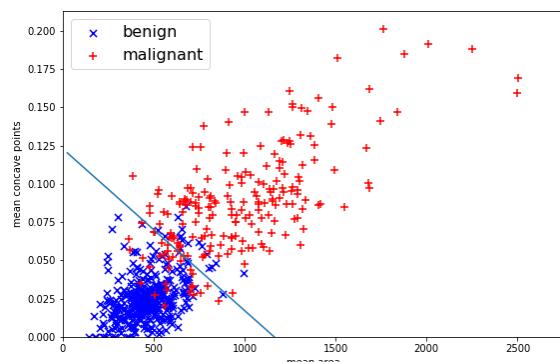
- variabile da predire **discreta**

$$y = \begin{cases} -1 & \text{if benign} \\ +1 & \text{if malign} \end{cases}$$

- separazione lineare delle due classi

- la retta di separazione è $b + w_1x_1 + w_2x_2 = 0$ con w_1, w_2, b da apprendere
- le coppie $x = (x_1, x_2)$ tali che $b + w_1x_1 + w_2x_2 < 0$ sono cellule benigne
- quelle tali che $b + w_1x_1 + w_2x_2 \geq 0$ sono maligne, ossia:

$$y = \begin{cases} -1 & \text{if } b + w \cdot x < 0 \\ +1 & \text{if } b + w \cdot x \geq 0 \end{cases}$$



Gianluca Moro - DISI, Università di Bologna

8



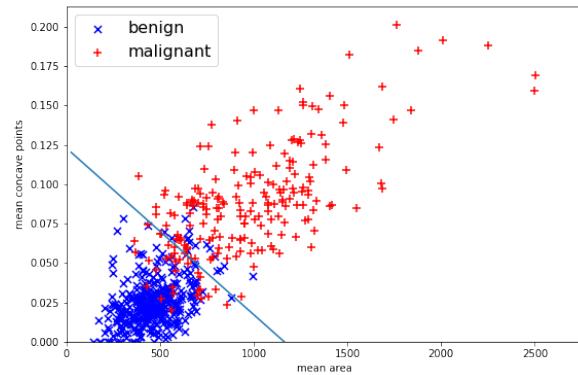
Classificazione di Istanze con Iperpiano

- variabili di input

$$\begin{aligned}x_1 &= \text{mean_area} \\x_2 &= \text{mean_concave_points}\end{aligned} \quad y = \begin{cases} -1 & \text{if } b + w \cdot x < 0 \\ +1 & \text{if } b + w \cdot x \geq 0\end{cases}$$

- Iperpiano individuato (i.e. retta)

- $0.007\text{mean_area} + 67.443\text{mean_concave_points} - 8.287 = 0$



- Classifichiamo due nuove cellule

- $x^{(1)} = (\text{mean_area} = 500, \text{mean_concave_points} = 0.025)$
- $0.007 \cdot 500 + 67.443 \cdot 0.025 - 8.287 = -3.035$ cellula benigna
- $x^{(2)} = (\text{mean_area} = 500, \text{mean_concave_points} = 0.075)$
- $0.007 \cdot 500 + 67.443 \cdot 0.075 - 8.287 = 0.271$ cellula maligna (border line)



Come Trovare l'Iperpiano di Separazione ?

- ogni istanza è etichettata con -1 o +1
- forma compatta equivalente di separazione $-y(b + w \cdot x) < 0$
 - l'istanza correttamente classificata da valore negativo, altrimenti positivo
 - e.g. con $y = -1$ e $b + w \cdot x = -1$ si ha $1 \cdot -1 = -1$ analogamente con istanza $y=1$
- $-y(b + w \cdot x) < 0$ è equivalente a $\max(0, -y(b + w \cdot x)) = 0$
 - l'espressione $\max(0, -y(b + w \cdot x))$ restituisce 0 se l'istanza è classificata correttamente, altrimenti il valore è positivo

- perciò minimizzando, rispetto a b , w , la somma di questa espressione sulle m istanze di training, si minimizza l'errore

Somma dei loss = errore complessivo

$$\underset{b,w}{\text{minimize}} \sum_{i=1}^m \max(0, -y_i \cdot h_w(x_i)) \quad \text{dove} \quad h_w(x_i) = b + w \cdot x_i$$

Minimizzare significa ridurre al minimo il conteggio (e la gravità) degli errori su tutto il training set.

- la funzione è continua e convessa ma non derivabile, perciò il metodo di discesa del gradiente è inapplicabile, inoltre ha un min fittizio per $b=w=0$



(2)

Iperpiano di Separazione: Logistic Loss

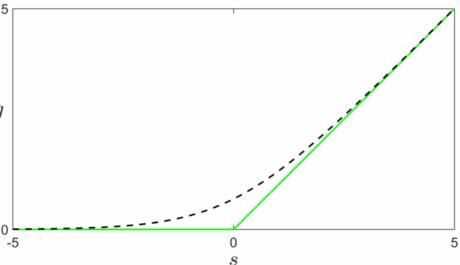
- sostituiamo $\max(0, s)$ in verde con una funzione derivabile
- una sua approssimazione è nota come $\text{softmax}(0, s) = \log(1+e^s)$
 - convessa, continua e derivabile
- perciò minimizzando la somma rispetto a b e w della softmax su tutte le istanze, si minimizza l'errore

$$\underset{b,w}{\text{minimize}} \sum_{i=1}^m \log\left(1 + e^{-y_i \cdot h_w(x_i)}\right)$$

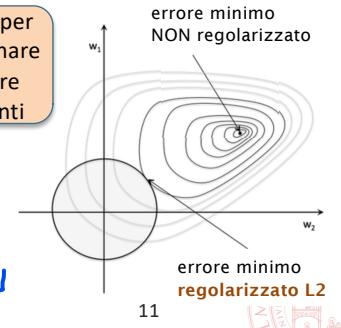
- formulazione nota come *logistic loss* a cui si può aggiungere la regolarizzazione L2 (o L1) dei parametri w con peso λ

$$\underset{b,w}{\text{minimize}} \sum_{i=1}^m \log\left(1 + e^{-y_i \cdot h_w(x_i)}\right) + \frac{\lambda}{2} \|w\|_2^2$$

Gianluca Moro - DISI, Università di Bologna Reg L2



anche per
selezionare
feature
rilevanti



11



Regressione Logistica

- derivata della logistic loss con norma L2 in forma compatta (i.e. senza b)

$$\text{E.g. } \nabla_t (\log 1 + e^{-t c}) = \partial_t (\log 1 + e^{-t c}) \cdot \partial_t (1 + e^{-t c}) = 1/(1 + e^{-t c}) - e^{-t c} c = -c/(1 + e^{t c})$$

$$\tilde{x} = (1, x) \quad \tilde{w} = (b, w) \quad \nabla_{\tilde{w}} \sum_{i=1}^m \log\left(1 + e^{-y_i \cdot h_{\tilde{w}}(\tilde{x}_i)}\right) + \frac{\lambda}{2} \|\tilde{w}\|_2^2 = \sum_{i=1}^m -\frac{y_i \tilde{x}_i}{1 + e^{y_i \cdot h_{\tilde{w}}(\tilde{x}_i)}} + \lambda \tilde{w}$$

- determinazione dei parametri w migliori con discesa del gradiente

$$\tilde{w} = \tilde{w} - \eta \left(\sum_{i=1}^m -\frac{y_i \tilde{x}_i}{1 + e^{y_i \cdot h_{\tilde{w}}(\tilde{x}_i)}} + \lambda \tilde{w} \right)$$

- il **classificatore lineare** appreso $h_{\tilde{w}}(\tilde{x})$ è utilizzato nella funzione seguente nota come **Regressione Logistica** (o sigmoide), versione moderna del Perceptron

$$\sigma(h_{\tilde{w}}(\tilde{x})) = \frac{1}{1 + e^{-h_{\tilde{w}}(\tilde{x})}} = \frac{1}{1 + e^{-(b + w \cdot x)}} \quad \text{dove } b + w \cdot x \text{ è l'iperpiano individuato}$$

Gianluca Moro - DISI, Università di Bologna

12



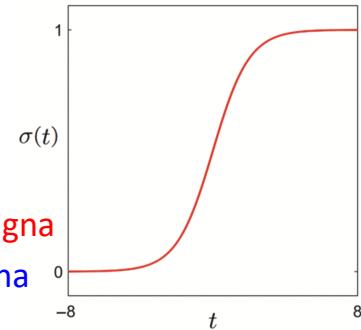
Regressione Logistica: Caratteristiche

- $\sigma(t) = \frac{1}{1 + e^{-(b+w \cdot t)}}$ Dominio e Codominio: $R \rightarrow [0, 1]$
- il risultato è interpretabile come probabilità di appartenenza di ogni istanza t ad una delle classi (es. in figura con una variabile di input)
 - e.g. $P(Y=+1 | X=x^{(i)}) = \text{prob. che } x^{(i)} \text{ sia una cellula maligna}$
 - $P(Y=-1 | X=x^{(i)}) = 1 - P(Y=+1 | X=x^{(i)}) = \text{prob. } x^{(i)} \text{ cell. benigna}$
- approssima una funzione a gradino

se x è sull'iperpiano allora $w \cdot x + b = 0 \rightarrow \sigma(x) = \frac{1}{1 + e^0} = 0.5$

se x è t.c. $w \cdot x + b > 0 \rightarrow \sigma(x) = \frac{1}{1 + e^{-(b+w \cdot x>0)}} > 0.5$ (Classe 1)

se x è t.c. $w \cdot x + b < 0 \rightarrow \sigma(x) = \frac{1}{1 + e^{-(b+w \cdot x<0)}} < 0.5$ (Classe -1)



- tutto ciò vale sia con più variabili di input dove $x = (x_1, \dots, x_n)$, sia con più di 2 classi

Gianluca Moro - DISI, Università di Bologna

14



Regressione Logistica: Esempio di Classificazione con 2 Variabili di Input

- Classificazione di cellule in benigne e maligne: iperpiano individuato

$$0.007\text{mean_area} + 67.443\text{mean_concave_points} - 8.287 = 0$$

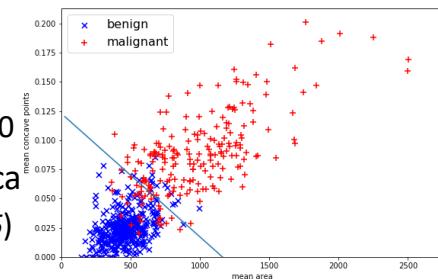
e.g. 2 cellule $x^{(1)}$ e $x^{(2)}$, con iperpiano e con logistica

$x^{(1)} = (\text{mean_area} = 500, \text{mean_concave_points} = 0.075)$

$$0.007 \cdot 500 + 67.443 \cdot 0.075 - 8.287 = 0.337 > 0 \text{ (maligna)}$$

$$\sigma(500, 0.075) = \frac{1}{1 + e^{-(0.007 \cdot 500 + 67.443 \cdot 0.075 - 8.287)}} = 0.567$$

$$\begin{aligned} \sigma(500, 0.075) &= P(Y=+1 | X=x^{(1)}) = 0.567 \text{ Classe } y^{(1)} = +1 \\ P(Y=-1 | X=x^{(1)}) &= 1 - 0.567 = 0.433 \end{aligned}$$

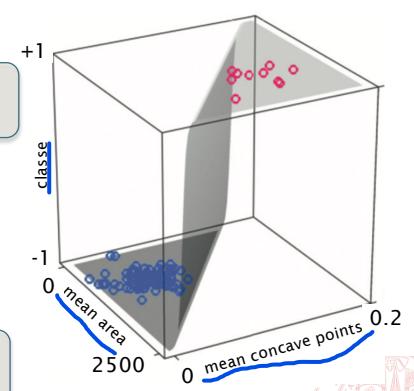


$x^{(2)} = (\text{mean_area} = 500, \text{mean_concave_points} = 0.025)$

$$0.007 \cdot 500 + 67.443 \cdot 0.025 - 8.287 = -3.1 < 0 \text{ (benigna)}$$

$$\sigma(500, 0.025) = \frac{1}{1 + e^{-(0.007 \cdot 500 + 67.443 \cdot 0.025 - 8.287)}} = 0.046$$

$$\begin{aligned} \sigma(500, 0.025) &= P(Y=+1 | X=x^{(2)}) = 0.046 \text{ Classe } y^{(2)} = -1 \\ P(Y=-1 | X=x^{(2)}) &= 1 - 0.046 = 0.954 \end{aligned}$$



Gianluca Moro - DISI, Università di Bologna

15



Regressione Logistica in Due Classi: Multivariata Lineare e Non Lineare

- Sia $\mathbf{x} \in \mathbb{R}^n$ un'istanza $\mathbf{x} = (x_1, \dots, x_n)$ in n variabili, la funzione di classificazione lineare multivariata è

$$\sigma(x_1, \dots, x_n) = \frac{1}{1 + e^{-h_w(x_1, \dots, x_n)}} \quad \text{dove } h_w(x_1, \dots, x_n) = w_1 x_1 + \dots + w_n x_n + b$$

è l'iperpiano di separazione

- L'individuazione dell'iperpiano avviene con il medesimo metodo già illustrato
- Regressione logistica non lineare: es con polinomio di grado 2

$$\sigma(x_1, \dots, x_n) = \frac{1}{1 + e^{-h_w^2(x_1, \dots, x_n)}} \quad \text{dove } h_w^2(x_1, \dots, x_n) = \sum_{i=1}^n w_{i,1} x_i^2 + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n w_{i,j} x_i x_j$$

- Sul numero dei termini generati in n variabili con grado g e non scalabilità, vale quanto già visto nella regressione non lineare, i.e. $\binom{n+g-1}{g}$ ma il problema è stato superato perché come vedremo esiste la kernel logistic regression (SVM)

Gianluca Moro - DISI, Università di Bologna

16



Regressione Logistica con Feature Non Lineari in Python

```

from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
...
model = Pipeline([
    ("scaler", StandardScaler()),
    # grado 5
    ("poly", PolynomialFeatures(degree=5)),
    # C regolarizzazione
    ("logreg", LogisticRegression(C=0.0011))
])
model.fit(X_train, y_train)
model.score(X_test, y_test)
# parametri w del modello appreso
model.named_steps['logreg'].coef_.T

```

Gianluca Moro - DISI, Università di Bologna

17



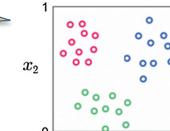
3

Classificazione Multiclasse con Iperpiani

- 2 metodi con C classi

1 One-Versus-All e 2 Multinomial

3 classi



argmax: Questa funzione sceglie l'indice della classe j per cui il punteggio è massimo. In altre parole, l'istanza viene assegnata alla classe per cui ha ottenuto il punteggio più alto.

1

- One-Versus-All in python `multiclass='ovr'`

individuazione di C iperpiani: uno per ogni classe da separare da tutte le altre

$$b_c + \mathbf{x}^T \mathbf{w}_c = 0, \quad c = 1, \dots, C.$$

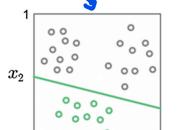
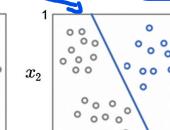
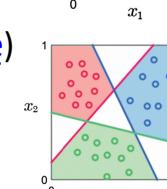
stesso metodo per l'individuazione di un singolo iper piano (è parallelizzabile)

Regola di fusione: ad ogni istanza \mathbf{x} si assegna la classe y corrispondente al piano j che massimizza

Così, anche se più iperpiani "dicono sì" (score positivo), sceglieremo quello che giudica x più decisamente come appartenente alla sua classe.

$$y = \operatorname{argmax}_{j=1, \dots, C} b_j + \mathbf{x}^T \mathbf{w}_j$$

Viene scelta la classe j che ha $b_j + \mathbf{x}^T \mathbf{w}_j$ con valore maggiore tra tutte le classi



- quindi `model.coef_.T` restituisce C array di parametri \mathbf{w}_c $c = 1, \dots, C$

Gianluca Moro - DISI, Università di Bologna

18

Per classificare una nuova istanza, l'istanza viene passata a tutti i C classificatori binari. Ogni classificatore restituisce un "punteggio di confidenza" o una probabilità che l'istanza appartenga alla sua classe "positiva". La classe finale predetta è quella associata al classificatore che ha restituito il punteggio di confidenza più alto.

Le "probabilità" o i punteggi di confidenza da diversi classificatori binari potrebbero non sommarsi a 1, rendendo difficile interpretarli come vere probabilità. Un'istanza potrebbe ottenere un punteggio alto da più classificatori, o basso da tutti, portando a incertezze.

2

Classificazione Multiclasse: Multinomial

in python `multiclass='multinomial'`

e.g. con 3 classi

individua congiuntamente C iperpiani minimizzando la regola di fusione

$$y = \operatorname{argmax}_{j=1, \dots, C} b_j + \mathbf{x}^T \mathbf{w}_j$$

quindi, ogni istanza \mathbf{x}_p è classificata correttamente nella propria classe c

se $b_c + \mathbf{x}_p^T \mathbf{w}_c = \max_{j=1, \dots, C} (b_j + \mathbf{x}_p^T \mathbf{w}_j)$ ossia in $\max_{j=1, \dots, C} (b_j + \mathbf{x}_p^T \mathbf{w}_j) - (b_c + \mathbf{x}_p^T \mathbf{w}_c) = 0$
la parte sinistra è > 0 solo con errori, perciò è la loss da minimizzare

La funzione Softmax trasforma i punteggi in un vettore di probabilità

per derivarla sostituiamo max con softmax (come in una variabile) e minimizziamo la seguente funzione (Ω_c set di istanze della classe c):

$$g(b_1, \dots, b_C, \mathbf{w}_1, \dots, \mathbf{w}_C) = \sum_{c=1}^C \sum_{p \in \Omega_c} \left[\max_{j=1, \dots, C} (b_j + \mathbf{x}_p^T \mathbf{w}_j) - (b_c + \mathbf{x}_p^T \mathbf{w}_c) \right] = -\sum_{c=1}^C \sum_{p \in \Omega_c} \log \left(\frac{e^{b_c + \mathbf{x}_p^T \mathbf{w}_c}}{\sum_{j=1}^C e^{b_j + \mathbf{x}_p^T \mathbf{w}_j}} \right)$$

- model.coef_.T restituisce C iperpiani come in 'ovr'



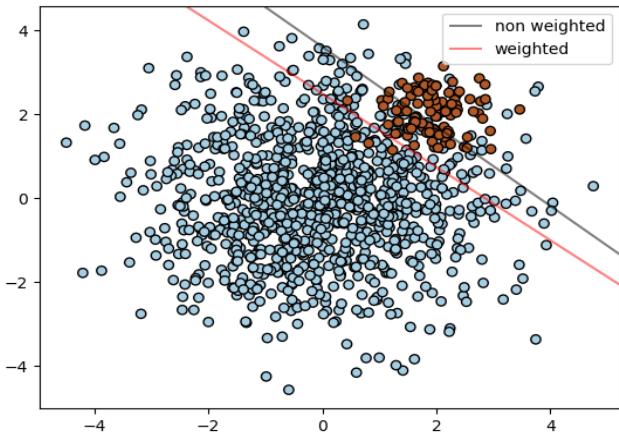
$$\text{soft}(s_1, \dots, s_C) = \log \left(\sum_{j=1}^C e^{s_j} \right)$$

multiclass softmax



Classificazione con Classi Sbilanciate

- In numerosi problemi reali la suddivisione di istanze tra classi è molto sbilanciata
 - e.g. transazioni con carte di credito: **lecite** >> **illecite**
 - diagnosi malattie: **negativo** >> **positivo**
 - churn analysis: clienti **fedeli** >> **infedeli**
- Molti più errori di classificazione sulla classe meno rappresentata, spesso con risultati inaccettabili
- Rimedi per ridurre il problema
- aumentare il peso degli errori sulla classe con meno istanze: parametro `class_weight` nella inizializzazione del modello, es. `class_weight={1: 10}` # peso 10 volte superiore della classe 1 (miglior peso individuabile con gridsearch) con multiclassa `class_weight = {0: 1., 1: 50., 2: 2.}`
 - undersampling* della classe numerosa, *oversampling* classe con meno dati mediante varie tecniche tra cui SMOTE `from imblearn.over_sampling import SMOTE`



Gianluca Moro - DISI, Università di Bologna

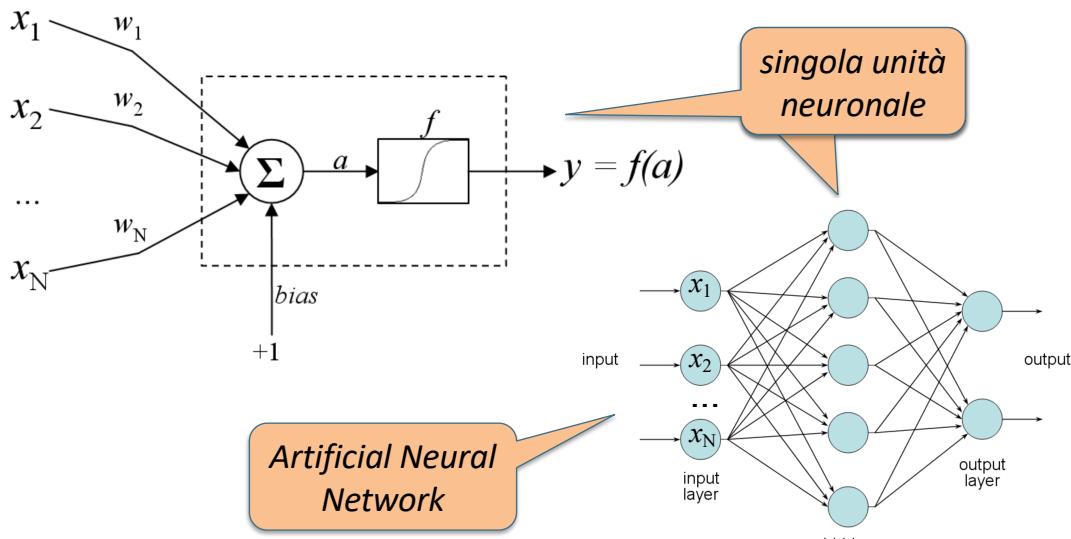
20



Regressione Logistica == Unità Neuronale

$$f(a) = \frac{1}{1+e^{-a}} \text{ dove } h_w(x_1, \dots, x_n) = w_1x_1 + \dots + w_nx_n + 1 = a$$

è l'iperpiano di separazione



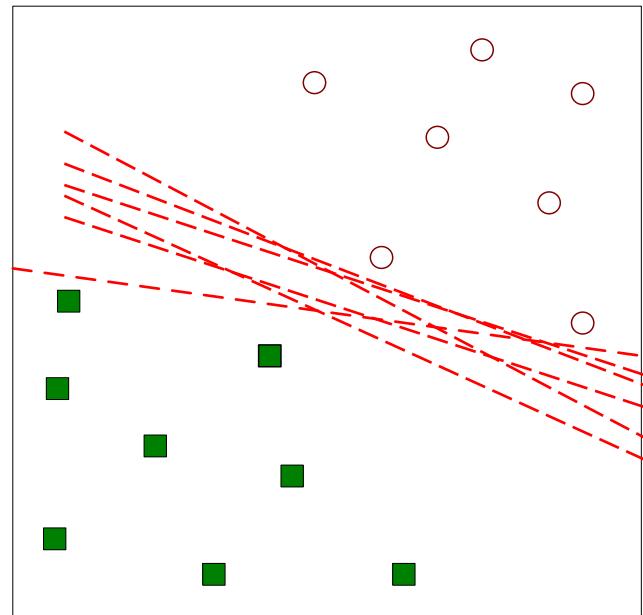
Gianluca Moro - DISI, Università di Bologna

21



Ritorniamo all'Individuazione dell'Iperpiano

- l'iperpiano individuato da Perceptron e Regressione Logistica è uno dei tanti possibili iperpiani di separazione
- manca un criterio per stabilire quale tra le soluzioni sia la migliore
 - al di là della minimizzazione del numero di errori sul training set
 - *tuttavia ricordiamo che la regolarizzazione vincola e riduce le possibili soluzioni*

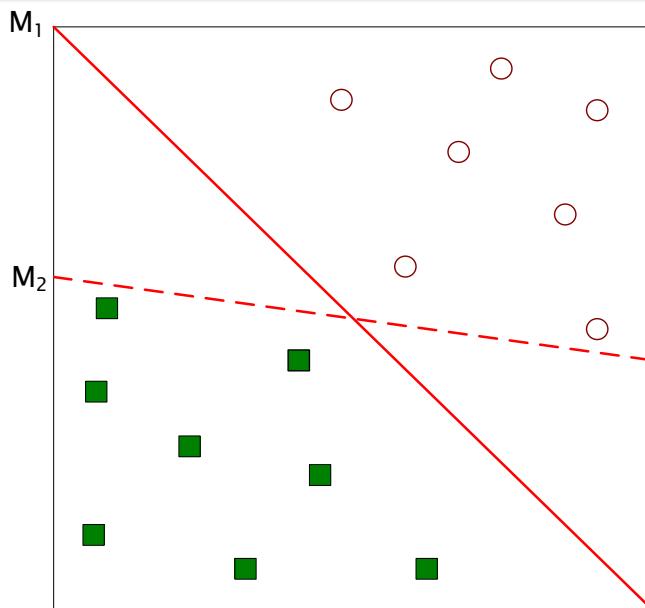


Gianluca Moro - DISI, Università di Bologna

22



Come Stabilire Quale Sia il Migliore ?



- Quale è il migliore, M1 o M2 ? Come definiamo il concetto di *migliore* ?

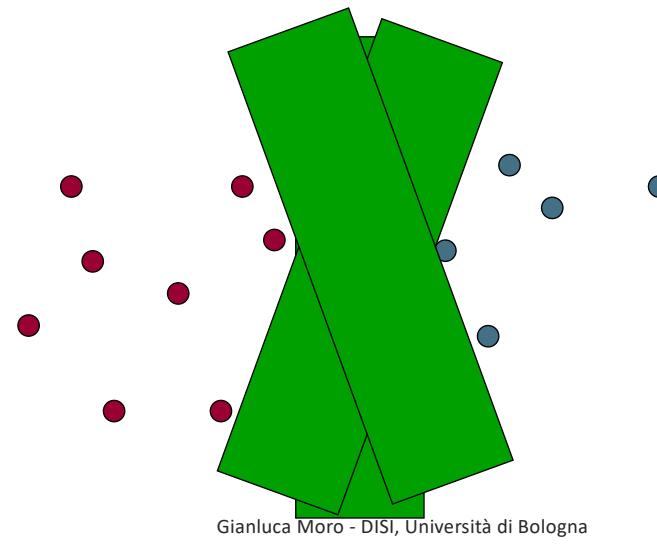
Gianluca Moro - DISI, Università di Bologna

23



Idea ...

- Il numero di soluzioni diminuisce se cerchiamo una separazione lineare con il maggiore margine possibile tra le istanze delle due classi



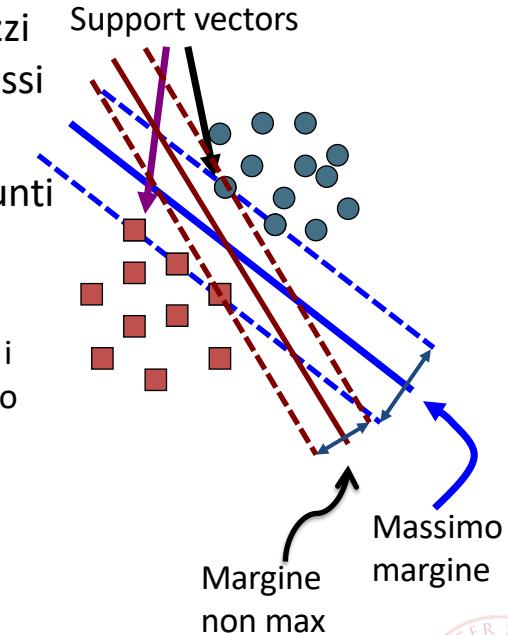
25



Iperpiano di Separazione Migliore

- individuare l'iperpiano che massimizzi il margine tra le istanze delle due classi
→ **minore overfitting**
- L'iperpiano migliore è definito dai punti “difficili” chiamati *support vectors*
 - punti più vicini al decision boundary
 - Se mancassero i restanti punti (i.e. tutti i non support vector) l'iperpiano calcolato sarebbe il medesimo
- Risolubile come problema di ottimizzazione quadratica →

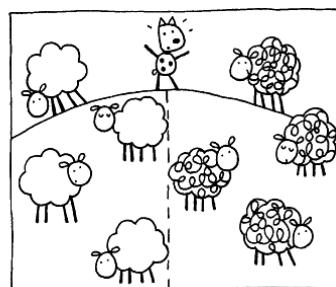
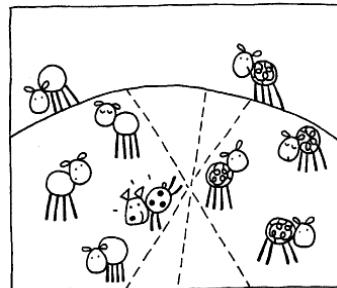
SUPPORT VECTOR MACHINES



SVM: Support Vector Machines

Argomento Facoltativo

- Metodo sviluppato da V. Vapnik (e suoi co-autori) negli anni 70 in Russia e diventato noto internazionalmente solo nel 1992
- Definito originariamente per la classificazione binaria
 - i.e. istanze appartenenti in modo esclusivo a due classi
- Obiettivo: individuare la separazione lineare ottimale (secondo un criterio geometrico) tra le istanze delle due classi
 - Adatto a domini ad elevata dimensionalità
 - Ritenuto tra i metodi più efficaci per il testo
 - più efficace di altri metodi con training set piccoli



Gianluca Moro - DISI, Università di Bologna

27



Definizione del Problema

- Sia $D=\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_r, y_m)\}$ l'insieme delle r istanze di **training**
- dove $\mathbf{x}_i = (x_1, \dots, x_n)$ è il **vettore dell'istanza i -esima** nello spazio di valori reali $X \subseteq \mathbb{R}^n$
- y_i è la **classe** di appartenenza di \mathbf{x}_i , con $y_i \in \{1, -1\}$
 - 1: *classe esempi positivi* -1: *classe esempi negativi*
- Individuare \mathbf{w} e b t.c. l'iperpiano $\mathbf{w}^\top \cdot \mathbf{x} + b = 0$ massimizzi la separazione tra i support vector delle 2 classi
- Il classificatore risultante è una funzione lineare

$$f(\mathbf{x}_i) = y_i \quad \text{dove} \quad y_i = \begin{cases} 1 & \text{se } \mathbf{w} \cdot \mathbf{x}_i + b \geq 0 \\ -1 & \text{se } \mathbf{w} \cdot \mathbf{x}_i + b < 0 \end{cases}$$

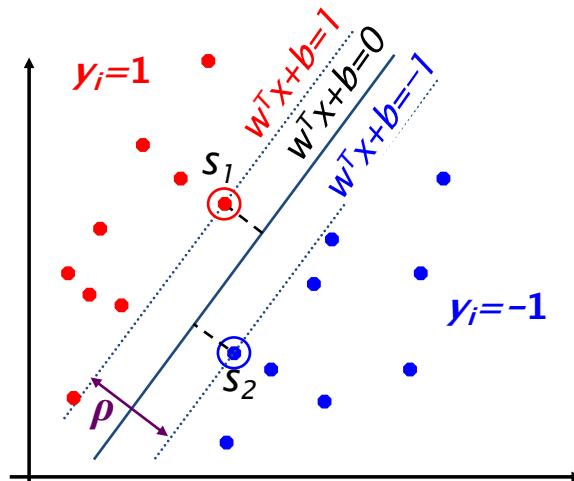
Gianluca Moro - DISI, Università di Bologna

40



Separazione Lineare

- i punti x_i sull'iperpiano di separazione soddisfano l'uguaglianza $w^T \cdot x_i + b = 0$
 - i punti x_i t.c. $w^T \cdot x_i + b > 1$ sono le istanze con $y_i = 1$ e
 - i punti x_i t.c. $w^T \cdot x_i + b < -1$ sono le istanze con $y_i = -1$
 - b è proporzionale alla distanza dello iperpiano dall'origine
- siano s_1 e s_2 due support vector delle 2 classi (punti più vicini all'iperpiano)
- definiamo i 2 iperpiani paralleli a $w^T \cdot x + b = 0$ passanti per s_1 e s_2
- $w^T \cdot s_1 + b = 1$ $w^T \cdot s_2 + b = -1$, ρ è il margine (distanza) tra loro
- Obiettivo:** calcolare gli s_j , w , b che massimizzino ρ



Gianluca Moro - DISI, Università di Bologna

29



Per Trovare il Margine ρ , Calcoliamo la Distanza di un Punto da un Iperpiano

- La distanza d del punto x dall'iperpiano di separazione è $d = \|x - x'\|$ norma-2 del vettore differenza tra i vettori x e x'
- se 2 vettori sono *ortogonali*, il loro prodotto scalare è 0
- siano x' e x'' 2 punti sull'iperpiano di separazione $\rightarrow w^T \cdot x' + b = 0$ e $w^T \cdot x'' + b = 0$
- il vettore differenza è $w^T \cdot x' - w^T \cdot x'' = 0$ ed è parallelo all'iperpiano di separazione ma $w^T \cdot (x' - x'') = 0 \rightarrow$ i 2 vettori sono ortogonali $\rightarrow w$ è ortogonale all'iperpiano
- $w / \|w\|$ è il vettore unitario, perciò $d \cdot w / \|w\| = x - x' \rightarrow x' = x - d \cdot w / \|w\|$ ma x' soddisfa $w^T \cdot x' + b = 0 \rightarrow w^T \cdot (x - d \cdot w / \|w\|) + b = 0 \rightarrow w^T \cdot x - d \cdot w^T \cdot w / \|w\| + b = 0$ e poiché $\|w\| = \text{radice_di}(w^T w)$ allora

$$\rightarrow w^T \cdot x - d \cdot \|w\| + b = 0 \rightarrow d = (w^T \cdot x + b) / \|w\|$$

Gianluca Moro - DISI, Università di Bologna

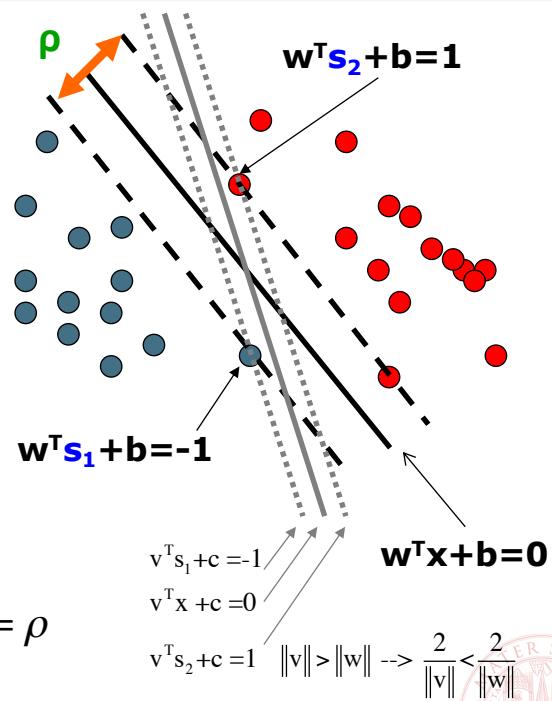
30



Calcolo Analitico del Margine ρ

- sia l'iperpiano di separazione
 $w^T x + b = 0$ t.c. $\forall (x_i, y_i) \in D$
 $y_i(w^T x_i + b) \geq 1$
- e 2 iperpiani paralleli a quello di separazione $|w^T x + b| = 1$
 - gli x_i che soddisfano questa equazione sono support vector
- siano s_1, s_2 due support vector
- la somma della loro distanza dall'iperpiano è il margine ρ

$$\frac{|w^T s_1 + b|}{\|w\|} + \frac{|w^T s_2 + b|}{\|w\|} = \frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|} = \rho$$



Gianluca Moro - DISI, Università di Bologna

31

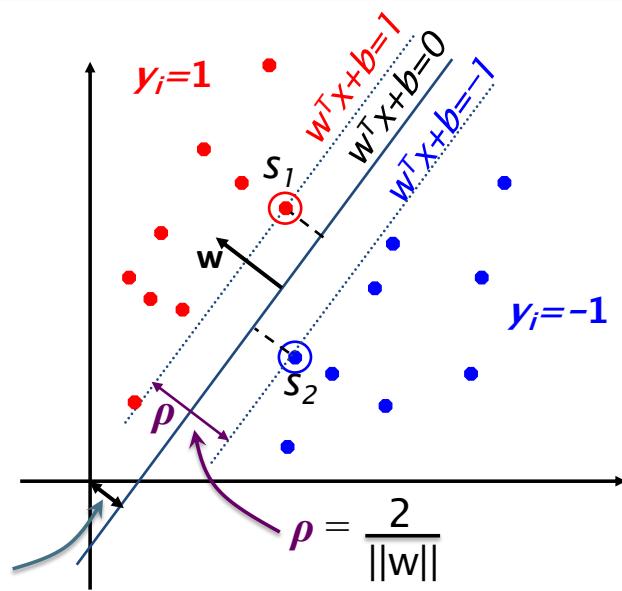


Iperpiano di Separazione: Il Parametro b

- iperpiano di separazione
 $w^T x + b = 0$
- quanto dista dall'origine ?
- la distanza di un punto x_i dall'iperpiano è

$$\frac{w^T x_i + b}{\|w\|}$$

- Le componenti del punto nell'origine valgono 0
- $\rightarrow w^T x_i = 0$, perciò dista $\frac{b}{\|w\|}$
- b determina la distanza dell'iperpiano dall'origine



Gianluca Moro - DISI, Università di Bologna

32



SVM Lineare: il Problema di Ottimizzazione

Calcolare \mathbf{w} e b tali che sia massimizzato il margine

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ con i vincoli, per ogni } \{(\mathbf{x}_i, y_i)\}$$

$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ se } y_i = 1$
 $\mathbf{w}^T \mathbf{x}_i + b \leq -1 \text{ se } y_i = -1$

o in forma compatta
 $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

SVM con
margine
hard

- Formulazione equivalente: $\max \frac{2}{\|\mathbf{w}\|} = \min \frac{\|\mathbf{w}\|}{2} = \min \frac{\sqrt{\sum_{j=1}^n w_j^2}}{2} \rightarrow \min \frac{\mathbf{w}^T \mathbf{w}}{2}$

è più semplice
minimizzare $\|\mathbf{w}\|^2$
di $\|\mathbf{w}\|$

Calcolare \mathbf{w} e b tali che sia minimizzata

$$\Phi(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{w}}{2} \text{ con i vincoli } y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \text{ per ogni } \{(\mathbf{x}_i, y_i)\}$$

soluzione
mediante inclusione
dei vincoli nella
funzione obiettivo
con moltiplicatori di
Lagrange



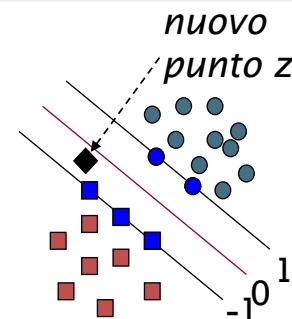
SVM: La Funzione di Classificazione Finale

- La funzione di classificazione ha questa forma:

$$f(z) = \text{sign}(\mathbf{w}^T z + b) \quad \text{dove}$$

$$\mathbf{w}^T z = \sum_{k \in \text{support vector}} y_k \alpha_k \mathbf{x}_k^T z \quad b = \frac{1}{y_k} - \mathbf{w}^T \mathbf{x}_k = y_k - \mathbf{w}^T \mathbf{x}_k$$

α_k calcolati con
moltiplicatori di Lagrange

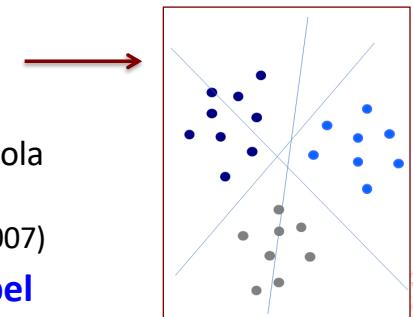
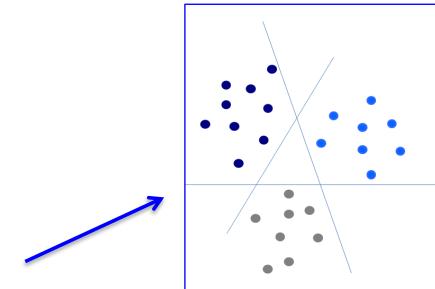


- data un'istanza z , $f(z)$ esegue il prodotto scalare tra z ed i support vector $\mathbf{x}_k \Rightarrow$ non occorre l'iperpiano di separazione
- se $f(z)$ ha segno positivo allora z è classificata come positiva, altrimenti è negativa
 - il costo teorico per determinare la funzione di classificazione include quello del prodotto scalare di ogni coppia delle istanze di training
 - algoritmi numerici risolvono il problema con complessità lineare rispetto al numero delle istanze di training (<https://link.springer.com/article/10.1007/s10994-009-5108-8>)



SVM: Classificazione Multi-Classe

- Metodo di classificazione binaria
 - i.e. esistono 2 classi ed ogni istanza appartiene ad una e una sola classe
- Tre modalità per applicare SVM a classificazioni con K classi:
 - **1-molti:** K data set a 2 classi, ciascuno contiene a turno una classe e l'unione delle restanti classi → generazione di K SVM
 - **1-1:** si genera una SVM per ogni coppia di classi → $K(K-1)/2$ data set e SVM
 - riformulazione del metodo SVM con una sola funzione obiettivo invece che K o $K(K-1)/2$ (Crammer 2002, Lee-Wabha 2004, Nanculef 2007)
 - **1-molti adatta anche a problemi multi-label**



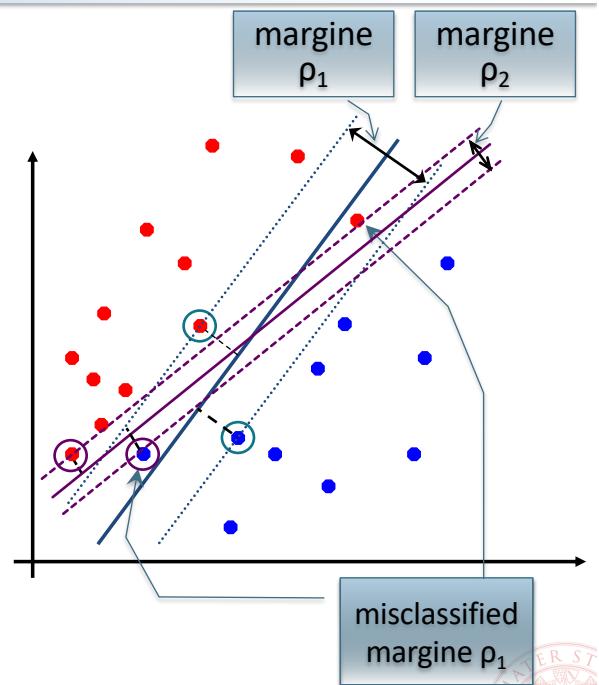
Gianluca Moro - DISI, Università di Bologna

35



Margine Soft e Istanze Misclassified

- maggiore è il margine, minore è l'overfitting
 - accuratezza più affidabile su nuove istanze
- margine ρ_2 senza errori
- margine $\rho_1 > \rho_2$ ma con errori
- **qual è l'iperpiano migliore ?**
- trade off tra *massimizzazione del margine* e *minimizzazione del numero di errori*
→ MARGINE SOFT
- Introduzione della tolleranza agli errori nel training set ma con penalizzazione



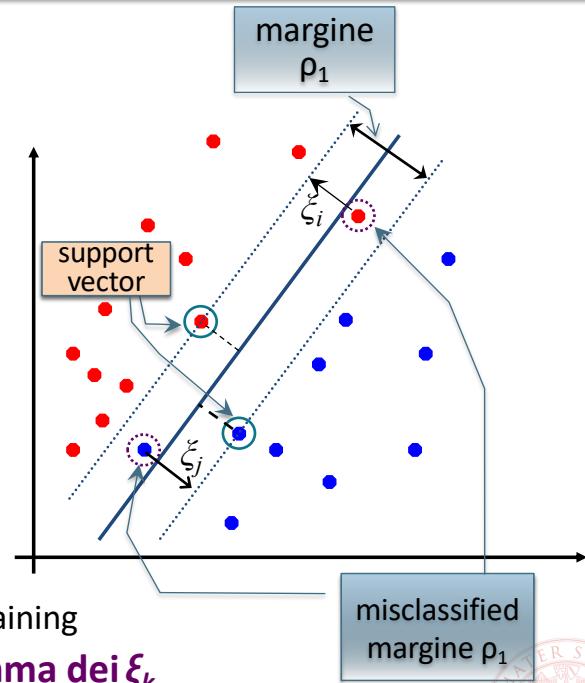
Gianluca Moro - DISI, Università di Bologna

36



Istanze Misclassified e Variabili Slack

- La formulazione illustrata non ammette errori nel training set
- si introducono altre variabili ξ chiamate *slack* per “spostare” le istanze misclassified
- Le variabili ξ “trasformano” gli errori, i.e. i punti, all’interno del margine in support vector
 - massimizzando solo il **margine** con tali trasformazioni, otterremmo soluzioni ottime degeneri
 - il margine diventerebbe così ampio che conterebbe tutte le istanze di training
- minimizziamo perciò anche la somma dei ξ_k**



Gianluca Moro - DISI, Università di Bologna

37



Margine Soft: Ottimizzazione con Variabili Slack

- Formulazione precedente con margine hard:

Calcolare \mathbf{w} e b tali che sia minimizzata

$$\Phi(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{w}}{2} \quad \text{con i vincoli, per ogni } \{(\mathbf{x}_i, y_i)\}, \\ y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

La soluzione ha questa forma: $f(\mathbf{x}) = \text{sign}(\sum y_i \alpha_i \mathbf{x}_i^T \mathbf{x} + b)$

- Formulazione con variabili slack (margine soft):

Calcolare \mathbf{w} e b tali che sia minimizzata

$$\Phi(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_i \xi_i \quad \text{con i vincoli, per ogni } \{(\mathbf{x}_i, y_i)\}, \\ y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \\ \text{con } \xi_i \geq 0 \text{ per ogni } i$$

- l’iperparametro $C > 0$ permette di controllare l’overfitting

Gianluca Moro - DISI, Università di Bologna

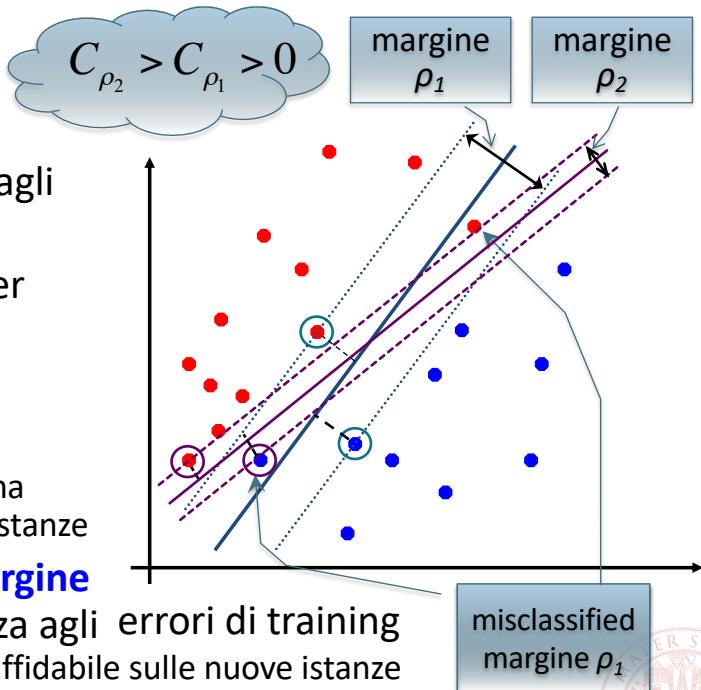
38



Margine Soft e l'Iperparametro C

- $\min \frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_i \xi_i$

- aumentando il valore di C ,** aumenta il peso assegnato agli errori di training
- **il margine si restringe** per ridurre gli errori di training
- più il margine è piccolo, più aumenta l'overfitting
 - più accuratezza sul training ma meno affidabile sulle nuove istanze
- riducendo C aumenta il margine** perché aumenta la tolleranza agli errori di training
→ accuratezza inferiore ma più affidabile sulle nuove istanze



Gianluca Moro - DISI, Università di Bologna

39



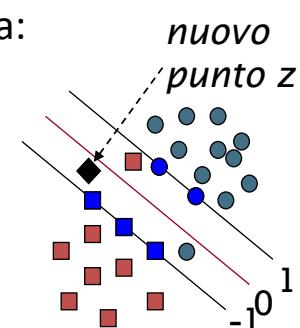
SVM con Variabili Slack: La Funzione di Classificazione Finale

- La funzione di classificazione ha la stessa forma:

$$f(z) = \text{sign}(\mathbf{w}^T z + b) \quad \text{dove}$$

$$\mathbf{w}^T z = \sum_{k \in \text{support vector}} y_k \alpha_k x_k^T z \quad b = \frac{1}{y_k} - \mathbf{w}^T x_k = y_k - \mathbf{w}^T x_k$$

α_k calcolati con moltiplicatori di Lagrange



- i.e. la classificazione avviene come nella formulazione senza variabili slack
 - data un'istanza z , $f(z)$ esegue il prodotto scalare tra z e i support vector $x_k \Rightarrow$ non occorre l'iperpiano di separazione
 - se $f(z)$ ha segno positivo allora z è classificata come positiva, altrimenti negativa → la classificazione lineare con margine soft è efficiente quanto la classificazione senza variabili slack

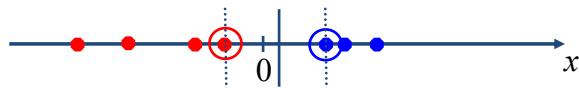
Gianluca Moro - DISI, Università di Bologna

40



SVM Non Lineari

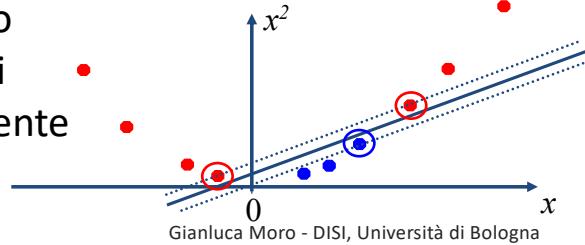
- I metodi precedenti individuano soluzioni per dati linearmente separabili (anche con misclassified):



- Come trattare dati non linearmente separabili, nemmeno con variabili slack, in modo efficace ?



- Mapping dei dati in uno spazio con maggiori dimensioni dove diventano separabili linearmente



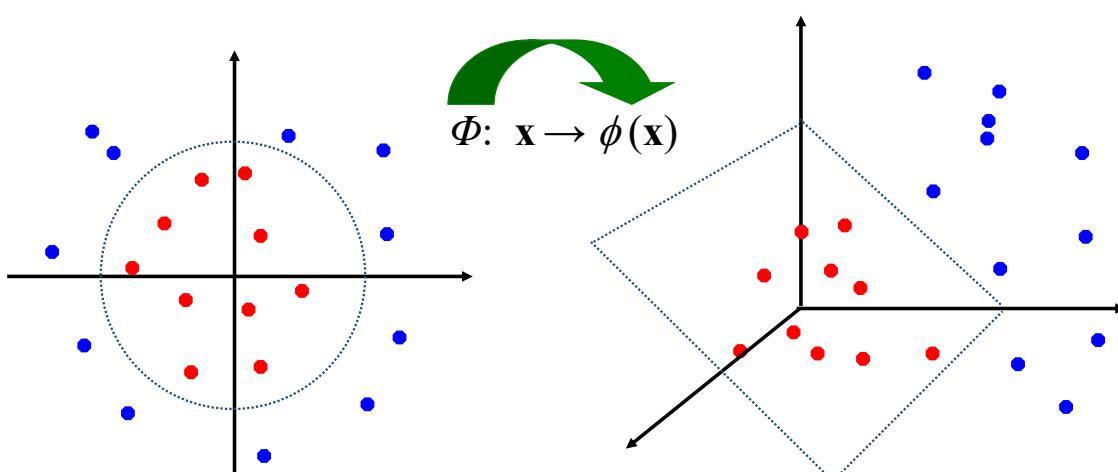
Gianluca Moro - DISI, Università di Bologna

41



SVM Non Lineari: Spazio dei Dati

- Mapping dello spazio dei dati, **con una funzione non lineare**, in uno spazio con più dimensioni dove il training set è separabile con **SVM lineari**



Gianluca Moro - DISI, Università di Bologna

42



Ottimizzazione nello Spazio Trasformato

- Dopo il mapping, il training set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_r, y_r)\}$ diventa $\{(\phi(\mathbf{x}_1), y_1), (\phi(\mathbf{x}_2), y_2), \dots, (\phi(\mathbf{x}_r), y_r)\}$

Calcolare \mathbf{w} e b tali che sia minimizzata

$$\Phi(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_i \xi_i \quad \text{con i vincoli, } \forall \{(\mathbf{x}_i, y_i)\},$$

$$y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \text{con } \xi_i \geq 0 \quad \forall i=1,\dots,r$$

soluzione
mediante inclusione
dei vincoli nella
funzione obiettivo
con moltiplicatori α
di Lagrange

Funzione di Classificazione $f(z) = \text{sign}\left(\sum_{k \in SV} y_k \alpha_k \phi(\mathbf{x}_k^T) \phi(z) + b\right)$ con SV insieme dei support vector

prodotto scalare dei vettori trasformati

Gianluca Moro - DISI, Università di Bologna

43



SVM non lineari: Esempio di Trasformazione

- Supponiamo di definire in un spazio di dati bidimensionale la trasformazione quadratica da 2 a 3 variabili:

$$(\mathbf{x}_1, \mathbf{x}_2) \mapsto (\mathbf{x}_1^2, \mathbf{x}_2^2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2)$$

- L'istanza di training $(2, 3)$ della classe -1 , i.e. $((2, 3), -1)$, è mappata, dallo spazio bidimensionale nello spazio delle feature come $((4, 9, 8.5), -1)$
- In generale questo approccio è soggetto al problema della “maledizione della dimensionalità” → soluzione non scalabile
 - In problemi reali, trasformazioni utili, anche partendo da un numero di attributi ragionevole nello spazio originale, generano un num. di dimensioni nello spazio delle feature, in generale, intrattabile*



SVM non lineari:

“Maledizione della Dimensionalità”

- Generalizziamo, ad esempio, la **trasformazione quadratica** precedente di un vettore **n-dimensionale**:

$$(x_1, \dots, x_n) \xrightarrow{2} (\sqrt{x_1^2 + x_2^2 + \dots + x_n^2}, \dots, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \dots, \sqrt{2}x_{i+1}x_{i+2}, \dots, \sqrt{2}x_{i+1}x_n, \dots, \sqrt{2}x_{n-1}x_n)$$

n(n+1)/2 dimensioni

- il prodotto scalare dei vettori x e z trasformati è il seguente:

$$\phi(x_1, \dots, x_n) \cdot \phi(z_1, \dots, z_n) = \sum_{i=1}^n x_i^2 z_i^2 + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n x_i x_j z_i z_j$$

- num. operazioni = $O(\frac{1}{2} n(n+1))$, n = dim. dello spazio iniziale**
- $\boxed{n} + \boxed{n(n-1)/2} = (2n + n(n-1))/2 = n(2+n-1)/2 = \mathbf{n(n+1)/2}$
- In generale trasformazioni con polinomi di grado g costano $\approx O(n^g/g!)$



Trasformazioni e Costi Computazionali

- Il training, per determinare l'iperpiano di separazione ottimale nello spazio dei dati trasformati, richiede **$r^2/2$ prodotti scalari** (con r = num. delle istanze di training)
- Con polinomi di grado **g** ed **n dimensioni** il costo è **$O(r^2 n^g / 2g!)$**
 - e.g. 100 dim., grado 4, 1000 istanze $\rightarrow 1000^2/2 \times 4421275 \approx \mathbf{2.21 \times 10^{12}}$
- Il costo di classificazione di ogni istanza con l'iperpiano ottimo

$$f(z) = sign \left(\sum_{k \in SV} y_k \alpha_k \phi(x_k^T) \phi(z) + b \right) \text{ con } SV = \text{support vector}$$

è **$O(|SV|^2 n^g / 2g!)$** E.g. come sopra: $\geq 100 \times 4421275 \approx \mathbf{4.42 \times 10^8}$

- Se il risultato di $\phi(x) \cdot \phi(z)$ si ottenessse direttamente da **$x \cdot z$ non occorrerebbero i vettori trasformati $\phi(x)$**



SVM non Lineari: Funzioni Kernel

- Kernel Polinomiali: $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^g$
 - Calcoliamo il kernel con grado $g = 2$ con uno spazio iniziale a 2 dimensioni: $\mathbf{x} = (x_1, x_2)$ e $\mathbf{z} = (z_1, z_2)$.
- $$\begin{aligned}
 (\mathbf{x} \cdot \mathbf{z})^2 &= (x_1 z_1 + x_2 z_2)^2 \\
 &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\
 &= (x_1^2, x_2^2, \sqrt{2}x_1 x_2) \cdot (z_1^2, z_2^2, \sqrt{2}z_1 z_2) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z})
 \end{aligned}$$
- costo computazionale $O(n)$**
 $n = \text{num. dimensioni}$
- costo comput. $O(\frac{1}{2} n(n+1))$**
- $(\mathbf{x} \cdot \mathbf{z})^2$ corrisponde al prodotto scalare nello spazio trasformato dove $\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2)$
 - Kernel Trick:** sostituzione di $\phi(\mathbf{x}) \cdot \phi(\mathbf{z})$ con $K(\mathbf{x}, \mathbf{z})$ nella funzione da minimizzare e di conseguenza nella funzione di classificazione
 - e.g. con $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^g$ con g iperparametro del kernel polinomiale



Confronto Costi Computazionali con Kernel

- Kernel polinomiale di grado g di vettori n -dimensionali

$$K(x, z) = (x^T z)^g = \left(\sum_{k=1}^n x_k z_k \right)^g = \phi(x) \cdot \phi(z)$$

$$\text{num. dim. del vettore trasformato } \phi(x) = \binom{n-1+g}{n-1} = \frac{1}{g!} \prod_{i=0}^{g-1} (n+i)$$

- invece il num. di termini con funzione Kernel è solo $O(n)$

Grado del polinomio di trasformazione	$\phi(\mathbf{x}) \cdot \phi(\mathbf{z})$	$\phi(\mathbf{x})$: num. dimensioni con n dimensioni iniziali	E.g. con 100 dimensioni ed $r^2/2$	Con Kernel Trick	E.g. con 100 dimensioni
2	$(\mathbf{x} \cdot \mathbf{z})^2$	$n(n+1)/2$	$2525 r^2$	$n r^2/2$	$50 r^2$
3	$(\mathbf{x} \cdot \mathbf{z})^3$	$n(n+1)(n+2)/6$	$85850 r^2$	$n r^2/2$	$50 r^2$
4	$(\mathbf{x} \cdot \mathbf{z})^4$	$n(n+1)(n+2)(n+3)/24$	$2.21 \times 10^6 r^2$	$n r^2/2$	$50 r^2$



Come Riconoscere le Funzioni Kernel ?

- Funzione kernel:
 - è una funzione con dominio $R^m \times R^m$ il cui risultato corrisponde al prodotto scalare dei vettori del dominio in un qualche spazio delle feature R^n con $n > m$
 - $K(a, b) = \phi(a) \cdot \phi(b)$
 - non tutte le funzioni sono kernel, e.g. $K(a,b) = (a \cdot b + 1)^3$ è kernel ?
- Come riconoscere se una funzione è una funzione kernel ?
- Teorema di Mercer:
 - $K(x,y)$ è kernel se e solo è una funzione *semi-definita positiva*, i.e., per ogni funzione $f(x)$ il cui $\int f^2(x)dx$ è finito, deve valere la seguente:

$$\int K(x,y)f(x)f(y)dx dy \geq 0$$



SVM: Esempi di Funzioni Kernel

- Kernel non lineari possono separare data set altrimenti non separabili
- Efficienti grazie al kernel trick
- Limite: possono generare modelli affetti da overfitting

Proprietà delle Funzioni Kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})$

Polinomiale: $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + \theta)^g$

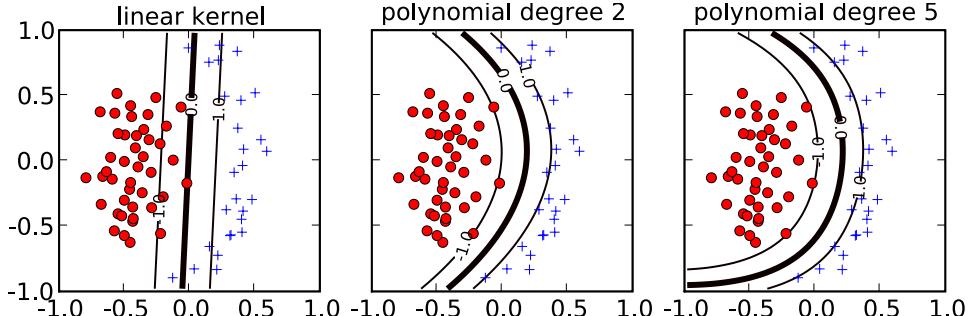
Gaussian Radial Basis: $K(\mathbf{x}, \mathbf{y}) = e^{(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)}$ con $\gamma = \frac{1}{2\sigma^2}$

Sigmoidale: $K(\mathbf{x}, \mathbf{y}) = \tanh(k \mathbf{x} \cdot \mathbf{y} - \delta)$

dove $0 \leq \theta \in R, g \in N, k, \delta \in R$



Iperparametri SVM: Grado del Polinomio



funzione di classificazione con kernel polinomiale

$$f(z) = \text{sign}(w^T z + b) \text{ dove } w^T z = \sum_{k \in SV} y_k \alpha_k K(x_k, z) \quad b = \sum_{k \in SV} y_k - K(x_k, x_k) \quad K(a, b) = (a \cdot b + \theta)^g$$

- nel data set in figura, il decision boundary con un kernel lineare, e.g. polinomio di grado 1, genera dei misclassified
- lasciando invariato l'iperparametro C e aumentando il grado del polinomio, aumenta la curvatura del decision boundary
 - in questo caso migliora la separazione tra le due classi

Gianluca Moro - DISI, Università di Bologna

51



Iperparametri SVM: Kernel Gaussiano

- funzione di classificazione con Kernel Gaussiano (RBF)

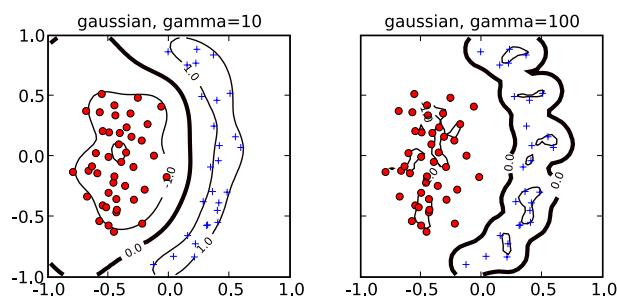
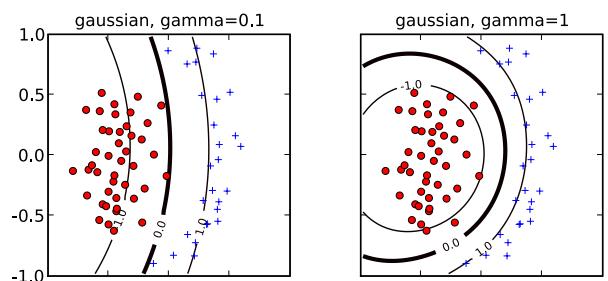
$$f(z) = \text{sign} \left(\sum_{k \in SV} y_k \alpha_k e^{(-\gamma \|x_k - z\|^2)} + b \right)$$

$$\text{con } SV = \text{support vectors}, \quad \gamma = \frac{1}{2\sigma^2}$$

corrisponde ad una somma di gaussiane centrate sui support vectors

- aumentando gamma, con C costante, aumenta la flessibilità del decision boundary

- poiché diminuiscono le devstd delle gaussiane e quindi anche le reciproche influenze date dalla sommatoria → caso estremo ogni punto è decision boundary



Gianluca Moro - DISI, Università di Bologna

52



Iperparametri Kernel Gaussiano: Gamma e C

- $$f(z) = \text{sign} \left(\sum_{k \in SV} y_k \alpha_k e^{(-\gamma \|x_k^T - z\|^2)} + b \right)$$

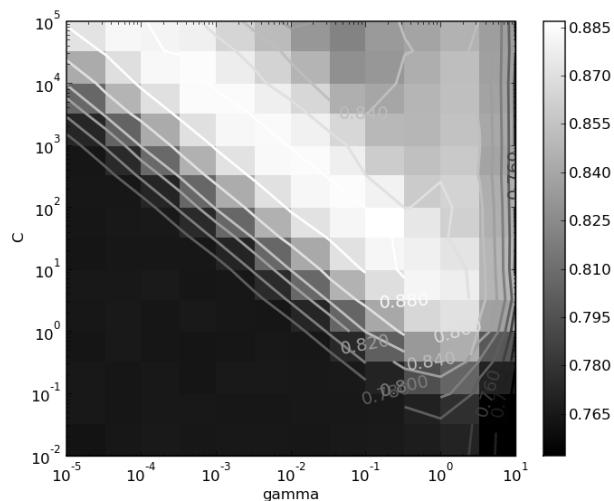
con SV =support vectors, $\gamma = \frac{1}{2\sigma^2}$

- lo spazio di ricerca del modello migliore (i.e. accurato) con kernel polinomiale e gaussiano è bidimensionale

- parametri C e γ oppure C e gamma che variano su scala logaritmica

- proprietà:

- le curve di livello nel grafico collegano combinazioni dei parametri gamma e C che producono accuratezze equivalenti
- infatti C e gamma influenzano entrambi la flessibilità del decision boundary



Bibliografia

- C. J. C. Burges. 1998. A Tutorial on Support Vector Machines for Pattern Recognition
- S. T. Dumais. 1998. Using SVMs for text categorization, IEEE Intelligent Systems, 13(4)
- S. T. Dumais, J. Platt, D. Heckerman and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. *CIKM '98*, pp. 148-155.
- Y. Yang, X. Liu. 1999. A re-examination of text categorization methods. 22nd SIGIR
- Tong Zhang, Frank J. Oles. 2001. Text Categorization Based on Regularized Linear Classification Methods. *Information Retrieval* 4(1): 5-31
- Trevor Hastie, Robert Tibshirani and Jerome Friedman. *Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, New York.
- T. Joachims, *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.
- Fan Li, Yiming Yang. 2003. A Loss Function Analysis for Classification Methods in Text Categorization. *ICML 2003*: 472-479.
- Tie-Yan Liu, Yiming Yang, Hao Wan, et al. 2005. Support Vector Machines Classification with Very Large Scale Taxonomy, *SIGKDD Explorations*, 7(1): 36-43.
- H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. J. C. H. Watkins (2002). Text Classification using String Kernels. *Journal of Machine Learning Research*. 2:419-444.
- ‘Classic’ Reuters-21578 data set: <http://www.daviddlewis.com/resources/testcollections/reuters21578/>

