Data Mining and Machine Learning Lab. Lezione 8 Master in Data Science for Economics, Business and Finance 2018

18.05.18

Marco Frasca

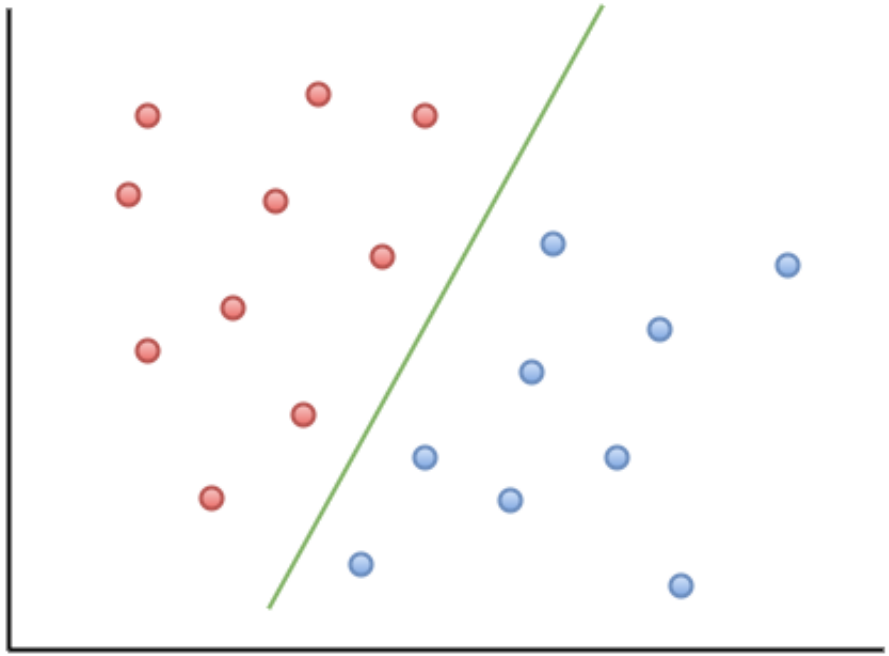
Università degli Studi di Milano

SVM - Richiami

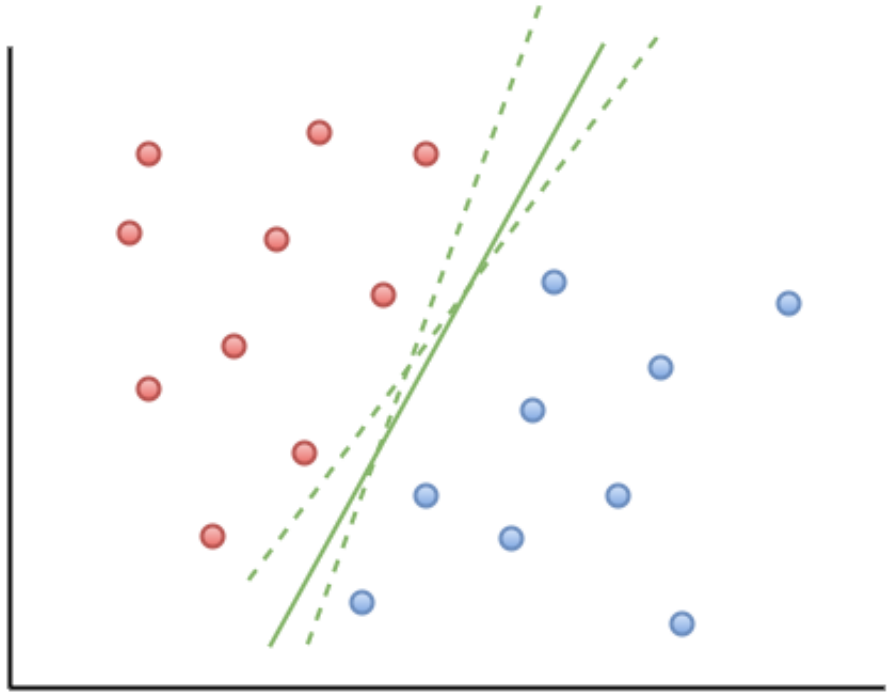
- La Support Vector Machine (SVM) è un classifcatore lineare per dati con etichetta binaria
- **Dati di input**: coppie $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$, con $\boldsymbol{x}_i \in \mathbb{R}^d$ e $y_i \in \{-1, 1\}$, per ogni $i \in \{1, 2, \ldots, n\}$
- Le istanze sono vettori reali d-dimensionali con etichetta binaria (-1, +1)
- Sottoinsieme $S \subset D$ delle istanze di cui è nota l'etichetta
- Sottoinsieme $T := D \setminus S$ delle istanze non etichettate
- **Obiettivo**: Apprendere una classificatore binario $f : \mathbb{R}^d \longrightarrow \{-1, 1\}$ che permetta di classificare le istanze in T (e in generale in $\mathbb{R}^d$)

Esempio di dataset linearmente separabile

Esempio nel piano: le due classi sono separabili con una retta



Ne esistono però diverse. Quale scegliere?



SVM - Richiami

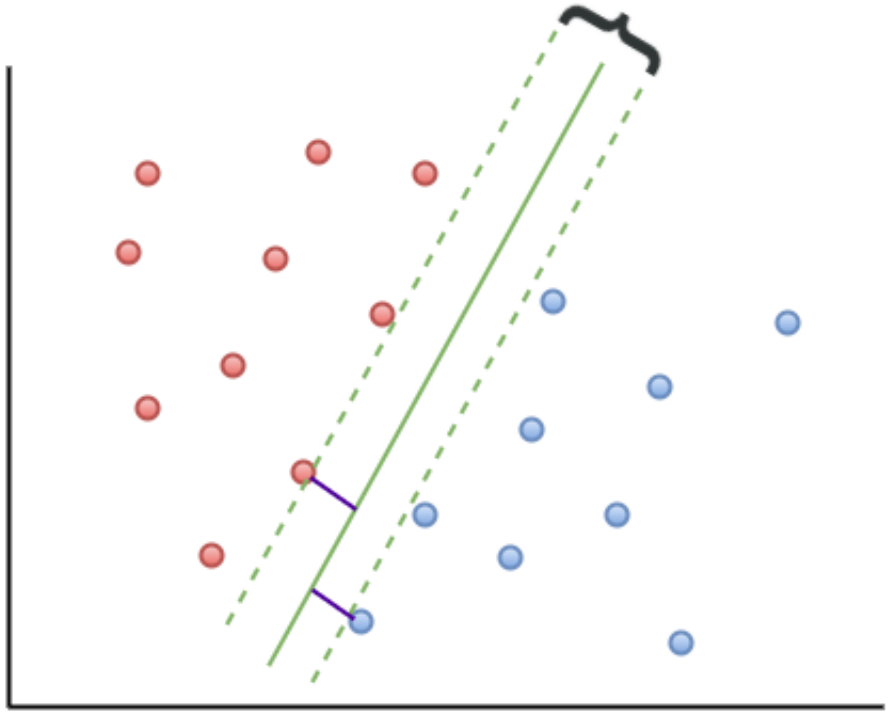
- Una SVM apprende l'iperpiano separatore $\boldsymbol{\omega}^* \in \mathbb{R}^d$ unica soluzione del problema di ottimizzazione convessa

$$\begin{aligned} \min_{\boldsymbol{\omega} \in \mathbb{R}^d} \; & \frac{1}{2}\|\boldsymbol{\omega}\|^2 + \; C \sum_i \xi_i \\ \text{s.t.} \quad & y_i \boldsymbol{\omega}^\top \boldsymbol{x}_i \geq 1 - \xi_i \quad i \in S \\ & \xi_i \geq 0 \qquad\qquad\quad i \in S \; . \end{aligned} \tag{1}$$

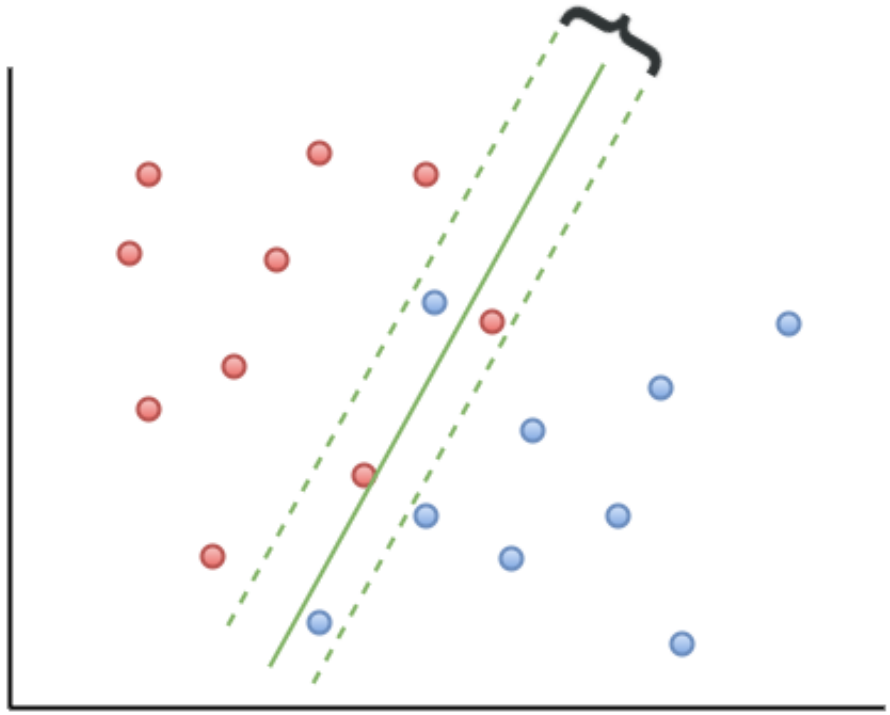
- Già sapete che $\boldsymbol{\omega}^*$ è l'iperpiano separatore a margine massimo (il margine è la distanza tra l'iperpiano e i punti a esso più vicini)
- Le quantità ξ_i le *variabili di slack*, che misurano di quanto ciascun vincolo di margine è violato da una potenziale soluzione $\boldsymbol{\omega}$, cioè $\xi_i = 1 - y_i \boldsymbol{\omega}^T \boldsymbol{x}_i$ (soft margin). Nel caso linearmente separabile tali variabili assumeranno valore 0
- $C > 0$ è una costante che regola quanto penalizzare le violazioni. In genere viene appreso mediante cross validation interna
- Sappiamo che tra i vettori di training ce ne sono alcuni particolari, detti vettori di supporto, che sono gli $\boldsymbol{x}_i$ su cui $\boldsymbol{\omega}^*$ ha margine 1, cioè $y_i {\boldsymbol{\omega}^*}^T \boldsymbol{x}_i = 1$

Esempio di dataset linearmente separabile

Iperpiano separatore a margine massimo

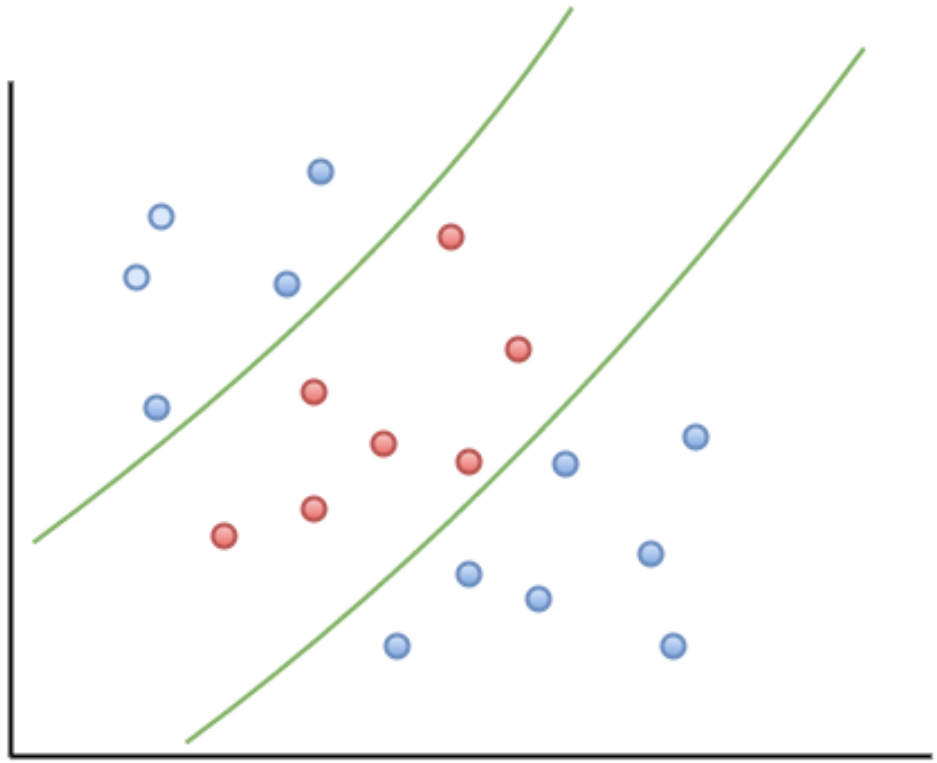


Iperpiano separatore per il problema che include le variabili di slack per i casi non linearmente separabili



Dataset non linearmente separabile

Spesso i dati non sono linearmente separabili, neanche con l'introduzione del margine 'soft'



Kernel trick - gestire la non linearità

- Tuttavia i dati potrebbero essere 'facilmente' separabili se proiettati in spazi oppurtuni di dimensione maggiore, in cui sia nota una metrica che definisca la distanza tra le istanze nel nuovo spazio
- I *kernel* sono funzioni che realizzano il prodotto interno (che è una metrica) $K(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi_K(\boldsymbol{x}), \phi_K(\boldsymbol{z}) \rangle$ di due vettori nello spazio originario $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^d$ in uno spazio $\mathcal{H}_K$ secondo una feature map $\phi_K : \mathbb{R}^d \to \mathcal{H}_K$
- Non ci interessa conoscere ϕ_K, ma K
- Un esempio di kernel è quello radiale o Gaussiano, definito come $K_\gamma(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\frac{1}{2\gamma} \parallel \boldsymbol{x} - \boldsymbol{z} \parallel^2)$
- K_γ tende a creare iperpiani sepratori che classificano allo stesso modo punti vicini secondo la distanza Euclidea nello spazio originale
- $K_\gamma(\boldsymbol{x}, \boldsymbol{z})$ va a zero (decade) man mano che i due vettori $\boldsymbol{x}$ e $\boldsymbol{z}$ si allontanano
- γ regola la velocità con cui il kernel decade

SVM in R

- Anche le Support Vector Machine sono già implementare in R, per esempio nel package `e1071`
- La funzione che apprende una SVM è `svm`, che ha come argomenti principali un oggetto di classe formula e un dataframe contente i dati di input

```
library(e1071); library(ROSE)
data(hacide)
SVM <- svm(formula=cls ~ . , data=hacide.train, scale=FALSE)
print(SVM)
 svm(formula = cls ~ ., data = hacide.train)
 Parameters:
    SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.5
 Number of Support Vectors:  51
```

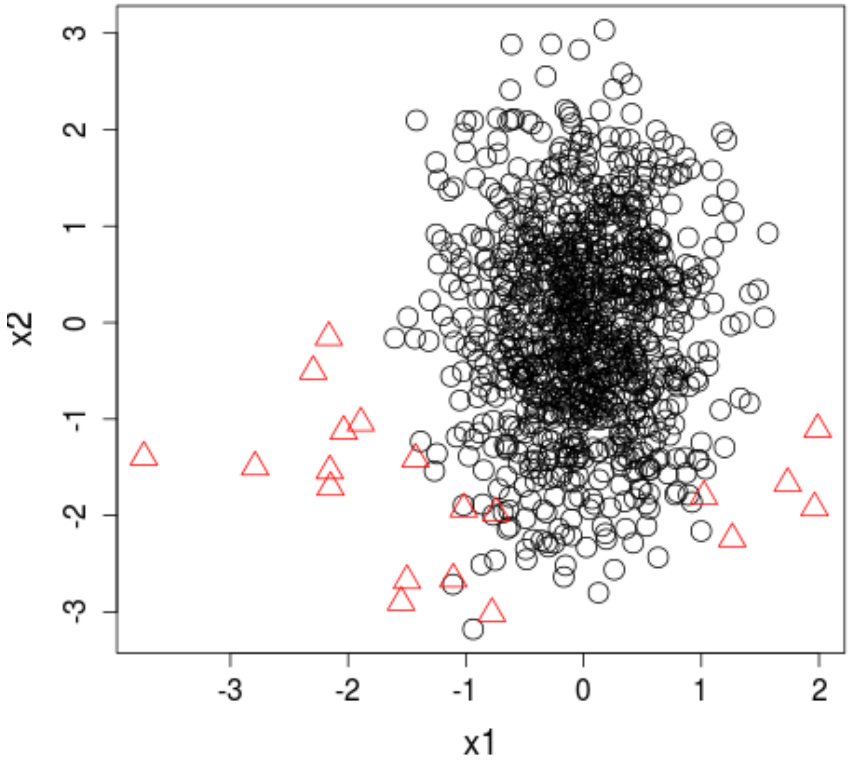
- La descrizione del modello riporta il tipo (classificazione), il kernel utilizzato, i valori attribuiti ai parametri – `cost` equivale a C, `gamma` è il parametro γ del kernel Gaussiano
- Inoltre notiamo che sono stati individuati 51 vettori di supporto

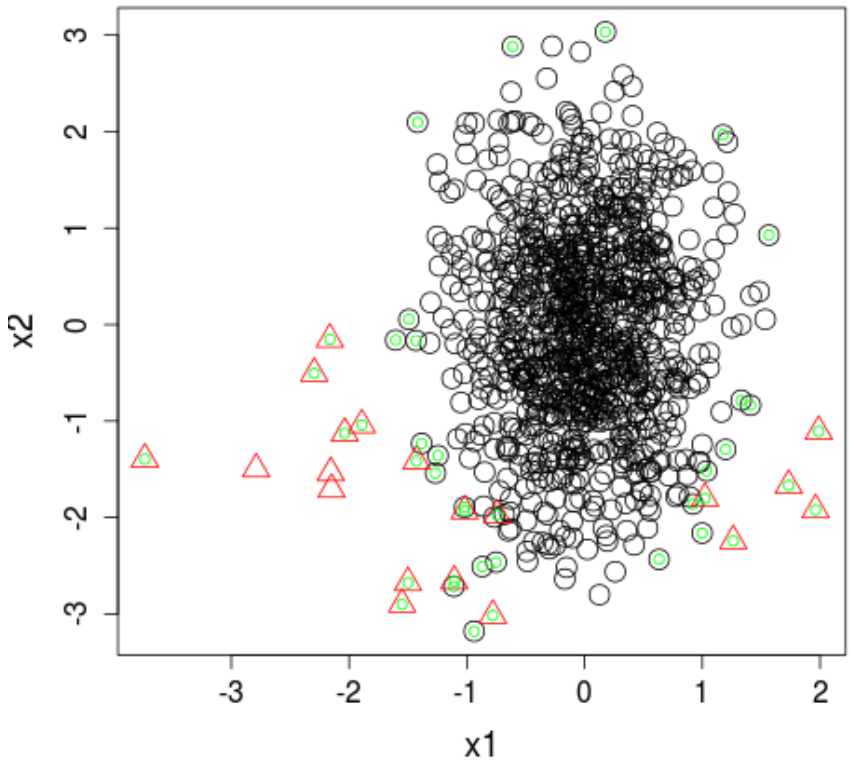
Vettori di Supporto

Tracciamo il grafico dei punti del dataset

I vettori di supporto sono nella componente `SV` del modello

```
points(SVM$SV, col = "green")
```





Predire le istanze del test

- Possiamo predire ancora una volta mediante la funzione `predict`

```
source("perf.R")
pred <- predict(SVM,hacide.test)
perfSVM <- do.perf(hacide.test$cls, pred)
print(round(perfSVM, 3))
    A  Prec   Rec     F
0.988 1.000 0.400 0.571
```

- La prestazione è in linea con i quella degli alberi di decisione cost sensitive
- Esistono delle varianti cost sensitive delle SVM, per esempio [1], che distinguono due variabili costo nel problema (1), una per le istanze positive e una per quelle negative
- La funzione SVM inoltre opera anche la classificazione multiclasse (se la variabile di outcome ha più di die classi). Usa l'approccio AVA

Tuning dei parametri

- La funzione `tune` permette di apprendere i parametri mediante cross validation
- L'argomento `tunecontrol` da settare mediante la funzione `tune.control` permette di specificare alcune caratteristiche dell'ottimizzazione, come il numero di folds (def. 10), la funzione di errore (def. errore assoluto) etc.
- L'oggetto restituito contiene i valori ottimali nella componente `best.parameters`
- In questo caso, non migliora l'errore sul test. Ma questo dipende molto dal dataset

```
svm_tune<-tune(method=svm, cls ~ .,
       data=hacide.train,
       kernel="radial",
       ranges=list(cost=seq(0.1, 10, length=15),
         gamma=seq(10^-2, 10, length=15)),
       scale=FALSE)
print(svm_tune)
   Parameter tuning of 'svm':

    - sampling method: 10-fold cross validation

    - best parameters:
         cost     gamma
     2.928571 0.7235714

   - best performance: 0.008

SVM2<-svm(formula=cls ~ . , data=hacide.train,
    cost=as.numeric(svm_tune$best.parameters["cost"]),
    gamma=as.numeric(svm_tune$best.parameters["gamma"]))
pred <- predict(SVM2,hacide.test)
perfSVMtune <- do.perf(hacide.test$cls, pred)
print(round(perfSVMtune, 3))
    A  Prec   Rec     F
0.988 1.000 0.400 0.571
```

References I

[1] K. Morik, P. Brockhausen, and T. Joachims.
Combining statistical learning with a knowledge-based approach – a case study in intensive care monitoring.
In *International Conference on Machine Learning (ICML)*, pages 268–277, Bled, Slowenien, 1999.