Machine Learning

Università Roma Tre
Dipartimento di Ingegneria

Anno Accademico 2021 - 2022

Esercitazione: SVM (Ex 14)

Sommario

- Scikit-learn e SVM
- SVM e Iris dataset
- Use case: Stock forecasting
- Use case: Sentiment Analysis

Support Vector Machines

- L'algoritmo SVM è impiegato in ambito di classificazione e regressione.
- Ha molti vantaggi tra cui:
  - Efficace in spazi con molte dimensioni (cioè features)
  - Può trattare casi in cui le dimensioni sono maggiori delle istanze
  - È efficiente in termini di spazio di memoria richiesto
- Attenzione:
  - Se le dimensioni sono molto maggiori delle istanze, la scelta della funzione kernel e la regolarizzazione sono fondamentali.
  - SVM non restituisce direttamente probabilità.

Scikit-learn: Support Vector Machines

- I dati in input supportati in scikit-learn sono sia *dense* (es. numpy.ndarray, numpy.asarray) sia sparsi (qualsiasi scipy.sparse)

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
SVC()

>>> clf.predict([[2., 2.]])
array([1])

>>> # support vectors
>>> clf.support_vectors_
array([[0., 0.],
       [1., 1.]])
>>> # indici dei support vectors
>>> clf.support_
array([0, 1]...)
>>> # numero dei support vectors per ogni classe
>>> clf.n_support_
array([1, 1]...)
```

Scikit-learn: Support Vector Machines

- Esempio IRIS dataset:

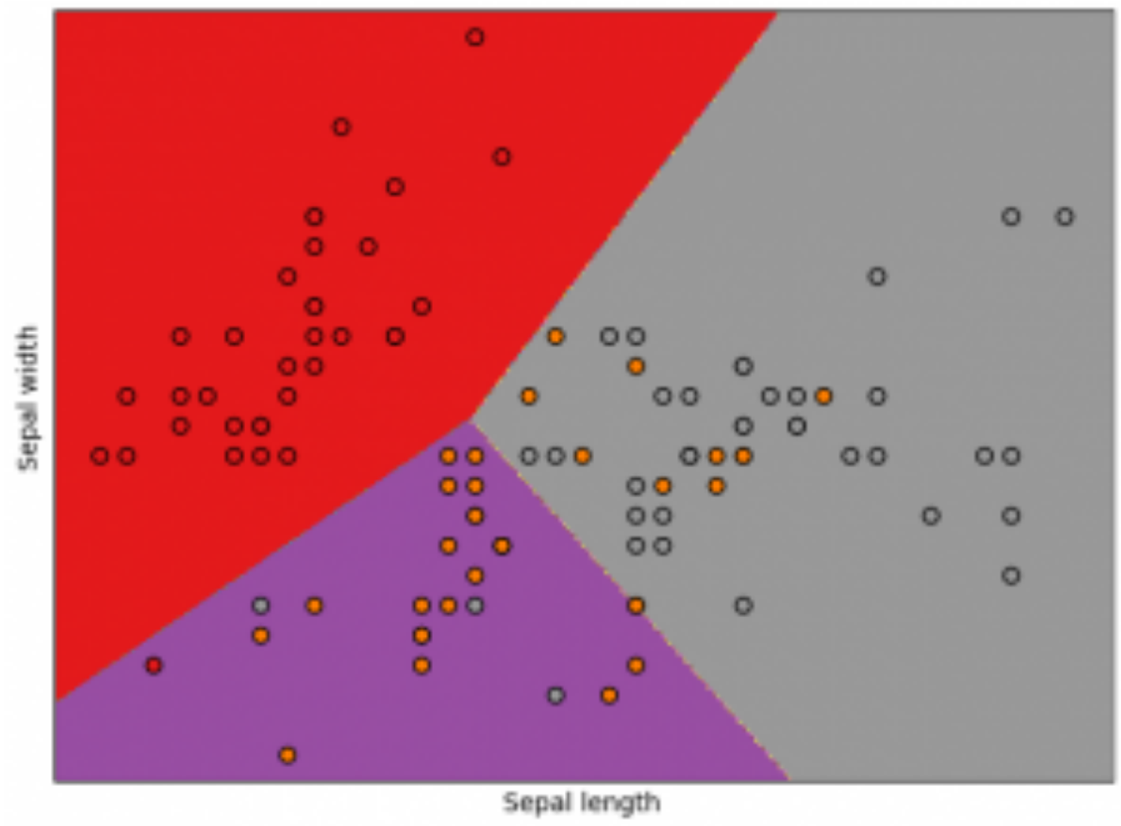
```
# carico il dataset IRIS
iris = load_iris()

# uso solo le prime due features
X = iris.data[:, :2]
Y = iris.target

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=0)

scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)

# equivale a SVC(kernel="linear")
svm = LinearSVC()
svm.fit(X_train_std, Y_train)

print("Accuracy Train Set:", svm.score(X_train_std, Y_train))
print("Accuracy Test Set:", svm.score(X_test_std, Y_test))

>> Accuracy Train Set: 0.8285714285714286
>> Accuracy Test Set: 0.6888888888888889
```

- Cosa possiamo dire?

Scikit-learn: Support Vector Machines

```
>> Accuracy Train Set: 0.8285714285714286
>> Accuracy Test Set: 0.6888888888888889
```

- Cosa possiamo dire?
  - Il modello soffre di overfitting.



- Esercizio: prova ad impiegare tutte le features del dataset.

Scikit-learn: Support Vector Machines

- Esercizio: prova ad impiegare tutte le features del dataset.

```
>> Accuracy Train Set: 0.9428571428571428
>> Accuracy Test Set: 0.9555555555555556
```

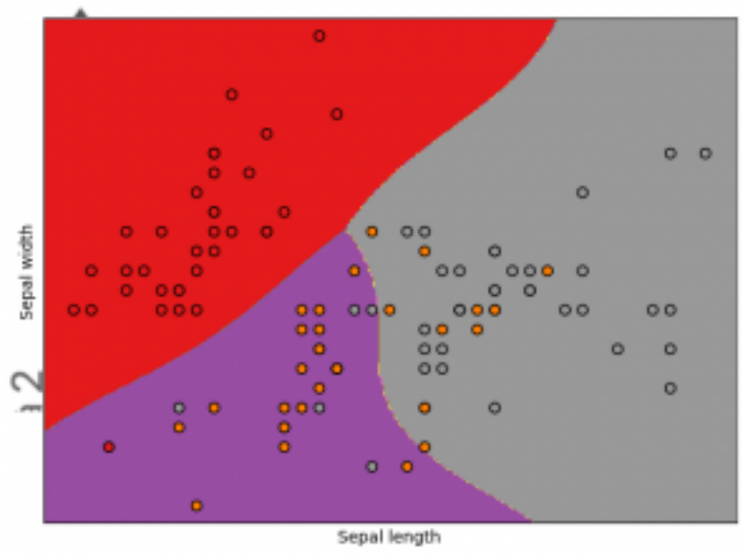
- Esercizio: cambia il parametro *kernel* di SVC() e testa le altre funzioni oltre alla **linear** cioè **rbf**, **sigmoid** e **poly** impiegando sempre 2 features.

Scikit-learn: Support Vector Machines

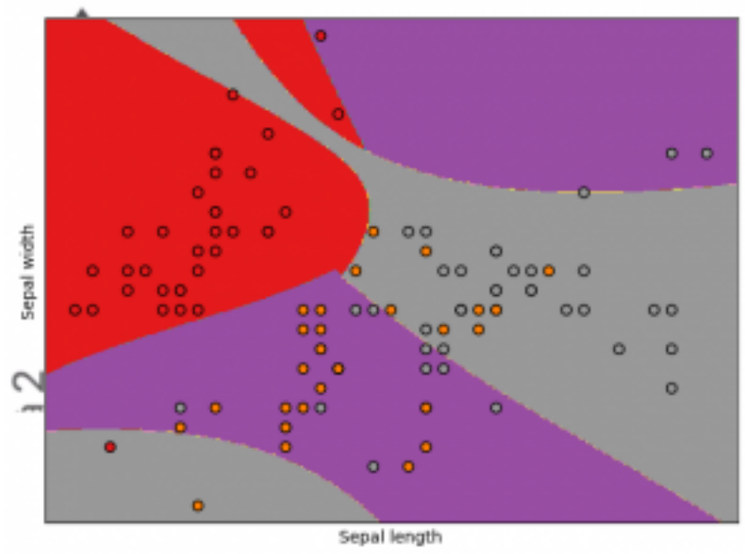
- Esercizio: prova ad impiegare tutte le features del dataset.

```
>> Accuracy Train Set: 0.9428571428571428
>> Accuracy Test Set: 0.9555555555555556
```

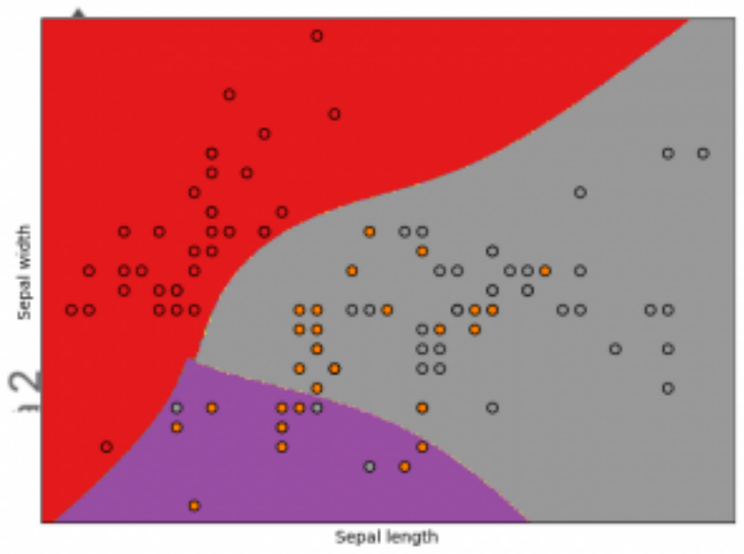
- Esercizio: cambia il parametro *kernel* di SVC() e testa le altre funzioni oltre alla **linear** cioè **rbf**, **sigmoid** e **poly** impiegando sempre 2 features.




```
Accuracy Train Set: 0.81
Accuracy Test Set: 0.78
```




```
Accuracy Train Set: 0.72
Accuracy Test Set: 0.8
```




```
Accuracy Train Set: 0.76
Accuracy Test Set: 0.67
```

Stock forecasting

- Alcuni servizi web rendono disponibili gli andamenti di titoli azionari via APIs, es:
  - Open: Starting price at which a stock is traded in a day.
  - Close: Closing price.
  - High: The highest price of equity symbol in a day.
  - Low: The lowest price of the share in a day
  - VWAP: Volume weighted average price
  - Volume: Total volume of stocks traded on a particular day.
- I dati possono essere interpretati come *time series*, cioè sequenze di valori ordinati temporalmente.
  - Per approfondimenti: https://www.kaggle.com/code/parulpandey/getting-started-with-time-series-using-pandas/notebook
- Il task della stock price forecasting è predire il valore futuro (es. intraday, giornalieri, mensile, etc). di un titolo in base ai valori passati.

Esercitazione: stock forecasting

- Al seguente indirizzo trovi i dati storici del titolo RELIANCE:
  - https://storage.googleapis.com/kaggle-forum-message-attachments/894813/16059/RELIANCE.csv
- Possiamo impiegare l'istanza attuale come input e tentare di fare predizione sul comprare (+1) oppure no (0).
- In ambito azionario è utile definire nuove features che combinano quelle attuali, es. Open-Close o High-Low:

```
df['Open-Close'] = df.Open - df.Close
```

- La variabile target puoi essere approssimare nel seguente modo:

```
y = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)
```

- Il ritorno cumulato può essere ottenuto nel seguente modo:

```
df['Return'] = df.Close.pct_change() # variazione percentuale rispetto al prec
df['Strategy_Return'] = df.Return * df.Predicted_Signal.shift(1)
df['Cum_Ret'] = df['Return'].cumsum()
df['Cum_Strategy'] = df['Strategy_Return'].cumsum()
```

- Esercizio: impiega l'algoritmo SVM per la predizione e valuta l'accuratezza.

Esercitazione: stock forecasting

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')

import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('RELIANCE.csv')
df.index = pd.to_datetime(df['Date'])
df = df.drop(['Date'], axis='columns')

... (completa)
```

Esercitazione: stock forecasting

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')

import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('RELIANCE.csv')
df.index = pd.to_datetime(df['Date'])
df = df.drop(['Date'], axis='columns')

df['Open-Close'] = df.Open - df.Close
df['High-Low'] = df.High - df.Low

# per ora uso solo 2 valori
X = df[['Open-Close', 'High-Low']]

y = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)
>> [1 1 1 ... 1 0 0]

... (segue)
```

Esercitazione: stock forecasting

```
split_percentage = 0.8
split = int(split_percentage*len(df))

# Train data set
X_train = X[:split]
y_train = y[:split]

# Test data set
X_test = X[split:]
y_test = y[split:]

cls = SVC().fit(X_train, y_train)
```

```
df['Predicted_Signal'] = cls.predict(X)

df['Return'] = df.Close.pct_change()
df['Strategy_Return'] = df.Return *df.Predicted_Signal.shift(1)
df['Cum_Ret'] = df['Return'].cumsum()
df['Cum_Strategy'] = df['Strategy_Return'].cumsum()

import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(df['Cum_Ret'],color='red')
plt.plot(df['Cum_Strategy'],color='blue')
```

Esercitazione: stock forecasting

Date	%Deliverble	Open-Close	High-Low	Predicted_Signal	Return
2000-01-03	NaN	-14.20	14.20	1	NaN
2000-01-04	NaN	-13.45	20.55	1	0.080056
2000-01-05	NaN	-25.85	31.25	1	0.039176
2000-01-06	NaN	-5.35	11.70	1	0.041947
2000-01-07	NaN	-19.55	24.90	1	0.068626
...	...	...	...	...	...
2020-05-22	0.2339	20.25	31.50	1	-0.006730
2020-05-26	0.4852	24.10	33.40	1	-0.005239
2020-05-27	0.3964	-14.55	42.00	1	0.015098
2020-05-28	0.4523	-17.25	30.75	1	0.018470
2020-05-29	0.5572	3.60	19.35	1	-0.005332

Date	Strategy_Return	Cum_Ret	Cum_Strategy
2000-01-03	NaN	NaN	NaN
2000-01-04	0.080056	0.080056	0.080056
2000-01-05	0.039176	0.119232	0.119232
2000-01-06	0.041947	0.161179	0.161179
2000-01-07	0.068626	0.229804	0.229804
...	...	...	...
2020-05-22	-0.006730	3.375374	4.574602
2020-05-26	-0.005239	3.370135	4.569363
2020-05-27	0.015098	3.385233	4.584461
2020-05-28	0.018470	3.403704	4.602932
2020-05-29	-0.005332	3.398372	4.597600

Esercitazione: stock forecasting

- L'algoritmo genera un ritorno del 18.87% in un 1 anno, rispetto al 5.97% del titolo azionario.

- La funzione accuracy_score() restituisce una accuracy del 62.07% sul train set e 50.67 sul test set.
- Esercizi: (1) crea il target value a distanza di più giorni dall'istanza corrente; (2) usa gli ultimi 15 valori Close come istanza di input per predire il successivo; (3) impiega altri kernel (es. rbf).

Esercitazione: Sentiment Analysis

- Tecnica molto popolare per classificare brani di testo, micropost o frasi in linguaggio naturale in base al sentimento (es. positivo, negativo, neutro).
- Supponiamo di impiegare le review di film, es:
  - http://www.cs.cornell.edu/people/pabo/movie-review-data/
  - Movie-review data for use in sentiment-analysis experiments. Available are collections of movie-review documents labeled with respect to their overall *sentiment polarity* (positive or negative) or *subjective rating* (e.g., "two and a half stars") and sentences labeled with respect to their *subjectivity status* (subjective or objective) or *polarity*

```
import pandas as pd

trainData = pd.read_csv("https://raw.githubusercontent.com/Vasistareddy/
sentiment_analysis/master/data/train.csv")
testData = pd.read_csv("https://raw.githubusercontent.com/Vasistareddy/
sentiment_analysis/master/data/test.csv")

trainData.sample(frac=1).head(5) # shuffle the df and pick first 5
      Content                                                  Label
56    jarvis cocker of pulp once said that he wrote ...   pos
1467  david spade has a snide , sarcastic sense of h...   neg
392   upon arriving at the theater during the openin...   pos
104   every once in a while , a film sneaks up on me...   pos
1035  susan granger's review of " american outlaws "...   neg
```

Esercitazione: Sentiment Analysis

- Per impiegare il testo come input agli algoritmi di ML spesso si effettua una pipeline di processamento per trasformare parole o frasi in vettori numerici.
  - Per dettagli: https://medium.com/@vasista/preparing-the-text-data-with-scikit-learn-b31a3df567e e https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```
from sklearn.feature_extraction.text import TfidfVectorizer

# ignora i termini che compaiono in meno di 5 documenti
# e i termini che compaiono in > 80% dei documenti;
# abilita l'inverse document frequency per pesare i termini
vectorizer = TfidfVectorizer(min_df = 5,
                             max_df = 0.8,
                             sublinear_tf = True,
                             use_idf = True)
train_vectors = vectorizer.fit_transform(trainData['Content'])
test_vectors = vectorizer.transform(testData['Content'])
```

- Esercizio: completa il codice impiegando SVM lineare e valutane l'accuratezza. Testa la predizione su review di Amazon.

Esercitazione: Sentiment Analysis

```
from sklearn import svm
from sklearn.metrics import classification_report

classifier_linear = svm.SVC(kernel='linear')
classifier_linear.fit(train_vectors, trainData['Label'])
prediction_linear = classifier_linear.predict(test_vectors)
time_linear_train = t1-t0
time_linear_predict = t2-t1

report = classification_report(testData['Label'], prediction_linear,
output_dict=True)
print('positive: ', report['pos'])
print('negative: ', report['neg'])

positive:  {'precision': 0.9191919191919192, 'recall': 0.91, 'f1-score':
0.9145728643216081, 'support': 100}
negative:  {'precision': 0.910891089108911, 'recall': 0.92, 'f1-score':
0.9154228855721394, 'support': 100}


review = """Very good picture and sound, very glad I chose this unit"""
review_vector = vectorizer.transform([review]) # vectorizing
print(classifier_linear.predict(review_vector))
```

Testi di Riferimento

- Andreas C. Müller, Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media 2016
- Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media 2017
- https://www.kaggle.com/code/parulpandey/getting-started-with-time-series-using-pandas/notebook
- https://medium.com/@vasista/preparing-the-text-data-with-scikit-learn-b31a3df567e
- https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
- Tutorial: https://www.geeksforgeeks.org/predicting-stock-price-direction-using-support-vector-machines/?ref=rp
- Tutorial: https://medium.com/@vasista/sentiment-analysis-using-svm-338d418e3ff1