UNIVERSITÀ DEGLI STUDI DI NAPOLI

FEDERICO II

Confronto tra SVM e KNN applicando i test di significatività statistica discussi in *"The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing"*

Prof.ssa Anna Corazza

De Lucia Gianluca (N97000311)

A.A. 2019/2020

Sommario

1. Introduzione

Con questo esperimento vogliamo mettere a confronto due algoritmi di classificazione quali Knn e SVM.

Svm (Space Vector Machine): divide uno spazio vettoriale con un iperpiano, definendo lo spazio delle classi.

Knn (K nearest neighbor): tramite k vicini e vedendo la maggioranza di essi a quale classe appartiengono, riesce a classificare un nuovo documento.

Per valutare e confrontare due algoritmi come questi utilizziamo il test di significatività statistica per la NPL [1].

Ma come si può conoscere la distribuzione statistica dei test? Una possibilità è quella di applicare prove progettate per valutare la distribuzione di un campione di osservazioni.

Esistono vari test, che si dividono in parametrici e non parametrici.

PARAMETRICI:

- Ad esempio, il test di Shapiro-Wilk (Shapiro e Wilk, 1965) verifica l'ipotesi nulla che un campione provenga da una popolazione normalmente distribuita;
- il test di Kolmogorov-Smirnov quantifica la distanza tra la funzione di distribuzione empirica del campione e la funzione di distribuzione cumulativa della distribuzione di riferimento;
- il test Anderson-Darling (Anderson e Darling, 1954) verifica se un dato campione di dati viene estratto da una data distribuzione di probabilità;
- il t-test Questo test valuta se la popolazione media di due serie di misurazioni differisce l'una dall'altra e si basa sul presupposto che entrambi i campioni provengano da una distribuzione normale (Fisher, 1937). calcola il numero medio di previsioni corrette per esempio di input. i punteggi sono significativamente più grandi di zero, il che può servire da indicazione che un parser è migliore dell'altro.

NON PARAMETRICI:

Quando la distribuzione statistica del test è sconosciuta, è necessario utilizzare test di significatività non parametrici.

Ci sono due famiglie dei non parametrici:

- La prima famiglia è composta da test che non tengono conto dei valori effettivi delle misure di valutazione.
- La seconda famiglia considera i valori delle misure: verifica ripetutamente il campione dai dati del test e stima il valore p in base ai valori statistici del test nei campioni.

Ci riferiamo alla prima famiglia come famiglia di test senza campionamento e alla seconda come famiglia di test basati su campionamento. Sono i segurnti test:

- McNemar: Questo test è progettato per osservazioni nominali accoppiate (etichette binarie). Il test viene applicato a una tabella di contingenza 2×2, che tabula i risultati di due algoritmi su un campione di n esempi. L'ipotesi nulla per questo test afferma che la probabilità marginale per ciascun risultato (etichetta uno o etichetta due) è la stessa per entrambi gli algoritmi. Cioè, quando applichiamo entrambi gli algoritmi sugli stessi dati, ci aspetteremmo che siano corretti / errati sulla stessa proporzione di elementi;

- Wilcoxon: Questo test viene utilizzato quando si confrontano due campioni corrispondenti. La sua ipotesi nulla è che le differenze seguano una distribuzione simmetrica attorno allo zero;

- Permutation: Questo test stima la distribuzione delle statistiche di test sotto l'ipotesi nulla calcolando i valori di questa statistica sotto tutte le possibili etichettature (permutazioni) del set di test. Il valore p (bilaterale) del test viene calcolato come la proporzione di queste permutazioni in cui la differenza assoluta era maggiore o uguale al valore assoluto della differenza nell'output dell'algoritmo;

- Bootstrap: Questo test è molto simile alla randomizzazione approssimativa del test di permutazione, con la differenza che il campionamento viene effettuato con sostituzioni (ad esempio, un esempio dai dati del test originale può apparire più di una volta in un campione). L'idea di bootstrap è di usare i campioni come popolazioni surrogate, allo scopo di approssimare la distribuzione campionaria della statistica. Il valore p viene calcolato in modo simile al test di permutazione. Il test è meno efficace per piccoli set di test, poiché presuppone che la distribuzione del set di test non si discosti troppo dalla distribuzione della popolazione.

Quando il test risulta significante possiamo affermare con certezza la superiorità di un algoritmo rispetto all’altro.

2. Esperimento

Ho raccolto in due file i valori di knn di svm applicati a 4 dataset differenti [2].

Secondo i test svolti:

DataSet	Accuratezza KNN (%)	Accuratezza SVM (%)
Data 1	92.5925	98.1481
Data 2	87.0370	90.7407
Data 3	81.8182	84.0909

Notiamo subito che SVM risulta essere più accurato di KNN.

Ho applicato il test di significatività su questi valori di entrambi gli algoritmi utilizzando uno script python allegato all'articolo scientifico in considerazione [2].

Per valutare la prestazione di due algoritmi, A e B, su un set di dati X, si confrontano usando una misura di valutazione M. Indichiamo M (ALG, X) come valore della misura di valutazione M quando l'algoritmo ALG viene applicato al set di dati X.
Senza perdita di generalità, assumiamo che valori più alti della misura siano migliori.
Definiamo la differenza nelle prestazioni tra i due algoritmi secondo la misura M sul set di dati X come:

$$\delta (X) = M (A, X) - M (B, X).$$

Consideriamo $\delta (X)$ come la nostra statistica di prova.
Usando questa notazione formuliamo il seguente problema di verifica dell'ipotesi statistica:

$$H0: \delta (X) \leq 0$$
$$H1: \delta (X) > 0$$

Per decidere se rifiutare o meno l'ipotesi nulla, che sta raggiungendo la conclusione che $\delta (X)$ è effettivamente maggiore di 0, di solito calcoliamo un *p-value* per il test. Il valore p è definito come la probabilità, sotto l'ipotesi nulla H0, di ottenere un risultato uguale o più estremo di quanto effettivamente osservato.

Indicando con α il valore di soglia da utilizzare per considerare la significatività e quindi il livello di significatività e con *p-value* il valore:

$$P r(\delta(X) \geq \delta_{observed} | H_0)$$

Quando il valore $p < \alpha$ allora l'ipotesi nulla viene respinta.

Applicazione del Test di significatività statistica con $\alpha = 0.5$:

Tipo di Test	Significativo (Si/No)	P-Value
T-test	SI	0.028013616053
Wilcoxon	SI	0.108809430041
Permutation	SI	0.124487551245
Bootstrap	SI	0.0

Applicazione del Test di significatività statistica con $\alpha = 0.8$:

Tipo di Test	Significativo (Si/No)	P-Value
T-test	SI	0.028013616053
Wilcoxon	SI	0.108809430041
Permutation	SI	0.122587741226
Bootstrap	SI	0.0

È cambiato solo il p-value del test di Permutation.

McNemar non passa il test poiché accetta solo etichettature binarie.
Bootstrap per funzionare al meglio dovrebbe essere eseguiti su molti più valori per poter essere accertato.

Per quanto riguarda i test di Shapiro-Wilk, di Kolmogorov-Smirnov e di Anderson-Darling, non avendo una distribuzione normale dei dati, il test risulta essere non significativo in tutti i casi.

A conclusione di tutto possiamo affermare che l'algoritmo SVM è migliore rispetto a KNN in maniera significativa in accordo con il t-test, Wilcoxon test, Permutation test e Bootstrap test.

3. Riferimenti

[1]: “The Hitchhiker’s Guide to Testing Statistical Significance in Natural Language Processing” Rotem Dror Gili Baumer Segev Shlomov Roi Reichart Faculty of Industrial Engineering and Management, Technion, IIT

[2]: http://archive.ics.uci.edu/ml/datasets.php

[3]: https://github.com/rtmdrr/ testSignificanceNLP.git

[4]: https://gitlab.com/gigernau/cvs-classification.git

4. Appendice

Il codice utilizzato per valutare l'accuratezza dei due algoritmi è il seguente (python 3):

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import accuracy_score

# Importing the dataset
dataset = pd.read_csv('wine_data.csv')
X = dataset.iloc[:, 1:13].values
y = dataset.iloc[:, 0].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

"""#linear discriminant analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components=5)
X_train = lda.fit_transform(X_train,y_train)
X_test = lda.fit_transform(X_test,y_test)
#explained_variance=pca.explained_variance_ratio_
          or

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train = pca.fit_transform(X_train)
X_test = pca.fit_transform(X_test)
#explained_variance=pca.explained_variance_ratio_"""


# Fitting KNN to the Training set

from sklearn.neighbors import KNeighborsClassifier


classifier = KNeighborsClassifier(n_neighbors=5)
trained_model=classifier.fit(X_train,y_train)
trained_model.fit(X_train,y_train )

# Predicting the Test set results

y_pred = classifier.predict(X_test)

```

```
# Making the Confusion Matrix

from sklearn.metrics import confusion_matrix

cm_KNN = confusion_matrix(y_test, y_pred)
print(cm_KNN)
print("Accuracy score of train KNN")
print(accuracy_score(y_train, trained_model.predict(X_train))*100)

print("Accuracy score of test KNN")
print(accuracy_score(y_test, y_pred)*100)
TotalAmount = accuracy_score(y_test, y_pred)*100




with open("resA.txt", "a+") as text_file:
    print(f"{TotalAmount}", file=text_file)


# # Fitting SVM to the Training set
from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random_state = 0)
trained_model=classifier.fit(X_train,y_train)
trained_model.fit(X_train,y_train )


# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm_SVM = confusion_matrix(y_test, y_pred)
print(cm_SVM)
print("Accuracy score of train SVM")
print(accuracy_score(y_train, trained_model.predict(X_train))*100)

print("Accuracy score of test SVM")
print(accuracy_score(y_test, y_pred)*100)
TotalAmount2 = accuracy_score(y_test, y_pred)*100

with open("resB.txt", "a+") as text_file2:
    print(f"{TotalAmount2}", file=text_file2)

```