Instituto Superior Técnico

Aprendizagem Automática

Laboratório 6 - Evaluation and Generalization

Autores:

Diogo Moura - nº 86976

Diogo Alves - nº 86980

Turno:

Terça 17h-18h30m

10 de Dezembro de 2019

Trabalho Prático

Métodos utilizados e código

Os seguintes métodos foram utilizados:

- SVM polynomial
- SVM RBF
- Nearest Neighbours
- Multinomial Naive Bayes
- Gaussian Naive Bayes
- Decision Tree

A precisão dos métodos varia e é analisada na secção que se segue.

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow.keras as keras
import sklearn.metrics as skmetrics
import sklearn.linear_model as lm
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import sklearn.naive_bayes as nb
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
import math
import sklearn.model_selection as ms
from sklearn.metrics import balanced_accuracy_score

import warnings

warnings.filterwarnings('ignore', 'Solver terminated early.*')

for dataset_number in range(1,3):
    if dataset_number == 1:
        print("dataset1:")
        dataset_xtrain=np.load('dataset1_xtrain.npy')
        dataset_ytrain=np.load('dataset1_ytrain.npy')
        dataset_xtest=np.load('dataset1_xtest.npy')
        dataset_ytest=np.load('dataset1_ytest.npy')
    elif dataset_number == 2:
        print("dataset2:")
```

```
    dataset_xtrain=np.load('dataset2_xtrain.npy')
    dataset_ytrain=np.load('dataset2_ytrain.npy')
    dataset_xtest=np.load('dataset2_xtest.npy')
    dataset_ytest=np.load('dataset2_ytest.npy')

X_train, X_validation, y_train, y_validation = ms.train_test_split
(dataset_xtrain, dataset_ytrain, test_size=0.3)

num_tests = 100
y_predict=np.zeros((y_validation.size,1))
scores=np.zeros(num_tests)
c0=0
for j in range(num_tests):
    #fit the SVM to the data with different polynomial orders j
    clf = SVC(kernel='poly',max_iter=100000,degree=j,coef0=c0,gamma='auto')
    clf.fit(X_train,np.ravel(y_train))
    y_predict=clf.predict(X_validation)
    scores[j]=accuracy_score(y_validation,y_predict)
    clf=None
max_degree = np.argmax(scores)
clf = SVC(kernel='poly',max_iter=100000,degree=max_degree,coef0=c0,gamma='auto')
clf.fit(dataset_xtrain,np.ravel(dataset_ytrain))
y_predict=clf.predict(dataset_xtest)
score=accuracy_score(dataset_ytest,y_predict)
balanced_score=balanced_accuracy_score(dataset_ytest,y_predict)
confusion_matrix = skmetrics.confusion_matrix(dataset_ytest,y_predict)
print("SVM Polynomial with degree",max_degree,"\t Score: ",score,"\t Balanced Score: ",
balanced_score, "\t Confusion Matrix: ",confusion_matrix[0], confusion_matrix[1])
clf = None

gammas=np.logspace(start=-8,stop=1,num=num_tests)#create various values of gamma to test
for j in range(num_tests):
    #fit the SVM to the data with different polynomial orders j
    clf = SVC(kernel='rbf',max_iter=100000,gamma=gammas[j])
    clf.fit(X_train,np.ravel(y_train))
    y_predict=clf.predict(X_validation)
    scores[j]=accuracy_score(y_validation,y_predict)
    clf=None
max_gamma_index = np.argmax(scores)
clf = SVC(kernel='rbf',max_iter=100000,gamma=gammas[max_gamma_index])
clf.fit(dataset_xtrain,np.ravel(dataset_ytrain))
```

```
y_predict=clf.predict(dataset_xtest)
score=accuracy_score(dataset_ytest,y_predict)
balanced_score=balanced_accuracy_score(dataset_ytest,y_predict)
confusion_matrix = skmetrics.confusion_matrix(dataset_ytest,y_predict)
print("SVM RBF with gamma",gammas[max_gamma_index],"\t Score: "
,score,"\t Balanced Score: ",balanced_score, "\t Confusion Matrix: "
,confusion_matrix[0], confusion_matrix[1])
clf = None

for j in range(1,num_tests):
    #fit the SVM to the data with different polynomial orders j
    clf = KNeighborsClassifier(j)
    clf.fit(X_train,np.ravel(y_train))
    y_predict=clf.predict(X_validation)
    scores[j]=accuracy_score(y_validation,y_predict)
    clf=None
max_neighbours = np.argmax(scores)
clf = KNeighborsClassifier(max_neighbours)
clf.fit(dataset_xtrain,np.ravel(dataset_ytrain))
y_predict=clf.predict(dataset_xtest)
score=accuracy_score(dataset_ytest,y_predict)
balanced_score=balanced_accuracy_score(dataset_ytest,y_predict)
confusion_matrix = skmetrics.confusion_matrix(dataset_ytest,y_predict)
print("Nearest Neighbour with neighbours ",max_neighbours,"\t Score: ",score,
"\t Balanced Score: ",balanced_score, "\t Confusion Matrix: "
,confusion_matrix[0], confusion_matrix[1])
clf = None


clf = nb.MultinomialNB(alpha = 1, fit_prior = False)
clf.fit(dataset_xtrain,np.ravel(dataset_ytrain))
y_predict=clf.predict(dataset_xtest)
score=accuracy_score(dataset_ytest,y_predict)
balanced_score=balanced_accuracy_score(dataset_ytest,y_predict)
confusion_matrix = skmetrics.confusion_matrix(dataset_ytest,y_predict)
print("Multinomial Naive Bayes \t Score: ",score,"\t Balanced Score: "
,balanced_score, "\t Confusion Matrix: ",confusion_matrix[0], confusion_matrix[1])
clf=None

clf = nb.GaussianNB()
clf.fit(dataset_xtrain,np.ravel(dataset_ytrain))
```

```
y_predict=clf.predict(dataset_xtest)
score=accuracy_score(dataset_ytest,y_predict)
balanced_score=balanced_accuracy_score(dataset_ytest,y_predict)
confusion_matrix = skmetrics.confusion_matrix(dataset_ytest,y_predict)
print("Gaussian Naive Bayes \t Score: ",score,"\t Balanced Score: ",
balanced_score, "\t Confusion Matrix: ",confusion_matrix[0], confusion_matrix[1])
clf=None

clf = tree.DecisionTreeClassifier()
clf.fit(dataset_xtrain,np.ravel(dataset_ytrain))
y_predict=clf.predict(dataset_xtest)
score=accuracy_score(dataset_ytest,y_predict)
balanced_score=balanced_accuracy_score(dataset_ytest,y_predict)
confusion_matrix = skmetrics.confusion_matrix(dataset_ytest,y_predict)
print("Decision Tree \t Score: ",score,"\t Balanced Score: ",balanced_score,
"\t Confusion Matrix: ",confusion_matrix[0], confusion_matrix[1])
clf = None
```

Para os classificadores em que não é necessário estimar hiperparâmetros (Multinomial Naive Bayes, Gaussian Naive Bayes e Decision Tree), foi utilizado o conjunto de treino para treinar o classificador e o conjunto de teste para avaliar a sua performance. Para os restantes classificadores (SVM polinomial, SVM RBF e Nearest Neighbour), em que é preciso estimar hiperparametros, o conjunto de treino foi dividido em dois conjuntos disjuntos: (X_train, y_train) e (X_validation, y_validation), em que o conjunto de validação tem 30% do tamanho do conjunto original. Depois, foi usado o conjunto (X_train, y_train) para treinar o classificador e o conjunto (X_validation, y_validation) para testar a performance para cada valor do hiperparâmetro e foi escolhido o valor que deu origem à máxima precisão no conjunto de validação. Com este escolhido, foi utilizado o conjunto de treino original para treinar o classificador e o conjunto de teste para avaliar a performance.

Análise de Resultados

Classifier / Evalutor	SVM polynomial (Degree=1)	SVM RBF ($\gamma = 0.0023$)	Nearest Neighbours ($neighbours = 10$)	Multinomial Naive Bayes	Gaussian Naive Bayes	Decision Tree
Accuracy	0.765	0.709	0.652	0.513	0.565	0.687
Balanced Accuracy	0.771	0.713	0.657	0.511	0.588	0.682
Confusion Matrix	$\begin{bmatrix} 94 & 14 \\ 40 & 82 \end{bmatrix}$	$\begin{bmatrix} 84 & 24 \\ 43 & 79 \end{bmatrix}$	$\begin{bmatrix} 79 & 29 \\ 51 & 71 \end{bmatrix}$	$\begin{bmatrix} 52 & 56 \\ 56 & 66 \end{bmatrix}$	$\begin{bmatrix} 103 & 5 \\ 95 & 27 \end{bmatrix}$	$\begin{bmatrix} 65 & 43 \\ 29 & 93 \end{bmatrix}$

Tabela 1: Tabela com precisões e matrizes de confusão para os diferentes métodos utilizados para o Dataset 1

Classifier / Evalutor	SVM polynomial (Degree=4)	SVM RBF ($\gamma = 2.31 \times 10^{-8}$)	Nearest Neighbours ($neighbours = 5$)	Multinomial Naive Bayes	Gaussian Naive Bayes	Decision Tree
Accuracy	0.838	0.869	0.838	0.758	0.657	0.788
Balanced Accuracy	0.820	0.817	0.812	0.762	0.715	0.731
Confusion Matrix	$\begin{bmatrix} 24 & 7 \\ 9 & 59 \end{bmatrix}$	$\begin{bmatrix} 21 & 10 \\ 3 & 65 \end{bmatrix}$	$\begin{bmatrix} 23 & 8 \\ 8 & 60 \end{bmatrix}$	$\begin{bmatrix} 24 & 7 \\ 17 & 51 \end{bmatrix}$	$\begin{bmatrix} 27 & 4 \\ 30 & 38 \end{bmatrix}$	$\begin{bmatrix} 18 & 13 \\ 8 & 60 \end{bmatrix}$

Tabela 2: Tabela com precisões e matrizes de confusão para os diferentes métodos utilizados para o Dataset 2

Em relação ao primeiro dataset, a maior precisão obtida foi de cerca de 76% com o classificador SVM linear (polinomial de ordem 1). Os restantes classificadores tiveram precisões inferiores, sendo a mais baixa de 51% para o classificador Multinomial Naive Bayes. Sendo que o conjunto de teste do dataset 1 tem quase o mesmo número de exemplos para ambas as classes (122 para a classe 1 e 108 para a classe 0), os valores obtidos para a precisão equilibrada são bastante próximos dos obtidos para a precisão, para cada classificador. Observando as matrizes de confusão, verificamos que a maioria dos classificadores (SVM polinomial, SVM RBF, Nearest neighbour e Gaussian Naive Bayes) tiveram maior precisão na classificação dos exemplos da classe 0, sendo o algoritmo mais eficaz a fazê-lo o Gaussian Naive Bayes, que classificou corretamente 103 dos 108 exemplos da classe 0. No entanto, este classificador também classificou na classe 0 95 dos 122 exemplos da classe 1, daí a sua baixa precisão. Os restantes classificadores (Multinominal Naive Bayes e Decision Tree) classificaram corretamente mais exemplos da classe 1.

Para o segundo dataset, verificamos que a maior precisão (cerca de 87%) é dada pelo classificador SVM RBF com $\gamma \approx 2.31 * 10^{-8}$, no entanto, como o conjunto de teste é desproporcional (existem 68 exemplos da classe 1 e apenas 31 da classe 0), não é este o classificador que apresenta maior precisão equilibrada, mas sim o SVM polinomial com grau 4, que apresenta uma precisão equilibrada de 82%. Verificamos que os valores das precisões dos vários classificadores variam entre 66% e 87%, no entanto os valores das precisões equilibradas variam apenas entre 72% e 82%. Mais uma vez, isto deve-se ao facto de o conjunto de teste ser desproporcional. É possivel verificar através das matrizes de confusão que os classificadores SVM polinomial, Nearest Neighbour e SVM RBF, que foram os que obtiveram maiores precisões equilibradas, foram também aqueles que tiveram menos erros de classificação. É curioso observar que, de forma semelhante ao dataset 1, o classificador Gaussian Naive Bayes foi aquele que obteve maior número de exemplos corretamente classificados na classe 0, mas em contrapartida foi também o que obteve maior número de exemplos que pertenciam à classe 1 incorretamente classificados na classe 0. Isto mostra que este classificador tende a classificar uma maior percentagem de dados em uma classe específica.