Aprendizado de Máquina – Exercício 2

Thiago Amendola – 148062

Introdução

Foi feita a leitura de um conjunto de 476 dados de 166 dimensões, com suas respectivas classificações em 2 grupos. Foi então feito um treinamento de uma SVM com kernel RBF, utilizando 5-fold estratificado e 3-fold interno. Para encontrar valores factíveis para os hiperparâmetros C e gamma (provenientes das funções de SVM e do kernel RBF, respectivamente), foi feita uma Grid Search entre possíveis valores de C e gamma apresentados a seguir:

C = [2^-5, 2^-2, 2^0, 2^2, 2^5]
gamma = [2^-15, 2^-10, 2^-5, 2^0, 2^5]

Resolução

Primeiramente, realizamos a importação das bibliotecas que serão utilizadas nesta aplicação:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
import sklearn
```

Podemos criar desde já um vetor com os valores que serão utilizados por C e gamma na aplicação

```
c_values = [2**-5, 2**-2, 2**0, 2**2, 2**5]
gamma_values = [2**-15, 2**-10, 2**-5, 2**0, 2**5]
```

Realizamos a leitura dos dados e separação em conjunto de dados de entrada do problema e a respectiva classificação:

```
raw_data = pd.read_csv('data1.csv')
raw_array = raw_data.values

datas = raw_array[:,0:165]
labels = raw_array[:,166]
```

É criada então o 5-Fold estratificado externo usando os dados obtidos. Dentro do loop para cada fold, é realizada a separação entre os conjuntos de dados e labels de treinamento e teste.

```
ext_skf = StratifiedKFold(5)
```

```
    for ext_tr, ext_te in ext_skf.split(datas,labels):
        #Preparations for internal k-fold
        d_tr, d_te = datas[ext_tr], datas[ext_te]
        l_tr, l_te = labels[ext_tr], labels[ext_te]
```

Em seguida, é criado um 3-Fold interno, com o intuito de encontrar valores factíveis para os hiperparâmetros, separando o atual conjunto de treinamento em um novo conjunto de treinamento e teste internos:

```
    ext_skf = StratifiedKFold(3)
    for int_tr, int_te in ext_skf.split(d_tr,l_tr):
        #Preparations for training
        tr, te = d_tr[int_tr], d_tr[int_te]
        ltr, lte = l_tr[int_tr], l_tr[int_te]
```

Dentro deste 2° K-Fold, inicia-se a busca pelos valores dos hiperparâmetros, iterando entre todos os valores possíveis.

```
    # Grid Search through hiperparameters C and gamma
    for C in c_values:
        for gamma in gamma_values:
```

Para cada valor de C e gamma obtido pelos loops, é feita uma nova SVM. Esta SVM é treinada com os conjuntos de dados e labels de treino internos e testados ao predizer classificações para os conjuntos de dados de teste internos. É então feita uma comparação com o conjunto de labels de teste internos para avariar a acurácia das predições:

```
    #Training
    clf = SVC(C=C, kernel='rbf', gamma=gamma)
    clf.fit(tr, ltr)
    #Testing
    pred = clf.predict(te)
    ac = pred_accuracy(pred, lte)
```

A função pred_accuracy realiza o cálculo da acurácia da predição ao receber como dados de entrada o vetor de predições do SVM e o vetor de testes com os valores reais, retornando um valor decimal de 0 a 1.

```
def pred_accuracy(pred, lte):
    hits = [None] * len(pred)
    for i in range(0,len(pred)):
        if pred[i] == lte[i]:
            hits[i] = 1
        else:
            hits[i] = 0
    return sum(hits)/len(pred)
```

Após obter a acurácia, checa-se se a acurácia atual é a melhor dentro do 3-fold. Da melhor acurácia do 3-fold são armazenados os valores de C e gamma.

```
#Check if result's the best
if ac > int_ac:
    int_ac, int_C, int_gamma = ac, C, gamma
```

Retornando ao bloco do 5-fold estratificado, com o fim da execução do 3-fold, temos valores factíveis para nossos hiperparâmetros. É possível fazer mais uma seção de treinamento com SVM e testes com as predições geradas, desta vez utilizando os conjuntos de treino e testes do 5-fold estratificado externo

```
#Training
clf = SVC(C=int_C, kernel='rbf', gamma=int_gamma)
clf.fit(d_tr, l_tr)
#Testing
pred = clf.predict(d_te)
ac = pred_accuracy(pred, l_te)
```

Após as 5 iterações do 5-fold, é obtida a média das 5 acurácias e são registrados os valores de C e gamma para a maior acurácia obtida entre elas

```
aver_accur += ac
if ac>best_accur:
    best_accur, best_C, best_gamma = ac, int_C, int_gamma
```

Obtida a acurácia média e os melhores valores de C e gamma, é então feito mais um 3-fold final com todo o conjunto fornecido como entrada para a aplicação:

```
aver_accur = 0
fin_skf = StratifiedKFold(3)
for tr, te in fin_skf.split(datas,labels):
    #Training
    clf = SVC(C=best_C, kernel='rbf', gamma=best_gamma)
    clf.fit(datas[tr], labels[tr])
    #Testing
    pred = clf.predict(datas[te])
    ac = pred_accuracy(pred, labels[te])
```

Resultados

Foi obtida, para as acurácias da etapa de 5-fold estratificado, uma média de **0.9077164986935424**.

Os resultados mais acurados na etapa de 5-fold estratificado foram obtidos com **C=4** e **gamma=0.03125**.

Para o 3-fold final, a média de acurácia obtida foi de *0.920176206777592*.

Código completo

```
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
import sklearn


c_values = [2**-5, 2**-2, 2**0, 2**2, 2**5]
gamma_values = [2**-15, 2**-10, 2**-5, 2**0, 2**5]


def pred_accuracy(pred, lte):
    hits = [None] * len(pred)
    for i in range(0,len(pred)):
        if pred[i] == lte[i]:
            hits[i] = 1
        else:
            hits[i] = 0
    return sum(hits)/len(pred)


if __name__=='__main__':

    raw_data = pd.read_csv('data1.csv')
    raw_array = raw_data.values

    datas = raw_array[:,0:165]
    labels = raw_array[:,166]

    aver_accur = 0
    int_C = 1
    int_gamma = 1
    best_accur = 0
    best_C = 1
    best_gamma = 1

    #External stratified 5-fold
    print("Starting stratified 5-Fold with internal 3-Fold and Grid
    ext_skf = StratifiedKFold(5)
    for ext_tr, ext_te in ext_skf.split(datas,labels):
        #Preparations for internal k-fold
        d_tr, d_te = datas[ext_tr], datas[ext_te]
        l_tr, l_te = labels[ext_tr], labels[ext_te]
        int_ac = 0
        int_C = 1
        int_gamma = 1

        #Obtain hiperparameters: 3-fold training
```

```
    ext_skf = StratifiedKFold(3)
    for int_tr, int_te in ext_skf.split(d_tr,l_tr):
        #Preparations for training
        tr, te = d_tr[int_tr], d_tr[int_te]
        ltr, lte = l_tr[int_tr], l_tr[int_te]

        ### Grid Search through hiperparameters C and gamma
        for C in c_values:
            for gamma in gamma_values:
                #Training
                clf = SVC(C=C, kernel='rbf', gamma=gamma)
                clf.fit(tr, ltr)
                #Testing
                pred = clf.predict(te)
                ac = pred_accuracy(pred, lte)
                #Check if result's the best
                if ac > int_ac:
                    int_ac, int_C, int_gamma = ac, C, gamma
    #Training
    clf = SVC(C=int_C, kernel='rbf', gamma=int_gamma)
    clf.fit(d_tr, l_tr)
    #Testing
    pred = clf.predict(d_te)
    ac = pred_accuracy(pred, l_te)
    print("C="+str(int_C)+"; gamma="+str(int_gamma))
    print("Accuracy="+str(ac))
    aver_accur += ac
    if ac>best_accur:
        best_accur, best_C, best_gamma = ac, int_C, int_gamma

#Get average accuracy
aver_accur /= 5
print("========================")
print("Average accuracy="+str(aver_accur))
print("========================")
print("")

#Final 3-fold
aver_accur = 0
print(">>Starting final 3-Fold")
print(">>>>C="+str(best_C)+"; gamma="+str(best_gamma))
fin_skf = StratifiedKFold(3)
for tr, te in fin_skf.split(datas,labels):
    #Training
    clf = SVC(C=best_C, kernel='rbf', gamma=best_gamma)
    clf.fit(datas[tr], labels[tr])
    #Testing
    pred = clf.predict(datas[te])
    ac = pred_accuracy(pred, labels[te])
    print(">>Accuracy="+str(ac))
```

```
    aver_accur += ac
aver_accur /= 3
print("========================")
print("Final average accuracy="+str(aver_accur))
print("========================")
```