Titulo: Exercício 2 - Grid Search, Stratified K-Folds e SVM
Autor: Juan Sebastián Beleño Díaz
Data: 3 de Outubro de 2016

Introdução

Neste trabalho é achado os valores optimizados de gamma e C=1/alpha para um SVM com kernel RBF, usando K-folds estratificados externos para achar a acurácia do SVM e K-Folds estratificados internos para achar os hiperparâmetros gamma e C.

Dados

O arquivo base deste trabalho é data1.csv; o arquivo contém 167 colunas e 476 filas. As primeiras 166 colunas do conjunto de dados tem um nome f{n} onde n é um número incremental desde 1 até 166; a coluna 167 é a clase à que pertence cada fila.

Preparação dos dados

Antes de começar a trablahar com os dados é preciso incluir as dependecias do projeto:

```
# Loading the libraries
import numpy as np
import pandas as pd

from sklearn.cross_validation import StratifiedKFold
from sklearn.svm import SVC
```

Existem muitas maneiras de abrir o arquivo csv e obter os dados, mas neste caso vamos usar pandas para obter o dataframe diretamente desde a URL.

```
# Reading the csv file with the raw data
df = pd.read_csv('http://www.ic.unicamp.br/~wainer/cursos/2s2016/ml/data1.csv')
```

Separar os dados para obter os parâmetros e os resultados ou classes dos dados.

```
ncolumns = df.shape[1] # 167 columns
ncolumns_without_class = ncolumns - 1 # 166 columns

# Removing the column 'clase' from the dataset
df_params = df.iloc[:, 0:ncolumns_without_class]

# Getting the column 'clase' from the dataset
df_result = df.iloc[:,ncolumns_without_class:ncolumns]
df_result = np.ravel(df_result) # convert a column vector to vector
```

Definir um conjunto de variáveis que vão ser usadas depois.

```
# Declare important variables to use later
n_external_folds = 5
n_internal_folds = 3

gamma_values_set = [2**-15, 2**-10, 2**-5, 2**0, 2**5]
c_values_set = [2**-5, 2**-2, 2**0, 2**2, 2**5]
optimal_gamma = 0
optimal_c = 0

final_accuracy = 0
best_accuracy = 0
```

Processamento dos dados

Implementamos um K-Fold estratificado externo de 5 folds para separar conjuntos de treino e conjunto de testes, achando assim o valor médio da acurácia esperada de um SVM com kernel RBF em cada fold. Além disso, é implementado um K-fold interno com 3 folds para cada fold externo, achando os melhores valores de gamma e C, iterando sobre conjunto fixos de dados. O valor do C itera sobre valores contidos no array [2^-5, 2^-2, 2^0, 2^2, 2^5]. O valor do gamma itera sobre os valores contidos no array [2^-15, 2^-10, 2^-5, 2^0, 2^5]. Os valores finais de gamma e C correspondem à maior acurácia obtida nos folds internos. Para calcular a acurácia de cada fold é usado um SVM com kernel RBF que trabalha com validação interna dentro dos dados externos de treino. A acurácia de cada fold é somada a variável *final_accuracy*.

```
# Define the external K-Fold Stratified
external_skf = StratifiedKFold(df_result, n_folds = n_external_folds)

# Iterate over several folds to find a good accuracy in the SVM
for external_train_index, external_test_index in external_skf:

    # Declare external variables
    external_accuracy = 0
    external_gamma_value = 0
    external_c_value = 0

    # Split the external training set and the external test set
    external_params_train = df_params.iloc[external_train_index, :]
    external_results_train = df_result[external_train_index]
    external_params_test = df_params.iloc[external_test_index, :]
    external_results_test = df_result[external_test_index]

    # Define the internal K-Fold Stratified
    internal_skf = StratifiedKFold(external_results_train, n_folds =
n_internal_folds)

    # Iterate over several internal folds
    for internal_train_index, internal_test_index in internal_skf:

        # Declare internal variables
        internal_accuracy = 0
        internal_gamma_value = 0
```

```
        internal_c_value = 0

        # Split the internal training set and the internal test set
        internal_params_train = external_params_train.iloc[internal_train_index, :]
        internal_results_train = external_results_train[internal_train_index]
        internal_params_test = external_params_train.iloc[internal_test_index, :]
        internal_results_test = external_results_train[internal_test_index]

        # Iterate over gamma and C values to get best results in internal folds
        for gamma_value in gamma_values_set:
            for c_value in c_values_set:

                # Set up the internal classifier
                internal_classifier = SVC(C = c_value, kernel = 'rbf', gamma =
gamma_value)
                internal_classifier.fit(internal_params_train,
internal_results_train)

                # Getting the accuracy of the internal classifier for experimental
                # values for gamma and C= 1/alpha
                temporal_accuracy = internal_classifier.score(internal_params_test,
internal_results_test)

                # Looking for the best internal accuracy
                if temporal_accuracy > internal_accuracy:
                    internal_accuracy = temporal_accuracy
                    internal_gamma_value = gamma_value
                    internal_c_value = c_value


                # Looking for the best gamma and C values
                if temporal_accuracy > best_accuracy:
                    best_accuracy = temporal_accuracy
                    optimal_gamma = gamma_value
                    optimal_c = c_value

        # Compare and update the fold accuracy
        if(internal_accuracy > external_accuracy):
            external_accuracy = internal_accuracy
            external_gamma_value = internal_gamma_value
            external_c_value = internal_c_value


    # Calculate the fold accuracy
    external_classifier = SVC(C = external_c_value, kernel = 'rbf', gamma =
external_gamma_value)
    external_classifier.fit(external_params_train, external_results_train)

    fold_accuracy = external_classifier.score(external_params_test,
external_results_test)

    # Perform a sum over the fold accuracy
    final_accuracy = final_accuracy + fold_accuracy
```

Para obter a previsão da acurácia do SVM com kernel RBF neste conjunto de dados é preciso obter a média das acurácias dos folds externos, o que é feito no seguiente código:

```
# Divide the final_accuracy over the number of folds to get the mean accuracy
final_accuracy = final_accuracy/n_external_folds
```

Resultados

Finalmente são apresentados os resultados do nosso enfoque, mostrando a acurácia média do nosso SVM e os valores finais de gamma e C.

```
# Print results
print('Acurácia média do SVM: ', final_accuracy)
print('Valor final do gamma: ', optimal_gamma)
print('Valor final do C: ', optimal_c)
```

```
Acurácia média do SVM:  0.907716498694
Valor final do gamma:  0.03125
Valor final do C:  1
```