Nome: Nathana Facion

Exercício 2
Aprendizado de Máquina

---

1. Bibliotecas:

```
import numpy as np
from sklearn import datasets, preprocessing
from numpy import genfromtxt
from sklearn.svm import SVC
from sklearn.cross_validation import StratifiedKFold
from sklearn.metrics import confusion_matrix
import itertools
```

2. Funções usadas:

```
def accuracy(matrix):
  return round(float(matrix[0][0] + matrix[1][1]) / float(matrix.sum()), 4)


def createGridSearch():
  hparameters = {'C': [2 ** -5, 2 ** -2, 2 ** 0, 2 ** 2, 2 ** 5],
           'gamma': [2 ** -15, 2 ** -10, 2 ** -5, 2 ** 0, 2 ** 5]}
  h = [hparameters[k] for k in hparameters]
  i = itertools.product(*h)
  return list(i)


def meanFinal(acfinal, n_folds):
  return round(float(acfinal / n_folds), 4)
```

3. K-Fold Interno

```
def kfoldIntern(grid, trainning_x, trainning_y):
  # Para cada conjunto de treino da validacao externa faca um 3-fold para escolher os melhores hiperparametros para C (cost) e gamma.
  n_folds = 3
  skf = StratifiedKFold(trainning_y, n_folds)
  acmax = 0
  hparameter_max = 0

  for g in grid:
    ac = 0
    for training_index, test_index in skf:
      train_x_inside, test_x_inside = trainning_x[training_index], trainning_x[test_index]
      train_y_inside, test_y_inside = trainning_y[training_index], trainning_y[test_index]
      model = SVC(C=g[0], gamma=g[1], kernel='rbf')
      model.fit(train_x_inside, train_y_inside)
      predicted = model.predict(test_x_inside)
      c_matrix = confusion_matrix(test_y_inside, predicted)
      ac = ac + accuracy(c_matrix)
    if ac > acmax:
      acmax = ac
      hparameter_max = g
  print(hparameter_max)
  return hparameter_max
```

4. Main

```
def main():
  # Leia os dados do arquivo data1.csv A classe de cada dado e o valor da ultima coluna (0 ou 1).
  fileName = "//home//nathana//AM//data1.csv"
  # todos dados
  data = genfromtxt(fileName, delimiter=',')[1:]
  # apenas a classe
  classData = np.array([d[len(d) - 1] for d in data])
  # dados sem a classe
  data =  np.array([d[:len(d) - 1] for d in data])
  # Treine um SVM com kernel RBF nos dados do arquivos.
  # A validacao externa deve ser 5-fold estratificado.
  X = data
  Y = classData
  n_folds = 5
  external_skf = StratifiedKFold(Y, n_folds)

  # Faca um grid search de para o C nos valores 2**-5, 2**-2, 2**0, 2**2, e 2**5 e gamma nos valores 2**-15, 2**-10,
2**-5, 2**0, e 2**5
  grid = createGridSearch()
  acxFinal = 0

  for training_index, test_index in external_skf:
    X_train, X_test = X[training_index], X[test_index]
    Y_train, Y_test = Y[training_index], Y[test_index]

    hparameter_max = kfoldIntern(grid, X_train, Y_train)
    model = SVC(kernel='rbf',C=hparameter_max[0], gamma=hparameter_max[1])
    model.fit(X_train, Y_train)
    predicted = model.predict(X_test)

    c_matrix = confusion_matrix(Y_test, predicted)
    ac = accuracy(c_matrix)
    acxFinal = ac + acxFinal

    #print("Acuracia:", ac)

  # Qual a accuracia media (na validacao de fora).
  print "Media Final", meanFinal(acxFinal, n_folds)

  # Quais os valores de C e gamma a serem usados no classificador final (fazer o 3-fold no conjunto todo).
  hparameter_max = kfoldIntern(grid,X,Y)
  print("Valor Hiperparametro",hparameter_max)

if __name__ == '__main__':
  main()
```

5. Saídas

As 5 primeiras linhas são os valores de hiperparâmetro que foram retornado do k-fold interno. Após isso temos a média de fora e o valor de hiperparametro final.

```
(32, 0.0009765625)
(4, 0.03125)
(1, 0.03125)
(4, 0.03125)
(4, 0.03125)
Media Final 0.9119
(4, 0.03125)
('Valor Hiperparametro', (4, 0.03125))
```

6. Respostas

1) Qual a accuracia media (na validação de fora).
0.9119
2) Quais os valores de C e gamma a serem usados no classificador final (fazer o 3-fold no conjunto todo).
Na sequência C é o primeiro e Gamma é o segundo.
('Valor Hiperparametro', (4, 0.03125))