Exercício 2 - MO444 - Aprendizado de máquina e reconhecimento de padrões

Renato Lopes Moura - 163050

Importando os módulos que serão utilizados no exercício.

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.cross_validation import StratifiedKFold
        from sklearn.svm import SVC
        import itertools
```

Preparação dos dados para validação.

```
In [2]: #Carregando o conjunto de dados do csv usando o pandas
        data = pd.read_csv('data1.csv')

        #Conversao para arrays do numpy
        array = data.values

        #Separando os dados em atributos e classes
        X = array[:,0:165]
        Y = array[:,166]

        #Definicao dos hiperparametros C e gamma
        C = [2**(-5),2**(-2),2**0,2**2,2**5]
        gamma = [2**(-15),2**(-10),2**(-5),2**0,2**5]

        #Lista com o produto cartesiano hiperparametros no formato [C,gamma]
        hyperparameters = list(itertools.product(C,gamma))

        #Inicializacao da variavel de acuracia do loop externo
        accuracy_ext = 0

        #Instanciacao do StratifiedKFold no loop externo (5 folds)
        external = StratifiedKFold(Y,n_folds=5)
```

Loop para determinação da acurácia média.

```
In [3]: for train_ext, test_ext in external:
            #Separando os dados de treino do loop externo em atributos e classes
            X_train_ext = X[train_ext]
            Y_train_ext = Y[train_ext]

            #Separando os dados de teste do loop externo em atributos e classes
            X_test_ext = X[test_ext]
            Y_test_ext = Y[test_ext]

        #Inicializacao do array que armazenara os melhores hiperparametros
        # do loop interno e da variavel de acuracia do loop interno
            h_max = []
            accuracy_int = 0

            #Instanciacao do StratifiedKFold no loop interno (3 folds)
            internal = StratifiedKFold(Y_train_ext,n_folds=3)

            for parameters in hyperparameters:
                for train_int, test_int in internal:

                        #Separando os dados de treino do loop interno
                    # em atributos e classes
                        X_train_int = X_train_ext[train_int]
                        Y_train_int = Y_train_ext[train_int]

                        #Separando os dados de teste do loop interno
                    # em atributos e classes
                        X_test_int = X_train_ext[test_int]
                        Y_test_int = Y_train_ext[test_int]

                        #Instanciacao do SVM com kernel RBF e hiperparametros
                    # definidos
                        svm_int = SVC(C=parameters[0],gamma=parameters[1],
                          kernel='rbf')

                    #Treinamento do classificador no conjunto de dados interno
                        svm_int.fit(X_train_int, Y_train_int)

                    #Verificacao da acuracia para cada par de hiperparametros
                    # (C e gamma) e escolha do par que apresenta a melhor acuracia
                        if accuracy_int < svm_int.score(X_test_int, Y_test_int):
                            accuracy_int = svm_int.score(X_test_int, Y_test_int)
                            h_max = parameters

        #Instanciacao do SVM com kernel RBF e hiperparametros obtidos
        # no loop interno
            svm_ext = SVC(C=h_max[0],gamma=h_max[1],kernel='rbf')

        #Treinamento do classificador no conjunto de dados externo
            svm_ext.fit(X_train_ext, Y_train_ext)

        #Soma das acuracias obtidas em cada um dos 5 folds
            accuracy_ext = accuracy_ext + svm_ext.score(X_test_ext, Y_test_ext)

    #Calculo da acuracia media na validacao externa
    print "Acuracia media (na validacao de fora): "+str(accuracy_ext/5)
```

```
Acuracia media (na validacao de fora): 0.907716498694
```

Loop para determinação dos melhores hiperparâmetros para o classificador final.

```
In [4]: #Instanciacao do StratifiedKFold no conjunto de dados completo (3 folds)
        final = StratifiedKFold(Y,n_folds=3)

        #Inicializacao do array que armazenara os melhores hiperparametros
        # do classificador final e da variavel de acuracia do classificador final
        accuracy = 0
        h_final = []

        for parameters in hyperparameters:
                for train, test in final:

                        #Separando os dados de treino em atributos e classes
                        X_train = X[train]
                        Y_train = Y[train]

                        #Separando os dados de teste em atributos e classes
                        X_test = X[test]
                        Y_test = Y[test]

                    #Instanciacao do SVM com kernel RBF e hiperparametros definidos
                        svm = SVC(C=parameters[0],gamma=parameters[1],kernel='rbf')

                #Treinamento do classificador no conjunto de dados completo
                        svm.fit(X_train, Y_train)

                #Verificacao da acuracia para cada par de hiperparametros (C e gamma)
                # e escolha do par que apresenta a melhor acuracia
                        if accuracy < svm.score(X_test, Y_test):
                                accuracy = svm.score(X_test, Y_test)
                                h_final = parameters

        print "Hiperparametros do classificador final: C = "+str(h_final[0])+ \
        " gamma = "+str(h_final[1])
```

```
Hiperparametros do classificador final: C = 4 gamma = 0.03125
```

Os hiperparâmetros a serem utilizados no classificador final são

$$C = 2^2$$

e

$$gamma = 2^{-5}$$