Exercício 6 - MO444 - Aprendizado de máquina e reconhecimento de padrões

Renato Lopes Moura - 163050

1 Código

1.1 Leitura de dados e preprocessamento

```
import numpy as np
from sklearn.datasets import load_files
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier

data = load_files('filesk/')

count_vect_bin = CountVectorizer(stop_words='english',strip_accents='ascii', min_df=2, binary=True)
count_vect = CountVectorizer(stop_words='english',strip_accents='ascii', min_df=2)

X_bag_of_words = count_vect_bin.fit_transform(data.data)
X_count = count_vect.fit_transform(data.data)

X_freq = TfidfTransformer(use_idf=False).fit_transform(X_count)
```

Ao instanciar os objetos *count_vect_bin* e *count_vect* são definidos os seguintes parâmetros da classe CountVectorizer:

- *stop_words*: idioma de referência das *stop_words*
- *strip_accents*: codificação (unicode/utf8) dos acentos a serem removidos
- *min_df*: número mínimo de ocorrências do termo para ser incluído na matriz/*bag of words*
- *binary*: montar uma *bag of words* binária ou não

Portanto, ao aplicar o método *fit_transform()* nos textos já é feito todo o pré-processamento e separação dos termos, além da montagem das *bag of words* no formato de matrizes esparsas.

Por fim, é gerada a matriz de frequências a partir da *bag of words* nao binária.

1.2 Naive Bayes e Logistic Regression

```
##############################################################################################
#Naive Bayes na matriz binaria
X_train, X_test, y_train, y_test = train_test_split(X_bag_of_words, data.target, test_size=1000)

naive_bayes = MultinomialNB().fit(X_train, y_train)
nb_predicted = naive_bayes.predict(X_test)

print "Acuracia do Naive Bayes na matriz binaria foi: "+str(metrics.accuracy_score(y_test, nb_predicted))

##############################################################################################
#Logistic Regression na matriz binaria
logistic = LogisticRegression(C=10000).fit(X_train, y_train)
log_predicted = logistic.predict(X_test)

print "Acuracia do Logistic Regression na matriz binaria foi: "+
   str(metrics.accuracy_score(y_test, log_predicted))

##############################################################################################
#Logistic Regression na matriz de term frequency
X_train_freq, X_test_freq, y_train_freq, y_test_freq = train_test_split(X_freq, data.target, test_size=1000)

logistic_freq = LogisticRegression(C=10000).fit(X_train_freq, y_train_freq)
log_freq_predicted = logistic_freq.predict(X_test_freq)

print "Acuracia do Logistic Regression na matriz de frequencias foi: "+
   str(metrics.accuracy_score(y_test_freq, log_freq_predicted))
```

1.3 PCA, SVM e GBM

```
#########################################################################################################
#Aplicacao do PCA na matriz de term frequency e separacao dos conjuntos de treino e teste
pca = PCA(0.99)
X_transf = pca.fit_transform(X_freq.toarray())

X_train_transf, X_test_transf, y_train_transf, y_test_transf = train_test_split(X_transf,
   data.target, test_size=1000)

#########################################################################################################
#SVM na matriz de frequencias reduzida pelo PCA

svm = SVC(C=2**(5), gamma=2**(-5), kernel='rbf')
svm.fit(X_train_transf, y_train_transf)

svm_predicted = svm.predict(X_test_transf)

print "Acuracia do SVM na matriz de frequencias reduzida foi: "+
   str(metrics.accuracy_score(y_test_transf, svm_predicted))

#########################################################################################################
#GBM na matriz de frequencias reduzida pelo PCA

gbm = GradientBoostingClassifier(n_estimators=70, learning_rate=0.1, max_depth=5)
gbm.fit(X_train_transf, y_train_transf)

gbm_predicted = gbm.predict(X_test_transf)

print "Acuracia do GBM na matriz de frequencias reduzida foi: "+
   str(metrics.accuracy_score(y_test_transf, gbm_predicted))
```

2 Outputs

```
Acuracia do Naive Bayes na matriz binaria foi: 0.788
Acuracia do Logistic Regression na matriz binaria foi: 0.808
Acuracia do Logistic Regression na matriz de frequencias foi: 0.86
Acuracia do SVM na matriz de frequencias reduzida foi: 0.87
Acuracia do GBM na matriz de frequencias reduzida foi: 0.823
```