

Título: Exercício 1 - PCA, regressão logística e LDA

Autor: Juan Sebastián Beleño Díaz

Data: 20 de Setembro de 2016

Neste trabalho são comparados diferentes métodos de classificação sobre um conjunto de dados de treinamento e de teste. Os métodos de classificação são Linear Discriminant Analysis(LDA) e Regressão Logística. No entanto, o conjunto de dados deste exercício contém um grande número de colunas; assim, vamos implementar um PCA sobre os dados, comparando a precisão dos métodos de classificação sobre os dados reduzidos por o PCA e sobre os dados iniciais.

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 classe à que pertence cada fila.

Transformação dos dados

Antes de começar a trabalhar com os dados é preciso incluir as dependências do projeto:

```
#!/home/juan/anaconda3/bin/python3.5
import numpy as np
import pandas as pd

from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

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

```
df = pd.read_csv('http://www.ic.unicamp.br/~wainer/cursos/2s2016/ml/data1.csv')
```

Definimos um conjunto de constantes que vão nos permitir trabalhar mais facilmente com o conjunto de treinamento e de teste.

```
ncolumns = len(df.columns) #167 columns
ncolumns_without_class = ncolumns - 1 # 166 columns
ntraining_rows = 200
ntest_rows = 276
```

Separamos o conjunto de dados de treinamento(primeiras 200 linhas) e de teste(últimas 276 linhas).

```

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

# Getting the training set from the first 200 lines
df_training_set = df_without_class[0:ntraining_rows]

# Getting the test data frame from the last 276 lines
df_test_set = df.iloc[ntraining_rows: (ntest_rows + ntraining_rows),
0:ncolumns_without_class]

# Getting the 'clase' column for the training data
results_training_set = df.iloc[0:ntraining_rows,ncolumns_without_class:ncolumns]
results_training_set = np.ravel(results_training_set) # convert column vector to vector

# Getting the 'clase' column for the test data
results_test_set = df.iloc[ntraining_rows: (ntest_rows +
ntraining_rows),ncolumns_without_class:ncolumns]
results_test_set = np.ravel(results_test_set) # convert column vector to vector

```

Principal Component Analysis (PCA)

O PCA é um método que serve para reduzir a dimensionalidade dos dados, baseando-se em transformações nos eixos das dimensões originais. Este método às vezes é muito útil para diminuir a complexidade de alguns problemas de classificação; assim, vamos implementar um PCA mantendo o 80% de variância sobre os nossos dados.

```

# Applying the PCA
pca = PCA(n_components= ncolumns_without_class)
pca.fit(df_training_set)

# Getting the cumulative variance
variance_acum = pca.explained_variance_ratio_.cumsum()

# Finding the number of components to keep the variance over 80%
ncomp = 0
var_max = 0.8

for i in range(0, ncolumns_without_class):
    if(variance_acum[i] >= var_max):
        ncomp = i + 1 # For this training data set ncomp = 12
        break

print('Número de componentes no PCA: ', ncomp)

# Applying the dimensionality reduction based on the variance for the training data
pca = PCA(n_components= ncomp)
pca.fit(df_training_set)
df_training_set_reduced = pca.transform(df_training_set) # Array != Data Frame

# Applying the dimensionality reduction based on the variance for the test dataset
pca = PCA(n_components= ncomp)
pca.fit(df_training_set)

```

```
df_test_set_reduced = pca.transform(df_test_set)
```

Número de componentes no PCA: 12

Regressão Logística

Um método de classificação executado sobre o conjunto de dados com o PCA e sobre os dados sem o PCA.

```
# setting up the regression models with and without PCA
model_with_pca = LogisticRegression().fit(df_training_set_reduced,
results_training_set)
model_without_pca = LogisticRegression().fit(df_training_set, results_training_set)

# Testing
score_rl_pca = model_with_pca.score(df_test_set_reduced, results_test_set) #
0.80072
score_rl_no_pca = model_without_pca.score(df_test_set, results_test_set) # 0.79710

# Print
print ('Acurácia Regressão Logística com PCA: ', score_rl_pca)
print ('Acurácia Regressão Logística sem PCA: ', score_rl_no_pca)
```

Acurácia Regressão Logística com PCA: 0.800724637681
Acurácia Regressão Logística sem PCA: 0.797101449275

Linear Discriminant Analysis (LDA)

Outro método de classificação executado sobre o conjunto de dados com o PCA e sobre os dados sem o PCA.

```
# Setting up the LDA with and without PCA
lda_model_with_pca = LinearDiscriminantAnalysis()
lda_model_with_pca.fit(df_training_set_reduced, results_training_set)

lda_model_without_pca = LinearDiscriminantAnalysis()
lda_model_without_pca.fit(df_training_set, results_training_set)

# Testing the LDA model accuracy with the test dataset
score_lda_pca = lda_model_with_pca.score(df_test_set_reduced, results_test_set) #
0.77898
score_lda_no_pca = lda_model_without_pca.score(df_test_set, results_test_set) #
0.67753

# Print
print ('Acurácia LDA com PCA: ', score_lda_pca)
print ('Acurácia LDA sem PCA: ', score_lda_no_pca)
```

```
Acurácia LDA com PCA: 0.778985507246
Acurácia LDA sem PCA: 0.677536231884
```

Conclusões

O melhor método neste conjunto de dados foi a Regressão Logística sobre dados com o PCA com uma acurácia de **80.072%**. De maneira geral, o uso de dados de dimensionalidade reduzida por o PCA neste conjunto de dados foi vantajoso para os dois métodos de classificação e o método de Regressão Logística teve maior acurácia do que o LDA.