

FACULDADE DE TECNOLOGIA DA BAIXADA SANTISTA RUBENS
LARA
CIÊNCIA DE DADOS
DISCIPLINA: ÁLGEBRA LINEAR
DOCENTE: ALEXANDRE GARCIA DE OLIVEIRA

ANÁLISE PCA
PEDRO JORGE DE SOUZA COLOMBRINO
HELENA VICTÓRIA DOS SANTOS BARBOZA
MARIO AMBROSIO DE SOUZA NETO

SANTOS, NOVEMBRO DE 2023

1 INTRODUÇÃO

Nesse relatório, será descrita uma análise de componentes principais(PCA) dos dados. A análise de componentes principais ou PCA é uma técnica multivariada que permite indentificar padrões e estruturas nos dados, reduzindo sua dimensionalidade e facilitando a interpretação.

2 BANCO DE DADOS

O conjunto de dados escolhido para este estudo foi adquirido por meio do site Kaggle.com. Este arquivo consiste em 16.130 imagens de alta qualidade, com uma resolução de 512×512 pixels. Ele está dividido em três categorias: Gato, Cachorro e Animais Selvagens, sendo que cada categoria contém aproximadamente 5.000 imagens. No entanto, para a análise deste estudo, serão utilizadas apenas as pastas referentes a Gato e Cachorro.

3 CODIFICAÇÃO

```
1  import os
2  import numpy as np
3  import pandas as pd
4  import matplotlib.pyplot as plt
5  from sklearn.decomposition import PCA
6  from sklearn.preprocessing import StandardScaler
7  from sklearn.model_selection import train_test_split
8  from sklearn.svm import SVC
9  from sklearn.metrics import accuracy_score
10 from PIL import Image
11 import time
12
13 # Inicia a contagem do tempo de execução
14 start_time = time.time()
15
16 # Carregamento de imagens de cachorros e gatos a partir dos
17 ↪ diretórios especificados
18 def load_images(folder):
19     images = []
20     for filename in os.listdir(folder):
21         img = Image.open(os.path.join(folder, filename))
22         if img is not None:
23             images.append(img)
24     return images
25
26 # Carregamento de imagens de cachorros
```

```

26 dogs = load_images('D:\\PCA GARCIA\\dog')
27 # Carregamento de imagens de gatos
28 cats = load_images('D:\\PCA GARCIA\\cat')
29
30 # Pré-processamento das imagens: redimensionamento, conversão
   ↳ para tons de cinza e achatar as imagens
31 def process_images(image_list):
32     processed = []
33     for img in image_list:
34         img = img.resize((128, 128))
35         img = img.convert('L')
36         img = np.array(img).flatten()
37         processed.append(img)
38     return processed
39
40 # Pré-processamento das imagens de cachorros
41 dogs = process_images(dogs)
42 # Pré-processamento das imagens de gatos
43 cats = process_images(cats)
44
45 # Criação de DataFrames pandas a partir das imagens
   ↳ processadas
46 df_dogs = pd.DataFrame(dogs)
47 df_dogs['label'] = 'dog'
48 df_cats = pd.DataFrame(cats)
49 df_cats['label'] = 'cat'
50 df = pd.concat([df_dogs, df_cats])
51
52 # Exibe o tamanho da matriz antes da classificação
53 print("O tamanho da matriz antes da classificação é: ",
   ↳ df.shape)
54
55 # Divisão dos dados em conjuntos de treinamento e teste
56 print("Dividindo os dados em conjuntos de treinamento e
   ↳ teste.")
57 X = df.drop('label', axis=1)
58 y = df['label']
59 X_train, X_test, y_train, y_test = train_test_split(X, y,
   ↳ test_size=0.2, random_state=42)
60
61 # Normalização dos dados
62 print("Normalizando os dados.")
63 scaler = StandardScaler()
64 X_train = scaler.fit_transform(X_train)
65 X_test = scaler.transform(X_test)
66

```

```

67 # Cálculo dos autovalores e autovetores da matriz de
   ↳ covariância
68 print("Calculando os autovalores e autovetores da matriz.")
69 eigenvalues, eigenvectors = np.linalg.eig(X_train.T @
   ↳ X_train)
70 print('Autovalores: ', eigenvalues)
71 print('Autovetores: ', eigenvectors)
72
73 # Aplicação do PCA (Análise de Componentes Principais) aos
   ↳ dados
74 print("Aplicando PCA aos dados.")
75 pca = PCA(n_components=3)
76 X_train_pca = pca.fit_transform(X_train)
77 X_test_pca = pca.transform(X_test)
78
79 # Exibe o tamanho da matriz após a classificação
80 print("O tamanho da matriz após a classificação é: ",
   ↳ X_train_pca.shape)
81
82 # Treinamento do classificador Support Vector Machine (SVM)
   ↳ nos dados de treinamento
83 print("Treinando o classificador SVC nos dados de
   ↳ treinamento.")
84 svc = SVC()
85 svc.fit(X_train_pca, y_train)
86
87 # Faz previsões nos dados de teste
88 print("Fazendo previsões nos dados de teste.")
89 y_pred = svc.predict(X_test_pca)
90
91 # Cálculo da acurácia da classificação
92 print("Calculando a acurácia da classificação.")
93 accuracy = accuracy_score(y_test, y_pred)
94 print('Acurácia da classificação: ', accuracy)
95
96 # Plotagem dos gráficos tridimensionais antes e depois da
   ↳ aplicação do PCA
97 print("Plotando os gráficos.")
98 fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8),
   ↳ subplot_kw={'projection': '3d'})
99
100 # Gráfico antes da classificação
101 ax1.set_xlabel('Feature 1', fontsize=15)
102 ax1.set_ylabel('Feature 2', fontsize=15)
103 ax1.set_zlabel('Feature 3', fontsize=15)

```

```

104 ax1.set_title('Dados originais - Antes da classificação',
    ↪ fontsize=20)
105
106 targets = ['dog', 'cat']
107 colors = ['b', 'r']
108 for target, color in zip(targets, colors):
109     indicesToKeep = df['label'] == target
110     ax1.scatter(df.loc[indicesToKeep, df.columns[0]],
111                df.loc[indicesToKeep, df.columns[1]],
112                df.loc[indicesToKeep, df.columns[2]],
113                c=color,
114                s=50)
115 ax1.legend(targets)
116 ax1.grid()
117
118 # Gráfico após a classificação
119 ax2.set_xlabel('Principal Component 1', fontsize=15)
120 ax2.set_ylabel('Principal Component 2', fontsize=15)
121 ax2.set_zlabel('Principal Component 3', fontsize=15)
122 ax2.set_title('3 component PCA - Após a classificação',
    ↪ fontsize=20)
123
124 df_pca = pd.DataFrame(data=X_train_pca, columns=['pc1',
    ↪ 'pc2', 'pc3'])
125 df_pca['label'] = y_train.values
126
127 for target, color in zip(targets, colors):
128     indicesToKeep = df_pca['label'] == target
129     ax2.scatter(df_pca.loc[indicesToKeep, 'pc1'],
130                df_pca.loc[indicesToKeep, 'pc2'],
131                df_pca.loc[indicesToKeep, 'pc3'],
132                c=color,
133                s=50)
134 ax2.legend(targets)
135 ax2.grid()
136
137 # Exibe os gráficos
138 plt.show()
139
140 # Exibe o tempo total de execução
141 print("--- %s seconds ---" % (time.time() - start_time))

```

4 RESULTADOS

```
In [1]: runfile('D:/PCA GARCIA/PCA.py', wdir='D:/PCA GARCIA')
Iniciando a operação de carregamento de imagens. Esta operação irá carregar todas as imagens dos
diretórios especificados.
Iniciando a operação de pré-processamento. Esta operação irá redimensionar as imagens, convertê-las
para tons de cinza e achatar as imagens.
Iniciando a operação de criação do dataframe. Esta operação irá criar um dataframe pandas a partir das
imagens processadas.
O tamanho da matriz antes da classificação é: (1000, 16385)
Dividindo os dados em conjuntos de treinamento e teste.
Normalizando os dados.
Calculando os autovalores e autovetores da matriz.
Autovalores: [ 2.30962029e+06+0.00000000e+00j  1.83411188e+06+0.00000000e+00j
 1.10366247e+06+0.00000000e+00j ... -1.01606163e-13+0.00000000e+00j
 -6.12542530e-14+7.51302111e-14j -6.12542530e-14-7.51302111e-14j]
Autovetores: [[-6.86632148e-03+0.00000000e+00j  1.34889057e-02+0.00000000e+00j
 9.87426481e-04+0.00000000e+00j ... 1.33225866e-03+0.00000000e+00j
 1.26508052e-04-2.37956403e-04j  1.26508052e-04+2.37956403e-04j]
[-7.11958733e-03+0.00000000e+00j  1.36483004e-02+0.00000000e+00j
 9.62590844e-04+0.00000000e+00j ... 1.58306631e-05+0.00000000e+00j
 -2.23682323e-05-5.31033970e-06j -2.23682323e-05+5.31033970e-06j]
[-6.96017262e-03+0.00000000e+00j  1.37019680e-02+0.00000000e+00j
 1.07718695e-03+0.00000000e+00j ... 7.97716510e-05+0.00000000e+00j
 3.14073338e-06-9.24333857e-06j  3.14073338e-06+9.24333857e-06j]
...
[-5.87713268e-03+0.00000000e+00j  8.23985370e-04+0.00000000e+00j
 -1.34567084e-02+0.00000000e+00j ... 1.44785679e-02+0.00000000e+00j
 -9.39003516e-03-1.00684613e-03j -9.39003516e-03+1.00684613e-03j]
[-5.88928114e-03+0.00000000e+00j  1.05597600e-03+0.00000000e+00j
 -1.32866794e-02+0.00000000e+00j ... -1.96799451e-03+0.00000000e+00j
 2.47934906e-03+3.39674213e-03j  2.47934906e-03-3.39674213e-03j]
[-5.78982359e-03+0.00000000e+00j  1.41586730e-03+0.00000000e+00j
 -1.34045566e-02+0.00000000e+00j ... -3.05525713e-03+0.00000000e+00j
 -3.81118709e-03-2.83252404e-03j -3.81118709e-03+2.83252404e-03j]]

Aplicando PCA aos dados.
O tamanho da matriz após a classificação é: (800, 3)
Treinando o classificador SVC nos dados de treinamento.
Fazendo previsões nos dados de teste.
Calculando a acurácia da classificação.
Acurácia da classificação: 0.595
Plotando os gráficos.

Important

Figures are displayed in the Plots pane by default. To make them also appear inline in the
console, you need to uncheck "Mute inline plotting" under the options menu of Plots.

--- 2143.0379877090454 seconds ---
```

Figura 1: Resultados do código

O código em Python realiza uma Análise de Componentes Principais (PCA) em um conjunto de imagens de cachorros e gatos, utilizando bibliotecas como NumPy, Pandas e Matplotlib. As etapas incluem o carregamento e processamento das imagens, organização dos dados em DataFrames, divisão em conjuntos de treinamento e teste, normalização dos dados e o cálculo de autovalores e autovetores da matriz de covariância.

O conjunto de dados consiste em 1000 imagens, cada uma com 16385 pixels. As imagens passam por pré-processamento, redimensionamento, conversão para tons de cinza e achatamento, sendo transformadas em um DataFrame do Pandas. O conjunto de treinamento é usado para treinar o modelo PCA, enquanto o conjunto de teste avalia o desempenho do modelo.

A normalização é crucial para garantir a correta execução da PCA, e os autovalores e autovetores da matriz de dados são calculados. A PCA reduz a dimensionalidade de 16385 para 3, representando cada imagem como uma combinação linear de três componentes principais.

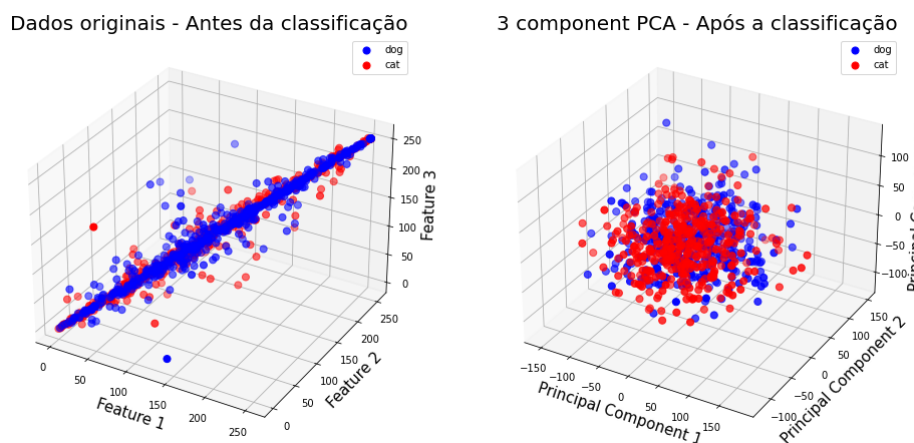


Figura 2: Separação de classes por PCA

Na fase de classificação, o código utiliza um classificador de suporte vetorial (SVC) para categorizar as imagens em duas classes: gato e cachorro. O SVC é um modelo de aprendizado de máquina supervisionado comum em tarefas de classificação de imagens.

A visualização dos dados é feita por meio de um gráfico de dispersão, onde pontos azuis representam imagens de cachorros e pontos vermelhos representam imagens de gatos. O gráfico revela uma sobreposição significativa entre os grupos, indicando características bastante semelhantes e dificuldade na distinção visual.

5 CONSIDERAÇÕES FINAIS

Em conclusão, a Análise de Componentes Principais (PCA) foi realizada com sucesso e os resultados plotados em um gráfico de dispersão. Apesar do sucesso na redução de dimensionalidade e classificação, a visualização revela uma sobreposição significativa entre as classes, indicando características semelhantes.