Análise de Components Principais
Compressão de Imagens
UFRGS - FIS01082

Aluno: Henrique Alexandre Boneto - 288744

Julho 2020

Análise de Components Principais

A Análise de Componentes Principais (ACP) é uma técnica utilizada para reduzir a dimensionalidade de dados com dimensão n para uma dimensão k onde $k < n$. A ideia principal é que possamos explicar esses dados com uma base de vetores ortogonais que melhor expliquem os dados originais, tais vetores são chamados de **componentes principais** e são encontrados de criando novas variáveis não correlacionadas com a imposição de de uma restrição de maximização da variância.

Aspectos Matemáticos

Dada uma matrix $\mathbf{X} \in \mathrm{IR}^{\mathrm{mxn}}$ com m linhas e n colunas que representa nosso conjunto de dados, queremos encontrar uma base $\mathbf{P}$ tal que $\mathbf{Y} = \mathbf{PX}$ onde cada uma das colunas de $\mathbf{P}$ seja uma das componentes principais.

Normalização

A técnica de ACP geralmente começa com a normalização dos dados já que estamos, utilizando-se:

$$z_{ij} = \frac{x_{ij} - \bar{x_j}}{s_j}$$

sendo s_j o desvio padrão da variável j.

Matriz de Covariância e Componentes Principais

A matriz de covariância é calculada da seguinte forma para:

$$\mathbf{C}_X = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T$$

essa matriz resultará em uma matriz $m \times m$, que vai conter todas as possíveis covariâncias entre nossas variáveis.

Lembrando que $\mathbf{Y} = \mathbf{PX}$ podemos demonstrar que [1]:

$$\mathbf{C}_Y = \frac{1}{n-1}\mathbf{D}$$

onde $\mathbf{D}$ é uma matriz diagonal com os autovalores da matriz $\mathbf{S} = \mathbf{X}\mathbf{X}^T$, já que $\mathbf{S}$ é uma matriz simétrica e portanto diagonalizável. Os autovalores da matriz $\mathbf{D}$ nos dizem sobre a importância de cada uma das componentes principais - maiores valores equivalem à maior importância, pois baixa variância pode estar relacionada ao ruído e, portanto devemos colocar os autovalores em ordem decrescente na matriz. As componentes principais nada mais são do que os autovetores da matriz $\mathbf{S}$.

Decomposição em Valores Singulares

A Decomposição em Valores Singulares (DVS) é uma fatorização tal que M de dimensão $m \times n$ pode ser escrita da seguinte maneira:

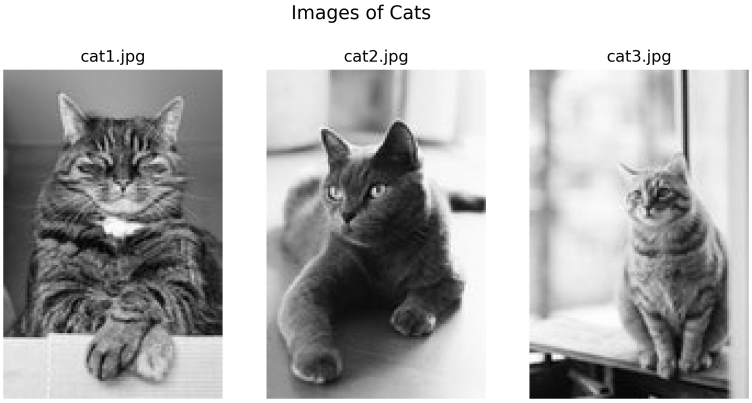
$$\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^T$$

onde $\mathbf{U}$ é uma matriz $m \times m$ ortonormal, Σ é diagonal $m \times n$ e $\mathbf{V}$ $n \times n$ também ortonormal. Podemos dizer que a ACP é um caso particular de SVD e que as componentes principais de $\mathbf{Z}$ ($\mathbf{X}$ já normalizada) são as colunas da matriz $\mathbf{V}$ ortogonal e portanto que $\mathbf{Y} = \mathbf{V}\mathbf{X}$, e os autovalores são dados pela matriz diagonal Σ [1][3]. Lembrando que, por convenção, a DVS já ordena decrescentemente os valores singulares das diagonais de Σ.

Prática

Para essa prática será feito a compressão de três imagens onde será utilizado a técnica de ACP, que será resolvida utilizando-se DVS. Cada uma das imagens foram convertidas para a escala cinza (por simplificação) sendo que cada imagem é representada por uma matriz X $m \times n$ onde m e n são respectivamente a altura (linhas) e largura (colunas), medidas em pixels. A a escala cinza varia numa escala de 0 a 256 em intensidade, pois em cada pixel da imagem é utilizado 8 bits ou 1 byte de informação.

As imagens escolhidas foram de diferentes gatos, retiradas do site **Pexels**, serão chamadass de cat1.jpg, cat2.jpg e cat3.jpg e todas possuem 150 pixels de altura por 100 pixels de largura. Abaixo é possível ver cada uma das imagens em suas qualidades originais:



Compressão de Imagens

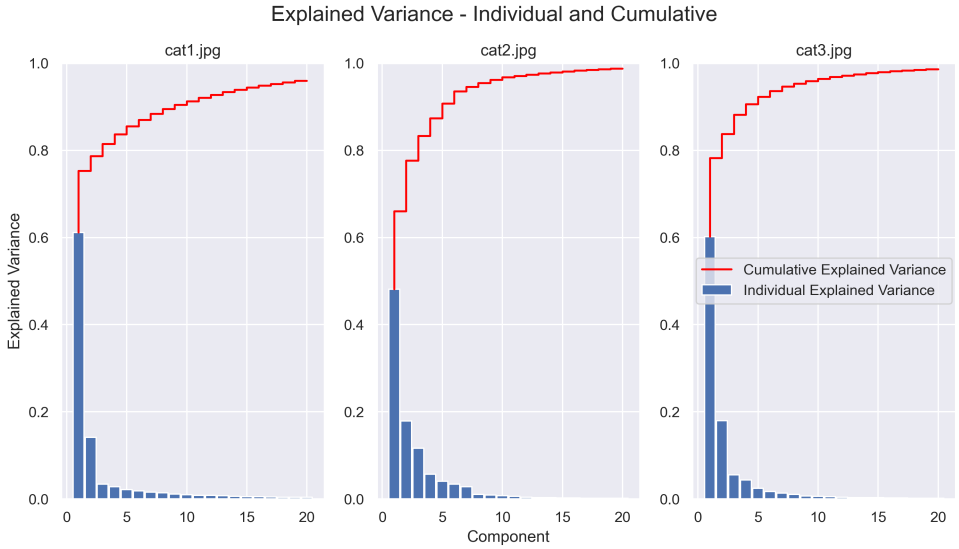
A compressão de cada imagem é feita aplicando-se ACP sobre a matriz $\mathbf{X}$ que representa a imagem utilizando o método de decomposição por valores singulares, obtendo-se as matrizes Σ e $\mathbf{V}$. Após isso escolhe-se um número k de componentes principais a partir dos autovalores de Σ e será feito o truncamento de $\mathbf{V}$ para $\mathbf{V}^{\tilde{}} \in \mathrm{IR}^{\mathrm{mxk}}$ utilizando-se somente as suas primeiras k colunas. Portanto a nova matriz $\mathbf{Y}^{\tilde{}} = \mathbf{V}^{\tilde{}}\mathbf{X}$ terá dimensões pertinentes $(m \times n)$ ao $\mathbf{X}$ original e assim, também pertinentes com as dimensões originais da imagem.
A implementação da ACP utilizando-se DVS e a compressão das imagens estão contidas na seção implementação nas funções definidas "PCA" "image_compression", respectivamente.

Variância Explicada

A variância explicada é uma medida feita sobre a importância de cada uma das componentes principais, calculada usando seus autovalores. Podemos calcular a variância explicada individual (o percentual que podemos explicar dos dados com uma componente principal) dividindo o autovalor pela soma dos autovalores e, também, calcular a variância explicada acumulada (o quanto, em percentual, que conseguimos explicar dos dados com certo número de componentes principais).

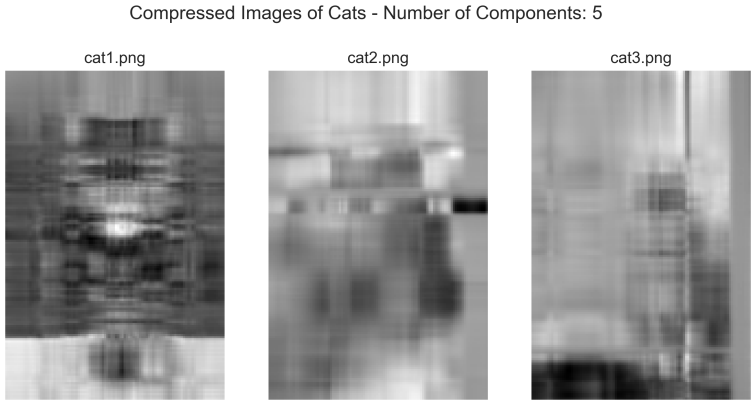
Resultados

Para a variância explicada plotou-se 20 componentes em ordem decrescente de importância para cada uma das imagens e resultou no seguinte:



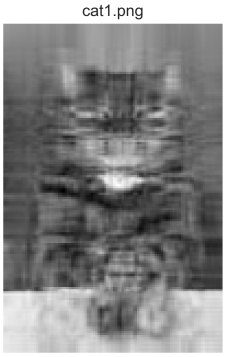
Pode-se perceber que com apenas 5 componentes principais, já conseguimos representar mais do que 80% da imagem original e com 20 componentes já estamos muito próximos dos 100%.

Para esse trabalho foram utilizados cinco valores diferentes para o número k de componentes, sendo eles 5, 10, 20, 40 e 80. Abaixo está o resultado:

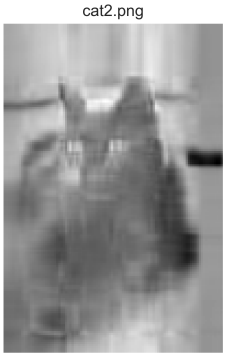


Compressed Images of Cats - Number of Components: 10

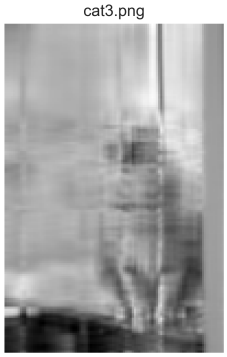
cat1.png



cat2.png

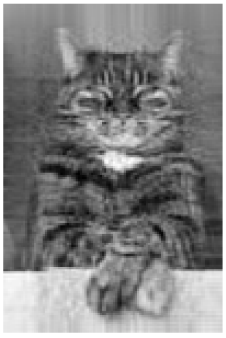


cat3.png



Compressed Images of Cats - Number of Components: 20

cat1.png

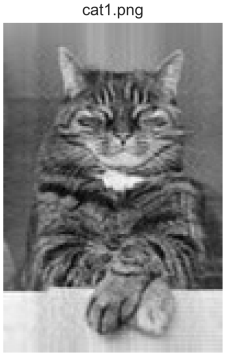


cat2.png

cat3.png

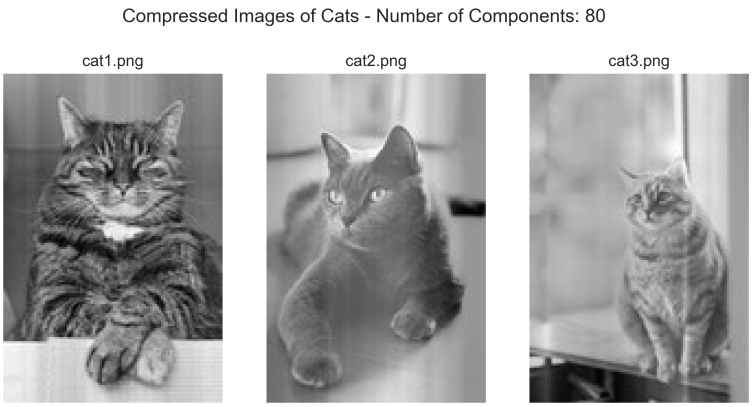
Compressed Images of Cats - Number of Components: 40

cat1.png



cat2.png

cat3.png



A partir das imagens conseguimos ver que utilizando-se 5 e 10 componentes o resultado não é tão satisfatório, apesar de que para as imagens cat2.jpg e cat3.jpg já é possível ver que o resultado é melhor do que para a imagem cat1.jpg, como esperado a partir da análise do gráfico da variância explicada. Também como esperado (variância acumulada maior do que 90%), com 20 componentes já é possível ver claramente os traços de cada imagem, apesar do ruído aparente.
Por fim usando-se 40 e 80 componentes, já não notamos diferenças expressivas comparando-se as imagens cat2.jpg e cat3.jpg, e para cat1.jpg notamos a redução de um ruído fraco restante na imagem.

Também foi feita a análise da diminuição do tamanho em bytes da imagem original para a imagem comprimida, como é possível ver na tabela abaixo:

Percentual Change from Original to Compressed File				
Compressed File	**Original File**	**Original Size (Bytes)**	**Compressed Size (Bytes)**	**% Change**
cat1_PCA_5.jpg	cat1.jpg	4428	2903	34,4%
cat1_PCA_10.jpg	cat1.jpg	4428	3277	26,0%
cat1_PCA_20.jpg	cat1.jpg	4428	3790	14,4%
cat1_PCA_40.jpg	cat1.jpg	4428	4051	8,5%
cat1_PCA_80.jpg	cat1.jpg	4428	4207	5,0%
cat2_PCA_5.jpg	cat2.jpg	3334	2214	33,6%
cat2_PCA_10.jpg	cat2.jpg	3334	2427	27,2%
cat2_PCA_20.jpg	cat2.jpg	3334	2687	19,4%
cat2_PCA_40.jpg	cat2.jpg	3334	2717	18,5%
cat2_PCA_80.jpg	cat2.jpg	3334	2755	17,4%
cat3_PCA_5.jpg	cat3.jpg	3523	2313	34,3%
cat3_PCA_10.jpg	cat3.jpg	3523	2652	24,7%
cat3_PCA_20.jpg	cat3.jpg	3523	2853	19,0%
cat3_PCA_40.jpg	cat3.jpg	3523	2911	17,4%
cat3_PCA_80.jpg	cat3.jpg	3523	2929	16,9%

As imagens comprimidas estão representadas com o seu número de componentes no nome e como esperado, temos uma diminuição muito maior para as imagens com 5 componentes. Anteriormente, vimos que com 20 componentes já é possível distinguir as imagens umas das outras, a tabela acima nos mostra uma diminuição de 14 a 19% para as compressões feitas com esse número de componentes, tornando tal resultado viável de aplicação real, como por exemplo o de diminuir o tempo de treinamento de determinado modelo que usa uma grande base de dados de imagens para reconhecimento de diferentes tipos de produtos, o que resultaria em um tempo mais rápido processamento e talvez com pouca perda em sua eficácia.

Implementação

O código feito em python com a implementação da ACP e compressão de imagens encontra-se abaixo:

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
from scipy.linalg import svd
from sklearn.preprocessing import StandardScaler
import pandas as pd

#reading images
path = 'data/cat_imgs/'
imgs = [Image.open(path + file).convert('L') for file in os.listdir(path)]
imgs = [np.array(img) / 255 for img in imgs]
#saving images grayscale
for i in range(len(imgs)):
    plt.imsave('figs/comp_cats_img/cat{}.jpg'.format(i+1), imgs[i], cmap='gray')

def plot_images(imgs, title):
    ncols=len(imgs)
    plt.figure()
    fig, axs = plt.subplots(1, ncols, figsize=(10,5))
    fig.suptitle("Images of Cats ", fontsize=14)

    for i in range(ncols):
        axs[i].set_title('cat{}.jpg'.format(i+1))
        axs[i].imshow(imgs[i], cmap='gray')
        axs[i].axis('off')
    plt.savefig('figs/{}.png'.format(title), dpi=300)

plot_images(imgs, 'cats')

def PCA(X):

    #standardizing data
    scaler = StandardScaler()
    X_std = scaler.fit_transform(X)

    m, n = X.shape

    #covariance matrix
    cov_matrix = np.dot(X_std, np.transpose(X_std)) / (n - 1)

    #applying Singular Value Decomposition - SVD
    U, S, V = svd(cov_matrix)

    #explained variance
    explained_var = [eig_component / sum(S) for eig_component in S]
    cumulative_explained_var = np.cumsum(explained_var)

    return U, S, V, explained_var, cumulative_explained_var

#plotting explained variance for each image

def plot_explained_variance(X, title, n_comp = 20):

    sns.set()
    ncols=len(imgs)
    plt.figure()
    fig, axs = plt.subplots(1, ncols, figsize=(12,6))
    fig.suptitle('Explained Variance - Individual and Cumulative', fontsize=16)
```

```
    for i in range(ncols):

        axs[i].set_title('cat{}.jpg'.format(i+1))

        U, S, V, explained_var, cum_expained_var = PCA(X[i])

        if(i==0):
            axs[i].set_ylabel('Explained Variance')
        if(i==1):
            axs[i].set_xlabel('Component')

        axs[i].set_ylim([0, 1])
        #barplot attributes
        x = range(1, len(explained_var[:n_comp])+1)
        height = explained_var[:n_comp]
        width = 0.9

        axs[i].bar(x, height, alpha=1, width=width, label='Individual \
        Explained Variance')
        axs[i].step(x, cum_expained_var[:n_comp], color='red', \
        label='Cumulative Explained Variance')
    plt.legend(loc='center')
    plt.savefig('figs/{}.png'.format(title), dpi=300)


plot_explained_variance(imgs, 'explained_variance')

#performing compression using SVD and truncation of the matrix V
def image_compression(X, n_comp):
    compressed_imgs = []
    for x in X:
        U, S, V, explained_var, cum_expained_var = PCA(x)
        VV = V[:150][:n_comp] #truncate V (mxk, where k<m)
        YY = np.dot(VV, x) #YY (kxn)
        XX = np.dot(np.transpose(VV), YY) #compressed image X (mxn, same size as the original)
        compressed_imgs.append(XX)
    return compressed_imgs

#plotting compressed images
def plot_compressed_images(imgs, title, subtitle):
    ncols=len(imgs)
    plt.figure()
    fig, axs = plt.subplots(1, ncols, figsize=(10,5))
    fig.suptitle(subtitle, fontsize=14)

    for i in range(ncols):
        axs[i].set_title('cat{}.png'.format(i+1))
        axs[i].imshow(imgs[i], cmap='gray')
        axs[i].axis('off')
    plt.savefig('figs/comp_cats_img/{}.jpg'.format(title), dpi=300)

#performing comparisons and saving new compressed images
n_comps = [5, 10, 20, 40, 80] # number of components used to represent the data

for n_comp in n_comps:
    compressed_imgs = image_compression(imgs, n_comp)
    subtitle = 'Compressed Images of Cats - Number of Components: {}'.format(n_comp)
    plot_compressed_images(compressed_imgs, 'comp_cats_PCA_{}'.format(n_comp), subtitle)

    #saving individual compressed images
    j = 1
    for compressed_img in compressed_imgs:
```

```
        plt.imsave('figs/comp_cats_img/cat{}_PCA_{}.jpg'.format(j,  n_comp), \
        compressed_img, cmap='gray')
        j+=1

#imgs sizes in Bytes
path = 'figs/comp_cats_img/'
normal_sizes = pd.DataFrame(
    [{'file': file,
      'size': os.path.getsize(path+file)}
       for file in os.listdir(path) if 'PCA' not in file and 'jpg' in file]
)

compressed_sizes = pd.DataFrame(
    [{'file': file,
      'size': os.path.getsize(path+file)}
       for file in os.listdir(path) if 'jpg' in file and 'PCA' in file and 'comp' not in file]
)
```

Referências

[1] Richardson, M. [online] Dsc.ufcg.edu.br. Available at: http://www.dsc.ufcg.edu.br/~hmg/disciplinas/posgraduacao/rn-copin-2014.3/material/SignalProcPCA.pdf [Accessed 9 July 2020].

[2] Jolliffe, I. and Cadima, J., 2016. Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), p.20150202. [online] Available at: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2015.0202 [Accessed 9 July 2020]

[3] Dr. Sebastian Raschka. 2020. Principal Component Analysis. [online] Available at: https://sebastianraschka.com/Articles/2015_pca_in_3_steps.html [Accessed 9 July 2020]. [4] En.wikipedia.org. 2020. Singular Value Decomposition. [online] Available at: https://en.wikipedia.org/wiki/Singular_value_decomposition [Accessed 9 July 2020].

[5] En.wikipedia.org. 2020. Principal Component Analysis. [online] Available at: https://en.wikipedia.org/wiki/Principal_component_analysis [Accessed 9 July 2020].