# Projeto 3 - MS960/MT862

Fernando Ribeiro de Senna — RA 197019
Rodolfo da Silva Santos — RA 228711

11 de dezembro de 2020

Na primeira parte do projeto, ***COMPLETAR***.

Já na segunda parte do projeto, foi implementada ferramenta de Análise de Componentes Principais (PCA), com o objetivo de realizar redução de dimensionalidade para trabalhar com reconhecimento facial. A implementação foi feita no arquivo *PCA_part2.ipynb* e está descrita na Seção 3.

Foram utilizados funções e objetos das bibliotecas *scipy, numpy* e *matplotlib*.

Toda a fundamentação teórica se baseia em conteúdo oferecido em vídeo-aulas e *slides* pelo Professor João Batista Florindo em ocasião de oferecimento da disciplina MS960 no segundo semestre de 2020 pelo Instituto de Matemática, Estatística e Computação Científica (IMECC) da Universidade Estadual de Campinas (UNICAMP).

## 1 Documentação

*** COMPLETAR ***

## 2 *** — Parte I

## 3 Análise de Componentes Principais (PCA) — Parte II

As operações descritas nessa Seção estão no arquivo *PCA_part2.ipynb* e são relativas à segunda parte do projeto, que realiza Análise de Componentes Principais a fim de realizar redução de dimensionalidade para problemas de reconhecimento facial.

A Análise de Componentes Principais (PCA) tem por objetivo reduzir o excesso de atributos dos exemplos de treinamento a fim de diminuir os custos computacionais do treinamento a ser realizado. Para isso, é utilizado o conceito de projeção. A ideia básica é projetar o espaço de n atributos num espaço de menor dimensionalidade, com dimensão k, em que k é chamado número de componentes principais. O valor do número de componentes principais depende do problema a ser resolvido.

No caso do reconhecimento facial, essa projeção consiste em obter k "faces padrão", chamadas *eigenfaces*, e definir cada uma das imagens originais de faces como uma combinação linear dessas *eigenfaces.*

## 3.1 Importação e Visualização dos Exemplos de Treinamento

Inicialmente, os dados de treinamento foram extraídos do arquivo *dado3.mat*, utilizando a função *loadmat* da biblioteca *scipy.io.* A matriz de exemplos de treinamento é chamada de X, em que cada uma das 5000 linhas representa um exemplo de treinamento e cada uma das 1024 colunas representa um atributo. Nesse, caso, o vetor de atributos representa a linearização de imagens de dimensão $32 \times 32$.

Em seguida, foram selecionadas aleatoriamente 100 imagens utilizando a função *random.choice* da biblioteca *numpy*. A fim de visulizar uma fração dos exemplos de treinamento, essas 100 imagens foram ilustradas com auxílio de funções da biblioteca *matplotlib*. O resultado está na Figura 1.

Figura 1: Representação de 100 imagens do conjunto de exemplos de treinamento

## 3.2 Determinação e Visualização das *eigenfaces*

O processo de "treinamento" do PCA consiste em utilizar como base do novo espaço de atributos os autovetores da matriz de covariância empírica ($\Sigma$) dos atributos que correspondam aos k maiores autovalores de $\Sigma$.

Esse processo é realizado obtendo-se, inicialmente essa matriz de covariância através da Equação 1. Em seguida, utiliza-se a função *svd* da biblioteca *numpy.linalg* para obter uma matriz (U), em que cada coluna é um dos autovetores de $\Sigma$ já ordenados com base na ordem decrescente dos autovalores correspondentes, e um vetor (S) em que cada entrada é um valor singular de $\Sigma$ obedecendo a mesma ordem da matriz de autovetores (U).

$$\Sigma = \frac{1}{m} X^t X \tag{1}$$

A Figura 2 apresenta as 36 principais *eigenfaces*, isto é, a representação gráfica dos 36 autovetores de $\Sigma$ cujos correspondentes autovalores são os maiores.

Figura 2: Representação visual das 36 principais *eigenfaces*.

É interessante observar que as *eigenfaces* se parecem com imagens de faces com pouca nitidez e detalhes, o que é coerente com o que se espera, pois, com o PCA, deseja-se obter as *eigenfaces* a fim de ser possível representar uma imagem de face de uma pessoa como a combinação linear de faces consideradas comuns pelo algoritmo de treinamento.

## 3.3 Projeção das Faces

Uma forma útil de avaliar se de fato as *eigenfaces* obtidas são boas é visualizar as faces dos exemplos de treinamento como combinação linear das *eigenfaces*.

Para obter esse resultado, utilizamos as 100 principais *eigenfaces* e projetamos os exemplos de treinamento sobre elas. Isso pode ser feito definindo a matriz $U_{red}$ como as 100 primeiras colunas da matriz U obtida anteriormente e definindo a projeção das imagens do conjunto de treinamento como $z = XU_{red}$. A fim de ser possível visualizar essas projeções como imagens de faces, é necessário, então, restituí-las ao espaço de atributos original, o que pode ser feito fazendo $proj = zU^t_{red}$.

As Figuras 3 e 4 apresentam a comparação entre as 100 faces originais e os resultados obtidos com as projeções utilizando as 100 principais *eigenfaces*. Nessas figuras, as linhas ímpares apresentam as imagens originais e, abaixo, nas linhas pares, estão os resultados das projeções, a fim de facilitar a comparação. As faces utilizadas para realizar essa representação são as mesmas que as presentes na Figura 1, que foram selecionadas aleatoriamente.

É interessante notar como as imagens obtidas pela projeção são similares às originais, indicando bom desempenho do processo de redução de dimensionalidade realizado. É evidente que as imagens originais apresentam mais detalhes e são mais nítidas. Contudo, as imagens originadas por projeção são bastante claras e permitem (a um humano) o reconhecimento facial das pessoas com facilidade.

Considerando que o objetivo dessa redução de dimensionalidade é justamente possibilitar o treinamento de algoritmos de reconhecimento facial com menor custo computacional, a possibilidade de reconhecer as pessoas a partir das imagens projetas em um espaço de dimensionalidade de menos de 10% das dimensões do espaço original é altamente significativa, devido à grande redução de custo computacional que isso implica.

É importante, todavia, ressaltar que seria necessário realizar testes sobre os algoritmos de reconhecimento facial para verificar qual é o número de componentes principais ideal que permita ao algoritmo reconhecer as faces com custo computacional adequado. A Seção 3.4 discute um pouco essa questão.

## 3.4 Fração da Variância Original Mantida

Uma forma de medir a perda de informação gerada pelo PCA é avaliar a fração da variância original mantida. Quanto maior for essa fração, menor a perda de informação. Essa fração pode ser obtida pela equação 2, em que k é o número de componentes principais utilizado e n é o número de atributos do dado original.

Figura 3: Comparação entre imagens originais e obtidas a partir da projeção com 100 *eigenfaces.*

Figura 4: Comparação entre imagens originais e obtidas a partir da projeção com 100 *eigenfaces.*

$$\frac{\sum_{i=1}^{k} S_i}{\sum_{i=1}^{n} S_i} \tag{2}$$

A Figura 5 apresenta um gráfico da evolução da fração da variância original mantida em função do número de componentes principais. Conforme esperado, quanto maior o número de componentes principais, maior essa fração e, portanto, menor a perda de informação. É interessante observar, contudo, como um número relativamente baixo de componentes principais já garante um valor grande dessa fração e consequente pequena perda de informação. Para k=100, por exemplo, como utilizado na Seção 3.3, há cerca de 94% da manutenção da variância original. Com 150 componentes principais, o que corresponde a pouco mais do que 10% da dimensão original dos dados, mais de 96% da variância original é mantida e, a partir de 350 *eigenfaces*, que representa cerca de um terço do número original de atributos, já há manutenção de mais do que 99% da variância original.

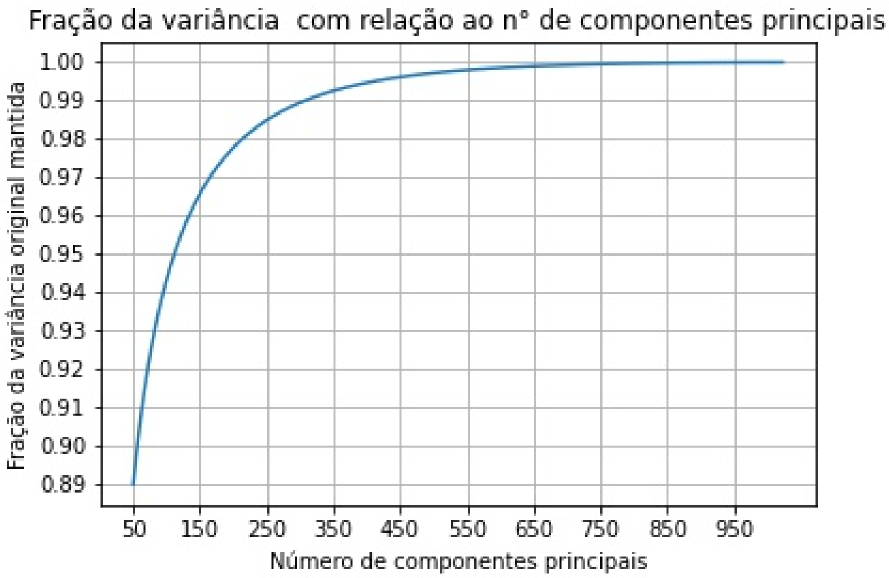


Figura 5: Evolução da fração da variância original mantida em relação ao número de componentes principais.

Portanto, há grande potencial de aplicação da técnica de PCA para facilitar o treinamento de algoritmos para realização de reconhecimento facial (e de fato essa técnica é muito utilizada na prática).

# 4 Referências

Vídeo-aulas e *slides* pelo Professor João Batista Florindo em ocasião de oferecimento da disciplina MS960 no segundo semestre de 2020 pelo Instituto de Matemática, Estatística e Computação Científica (IMECC) da Universidade Estadual de Campinas (UNICAMP).