Universidade Estadual do Ceará
Centro de Ciências e Tecnologia (CCT)

Reconhecimento Facial utilizando o algoritmo de Decomposição em Valor Singular (SVD), em Python, com o banco de dados "Extended Yale-B".

Equipe α_2: Arthur David (1538730), Emerson Lucena (1538695), Gustavo Cesar (1538617) e Yaslim Soares (1538806).

Professor orientador: Thelmo de Araújo

Fevereiro
2021

Universidade Estadual do Ceará
Centro de Ciências e Tecnologia (CCT)

Relatório

Relatório produzido pelos integrantes da equipe α_2, do curso de Ciência da Computação - UECE, sobre reconhecimento facial aplicando redução de dimensionalidade com Decomposição em Valor Singular (SVD).

Equipe α_2: Arthur David (1538730), Emerson Lucena (1538695), Gustavo Cesar (65431)
e Yaslim Soares (1538617).

Professor orientador: Thelmo de Araújo

Fevereiro
2021

Conteúdo

1 Resumo

Uma imagem, dependendo de sua resolução, possui milhares de pixels. Quando se tem o objetivo de analisá-los computacionalmente, eles são armazenados em uma matriz que pode ter milhares, talvez milhões, de colunas e linhas, o que dificulta o seu processamento. Sendo assim, a aplicação do algoritmo de Decomposição em Valores Singulares, do inglês Singular Value Decomposition (SVD), reduz as dimensões, por meio de arranjos matemáticos, das matrizes de dados, viabilizando o processamento destas. Este trabalho tem como objetivo realizar o reconhecimento facial, classificando fotografias do banco de dados "The Extended Yale Face Database B", por intermédio da linguagem de programação Python, com a melhor acurácia possível, utilizando o SVD. Este projeto obteve uma acurácia de 77.24% na classificação dos indivíduos utilizando cerca de 10% dos dados, o que reduziu o problema de 1689 colunas à 169, e utilizando apenas uma variabilidade acumulada de 45%.

Palavras-chave: Decomposição em Valores Singulares. SVD. Reconhecimento Facial. The extended Yale Face Database B. Python.Singular Value Decomposition,

2 Introdução

Reconhecimento facial não é um tema atual. Já na década de 60 [1], engenheiros iniciavam seus trabalhos na área, mesmo que os estudos tenham se consolidado dez anos depois [2]. Apesar dos anos de pesquisas sobre o reconhecimento de faces, ainda assim é um processo difícil e delicado desenvolver modelos com ínfimas taxas de erro, tendo em vista que, devido às condições adversas das imagens, como iluminação, expressões faciais e forma de captura do perfil, dificultam a existência de um único algoritmo de resolução com taxa de acerto alta e constante.

De acordo com Juliano Oliveira [3], a análise de componentes principais, comumente chamada de PCA, do inglês Principal Component Analysis, é uma das mais fundamentais ferramentas de uma análise multivariada pois ela constitui uma base para outros métodos multivariados de análise de dados. Além disso, também nos é revelado se existe ou não relação entre as variáveis da nossa problemática, informação essa que é crucial para o desenvolvimento da nossa aplicação. Basicamente, o que é importante para o nosso problema, é compreender que as variáveis ou características, dos dados que estamos analisando, tem uma relação entre si, e para fazermos uma análise de dados convincente e coerente, devemos eliminar essa relação. Uma vez sem relação, é possível examinar com mais precisão, o que torna cada característica determinante, e desenvolver melhores deduções a respeito do problema.

É importante ressaltar que, por meio de métodos computacionais, é possível fazer uma correlação entre imagens e matrizes. Uma imagem é basicamente uma matriz de dimensão m x n, na qual M e N, são os pixels da imagem, ou

seja, podemos "convertê-la"para uma matriz de dados m x n, que, será mais útil para as operações que iremos realizar.

Quando trabalhamos com conjuntos de dados de grandes dimensões, as variáveis possuem dependências entre si, o que se configura uma adversidade ao se tratar de análise de dados envolvendo reconhecimento facial, tendo em vista que imagens possuem milhares pixels a serem analisados, o que demanda bastante tempo de processamento, e, além disso, o número significante de características comuns podem atrapalhar o resultado final.

Com base nisso, uma das soluções para a redução de dimensionalidade, é a decomposição em valor singular, ou comumente abreviada, SVD. A SVD, a grosso modo, nos entrega uma matriz "reduzida"mais aproximada possível dos dados originais, viabilizando a manipulação da matriz original.

Em síntese, neste trabalho, iremos utilizar a SVD, com acompanhamento da linguagem de programação Python (manuseando a biblioteca **numpy**), para analisarmos as imagens, presentes no banco de dados "Extended Yale-B", em forma de matriz de dados, e realizarmos o reconhecimento de faces com uma acurácia significativa.

3 Fundamentação Teórica

O reconhecimento facial é baseado em características únicas em diferentes pessoas, que podem ser combinadas ou não. Imagens com pixels muito semelhantes não são fatores determinantes para reconhecer faces, tendo em vista que a semelhança pode ser facilmente prevista, já que elas tendem a seguir um padrão. Sendo assim, se torna mais viável retirar esses dados repetitivos e diminuir a matriz de dados, agilizando o seu processamento.

Sendo assim, faremos isso utilizando Decomposição em Valores Singulares. Para isso, ante o empecilho de que qualquer matriz quadrada não é diagonalizável, encontraremos uma matriz T, triangular, similar a uma matriz quadrada A e U uma matriz ortogonal, tendo em vista que os dados são números reais, [4] de forma que:

$$A = UTU^t.$$

Todavia, matrizes diagonais se tornam mais fáceis de manipular, o que nos faz adotar a seguinte notação como a decomposição de uma matriz A_{mxn} [3]:

$$A = U\Sigma V^t.$$

3.1 Decomposição em Valores Singulares

A Decomposição em Valores Singulares, do inglês Singular Value Decomposition (SVD), é uma fatoração de uma matriz qualquer que nos retorna três outras componentes que multiplicadas retornam à matriz original, e por isso, nos fornecem informações muito utéis. Essa técnica possui diversas aplicações,

seja ela em resultados retirados das suas componentes, ou como passo de um determinado algoritmo [3].

A decomposição em valores singulares possui dois tipos, a decomposição em valores singulares completa e a reduzida [4].

3.1.1 Decomposição em valores singulares reduzida

Dada uma matriz A_{mxn} com $m \geq n$, a decomposição em valores singulares de A (SVD de A) é do tipo reduzida e possui a fatorização $A = \hat{U}\hat{\Sigma}V^t$ [3], na qual:

- $\hat{U}$ é uma matriz m x n ortogonal;
- V é uma matriz n x n ortogonal;
- $\hat{\Sigma}$ é uma matriz n x n diagonal.

Conforme a Figura 1, os valores da diagonal principal de $\hat{\Sigma}$ $\sigma_1, \sigma_2, ..., \sigma_n$ são chamados de valores singulares de A, e vale ressaltar que: $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$. As colunas de $\hat{U}$, $u_1, u_2, ..., u_m$, são denominadas de vetores singulares à esquerda de A. E as colunas da matriz V, $v_1, v_2, ..., v_n$, são denominadas de vetores singulares à direita de A [3].

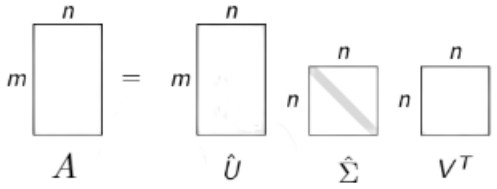

Figura 1: Decomposição em valores singulares reduzida [4]

Quando m < n, devemos realizar algumas manipulações nas componentes derivadas da SVD. Primeiramente devemos "consertar"a dimensão da matriz $\hat{U}$, para isso, devemos completar as colunas de $\hat{U}$, até termos uma base do $\mathbb{R}^m$, e tornaremos a base ortogonal aplicando Gram-Schmidt. Após isso devemos "consertar"as dimensões de $\hat{\Sigma}$, para fazer isso basta adicionar $(m - n)$ linhas de zero até obtermos a dimensão desejada. Feito esses passos, obtemos o que chamamos de SVD completa.

3.1.2 Decomposição em valores singulares completa

Dado uma matriz A_{mxn}, a Decomposição em Valores Singulares de A (SVD de A) é a fatorização $U\Sigma V^t$ na qual:

- U é uma matriz m x m ortogonal;

- V é uma matriz n x n ortogonal;
- Σ é uma matriz m x n.

Na SVD completa, temos a mesma decomposição, o que muda são apenas as dimensões de suas componentes, a nomenclatura continua a mesma, conforme a Figura 2.

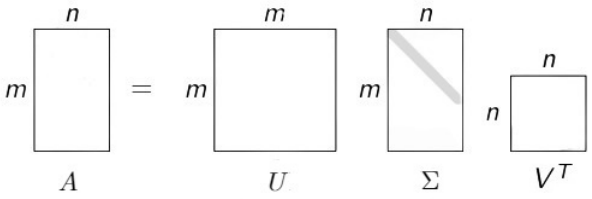

Figura 2: Decomposição em valores singulares completa [4]

3.2 Aplicação

3.2.1 Leitura computacional das imagens

Uma imagem é armazenada em um computador em forma de uma matriz com células preenchidas por pixels. O pixel é um ponto que corresponde a uma cor na figura que em uma matriz equivale a um valor entre 0 e 255, e, no caso de imagens coloridas, cada pixel pode conter outras três componentes de cores vermelha, verde e azul, se esta corresponder ao famoso RGB (Red, Green, Blue) [3].

Além disso, a dimensão da imagem corresponde ao tanto de pixels que ela possui. Sendo assim, a leitura de uma figura com dimensões m x n é feita e armazenada em uma matriz de dimensão m x n. Esta matriz, possui pontos com valores de 0 à 255, que, por convenção, será representada como tonalidades de cinza, de modo que 0 equivale a preto e 255 a branco [3].

É importante ressaltar que os dados utilizados no reconhecimento de faces, ou seja, as matrizes correspondentes as fotografias dos rostos das diferentes pessoas, serão divididas em conjuntos de treino e de teste. A matriz de treino equivale a uma única matriz M correspondente à todas as outras matrizes de imagens do conjunto de treino enfileiradas.

3.2.2 Centralizando os dados

Diante da irrelevância de características não determinantes em uma face, como a testa e as bochechas, é interessante remover estes dados com o fito de comprimir mais ainda a imagem e analisá-la de forma eficiente. Para fazer a centralização de dados, a matriz M, a nossa matriz de teste, será centralizada

realizando uma média das características dela e subtraindo de cada célula, removendo assim os pontos mais comuns entre todas elas, o que conservará as regiões mais relevantes para o reconhecimento de faces.

3.2.3 Redução de Dimensionalidade

Tendo em vista os conceitos acima, podemos aplicar a SVD em uma situação que precisamos reduzir a dimensionalidade da nossa matriz de informações.

Suponha que exista uma matriz de dados A_{mxn} e queremos reduzi-la. O "algoritmo"adotado nesta problemática consiste em algumas etapas. Após realizar a centralização dos dados da matriz A, podemos aplicar SVD em A. Como visto anteriormente, essa decomposição retorna as componentes U, Σ e V^t, e, com essas informações, será feita a redução de dimensionalidade. Para reduzirmos essa matriz, devemos escolher a quantidade de vetores singulares esquerdos, que será chamado de r. Após isso, iremos compor uma nova matriz U com os vetores singulares esquerdos até o r. Ao final dos passos acima, obtemos: $U = \begin{bmatrix} u_1 & u_2 & ... & u_r \end{bmatrix}$. Está feita a redução de dimensionalidade na matriz A_{mxn}.

Com o número de vetores singulares esquerdos que utilizamos, também temos a quantidade de valores singulares que iremos utilizar, com isso em mente, podemos inferir a energia acumulada que eles representam com a seguinte fórmula:

$$E(r) = \frac{\sigma_1^2 + \sigma_2^2 + \sigma_1^3 + ... + \sigma_r^2}{\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + ... + \sigma_n^2}$$

Em alguns projetos, utiliza-se cerca de 85% ou 90% de energia acumulada, podendo assim manipular, a quantidade de vetores singulares esquerdos que iremos utilizar a partir dessa informação.

4 Metodologia e Experimentos

Para realizar o reconhecimento e a classificação das faces do banco de dados *Extended Yale-B*, nós precisamos primeiramente carregar as imagens (em formato.pgm) do mesmo. Para isso, transformamos todas as fotos (2414 imagens de 38 pessoas, com aproximadamente 64 fotos em diferentes iluminações cada), que originalmente possuem dimensão 192 por 168 pixels, em vetores de 32256 elementos. Posteriormente, separamos em dois grupos de vetores, de treino, que o computador irá usar para "aprender a reconhecer um rosto", e de teste, que iremos usar para fazer o reconhecimento e a classificação – em uma proporção de **70%** para treino e **30%** para teste (1689 imagens de treino e 725 de teste).

Após a divisão, organizaremos os vetores de treino lado a lado para criar uma matriz (**M**) com os dados. A matriz **M** terá dimensão 32256 por 1689. O próximo passo do processo é centralizar os dados, que é feito subtraindo a média das características (uma "face média") de todas as faces de treino. Em seguida, usaremos a função **numpy.linalg.svd()** para realizar a decomposição da nossa matriz de dados. Com isso, teremos então duas matrizes e um vetor:

u (32256 por 1689), uma matriz com colunas ortonormais onde estão os vetores singulares esquerdos de M, **s** (1689 elementos), um vetor com os elementos da diagonal principal de uma matriz S, cujos elementos são os valores singulares de M, e por ultimo **vt** (1689 por 1689), uma matriz quadrada com colunas ortonormais onde estão os vetores singulares direitos de M.

Assim como transformamos imagens de 192 por 168 pixels em vetores de 32256 elementos, podemos fazer o processo inverso, transformando cada coluna de u em uma imagem com as dimensões originais. Com isso, obteremos as chamadas *"eigenfaces"*, que são rostos artificiais com destaques para diferentes fatores que caracterizam uma face. Segue exemplos de eigenfaces na Figura 3:

Figura 3: As 20 primeiras *eigenfaces*

Depois de tudo isso feito, estamos prontos para realizar a redução de dimensionalidade, que consiste em usar uma determinada quantidade (r, que no caso usamos r=169) de *eigenfaces* para classificar os indivíduos. Para isso, devemos encontrar uma espécie de "combinação linear"das imagens de teste em relação a uma "base"de vetores singulares (*eigenfaces*).

A forma de encontrar essa combinação é multiplicando as r primeiras colunas de vetores singulares de u transposto (Ur^t) por uma imagem em formato de vetor. Com isso, obteremos um vetor contendo os 169 coeficientes da combinação linear, que será utilizado para comparar a proximidade entre duas imagens, através da distância euclidiana dos seus respectivos vetores.

No nosso programa, geramos uma lista contendo todos os vetores de coeficientes da combinação linear (vamos chamar de **VCCL** para facilitar) de cada uma das imagens de treino com as 169 primeiras eigenfaces, afim de otimizar o processo.

Com essa lista gerada, percorremos nossas 725 imagens de teste realizando o processo: transformar em vetor; multiplicar Ur^t pelo vetor; calcular a distancia euclidiana entre nosso VCCL obtido e cada um dos **VCCLs** da lista. Ao fim, a imagem de treino cujo VCCL possuir a menor distância para o **VCCL** da imagem de teste deve corresponder ao indivíduo (classe) correto. Alguns exemplos estão presentes na Figura 4:

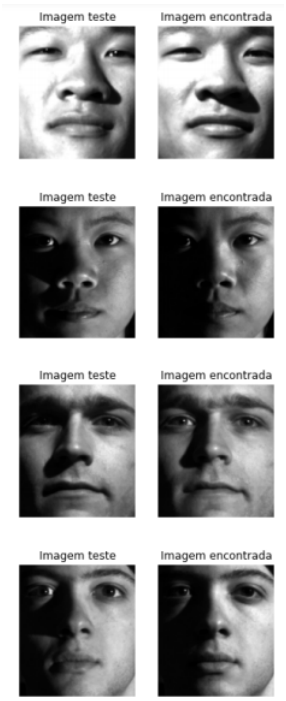

Figura 4: Experimentos

Vale ressaltar, porém, que nem sempre o programa acerta, principalmente em piores condições de iluminação, como podemos ver na Figura 5:

Figura 5: Exemplo em que ocorre um erro

5 Resultados

Após todo o procedimento ser realizado, chegamos a algumas conclusões e resultados. A quantidade r de eigenfaces que utilizamos impacta diretamente na precisão do programa. Isso se dá pelo fato de que, quanto mais *eigenfaces*, mais informações (parâmetros) para realizar a comparação temos, consequentemente variabilidade acumulada ("detalhes"). A seguir, na Figura 6, um gráfico com a relação entre quantidade de *eigenfaces* e variabilidade acumulada:

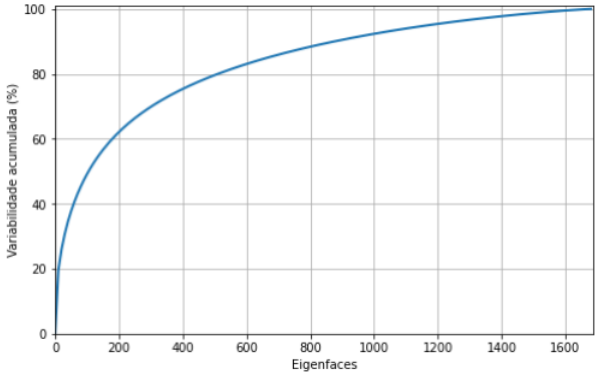

Figura 6: Variabilidade acumulada x Quantidade de *eigenfaces*

Ao escolhermos $r = 169$, estamos usando aproximadamente 10% das eigenfaces, o que resulta em 59% de variabilidade acumulada. Com apenas essa quantidade de informação, nosso programa obteve uma acurácia de 77.24% (560 acertos, 165 erros). A seguir, na Figura 7, um gráfico com a relação entre a variabilidade acumulada utilizada e a acurácia do programa:

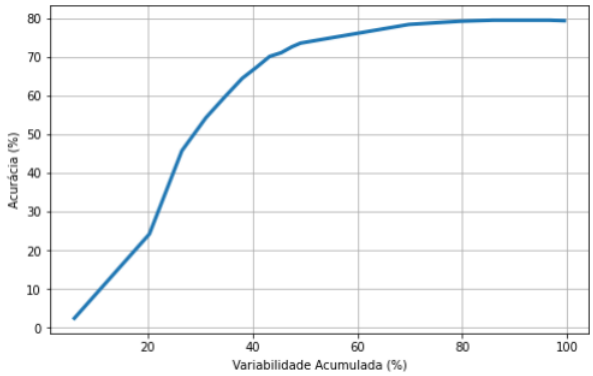

Figura 7: Variabilidade acumulada x Acurácia

6 Conclusão e Trabalhos Futuros

Em suma, obtemos uma acurácia relativamente alta de 77.24% utilizando r = 169 vetores singulares, o que corresponde a 59% de variabilidade acumulada e 10% dos vetores singulares equivalentes às *eigenfaces*. Isso se dá pelo fato de que as condições das imagens são bem variadas, de forma que algumas figuras, do banco de dados, possuem iluminação bem escassa, o que contribuiu para a diminuição da taxa de acerto. Acima de 59% de variabilidade acumulada, percebe-se que não é interessante utilizar mais vetores para um aumento de acurácia pouco significativo, o que gera um maior tempo de processamento utilizando quase a totatalidade das *eigenfaces* a troco de um aumento de, aproximadamente, 3% de acurácia. Sendo assim, fornecer mais características que as utilizadas neste programa não retorna uma melhora eficiente.

Em trabalhos futuros estudaremos os efeitos que a iluminação causa no reconhecimento facial, e, com essa informação, além de melhorar a acurácia, iremos aplicar a classificação em pessoas com acessórios - como óculos e máscaras - já que esses objetos influenciam na visibilidade do rosto por inteiro, e, na realidade atual, faz-se necessário a existência de um sistema capaz de fazer esse reconhecimento.

Referências

[1] M. Bledsoe., *The models method in facial recognition.* Technical Report PRI 15, Panoramic Research Inc, Palo, CA, 1964.

[2] M. Kelly., *Visual identification of people by computer.* Technical Report AI 130, Stanford, CA, 1970.

[3] J. V. d. Oliveira *et al.*, *Estudo da decomposição em valores singulares e análise dos componentes principais.* Volta Redonda, 2017.

[4] T. de Araujo, *Álgebra Linear: Teoria e Aplicações.* SBM. Rio de Janeiro, 2014.