2^o Trabalho de Álgebra Linear para Computação

Gerardo Rodrigues Carneiro Filho,
Maria Clara Mesquita Moura Ferreira,
Rafael de Almeida Menezes,
Lucas Campelo Santiago

20/04/2020

Resumo

Este trabalho tem por objetivo apresentar uma aplicação computacional da Álgebra Linear em compressão de imagens. Utiliza-se da técnica de decomposição de valor singular (ou, do inglês, a sigla SVD, abreviada de *singular value decomposition*), que, sendo esta válida para qualquer matriz quadrada e apresentando um custo operacional menor do que um processo de diagonalização comum, torna-se computacionalmente mais viável. O método utilizado possibilita que sejam geradas versões de uma mesma imagem com diferentes qualidades. O algoritmo fora desenvolvido na linguagem de programação Python, com o auxílio das bibliotecas NumPy e OpenCV. Ademais, é apresentado um estudo matemático que visa justificar a funcionalidade da técnica utilizada, expondo a relação entre esta e outros conceitos da Álgebra Linear e Estatística, como decomposição espectral, autovalores, autovetores, diagonalização e variabilidade.

1 Introdução

Diante do constante crescimento de informação que circula digitalmente na sociedade, desenvolver técnicas que gerenciam e transportam essas informações de maneira eficaz torna-se essencial. Imagens podem ser representadas digitalmente como grupos de milhares de *bits*, ou *pixels*. Visto que as ferramentas de manipulação estão cada vez mais sofisticadas, possibilitando a criação de imagens com grande qualidade, é inevitável a necessidade de técnicas que facilitem também, a circulação desse tipo de informação.

Como já dito anteriormente, imagens são grupos de milhares de *pixels*, grupos estes organizados como matrizes. Uma imagem com dimensões 640×480 *pixels*, por exemplo, é uma matriz com 640 colunas e 480 linhas. Um *pixel* é o menor elemento de uma imagem, cada um contendo três pontos — as cores azul, vermelho e verde — que podem assumir 256 tonalidades cada. Visto que esses valores são manipuláveis, devemos aliar técnicas computacionais e matemáticas, especificamente da Álgebra Linear, para trabalhar com estas matrizes.

A técnica a ser discutida será a *decomposição de valor singular*, também chamada de SVD, de *singular value decomposition*.[de Araujo, 2014] nos diz que o propósito da técnica é decompor uma matriz de forma tal que o esforço computacional seja reduzido ao máximo possível, evitando trabalhar com inversões matriciais e cálculos massivos, como em multiplição de matrizes.

2 Cálculo da decomposição SVD

A decomposição de valor singular faz parte de um conjunto de técnicas chamado *Análise de Componentes Principais*, muito utilizado no campo de análise de dados. De acordo com [de Araujo, 2014], essa decomposição objetiva decompor uma matriz $X_{m \times n}$ de valores reais na forma

$$X = U \Sigma V^T, \tag{1}$$

onde U e V são matrizes ortogonais e Σ é uma matriz diagonal, em que os elementos de sua diagonal principal são os *valores singulares* de X, geralmente dispostos em ordem decrescente. As colunas de U são chamadas *vetores singulares esquerdos*, enquanto as colunas de V são chamadas *vetores singulares direitos*. Isso se dá para toda matriz $X_{m \times n}$.

Suponhamos uma matriz $A = X^T X$ que seja simétrica, o que nos garante, pelo Teorema Espectral, que A pode ser decomposta em

$$A = V \Lambda V^T,$$

onde V é uma matriz ortogonal e Λ uma matriz diagonal. Vale ressaltar que os autovalores λ_i de A, sendo ela simétrica real, também serão reais e, ademais, não negativos. Dessa forma, temos que

$$V \Lambda V^T = A = X^T X = (U \Sigma V^T)^T (U \Sigma V^T) = (V^T)^T \Sigma^T U^T U \Sigma V^T = V \Sigma^T \Sigma V^T;$$

como Σ é diagonal, podemos dizer que $\Sigma^T = \Sigma$, e portanto

$$\Lambda = \Sigma^T \Sigma = \Sigma \Sigma = (\Sigma)^2.$$

Dessa forma concluimos que os autovalores de A são o quadrado dos valores singulares de X, além de que a matriz ortogonal de A é igual a matriz de vetores singulares direitos de X.

Finalmente, para descobrirmos o valor da matriz U, devemos efetuar o seguinte cálculo:

$$\begin{aligned} X &= U\Sigma V^T \\ XV &= U\Sigma. \end{aligned}$$

A seguir, aplicaremos esta técnica na compressão de imagens digitais.

3 Metodologia e Experimentos

Os experimentos foram realizados com o auxílio da linguagem de programação Python, juntamente com a biblioteca NumPy, que nos disponibiliza objetos de matrizes n-dimensionais, e a OpenCV, que nos fornece ferramentas para trabalhar processando imagens. Utilizaremos como objeto de estudo uma imagem de dimensões 512×512 *pixels* em escala de cinza.

De antemão, devemos transformar a imagem em uma matriz que possamos manipular. Isso se dá com um dos métodos providenciados pela biblioteca OpenCV (`imread()`, mais especificamente). Ao aplicar este método na imagem, obtemos uma matriz Y de dimensões iguais a da imagem, com diversos valores de 0 a 255, representando os níveis de cinza. A matriz obtida é

$$Y = \begin{bmatrix} 145 & 144 & \dots & 109 \\ 144 & 144 & \dots & 105 \\ \vdots & \vdots & \ddots & \vdots \\ 37 & 39 & \dots & 90 \end{bmatrix}.$$

Em seguida, devemos transformar todos os elementos da matriz em valores de 0 a 1, dividindo cada um por 255. Em Python isso é feito facilmente com o método `divide()` imbutido na biblioteca NumPy. Realizadas as operações, ficamos com a matriz

$$Y = \begin{bmatrix} 0.56862745 & 0.56470588 & \dots & 0.42745098 \\ 0.56470588 & 0.56470588 & \dots & 0.41176471 \\ \vdots & \vdots & \ddots & \vdots \\ 0.14509804 & 0.15294118 & \dots & 0.35294118 \end{bmatrix}.$$

Devemos agora centralizar as colunas de Y, o que consiste em em calcular a média dos elementos de cada coluna de Y e subtraí-la de sua respectiva

coluna, chamando a matriz resultante de X. Devemos, ademais, armazenar as médias calculadas em uma matriz unidimensional M. Ficamos, então, com as seguintes matrizes

$$X = \begin{bmatrix} 0.25068168 & 0.24689798 & \dots & -0.03383885 \\ 0.24676011 & 0.24689798 & \dots & -0.04952512 \\ \vdots & \vdots & \ddots & \vdots \\ -0.17284773 & -0.16486673 & \dots & -0.10834865 \end{bmatrix} \text{ e}$$

$$M = \begin{bmatrix} 0.31794577 & 0.3178079 & \dots & 0.46128983 \end{bmatrix}.$$

Seguidamente, devemos encontrar a decomposição SVD de X. Felizmente, há um método imbutido na biblioteca NumPy que nos garante isso, método este chamado `linalg.svd()`. O método nos retorna, obviamente, três matrizes; são estas:

$$U = \begin{bmatrix} -0.02455392 & 0.0062014 & \dots & -0.04419417 \\ -0.02449594 & 0.00672264 & \dots & -0.04419417 \\ \vdots & \vdots & \ddots & \vdots \\ -0.04686903 & -0.02755516 & \dots & -0.04419417 \end{bmatrix},$$

$$S = \begin{bmatrix} 3.92516050\text{e}{+01} & 0 & \dots & 0 \\ 0 & 3.28012645\text{e}{+01} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1.73956300\text{e}{-14} \end{bmatrix} \text{ e}$$

$$V^T = \begin{bmatrix} -0.02433759 & -0.02456757 & \dots & 0.07091606 \\ 0.0159537 & 0.01617359 & \dots & 0.08918372 \\ \vdots & \vdots & \ddots & \vdots \\ -0.00850785 & -0.03438893 & \dots & -0.01609229 \end{bmatrix}.$$

Vale ressaltar que a matriz S obtida, na execução do código, é simplificada em uma matriz unidimensional com os elementos da diagonal principal da matriz S, para facilitar cálculos posteriores. Deve-se notar que $S = \Sigma$ (ver a equação (1)).

Devemos agora confiar-nos ao conceito de *variabilidade acumulada*, dada pela função $E(k)$, onde

$$E(k) = \frac{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_k^2}{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2} = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2},$$

lembrando que σ_i representa um valor singular de X. Devemos encontrar com quais valores de k, em que $1 \leq k \leq n$, conseguimos determinadas compressões. Consideraremos as compressões de 23%, 90% e 99%, ou seja, $E(k) = 0.23$, $E(k) = 0.90$ e $E(k) = 0.99$.

Fazendo todos os 512 testes, descobrimos que $E(1) = 0.23$, $E(25) = 0.90$ e $E(110) = 0.99$. O próximo passo é gerar uma matriz

$$\hat{X} = \hat{U}\hat{S}\hat{V}^T + M,$$

onde:

- $\hat{U}$ é uma matriz $512 \times k$ em que suas colunas são as k-ésimas primeiras colunas de U;
- $\hat{S}$ é uma matriz quadrada de tamanho k onde sua diagonal principal é composta pelos k primeiros valores singulares de X;
- $\hat{V}$ é uma matriz $512 \times k$ em que, analogamente a $\hat{U}$, tem por colunas as k-ésimas primeiras colunas de V;

feito isso, armazenaremos os resultados para uma comparação posterior. Durante a execução do código, as imagens serão exibidas com o auxílio do método `imshow()` da biblioteca OpenCV.

Podemos fazer ainda um experimento aparentemente à parte. Realizaremos o cálculo de uma matriz

$$\tilde{X} = \tilde{U}\tilde{S}\tilde{V}^T + M,$$

onde:

- $\tilde{U}$ é uma matriz $512 \times (512 - k))$ em que suas colunas são as $(512 - k)$ últimas colunas de U;
- $\tilde{S}$ é uma matriz quadrada de tamanho $(512 - k)$ onde sua diagonal principal é composta por σ_{512-k}, $\sigma_{512-(k+1)}$, $\sigma_{512-(k+2)}$, $\ldots$, σ_n;
- $\tilde{V}$ é uma matriz $512 \times (512 - k)$ em que suas colunas são as $(512 - k)$ últimas colunas de V;

essa imagem será também exibida e salva para comparação.

4 Resultados e Conclusão

Realizado o experimento, obtivemos os seguintes resultados:

(a) original (b) $E(k) = 0.23$

(c) $E(k) = 0.90$ (d) $E(k) = 0.99$

Figura 1: A imagem `lena_gray.jpg` em diferentes qualidades.

Como podemos ver, a técnica apresentada torna-se extremamente eficaz na compressão de imagens, sem efetuar cálculos aritméticos enormes e inversões de matrizes. Vale lembrar do experimento "à parte" que fizemos no final da seção anterior. O resultado alcançado fora:

Figura 2: Resultado do segundo experimento.

Como observado, a imagem gerada não parece nada com a imagem original, o que nos leva a concluir que as informações tomadas para gerar esse resultado não têm grande relevância, pois formam apenas um aparente borrão preto. Se olhar mais de perto, será possível observar muito discretamente as formas do rosto de Lena, como o de uma pessoa em uma noite sem lua nem estrelas.

Referências

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