



Universidade Estadual de Campinas
Projeto MS512

Compressão de imagens utilizando SVD

Felipe Maia Lopes Sinoti - RA: 251014
Gabriel Cavalcanti de Arruda - RA:247101
José Roberto Samuel Junior - RA:175393
Mateus Lee Yu - RA:236235
Nicolas Toledo de Camargo - RA: 242524
Vécio Alves Packer - RA: 251681

Junho 2023

1 Introdução

Ao estudar matrizes em Álgebra Linear (e, certamente, em Análise Numérica), temos enorme diversidade de aplicações para certas manipulações matemáticas, como é o caso da *Decomposição via SVD (Singular Value Decomposition)*. Ao analisar uma imagem exibida computacionalmente, podemos expressá-la em preto e branco como uma matriz muito grande, em que cada pixel (que seria uma entrada na matriz) possui um número de 1 a N representando o quanto de branco essa imagem mostra, ou então, efetivamente, quão intenso deve ser o brilho do LED branco da tela para representar tal pixel da imagem. Outrossim, podemos imaginar 3 matrizes sobrepostas que indicam quais os valores de intensidade de LEDs Vermelhos, Verdes e Azuis cada pixel possui, em uma imagem colorida RGB (Red, Green, Blue - respectivamente, as cores citadas), uma das formas de representar imagens com cor computacionalmente. Enfim, deste modo, representamos com sucesso uma imagem via matriz, porém, de que modo sua decomposição SVD ajudaria em armazenar uma imagem?

Podemos imaginar que uma imagem, sendo formada de pequenos quadrados (pixels) ocupa muito espaço de memória, algo que pode ser manipulado a partir de seus valores singulares - quanto mais valores singulares são mostrados, melhor é a "qualidade" da imagem representada, apesar disso, nem sempre é bom ter a melhor qualidade para trabalhar computacionalmente com uma imagem. Com isso, veremos o método de Decomposição em Valores Singulares (ou SVD) e como ele pode ser usado para comprimir imagens computacionalmente, além de tangenciarmos o PCA (Análise de Componentes Principais), já que tal método de compressão pode se utilizar do método SVD para comprimir a imagem em si. Outrossim, será demonstrada a compressão de imagens no tópico "Simulações", onde colocaremos em prática um algoritmo de compressão e serão comparadas as clarezas das imagens antes e depois da compressão, para uma melhor visualização do projeto e, por fim, discorremos sobre aplicações mais avançadas com a utilização do método SVD.

Em suma, este documento apresenta, do básico ao intermediário, uma abordagem da Decomposição SVD, de seu ensinamento e didática até sua aplicação prática.

2 Decomposição SVD

A decomposição em valores singulares (*singular value decomposition* - SVD) é uma ferramenta matemática muito poderosa para fatorar uma matriz, essa técnica possui aplicações em compressão de imagens, aprendizado de máquina, análise de dados, processamento de sinal, etc.

Definição (Decomposição em Valores Singulares - "SVD Theorem"). *Seja A uma matriz não-nula $n \times m$ com posto $r \leq \min(n, m)$, então A pode ser decomposta como*

$$A = U\Sigma V^T,$$

onde U e V são matrizes ortogonais (suas colunas são ortonormais) e Σ uma matriz retangular "diagonal" contendo os valores singulares de A em (a_{ij}) com $i = j$ e zero nas entradas restantes,

logo

$$A = U\Sigma V^T = \underbrace{\left[\begin{array}{c|c|c|c} u_1 & u_2 & \cdots & u_m \end{array} \right]}_{U(n \times n)} \underbrace{\left[\begin{array}{ccccccc} \sigma_1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & & & & 0 & & 0 \\ \vdots & & \sigma_r & & & \vdots & & \vdots \\ 0 & & 0 & 0 & & 0 & & 0 \\ \vdots & & & \ddots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{array} \right]}_{\Sigma(n \times m)} \underbrace{\left[\begin{array}{c} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{array} \right]}_{V^T(m \times m)},$$

onde os valores singulares de A satisfazem $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$;

u_1, \dots, u_m são as colunas que compõe U , sendo chamadas de 'vetores singulares à esquerda';

v_1, \dots, v_n , colunas de V , ditas 'vetores singulares à direita'.

Definição ('Geometric SVD Theorem'). Seja A uma matriz não-nula $n \times m$ com posto $r \leq \min(n, m)$, então \mathbb{R}^m e \mathbb{R}^n têm bases ortonormais v_1, \dots, v_m e u_1, \dots, u_n , respectivamente, e existem $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ tais que:

$$A \cdot v_i = \begin{cases} \sigma_i \cdot u_i, i = 1, \dots, r \\ 0, i = r + 1, \dots, m \end{cases} \quad A^T \cdot u_i = \begin{cases} \sigma_i \cdot v_i, i = 1, \dots, r \\ 0, i = r + 1, \dots, m \end{cases}$$

Deste modo, peguemos como exemplo uma matriz $A = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}$, tomemos a matriz $A^T \cdot A$

$= \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$. Temos que as colunas de $A^T A$ são Linearmente Independentes, vamos normalizá-las:

Seja $w_1 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$, $w_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$. $\|w_1\| = \|w_2\| = \sqrt{2^2 + (-1)^2} = \sqrt{4+1} = \sqrt{5}$

$v_1 = \begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix}$, $v_2 = \begin{bmatrix} -1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$, portanto, temos $V = \begin{bmatrix} 2/\sqrt{5} & -1/\sqrt{5} \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}$. Os autovalores de $A^T A$ são 3 e 1. Temos $A^T A = (U\Sigma V^T)^T (U\Sigma V^T) = V(\Sigma^T \Sigma)V^T$. Portanto, o i-ésimo autovalor de A é a raiz do i-ésimo autovalor de $A^T A$. Assim, $\sigma_1 = \sqrt{3}$ e $\sigma_2 = 1$.

Deste modo, $\Sigma = \begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$. Para finalizar, usando o Teorema Geométrico SVD, temos que as colunas u_1 e u_2 de U são dadas por:

$$\begin{cases} u_1 = \frac{1}{\sigma_1} \cdot A \cdot v_1 \\ u_2 = \frac{1}{\sigma_2} \cdot A \cdot v_2 \end{cases}$$

$$\begin{cases} u_1 = \begin{bmatrix} -2/\sqrt{5} \cdot \sqrt{3} \\ 3/\sqrt{5} \cdot \sqrt{3} \\ -1/\sqrt{5} \cdot \sqrt{3} \\ 1/\sqrt{5} \\ -3/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix} \\ u_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{cases}$$

Portanto, temos as colunas u_1 e u_2 , realizaremos o procedimento

de Gram-Schmidt com um vetor $e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, temos p um novo vetor, que normalizado da origem a nova coluna u_3 :

$p = e_3 - \langle u_1 \cdot e_3 \rangle \cdot u_1 - \langle u_2 \cdot e_3 \rangle \cdot u_2$, perceba que não há normalização, pois u_1, u_2 já são normalizados.

$$p = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{1}{\sqrt{15}} \cdot \begin{bmatrix} -2/\sqrt{15} \\ 3/\sqrt{15} \\ -1/\sqrt{15} \end{bmatrix} - \frac{2}{\sqrt{5}} \cdot \begin{bmatrix} 1/\sqrt{5} \\ -3/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$$

$$p = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -2/15 \\ 3/15 \\ -1/15 \end{bmatrix} - \begin{bmatrix} 2/5 \\ -6/5 \\ 4/5 \end{bmatrix} = \begin{bmatrix} -8/15 \\ 7/5 \\ 2/15 \end{bmatrix}. \quad \text{Temos } \|p\| = \sqrt{\frac{(-8)^2}{15^2} + \frac{7^2}{5^2} + \frac{2^2}{15^2}} = \sqrt{509}/15. \quad \text{Assim, normalizando p e o chamando de } u_3, \text{ temos } u_3 = \begin{bmatrix} -8/\sqrt{509} \\ 21/\sqrt{509} \\ 2/\sqrt{509} \end{bmatrix}.$$

$$U = \begin{bmatrix} -2/\sqrt{15} & 1/\sqrt{5} & -8/\sqrt{509} \\ 3/\sqrt{15} & -3/\sqrt{5} & 21/\sqrt{509} \\ -1/\sqrt{15} & 2/\sqrt{5} & 2/\sqrt{509} \end{bmatrix}.$$

Assim,

$$A = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -2/\sqrt{15} & 1/\sqrt{5} & -8/\sqrt{509} \\ 3/\sqrt{15} & -3/\sqrt{5} & 21/\sqrt{509} \\ -1/\sqrt{15} & 2/\sqrt{5} & 2/\sqrt{509} \end{bmatrix} \cdot \begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 2/\sqrt{5} & -1/\sqrt{5} \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix} = U \cdot \Sigma \cdot V^T.$$

Uma maneira que talvez seja mais visual para a compressão via SVD segue o 'Teorema SVD' em sua forma matricial -

Podemos expressar a matriz A em forma de uma combinação de matrizes $n \times m$ a partir de uma manipulação por blocos das matrizes SVD:

$$A = \sigma_1 \cdot \begin{bmatrix} | \\ u_1 \\ | \end{bmatrix} \cdot [v_1^T] + \cdots + \sigma_r \cdot \begin{bmatrix} | \\ u_r \\ | \end{bmatrix} \cdot [v_r^T] + 0 + \cdots + 0$$

Deste modo, é perceptível que podemos escrever somente parte da decomposição, tendo, assim, uma versão condensada da SVD.

3 Compressão por SVD

Como vimos nas seções anteriores, dado uma imagem, podemos representá-la dentro do computador a partir de um conjunto de matrizes contendo a informação de cada pixel (seja para uma imagem colorida ou em escala de cinza). Não só isso, também vimos que podemos realizar a decomposição de uma matriz A , $n \times m$ com posto $r \leq \min(n, m)$ em outras três matrizes: U , Σ e V^T , na chamada Decomposição em Valores Singulares (SVD).

Isso nos dá uma noção intuitiva de que, se tivermos uma imagem representada como uma matriz no computador, também podemos aplicar este método de decomposição para fatorar a imagem

em outras três matrizes. Mas qual é a relevância disto?

Em termos computacionais, parece estranho pensar que transformar uma matriz em outras três pode nos gerar algum benefício, senão o gasto maior de memória para alocação destas outras três matriz. Porém, analisando atentamente o resultado que obtemos, podemos nos indagar: e se utilizarmos apenas um pedaço das matrizes fatoradas para reconstruir a imagem original? É na resposta desta pergunta que o método da compressão por SVD se mostra eficiente.

De fato, se tomarmos uma matriz e realizarmos sua decomposição em SVD, obteremos três outras matrizes. Deste resultado, o importante para a compressão é justamente reduzir o custo computacional de armazenamento dos valores dados. Sendo assim, ao invés de armazenarmos todos os valores obtidos, pegamos apenas um pedaço de cada uma das matrizes. Isto é, das três matrizes U , Σ e V^T , com posto r , salvamos apenas U_k , Σ_k e V_k^T , onde $k \leq r$ é o posto destas novas matrizes. Tal valor nos dirá a qualidade da aproximação da imagem. Isso significa que podemos pegar o valor de m que desejarmos para nossa aproximação - claro, se $k = r$, não estaremos comprimindo a imagem de fato.

Se temos uma imagem A com $n \times m$ elementos e posto r , teremos uma decomposição SVD tal que U possui $n \times r$ elementos, Σ possui r (apenas elementos da diagonal) e V^T possui $r \times m$. Então se escolhermos uma compressão considerando apenas k valores singulares, podemos armazenar os vetores de U_k , Σ_k e V_k^T , contendo $nk + k + km = k(n + 1 + m)$ elementos, e obter a imagem comprimida $A_{ap} = U_k \Sigma_k V_k^T$. Então a ideia é escolher este k para termos $k(n + 1 + m) \leq nm$, vamos chamar a taxa de compressão da imagem como $T = 1 - \frac{k(n+1+m)}{nm}$. O k é escolhido com base da qualidade requerida que se quer trabalhar, quanto mais valores singulares, melhor a qualidade mas menor a compressão.

Teremos mais exemplos práticos deste método na seção 5, porém vamos trazer uma simulação feita pelo professor Steve Brunton em um vídeo no youtube [1].

Primeiramente, ele pegou a imagem do seu cachorro na escala de cinza e realizou a fatoração SVD. Posteriormente, ele rodou um código para salvar apenas as matrizes U_k , Σ_k e V_k^T , e exibiu este valor na tela. O resultado foi o seguinte:



Figure 1: Imagem Original

É interessante ver como a nitidez da imagem quando $k = 100$ é boa. Vale ressaltar que $k = 100$ só utiliza 12% da matriz original.

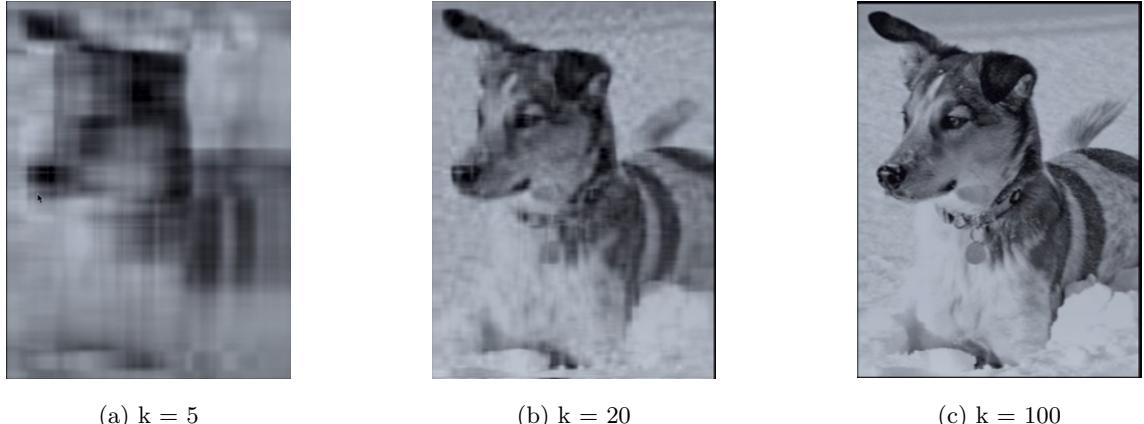


Figure 2: Imagens com compressão

4 Compressão por PCA

A compressão de imagens por meio da Análise de Componentes Principais (PCA) é uma técnica matemática que redimensiona matrizes, identificando padrões relevantes baseando-se na variância dos dados e descarta componentes irrelevantes. Essa abordagem é útil para reduzir o tamanho dos arquivos de imagem, economizar espaço de armazenamento e otimizar possíveis operações. Com a identificação e o descarte adequados dos componentes, é possível alcançar uma taxa eficiente de compressão, mantendo uma qualidade aceitável na reconstrução das imagens comprimidas. Neste relatório, iremos explorar a aplicação do PCA na compressão de imagens digitais com perdas mínimas.

Para iniciar o processo de compressão, é necessário preparar a imagem convertendo-a em uma matriz, na qual os valores $f(x,y)$ representam a intensidade dos pixels e as coordenadas x e y indicam sua localização.

Em seguida, é realizada a padronização dos dados, de modo a evitar que colunas específicas tenham uma influência desproporcional na obtenção dos componentes principais. Para isso, os dados são ajustados para que cada coluna tenha média zero e variância unitária, armazenando as médias e desvios padrão.

$$a_{ij} \leftarrow \frac{a_{ij} - \text{mean}_j}{\text{std}_j}, \text{ para cada linha } i \text{ de cada coluna } j.$$

A matriz de covariância C é calculada pela operação:

$$C = \frac{1}{n} A^T A$$

Em seguida, fazemos a decomposição SVD de C e a seleção dos primeiros k autovalores e seus autovetores correspondentes, descartando os demais, guardando U_k . Essa etapa possibilita a obtenção de uma maior taxa de eficiência na compressão, uma vez que são descartados os componentes que apresentam variação insignificante nos dados da imagem.

O resultado final do PCA é dado pela projeção dos dados originais nas componentes principais selecionadas, fazendo a multiplicação dos dados originais A por U_k e você terá uma matriz Z com

$n \times k$ elementos que explica a variância dos dados e pode ser usada em treinamento de algoritmos, pois ela leva a maior variação dos dados originais em uma forma mais compacta.

$$Z = AU_k$$

Note que para o PCA apenas temos uma compressão para reduzir dimensão com finalidade de treino, mas também podemos obter a reconstrução da imagem aproximada a partir de Z e U_k , basta multiplicar Z por U_k^T que voltamos para uma matriz $n \times m$.

$$A_{ap} = ZU_k^T$$

Depois 'desnormalizamos' as colunas, multiplicando pelos desvios padrões e somando as médias guardadas.

$$a_{ij} \leftarrow a_{ij} \text{std}_j + \text{mean}_j, \text{ para cada linha } i \text{ de cada coluna } j.$$

Então se aplicarmos PCA sobre uma imagem também podemos reduzir seu armazenamento, já que podemos guardar apenas Z , U_k e os vetores de média e desvio padrão das colunas. No total guardariamos $nk + mk + m + m = k(n + m) + 2m$ elementos. Portanto a ideia é escolher este k para termos $k(n + m) + 2m \leq nm$, vamos chamar a taxa de compressão da imagem como $T = 1 - \frac{k(n+m)+2m}{nm}$. O k é escolhido com base da qualidade requerida que se quer trabalhar, quanto mais componentes principais, melhor a qualidade mas menor a compressão.

5 Simulações

Para esta seção, baseado nos códigos de implementação, faremos simulações de compressões de imagens usando SVD diretamente e PCA.

Com estas simulações, analisaremos seus erros relativos quanto às imagens originais e suas taxas de compressão.

Se A é a matriz da imagem original e A_{ap} é a imagem comprimida, usaremos a medida de erro relativo $\frac{\|A - A_{ap}\|}{\|A\|}$.

5.1 Implementação

Implementamos os algoritmos em python, o script faz as compressões para todos os valores singulares que garantem uma redução do tamanho do arquivo e armazena os valores a cada 10% de redução de erro relativo e, a partir de 5% de erro, a cada 1%.

Ao rodar o arquivo main.py, o prompt fará algumas perguntas para o usuário inserir os inputs, e de outputs temos as imagens comprimidas, numero de valores singulares (componentes principais para PCA) utilizados, suas taxas de compressão e erros, sendo esses valores salvos na pasta save_files e também são plotadas as imagens e gráfico de taxa de compressão vs erro relativo .

5.2 Resultados

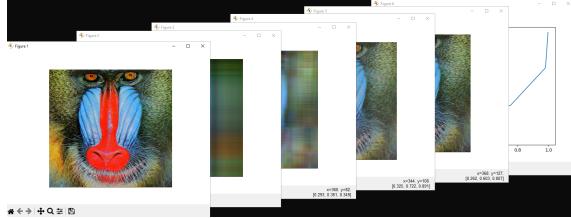
As imagens de referência para as simulações [2] são de um banco de dados de imagens comuns em testes de processamento de imagens.

```

Path dos arquivos? ex: C:\User\Documents\MS512\Projeto\main
C:\Users\accam\Documents\MS512\Projeto\main
gray ou rgb? ex: rgb
rgb
Arquivo da imagem? ex: baboon.ppm
baboon.ppm
Compressão por SVD ou PCA? ex: SVD
SVD

```

(a) Inputs.



```

k_relev_rgb_svd_baboon.ppm.npy
vetor_imagens_comprimidas_completo_svd_baboon.ppm.npy
vetor_numero_comp_por_camada_svd_baboon.ppm.npy
X_svd_baboon.ppm.npy

```

(b) Outputs 1.

(c) Outputs 2.

Figure 3: Script principal.

A primeira imagem, 'baboon', é uma imagem RGB e possui 480 x 500 elementos por camada de cor.

Quantidadde de valores singulares:	1	5	60	228
Taxa de compressão:	99%	97%	75%	7%
Erro relativo:	40%	28%	17%	6%

Table 1: Compressao de 'baboon' por SVD.

Quantidadde de componentes principais:	1	7	83	236
Taxa de compressão:	99%	97%	66%	3%
Erro relativo:	36%	27%	15%	6%

Table 2: Compressao de 'baboon' por PCA.

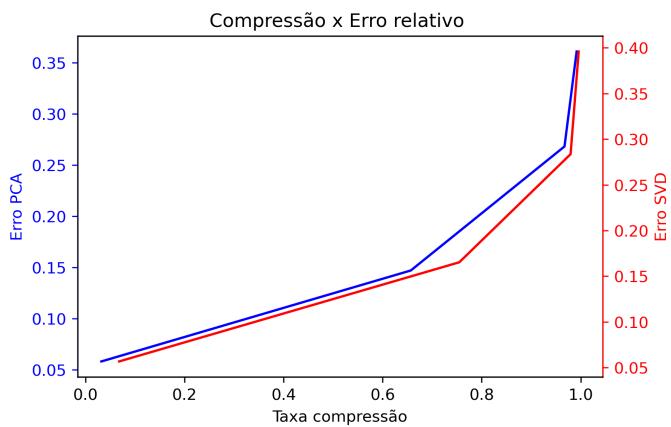


Figure 4: Compressão SVD x PCA para 'baboon'.

A segunda imagem, 'fingerprint1', é uma imagem em tons de cinza e possui 512 x 512 elementos.

Quantidadade de componentes principais:	1	5	27	202
Taxa de compressão:	99%	98%	89%	21%
Erro relativo:	90%	79%	48%	5%

Table 3: Compressao de 'fingerprint1' por SVD.

Quantidadade de componentes principais:	1	17	101	147
Taxa de compressão:	99%	93%	60%	42%
Erro relativo:	30%	18%	5%	3%

Table 4: Compressao de "fingerprint1" por PCA.

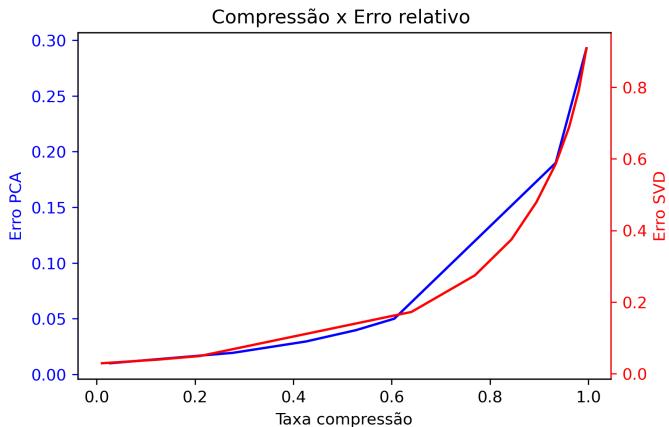


Figure 5: Compressão SVD x PCA para 'fingerprint1'.

Essas duas imagens captam bem a diferença entre os métodos de compressão. Para a imagem do face de um babuino, a compressão por SVD parece um pouco mais efetiva, conseguindo comprimir mais e usar menos valores singulares para apresentar uma imagem nítida. Já para a impressão digital, o PCA se apresentou superior, podendo comprimir mais a imagem até alcançar a nitidez.

Essa diferença decorre das construções dos métodos, o SVD reconstrói a imagem com base em informações mais gerais, com maior peso, da imagem, já a compressão por PCA reconstrói a imagem com base na variação dos dados.

Uma face possui muita redundância nos dados, e partes essenciais para seu reconhecimento, como o nariz, os olhos e a boca, contêm informações relevantes que são capturadas pelos vetores singulares mais importantes. Essas informações contribuem para o sucesso da compressão por SVD na reconstrução de uma face com qualidade adequada.

Por outro lado, a impressão digital não possui algumas características gerais de maior importância, é uma imagem com muita variância, com muitos detalhes relevantes. Neste caso, a reconstrução levando em conta a variação dos dados se torna mais poderosa, beneficiando o PCA.

6 Conclusão

Por fim, observamos que, a partir de matrizes completas, podemos expressar imagens por sua composição em valores singulares (SVD) e, com isso, realizar certa compressão em um nível onde temos qualidade o suficiente (menor erro relativo) e, também, menos informações computacionais armazenadas (maior taxa de compressão), isto é, queremos ocupar menos espaço na memória com uma qualidade de imagem suficientemente boa. Na prática, queremos que o novo posto da matriz k (com $k \leq r$, r posto original da matriz A) seja o menor possível, com um erro relativo $\frac{\|A - A_{ap}\|}{\|A\|}$ também pequeno, onde A é a matriz original e A_{ap} é nossa nova matriz condensada.

Deste modo, podemos usar tanto a Decomposição via SVD clássica ou o PCA, para reduzir o armazenamento de imagens ao custo de certo nível de qualidade. Escolhendo cada método com base em sua construção, sendo a decomposição SVD recomendada quanto temos uma imagem com algumas com a características dominantes, e a por PCA quando temos uma imagem com muita variação nos detalhes.

7 Aplicações avançadas

Com a técnica de SVD, podemos também abordar aplicações reais, como é o artigo "Measure of Singular Value Decomposition (MSVD) based Quality Assessment for Medical Images with Degradation" de Ersin Elbasi [3], que discute a importância da avaliação da qualidade da imagem em vários campos, incluindo a medicina. O texto propõe uma nova métrica de qualidade de imagem chamada Medida de Decomposição de Valor Singular (M-SVD) e avalia sua eficácia na avaliação da qualidade de imagens médicas com distorções.

Inicialmente, há o destaque da importância das imagens em áreas como militar, saúde, ciência e segurança, enfatizando a necessidade de métricas confiáveis de qualidade de imagem para medir a precisão e o nível de qualidade de imagens e vídeos - as métricas tradicionais podem não ser sensíveis o suficiente para detectar pequenas distorções em imagens médicas, o que pode ter um impacto crítico na tomada de decisões. A marca d'água, que pode ser considerada um tipo de distorção da imagem (normalmente para proteger informações médicas do paciente) é discutida como um fator potencial que afeta a qualidade da imagem. O artigo apresenta o conceito de medições subjetivas e objetivas de qualidade de imagem e categoriza medições objetivas com base na disponibilidade de imagens de origem.

Com a progressão do texto, são revisadas várias métricas existentes de qualidade de imagem, como: erro quadrático médio, relação sinal-ruído de pico, índice de similaridade estrutural e qualidade de imagem universal. Também é discutido os desafios e limitações dessas métricas, particularmente no contexto de imagens médicas e, em resposta a isso, o algoritmo M-SVD é proposto como uma nova métrica de qualidade de imagem. O algoritmo é baseado na Decomposição de Valor Singular e visa expressar numérica ou graficamente a qualidade de imagens distorcidas, incluindo imagens médicas com marca d'água. De acordo com o artigo, os resultados experimentais mostram resultados promissores ao comparar o algoritmo M-SVD com outras avaliações de qualidade comuns, como relação sinal-ruído de pico, erro quadrado médio, medidas de índice de similaridade estrutural e qualidade de imagem universal. O artigo é concluído discutindo a importância da medição da qualidade da imagem e as possíveis aplicações do algoritmo M-SVD na avaliação da qualidade das imagens médicas. Ele destaca a necessidade de métricas sensíveis que possam avaliar com precisão a qualidade das imagens médicas, pois mesmo pequenas distorções podem afetar significativamente as decisões médicas.

Agora, observemos as métricas de qualidade de imagem objetivas em si: Erro Quadrático Médio (MSE), Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), Universal Image Quality (UIQ), Gradient Similarity Measure (GSM) e Measure of Singular Value Decomposition (M-SVD), todas usadas para avaliar a diferença entre duas imagens, geralmente uma imagem de referência e uma imagem distorcida.

O MSE é uma medida absoluta do erro entre as duas matrizes de imagem, somando o quadrado da diferença pixel-a-pixel. O RMSE (Root Mean Square Error) é calculado a partir da raiz quadrada do MSE e fornece uma medida de qualidade de imagem mais fácil de interpretar.

O UIQ é um modelo que calcula a distorção total usando a distorção de luminância, distorção de contraste e perda de correlação. Ele combina essas medidas para fornecer uma avaliação geral da qualidade da imagem.

O GSM mede a similaridade entre duas imagens com base nas mudanças no contraste e na estrutura da imagem. É uma métrica mais robusta, eficiente e rápida em comparação com outras

métricas comuns de qualidade de imagem.

O M-SVD é uma medida de qualidade de imagem baseada na decomposição em valores singulares da imagem. Ele calcula as distâncias entre os valores singulares da imagem original e da imagem possivelmente distorcida.

Também são apresentados resultados experimentais da aplicação dessas métricas em imagens médicas, mostrando que o M-SVD tem uma precisão superior em comparação a outras métricas, como PSNR, MSE, SSIM e UIQ.

No geral, o artigo conclui que o M-SVD é uma métrica de qualidade de imagem promissora e eficaz, especialmente para imagens médicas. Sugere-se que pesquisas futuras possam explorar a aplicação do M-SVD em sequências de vídeo.

Referências

- [1] Steve Brunton - *SVD: Image Compression [Python]*. 2020. Disponível em: <<https://youtu.be/H7qMMudo3e8>>. Acesso em: 21 de junho de 2023.
- [2] Standard Test Images. Disponível em: <<https://www.kaggle.com/datasets/saeedehkamjoo/standard-test-images?resource=download>>. Acesso em: junho de 2023.
- [3] Ersin Elbasi - *Measure of Singular Value Decomposition (MSVD) based Quality Assessment for Medical Images with Degradation*. Disponível em: <The International Arab Journal of Information Technology, Vol. 18, No. 5, September 2021>. Acesso em: junho de 2023.
- [4] Watkins, David S. *Fundamentals of Matrix Computations*. John Wiley & Sons, Hoboken, NJ, 2002.