

Pedro Paulo Vezz  Campos

Compress o de imagens usando SVD

S o Paulo - SP, Brasil

10 de dezembro de 2012

Pedro Paulo Vezz  Campos

Compress o de imagens usando SVD

Terceiro exerc cio-programa apresentado para avalia  o na disciplina MAC0300, do curso de Bacharelado em Ci ncia da Computa  o, turma 45, da Universidade de S o Paulo, ministrada pelo professor Walter Figueiredo Mascarenhas.

DEPARTAMENTO DE CI NCIA DA COMPUTA  O
INSTITUTO DE MATEM TICA E ESTAT STICA
UNIVERSIDADE DE S O PAULO

S o Paulo - SP, Brasil

10 de dezembro de 2012

Sumário

1	Introdução	p. 3
2	Decomposição em Valores Singulares	p. 4
2.1	Algoritmo Clássico	p. 5
2.2	Algoritmo de Golub-Reinsch	p. 5
2.3	Análise de Complexidade do Algoritmo de Golub-Reinsch	p. 6
2.3.1	Primeira Fase: Bidiagonalização de Golub-Kahan	p. 6
2.3.2	Segunda Fase: Golub-Reinsch SVD	p. 6
2.4	Melhorias do Algoritmo de Golub-Reinsch ao Método Clássico	p. 6
3	Testes Realizados	p. 8
4	Conclusão	p. 10
	Referências Bibliográficas	p. 11

1 Introdução

Neste terceiro exercício-programa de MAC0300 - Métodos Numéricos da Álgebra Linear foi pedido que implementássemos um programa que fosse capaz de decompor uma matriz em seus valores singulares (Decomposição SVD) e aplicar tal algoritmo para a compressão de imagens. Neste relatório serão apresentados: Uma explicação e análise sobre SVD e seus algoritmos (Clássico e Golub-Reinsch), testes realizados e, por fim, será feita apresentada uma conclusão sobre o EP.

2 *Decomposição em Valores Singulares*

A Decomposição em Valores Singulares (SVD) é uma conhecida fatoração de uma matriz em três termos. Tal decomposição possui várias aplicações, variando desde processamento de sinais até estatística.

Uma SVD é definida para uma matriz M de dimensões $m \times n$ como sendo o produto

$$M = U\Sigma V^T$$

com as seguintes restrições:

- U é uma matriz $m \times m$. Suas colunas são conhecidas como *vetores singulares à esquerda* e são autovetores de MM^T
- Σ é uma matriz diagonal. Os valores da diagonal de Σ são conhecidos como *valores singulares de M* e são as raízes quadradas dos autovalores diferentes de zero de tanto $M^T M$ quanto MM^T .
- V^T é uma matriz $n \times n$. Suas colunas são conhecidas como *vetores singulares à direita* e são autovetores de $M^T M$

Para fins deste EP, houve a opção por implementar uma versão da SVD compacta. Nela, nem todos os autovetores são calculados, apenas os necessários para a reconstrução da matriz M a partir dos três fatores. No código implementado, a decomposição é descrita da seguinte forma:

- U é $m \times \min(m, n)$ e possui colunas ortogonais
- S é $\min(m, n) \times \min(m, n)$ e é uma matriz diagonal contendo na diagonal principal os valores singulares
- V é $n \times \min(m, n)$ e possui colunas ortogonais

2.1 Algoritmo Clássico

A decomposição em valores singulares possui como vantagem o fato que pode ser aplicado a qualquer matriz de dimensões $m \times n$ enquanto a decomposição em autovalores e autovetores só é possível para algumas matrizes quadradas. No entanto, podemos traçar paralelos entre ambas decomposições.

Dada uma SVD de uma matriz M , temos que:

$$M^T M = V \Sigma^T U^T U \Sigma V^T = V (\Sigma^T \Sigma) V^T$$

$$M M^T = U \Sigma V^T V \Sigma^* U^T = U (\Sigma \Sigma^T) U^T.$$

As expressões que se encontram no lado direito das igualdades descrevem decomposições em autovalores e autovetores das expressões do lado esquerdo. Isso traz como consequência que:

- As colunas de V são autovetores de $M^T M$.
- As colunas de U são autovetores de $M M^T$.
- Os valores diferentes de zero de Σ são as raízes quadradas dos autovalores diferentes de zero de $M^T M$ ou $M M^T$.

Este algoritmo para a obtenção da decomposição em valores singulares funciona corretamente para matrizes menores e quando os valores singulares são significativamente maiores que a precisão adotada nos cálculos. Por outro lado, há uma perda de precisão inerente ao algoritmo que será explicado na 2.4, o que nos induz a buscar um algoritmo que não faça uso do cálculo de autovalores e autovetores de $M^T M$, que será apresentado na seção 2.2.

2.2 Algoritmo de Golub-Reinsch

O Algoritmo de Golub-Reinsch é um método que faz uso extensivo de rotações e reflexões para obter a SVD sem calcular $M^T M$, o que poderia trazer problemas como será apresentado em seguida. O processo é dividido em duas etapas. Na primeira fase devemos reduzir a matriz M original à forma bidiagonal utilizando reflexões de Householder aplicadas alternadamente à esquerda para zerar as colunas abaixo da diagonal principal e à direita para zerar linhas acima da superdiagonal. A segunda fase é a computação propriamente dita da SVD. Através de rotações

de Givens vamos iterativamente zerando os elementos da superdiagonal enquanto acumulamos as rotações realizadas nas matrizes U e V . Este processo é repetido enquanto não for atingida uma precisão maior que uma especificada ($O \varepsilon$ da máquina por exemplo).

2.3 Análise de Complexidade do Algoritmo de Golub-Reinsch

2.3.1 Primeira Fase: Bidiagonalização de Golub-Kahan

Como vamos aplicando refletores alternadamente à esquerda e à direita, ao final da primeira fase do cálculo da SVD são necessários n refletores à esquerda e $n - 2$ refletores à direita. Podemos traçar um paralelo entre o processo de bidiagonalização e o de aplicar duas fatorações QR de Householder entrelaçadas, a primeira operando na matriz M de dimensões $m \times n$ e a outra operando na matriz M^T de dimensões $n \times m$. Assim, o custo total para a bidiagonalização é de $\sim 4mn^2 - \frac{4}{3}n^3$ flops. [1]

2.3.2 Segunda Fase: Golub-Reinsch SVD

Na segunda fase a princípio seria necessário um número infinito de rotações de Givens para que a matriz Σ convirja a uma matriz diagonal. Porém, a convergência é superlinear e em $O(n \log(|\log(\varepsilon)|))$ iterações atingimos a precisão ε da máquina. Na prática, sendo ε uma constante, a convergência é dada em $O(n)$ iterações. Ainda, como a matriz é originalmente bidiagonal, são necessários apenas $O(n)$ flops por iteração, totalizando $O(n^2)$ flops para a segunda fase. Como conclusão, na prática, a primeira fase do algoritmo de Golub-Reinsch é assintoticamente mais custosa que a segunda, ditando a complexidade final do algoritmo. [1]

2.4 Melhorias do Algoritmo de Golub-Reinsch ao Método Clássico

Apesar do algoritmo clássico ser relativamente barato computacionalmente ele possui como problema o fato que valores singulares pequenos serão calculados de maneira imprecisa. Isto é uma consequência do efeito de "perda de informação através da elevação ao quadrado" que acontece quando calculamos $M^T M$ a partir de A .

Nós podemos ter uma ideia desta perda de informação ao considerar um exemplo. Suponha que as entradas da matriz A são conhecidas com exatidão em seis casas decimais. Se A tem, digamos $\sigma_1 \approx 1$ e $\sigma_{17} \approx 10^{-3}$, então σ_{17} é razoavelmente menor que σ_1 , mas ainda assim acima

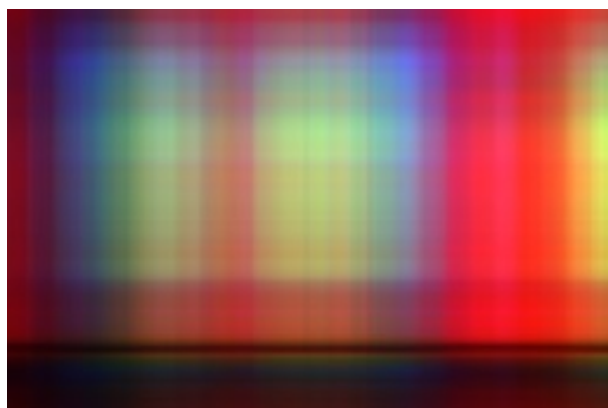
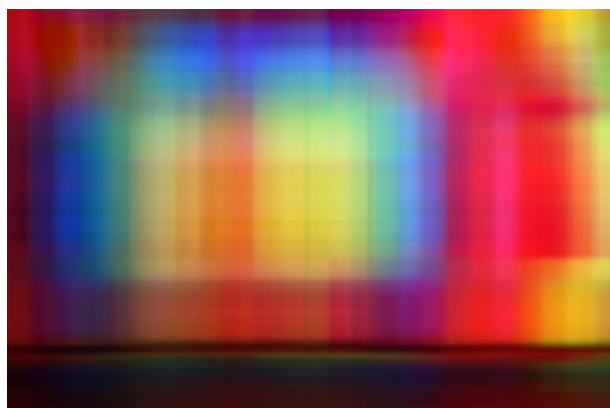
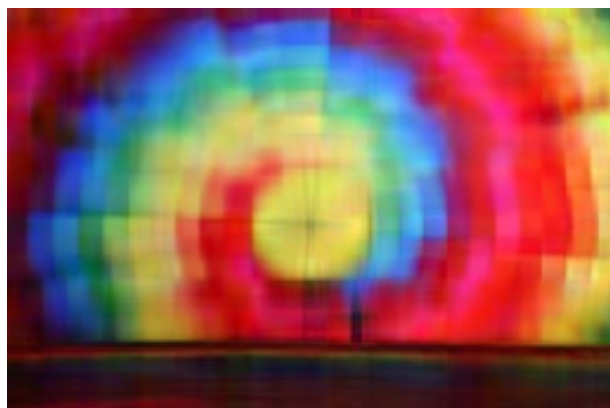
da precisão de $\varepsilon \approx 10^{-5}$ ou 10^{-6} . Nós gostaríamos de calcular σ_{17} com talvez duas ou três casas de precisão. As entradas de $M^T M$ tem também precisão de aproximadamente volta de seis casas decimais. Associada com os valores singulares σ_1 e σ_{17} , $M^T M$ tem autovalores $\lambda_1 = \sigma_1^2 \approx 1$ e $\lambda_{17} = \sigma_{17}^2 \approx 10^{-6}$. Note que λ_{17} tem a mesma magnitude que os erros nas entradas de $M^T M$. Portanto não podemos esperar que λ_{17} possa ser calculado de maneira precisa. [2]

3 *Testes Realizados*

Como foi pedido no enunciado, o algoritmo implementado foi utilizado para comprimir uma imagem RGB ao alterarmos o número do posto (*rank*) da decomposição obtida escolhendo apenas as k primeiras colunas de U , linhas e colunas de Σ e colunas de V . Os resultados condizem com o esperado, estando os valores singulares ordenados, os primeiros possuem um “peso” maior na composição da imagem final. O efeito prático é que com uma quantidade menor de termos na série que define a matriz da imagem original a partir da SVD podemos montar uma imagem comprimida muito próxima da original.



Figura 3.1: Imagem original (256×170)

(a) $k = 1$ (b) $k = 2$ (c) $k = 4$ (d) $k = 8$ (e) $k = 16$ (f) $k = 32$ (g) $k = 64$ (h) $k = 128$ Figura 3.2: Resultado das compressões para diferentes valores de k

4 *Conclusão*

Neste trabalho foram apresentados uma explicação sucinta sobre a decomposição em valores singulares (SVD). Posteriormente, foram vistos dois algoritmos distintos para obter tal decomposição e, em seguida, uma comparação foi traçada entre a eficácia de cada um dos métodos juntamente com uma análise de complexidade do algoritmo de Golub-Reinsch. Por fim, foram apresentados resultados experimentais do método de compressão de imagens apresentado no enunciado do EP.

O trabalho inovou ao introduzir aos alunos de Ciência da Computação a uma vasta área dentro do ramo que é a compressão de dados, tão relevante frente à crescente demanda, por exemplo, de conteúdos multimídia com resolução cada vez maior. Desta vez, a fundamentação teórica não é tão complexa depois que os alunos já estão habituados a conceitos como decomposições QR de Givens, Householder, autovalores e autovetores. Porém, a carga de programação foi bem mais intensa que antes, o algoritmo de Golub-Reinsch tem muitas subrotinas que devem ser implementadas.

Referências Bibliográficas

- [1] TREFETHEN, L. N.; BAU, D. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997. ISBN 0898713617. Disponível em: <<http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0898713617>>.
- [2] WATKINS, D. *Fundamentals of Matrix Computations*. Wiley, 2004. (Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts). ISBN 9780471461678. Disponível em: <<http://books.google.com.br/books?id=8t7mHZxNqIMC>>.
- [3] WIKIPEDIA. *Singular value decomposition* — *Wikipedia, The Free Encyclopedia*. 2012. [Online; accessed 10-December-2012]. Disponível em: <http://en.wikipedia.org/w/index.php?title=Singular_value_decomposition&oldid=526553712>.