PSI-3260 Aplicações de Álgebra Linear

Experiência 10

Decomposição em Valores Singulares 1 – Parte computacional

Comprimindo imagens com a SVD

Neste exercício, vamos usar a SVD para compressão de imagens. Inicialmente, definimos as medidas de energia e da taxa de compressão que serão usadas para avaliação da imagem comprimida. Em seguida, usamos a SVD para comprimir duas imagens.

1 Medidas de energia e taxa de compressão

Na compressão de uma imagem, deseja-se remover tantos dados quanto possível sem perder excessivamente a energia da imagem original. A diferença em termos de energia entre a imagem original e a imagem comprimida é uma das possíveis formas de se projetar uma medida de erro para imagens comprimidas. Entretanto, essa medida nem sempre é adequada, pois em muitos casos, a energia do erro medida pode ser pequena, enquanto o erro visual é grande. É necessário encontrar uma boa forma de medir o erro, tal que a medida coincida com nossos olhos.

Seja uma imagem digital representada por uma matriz $\mathbf{A}_{m\times n}$ com elementos A_{ij}, $i = 1\cdots m$ e $j = 1\cdots n$. A norma de Frobenius da matriz $\mathbf{A}$ é definida como

$$|||\mathbf{A}|||_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|A_{ij}|^2}.$$

Essa não é a norma de matrizes mais comum e nem sempre é a mais conveniente, mas está relacionada com funções da norma ℓ_2. A norma ℓ_2, por sua vez, está relacionada ao conceito de energia e por essa razão, a norma de Frobenius é útil em processamento de imagens. Note que quando usamos matrizes para representar imagens em escala de cinza, seus elementos representam pixels que pertencem ao intervalo $[0,\ 1]$.

Denotando a versão comprimida da imagem $\mathbf{A}$ pela matriz $\mathbf{A}_c$, definimos o erro relativo como

$$e = \frac{|||\mathbf{A} - \mathbf{A}_c|||_{\mathrm{F}}}{|||\mathbf{A}|||_{\mathrm{F}}}.$$

Uma outra medida importante para compressão de dados é a quantidade de informação que precisa ser armazenada. Sejam N_A e N_C números que representam respectivamente as

unidades de memória necessárias para representar uma imagem $\mathbf{A}$ e sua versão comprimida $\mathbf{A}_c$. Podemos medir a eficiência da compressão como

$$C = \frac{N_A}{N_C}.$$

Quando $C = 1$ não há compressão alguma e quando $C = 10$, $\mathbf{A}_c$ é armazenada com apenas 10% da memória usada para armazenar $\mathbf{A}$, ou seja, a taxa de compressão é de 10:1.

2 Comprimindo imagens

Seja $\mathbf{A}$ uma matriz $m \times n$ de posto r que representa uma imagem que se deseja comprimir. Neste caso, $\mathbf{A}$ não é considerada uma transformação linear e sim uma tabela com mn números. Desejamos encontrar uma aproximação que captura as principais características dos dados. Como o posto de uma matriz especifica o número de colunas (ou linhas) linearmente independentes, ele é uma medida de redundância. Uma matriz com posto pequeno tem uma grande quantidade de redundância e pode ser expressa de forma mais eficiente do que uma tabela com todos os seus elementos. Qualquer característica de larga escala na imagem será refletida por redundância nas colunas ou linhas da matriz e esperamos capturar essas características em uma aproximação por uma matriz de posto menor que $\min\{m, n\}$.

O caso extremo é quando a matriz tem posto $r = 1$. Se tivermos uma matriz desse tipo, todas as colunas são múltiplas umas das outras e o espaço coluna é unidimensional, podendo ser representado como um produto externo de dois vetores. Se uma matriz tiver posto $r = 5$, por exemplo, mas um de seus valores singulares for muito maior que os outros 4, podemos obter uma boa aproximação de posto um, zerando os 4 valores singulares de menor valor e reconstruindo a matriz aproximada via SVD. Isso também vale para aproximações de postos maiores, como veremos a seguir.

Usando a SVD para fatorar a matriz $\mathbf{A}$ de posto r, teremos $\mathbf{A} = \mathbf{USV}^T$. A aproximação de posto $p < r$ para a matriz $\mathbf{A}$ é obtida considerando seus p maiores valores singulares na diagonal da matriz $\hat{\mathbf{D}}$ para formar a matriz

$$\hat{\mathbf{S}} = \begin{bmatrix} \hat{\mathbf{D}}_{p\times p} & \mathbf{0}_{p\times(n-p)} \\ \mathbf{0}_{(m-p)\times p} & \mathbf{0}_{(m-p)\times(n-p)} \end{bmatrix}$$

e obtendo a aproximação para $\mathbf{A}$ através do produto $\mathbf{A}_c = \mathbf{U}\hat{\mathbf{S}}\mathbf{V}^T$. Assim, em vez de armazenar as $n \cdot m$ posições da imagem original $\mathbf{A}$, podemos armazenar p colunas da matriz $\mathbf{U}$, p valores singulares e p colunas da matriz $\mathbf{V}$, devido aos zeros que aparecem em $\hat{\mathbf{S}}$. Neste caso, teremos que armazenar dados em

$$N_C = p \cdot (1 + n + m)$$

unidades de memória e a taxa de compressão será de

$$\frac{n \cdot m}{p \cdot (1 + n + m)} : 1.$$

Dessa forma, podemos armazenar menos dados quando $n \cdot m/[p(1 + n + m)] > 1$. Em contrapartida, quando $n \cdot m/[p \cdot (1 + n + m)] < 1$ vale mais a pena armazenar os $n \cdot m$ pixels da imagem original do que usar a SVD para reconstruir a imagem.

3 Exercício com o Matlab

Vamos usar a SVD para comprimir duas imagens. Para isso, faça os itens enumerados a seguir:

1. Use a função imread.m do Matlab para ler a imagem "AlexandreIII.jpg" que pode ser obtida no Moodle. Note que os pixels dessa imagem estão no formato de 8 bits sem sinal (uint8 no Matlab). Para processar a imagem, precisamos convertê-la para o formato double, pois várias funções do Matlab (como svd.m) não aceitam o formato uint8. Para converter os pixels da imagem para double, use a função double.m. Quais as dimensões $m \times n$ dessa imagem? Use a função size.m para descobrir.

2. Veja a imagem usando os comandos

   ```
   image(I1)
   colormap(gray(256))
   ```

   sendo I1 a saída da função imread.m. A função colormap.m é importante para que a imagem seja mostrada com 256 níveis de cinza.

3. Use a função svd.m do Matlab para obter a decomposição em valores singulares da imagem I1. Apresente um gráfico desses valores singulares em função de $k = 1, 2, \cdots, r$, sendo r o posto da matriz. Apresente também um gráfico dos valores singulares normalizados.
   Dica: As funções max.m e diag.m podem ser úteis.

4. Faça um for para obter aproximações para essa imagem considerando os p maiores valores singulares. Considere $p \in [1,\ 150]$. Você não precisa obter as 150 aproximações! Adote por exemplo $p = 1, 5, 10, 15, 25, 50, \cdots, 150$. Para cada aproximação, veja a imagem aproximada e calcule a taxa de compressão e o erro relativo.
   Dicas: A norma de Frobenius pode ser calculada no Matlab usando a função norm.m com a opção 'fro'. Pode ser útil "imprimir" na figura as informações do erro relativo e da taxa de compressão. Use, por exemplo, a função num2str.m para converter os valores de p, do erro e da taxa para *strings* e as use no título da figura.

5. Apresente um gráfico do erro relativo em função de p. Esse erro está de acordo com sua percepção visual da imagem? Comente os valores desse erro para $p < 10$.

6. Quantos valores singulares você usaria para ter uma qualidade visual satisfatória? Qual a taxa de compressão neste caso?

7. Considere agora uma imagem com as mesmas dimensões de "AlexandreIII.jpg", mas com os pixels uniformemente distribuídos no intervalo [0, 255]. Gere a imagem usando a função rand.m do Matlab. Repita os itens anteriores, mas considerando $p \in [1,\ 500]$. Compare com o que você obteve para a imagem "AlexandreIII.jpg". O número de valores singulares que deve ser utilizado neste caso é maior ou menor? Por que?