Exercícios Práticos de Python: Decomposição SVD e Compressão de Imagens

1. Desenvolva um programa que:
   (a) Carregue uma imagem em escala de cinza usando `matplotlib.pyplot.imread()` ou `PIL.Image.open`
   (b) Converta para escala de cinza se necessário.
   (c) Exiba informações sobre a imagem:
      - Dimensões (altura × largura).
      - Tipo de dados (`dtype`).
      - Valores mínimo e máximo dos pixels.
      - Posto da matriz da imagem.
   (d) Plote a imagem original usando `plt.imshow()`.

2. Crie uma função `aproximacao_posto_r(imagem, r)` que:
   (a) Calcule a decomposição SVD da imagem.
   (b) Construa a aproximação de posto r usando apenas os primeiros r valores singulares.
   (c) Calcule o erro de Frobenius de duas formas:
      - **Método direto:** $\|A - A_r\|_F$ (norma da diferença das matrizes).
      - **Método teórico:** $\sqrt{\sigma_{r+1}^2 + \sigma_{r+2}^2 + \cdots + \sigma_n^2}$ (usando valores singulares).
   (d) Verifique se os dois métodos produzem o mesmo resultado.
   (e) Retorne a imagem aproximada e o erro calculado.

   Teste com diferentes valores de r: 1, 5, 10, 20, 50, 100 e compare os erros.

3. Desenvolva um programa que:
   (a) Carregue uma imagem de sua escolha.
   (b) Calcule aproximações de posto r para $r = [1, 5, 10, 25, 50, 100]$.
   (c) Crie um subplot com 2×3 imagens mostrando:
      - A imagem original no primeiro subplot.
      - As 5 aproximações nos demais subplots.
   (d) Para cada aproximação, inclua no título:
      - O valor de r.
      - O erro de reconstrução.
      - A porcentagem de compressão: $\left(1 - \frac{r(m+n+1)}{mn}\right) \times 100$.
   (e) Salve a figura como `comparacao_aproximacoes.png`.

   Desafio adicional: Adicione uma curva mostrando como o erro decresce com o aumento de r.

4. Implemente uma função `analisar_compressao(imagem)` que:

(a) Calcule aproximações para r variando de 1 até o posto máximo da imagem.

(b) Para cada r, calcule:

- Taxa de compressão: razao_compressao $= \frac{r(m+n+1)}{mn}$.
- Erro relativo: erro_relativo $= \frac{\|A-A_r\|_F}{\|A\|_F}$.
- PSNR (Peak Signal-to-Noise Ratio): PSNR $= 20 \log_{10}\left(\frac{255}{\text{RMSE}}\right)$.

(c) Plote três gráficos:

- Erro relativo vs. r.
- Taxa de compressão vs. r.
- PSNR vs. Taxa de compressão.

(d) Identifique o "ponto ótimo" onde se obtém boa qualidade com alta compressão.

Fórmula do RMSE: $\sqrt{\text{mean}((A - A_r)^2)}$.

5. Desenvolva um sistema de compressão para imagens coloridas:

(a) Carregue uma imagem colorida (RGB).

(b) Separe os canais R, G e B.

(c) Para cada canal, aplique SVD e mantenha apenas os primeiros r componentes.

(d) Reconstrua a imagem colorida combinando os canais comprimidos.

(e) Implemente uma função `comprimir_imagem_colorida(imagem, r_red, r_green, r_blue)` que permita diferentes níveis de compressão para cada canal.

(f) Compare visualmente:

- Imagem original.
- Compressão uniforme (mesmo r para todos os canais).
- Compressão adaptativa (r diferente por canal, baseado na variância de cada canal).

Dica: O canal verde geralmente contém mais informação visual importante.

6. Crie três métodos para determinar automaticamente o rank ótimo:

(a) **Método 1 - Critério de Energia:**

- Mantenha componentes até que 99% da "energia" (soma dos quadrados dos valores singulares) seja preservada.

(b) **Método 2 - Critério do Cotovelo:**

- Encontre o ponto de inflexão na curva dos valores singulares usando a segunda derivada.

(c) **Método 3 - Critério de Qualidade Visual:**

- Pare quando o PSNR atingir um limiar (ex: 30 dB).

Implemente uma função `rank_otimo(imagem, metodo='energia', limiar=0.99)` e compare os resultados dos três métodos em diferentes tipos de imagem (fotos, desenhos, texturas).

7. Conduza um estudo comparativo:

(a) Colete 4 tipos diferentes de imagem:

- Foto com muitos detalhes (paisagem).
- Imagem com poucas cores (logo, desenho).
- Textura repetitiva (padrão).
- Imagem com gradientes suaves (céu, pôr do sol).

(b) Para cada imagem, analise:

- Distribuição dos valores singulares (plot dos σ_i).
- Taxa de decaimento dos valores singulares.
- Rank efetivo (número de valores singulares ¿ 1% do máximo).
- Qualidade de reconstrução para diferentes valores de r.

(c) Crie um relatório automatizado que:

- Gere gráficos comparativos.
- Identifique qual tipo de imagem comprime melhor.
- Explique os resultados baseado nas características visuais.

8. **(Opcional)** Implemente um sistema de benchmark que:

(a) Teste a decomposição SVD em imagens de diferentes tamanhos (64×64, 128×128, 256×256, 512×512, 1024×1024).

(b) Meça e compare:

- Tempo de decomposição SVD.
- Tempo de reconstrução para diferentes valores de r.
- Uso de memória para armazenar U, Σ, V^T vs. imagem original.
- Tempo de I/O para salvar/carregar dados comprimidos.

(c) Investigue o impacto do tipo de dados (`float32` vs `float64`).

(d) Compare com outros métodos de compressão (JPEG) em termos de:

- Taxa de compressão.
- Qualidade visual (PSNR, SSIM).
- Tempo de processamento.

(e) Gere gráficos mostrando como a performance escala com o tamanho da imagem.