

Trabalho 5 - MO443 - Introdução ao Processamento de Imagem Digital

Aluno: Gabriel Bianchin de Oliveira RA: 230217

Junho 2019

1 Introdução

O objetivo deste trabalho consiste em aplicar algoritmos de agrupamento, ou seja, algoritmos não supervisionados, para reduzir (quantizar) o número de cores de uma imagem colorida, visando manter a qualidade da aparência da imagem de entrada.

O algoritmo de agrupamento utilizado foi o K-Means com variações de inicialização, sendo as variações K-Means++ e Random. A quantidade de cores para quantizar a imagem utilizadas foram 8, 16, 32, 64 e 128.

2 Código

O código foi implementado em Python 3.6.7, com as bibliotecas OpenCV 3.4.2, NumPy 1.13.3 e Sci-kit Learn 0.19.1.

2.1 Como Executar

Para a execução do código, deve-se executar o script `codigo.py`. O script recebe como argumento uma imagem no formato PNG ou JPG, a técnica de inicialização, a quantidade de cores para quantizar a imagem e o nome do arquivo de saída no formato PNG ou JPG. Um exemplo de execução é mostrado a seguir:

```
python3 codigo.py imagens/baboon.png k-means++ 16 saida.png
```

As opções de variações do K-Means são:

- **k-means++**: aplica o método K-Means++.
- **random**: aplica o método Random.

Caso nenhuma das opções válidas for informada, será enviada uma mensagem, via terminal, informando que não foi possível realizar a aplicação da técnica. Ao final, será salva a imagem resultante na pasta saida.

2.2 Entrada

A entrada de dados consiste em uma imagem no formato PNG ou JPG, a variação de inicialização do K-Means e a quantidade de grupos de imagens. As imagens utilizadas para testes estão na pasta imagens, e foram obtidas pelo seguinte diretório: http://www.ic.unicamp.br/helio/imagens_coloridas/

2.3 Saída

A saída de dados consiste em uma imagem resultante da aplicação da técnica de agrupamento utilizando a quantidade de cores indicada. A imagem será salva na pasta saída.

3 K-Means

O algoritmo K-Means utiliza medidas de distância para separar um conjunto de dados em K grupos. Para isso, o algoritmo segue os passos até convergir:

- Calcular a distância dos dados com os centróides dos K grupos;
- Atribuir o dado ao grupo com menor distância;
- Calcular os novos centróides utilizando a média dos dados que pertencem ao grupo.

Para a inicialização, o usuário deve informar a quantidade de grupos, a inicialização e a métrica de distância [2].

3.1 Random

O método de inicialização Random seleciona aleatoriamente os valores dos centróides dos grupos, como os algoritmos globais de agrupamento [3]. Neste trabalho foi utilizada a biblioteca Sci-kit Learn¹, onde a inicialização Random seleciona aleatoriamente K dados para serem os centróides iniciais.

3.2 K-Means++

O método de inicialização K-Means++ seleciona os centróides iniciais dos grupos de uma maneira mais eficiente. Com isso, o algoritmo K-Means++ se torna superior ao algoritmo com inicialização aleatória dos centróides, possuindo maior acurácia e rapidez [1].

¹<https://scikit-learn.org/stable>

4 Resultados

4.1 Leitura de Dados

A leitura das imagens de entrada é feita pela função **cv2.imread**, da biblioteca OpenCV², que carrega a imagem colorida. Para realizar o agrupamento, é necessário transformar a imagem de 3D, que possui altura, largura e valores RGB, para 2D, transformando uma dimensão em altura multiplicado por largura e a outra em valores de RGB. Para isso, foi utilizada a função **np.reshape**, da biblioteca NumPy³.

Para exemplificar os resultados obtidos, será utilizada a imagem baboon.png, mostrada na Figura 1.

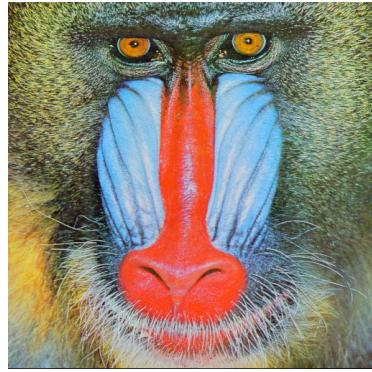


Figura 1: Figura utilizada para exemplificar os resultados

4.2 Agrupamento

A criação do algoritmo K-means é realizada pela função **KMeans**, da biblioteca Sci-kit Learn, passando como parâmetro a quantidade de grupos e o método de inicialização. Após a criação do K-means, é feito o ajuste de agrupamentos do algoritmo.

Terminado o ajuste, o grupo em que cada um dos pixels pertencem é predito e os centros dos grupos são salvos em um *codebook*.

4.3 Reconstrução da Imagem

Para reconstruir a imagem, é criada uma imagem nova em 3D, com largura, altura e valores RGB. Para cada valor predito dos pixels da imagem original, a imagem resultante recebe o valor de cor do pixel correspondente.

²<https://docs.opencv.org/3.0-beta/index.html>

³<https://www.numpy.org>

4.4 Comparação entre Random e K-Means++

Após reconstruir as imagens, é possível comparar os resultados obtidos pela inicialização Random e K-Means++.

A Figura 2 mostra as imagens com 8 cores. Visualmente, as duas imagens são muito semelhantes. O tempo necessário para computar o algoritmo com inicialização Random foi aproximadamente 12 segundos, enquanto o algoritmo com inicialização K-Means++ demorou aproximadamente 11 segundos.

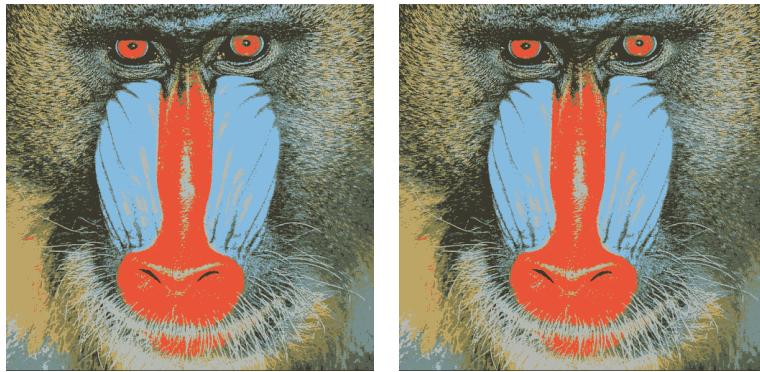


Figura 2: Figura com 8 cores. Imagem à esquerda utiliza inicialização Random e imagem à direita utiliza inicialização K-means++

A Figura 3 mostra as imagens com 16 cores. Novamente, as duas imagens são muito semelhantes. O tempo necessário para computar o algoritmo com inicialização Random foi aproximadamente 84 segundos, enquanto o algoritmo com inicialização K-Means++ demorou aproximadamente 57 segundos.

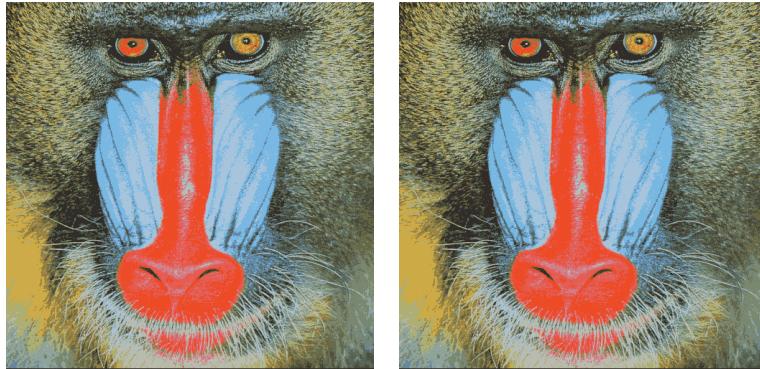


Figura 3: Figura com 16 cores. Imagem à esquerda utiliza inicialização Random e imagem à direita utiliza inicialização K-means++

A Figura 4 mostra as imagens com 32 cores. Mais uma vez, as duas imagens são muito semelhantes. O tempo necessário para computar o algoritmo com

inicialização Random foi aproximadamente 208 segundos, enquanto o algoritmo com inicialização K-Means++ demorou aproximadamente 191 segundos.

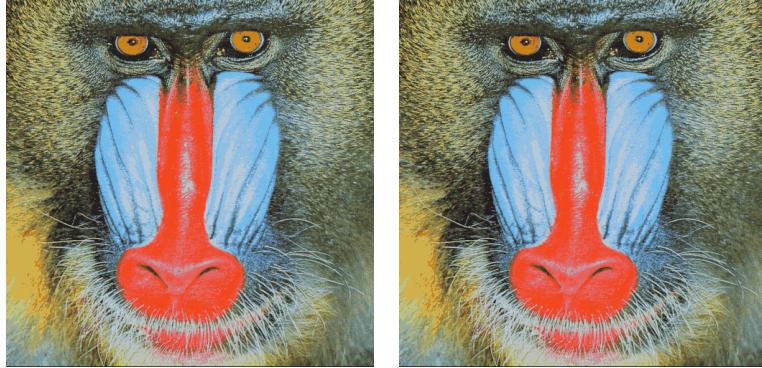


Figura 4: Figura com 32 cores. Imagem à esquerda utiliza inicialização Random e imagem à direita utiliza inicialização K-means++

A Figura 5 mostra as imagens com 64 cores. Outra vez, as duas imagens são muito semelhantes. O tempo necessário para computar o algoritmo com inicialização Random foi aproximadamente 499 segundos, enquanto o algoritmo com inicialização K-Means++ demorou aproximadamente 558 segundos.

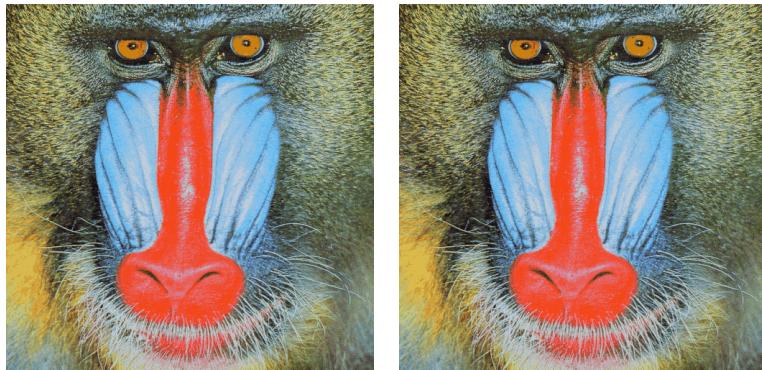


Figura 5: Figura com 64 cores. Imagem à esquerda utiliza inicialização Random e imagem à direita utiliza inicialização K-means++

A Figura 6 mostra as imagens com 128 cores. Visualmente, as duas imagens são muito semelhantes. O tempo necessário para computar o algoritmo com inicialização Random foi aproximadamente 1306 segundos, enquanto o algoritmo com inicialização K-Means++ demorou aproximadamente 1163 segundos.

A Figura 7 mostra a comparação entre os métodos de inicialização utilizando 8, 16, 32, 64 e 128 grupos. Como é possível visualizar, o método de inicialização K-Means++ é mais rápido que o método Random no geral.

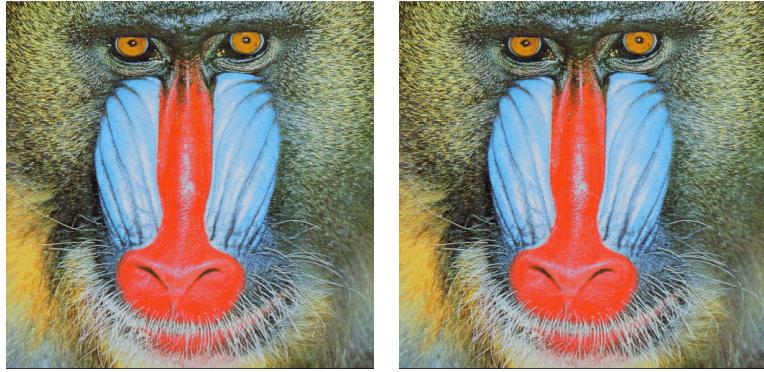


Figura 6: Figura com 128 cores. Imagem à esquerda utiliza inicialização Random e imagem à direita utiliza inicialização K-means++

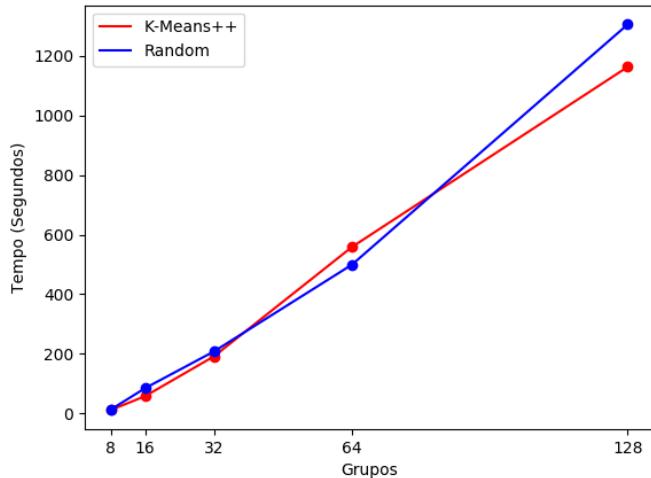


Figura 7: Comparativo de tempo de execução dos algoritmos

5 Conclusão

O método de agrupamento K-Means possui métodos de inicialização, como K-Means++ e Random. Para quantizar imagens, o resultado obtido por esses dois métodos são muito próximos, porém o tempo computacional do K-Means++ no geral é mais rápido. O que foi observado é que acima de 32 grupos, fica mais complicado notar diferenças entre as imagens.

Referências

- [1] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.
- [2] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.
- [3] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003.