Relatório - Trabalho 05

MO443 - Introdução ao Processamento de Imagem Digital

Vinicius Teixeira de Melo - RA: 230223

Universidade Estadual de Campinas
viniciusteixeira@liv.ic.unicamp.br

24 de Junho de 2019

I. Especificação do Problema

O objetivo deste trabalho é aplicar técnicas de agrupamento de dados (ou seja, técnicas de aprendizado de máquina não supervisionado) para reduzir (quantizar) o número de cores de uma imagem colorida, procurando manter a qualidade da aparência geral da imagem de entrada.

Os principais passos do processo de geração da imagem quantizada são:

1. ler a imagem colorida de entrada.
2. aplicar a técnica *k-means* de agrupamento de dados para encontrar grupos de cores mais representativas.
3. salvar dicionário (*codebook*) gerado pela técnica de agrupamento, ou seja, os centros dos grupos e os rótulos correspondentes a cada pixel da imagem.
4. reconstruir a imagem com cores reduzidas a partir do dicionário armazenado.

No passo (2), explore diferentes números de cores, por exemplo, 16, 32, 64 ou 128, para quantizar as imagens. Diferentes técnicas de agrupamento de dados também podem ser avaliadas.

A fundamentação e resolução deste trabalho estão descritas nas seções seguintes.

II. Entrada de Dados

O código fonte criado para a execução de todas as tarefas está no notebook **Trabalho 05.ipynb**. O código foi criado para aceitar imagens coloridas em diversos formatos, como exemplo: *PNG* (*Portable Network Graphics*), *JPG* (*Joint Photographic Group*), entre outros.

Para executar o notebook, basta iniciar o ambiente *Jupyter Notebook*, abrir o notebook **Trabalho 05.ipynb** e executar as células em ordem. Todo o algoritmo foi implementado na linguagem Python na versão 3.6.

As imagens de entrada utilizadas nso testes dos algoritmos foram retiradas da página do prof. Hélio Pedrini: Imagens. Na pasta **imgs/** estão as imagens coloridas utilizadas nos testes: **baboon.png** (512 x 512 *pixels*) e **monalisa.png** (256 x 256*pixels*).

Figura 1: *baboon.png*

Figura 2: *monalisa.png*

III. Dependências e Códigos

As bibliotecas utilizadas neste trabalho foram:

Biblioteca	Versão
numpy	1.16.2
cv2	3.4.2
matplotlib	3.0.3
scikit-learn	0.20.3
warnings	2.1

A leitura das imagens foi realizada utilizando uma função do **matplotlib** [3] chamada **pyplot.imread()**, a qual nescessita apenas do caminho da imagem, já que queremos as informações das imagens assim como estão.

Foi criada uma função para plotar os resultados finais utilizando também uma função do **matplotlib** chamada **pyplot.imshow()**. Nos passos (2) e (3) foram utilizadas funções da biblioteca **sklearn**, como **KMeans()** para a criação do modelo de agrupamento e **shuffle()** para a inicialização da amostragem das cores.

Por fim, no passo (4) foram utilizados os resultados dos passos anteriores para a recriação das imagens com a quantização de cores alteradas. Vale lembrar também que, inicialmente, as imagens foram transformadas de **uint8** para **float64** e seus níveis de intensidade foram normalizados para o intervalo [0,1], ao dividir por 255. E, no final, tiveram seus valores transformados novamente ao multiplicar os níveis de intensidade por 255.

IV. Fundamentação

i. Agrupamento de dados

O clustering ou agrupamento de dados é o conjunto de técnicas de prospeção de dados que visa fazer agrupamentos automáticos de dados segundo o seu grau de semelhança. O critério de semelhança faz parte da definição do problema e, dependendo, do algoritmo. A cada conjunto de dados resultante do processo dá-se o nome de grupo, aglomerado ou agrupamento (*cluster*) [4].

Normalmente o usuário do sistema deve escolher a priori o número de grupos a serem detectados. Alguns algoritmos mais sofisticados pedem apenas o número mínimo, outros tem a capacidade de subdividir um grupo em dois. Quando se tem um conjunto de objetos é natural olhar para eles tentando perceber o quão semelhantes ou diferentes eles são uns dos outros. Uma abordagem comum consiste em definir uma função de distância entre os objetos, com a interpretação de que objetos a uma distância menor são mais semelhantes uns aos outros.

i.1 KMeans

KMeans é um método de quantização vetorial, originalmente aplicado para processamento de sinais, que é popular para análise de agrupamentos em *data mining*. Essa técnica tem como objetivo particionar um conjunto de n objetos em k *clusters*, nos quais cada objeto pertence ao *cluster* com a média mais próxima.

O problema é definido formalmente como: Dado um conjunto de observações $(x_1, x_2, ..., x_n)$ onde cada observação é um vetor real de d dimensões, *kmeans* particiona as n observações em k conjuntos $S = \{S_1, S_2, ..., S_k\}$ de tal forma que minimize a soma dos erros quadráticos.

V. Saída de Dados

As imagens resultantes foram salvas dentro da pasta **resultados/** utilizando uma função da biblioteca **matplotlib** chamada **pyplot.imsave()** [3].

O formato dos nomes de saída estão da seguinte forma: primeiramente é o nome da imagem de entrada (baboon ou monalisa) concatenado com "_" e o número de *clusters* utilizados na criação do modelo de agrupamento *KMeans*.

VI. Resultados e Discussões

Nesta seção são apresentados os resultados obtidos após os passos descritos anteriormente e uma discussão sobre cada resultado.

A Figura 3 apresenta o resultado da aplicação da técnica de agrupamento de cores, utilizando 16 *clusters*. Podemos observar que, como a quantidade de cores é muito reduzida, as imagens resultantes contém regiões que se tornaram homogêneas, tendo assim muitas perdas de informações acerca dos detalhes contidos nas imagens.

A Figura 4 apresenta agora o resultado utilizando 32 *clusters*. Podemos observar que, diferente do resultado anterior, agora é possível notar mais detalhes nas imagens e as áreas homogêneas se tornaram bem menores, por conta de estarmos utilizando uma quantidade maior de cores para quantizar as imagens.

A Figura 5 mostra os resultados obtidos utilizando quantização com 64 *clusters*. Agora, já não é possível fazer uma distinção grande para o resultado com 32 *clusters*, pois o olho humano não distingue esse nível de detalhes. Mas podemos perceber que as regiões que estavam homogêneas no resultado anterior não existem mais, assim, o nível de detalhes foi elevado.

Por fim, a Figura 6 apresenta os resultados obtidos utilizando a técnica de agrupamento com 128 *clusters*. Já não é mais possível distinguir diferenças para o resultado anterior (64 *clusters*), o que nos diz que está próximo ou já chegou na quantidade de cores utilizadas na imagem anterior, assim, o agrupamento não altera mais as imagens de entrada.

Figura 3: *Resultado da quantização com 16 cores.*

VII. Conclusão

Podemos concluir que os resultados obtidos com as aplicações da técnica de quantização de imagem *KMeans* se mostrou eficiente e efetivo, apresentando resultados satisfatórios quanto a especificação do trabalho e se mostrou razoável para este objetivo, assim, consolida ainda mais os conceitos vistos em sala de aula.

Referências

[1] Documentation of scikit-learn. https://scikit-

Figura 4: *Resultado da quantização com 32 cores.*

Figura 5: *Resultado da quantização com 64 cores.*

learn.org/stable/documentation.html
Acesso em: 22/06/2019.

[2] Pedrini, Hélio, and William Robson Schwartz. Análise de imagens digitais: princípios, algoritmos e aplicações. Thomson Learning, 2008.

[3] Matplotlib Version 3.0.3 https://matplotlib.org/contents.html
Acesso em: 22/06/2019.

[4] Clustering https://pt.wikipedia.org/wiki/Clustering
Acesso em: 22/06/2019.

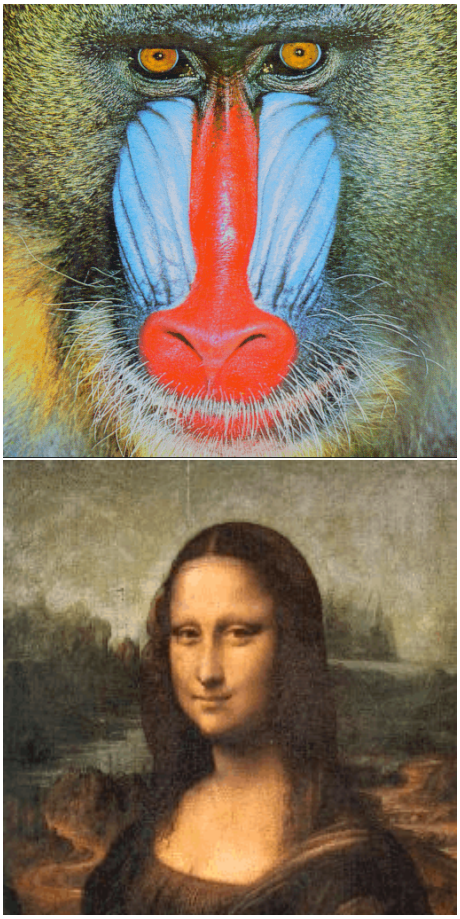

Figura 6: *Resultado da quantização com 128 cores.*