Computação da Matriz SCM em Imagens

1 Considerações Iniciais

Neste trabalho, você deverá implementar em Linguagem C a matriz de coocorrência estrutural (*Structural Co-occurrence Matrix*, SCM) para descrever as características de uma imagem de entrada e realizar um experimento de classificação utilizando a ferramenta Weka `https://www.cs.waikato.ac.nz/ml/weka/index.html`.

A matriz SCM deve ser computada entre uma imagem e sua versão suavizada. Ao final, os valores da matriz devem ser vetorizados e gravados em um arquivo para posterior processamento.

Maiores detalhes sobre a matriz SCM, consulte o artigo [*Ramalho et al*, 2016], disponível para *download* em nosso diretório no dropbox.

Título: *Rotation-invariant feature extraction using a structural co-occurrence matrix.*
`https://doi.org/10.1016/j.measurement.2016.08.012`

2 Computação da Matriz SCM - Simplificada

Sendo I uma imagem de entrada e $\bar{I}$ sua versão suavizada, considerando Q como uma Quantização em N níveis, a SCM pode ser computada conforme ilustrado a seguir:

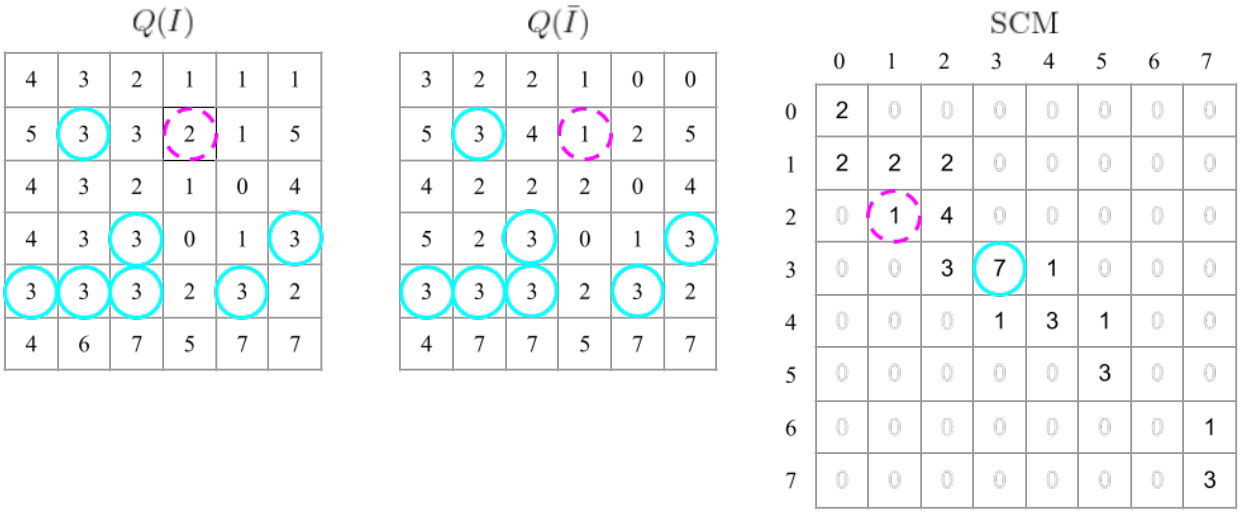

Figura 1: SCM com Quantização em N=8 níveis. A SCM tem dimensão NxN.

As etapas do algoritmo estão descritas a seguir:

1. Leitura das imagens em um diretório. Para cada imagem, gerar também a sua versão suavizada com o filtro da média (janelas 3x3, 5x5 e 7x7). As imagens serão disponibilizadas pelo professor no formato PGM (discutido na próxima seção deste documento).
2. Quantização da imagem e de sua versão suavizada em N níveis.
3. Computação da matriz SCM.

4. Vetorizar a matriz SCM e gravar os valores em um arquivo de características (arquivo texto), sendo uma linha para cada uma das imagens processadas. No final do vetor, incluir o rótulo da imagem (disponível no nome do arquivo, sendo 0 para *epithelium* e 1 para *stroma*). Base de dados a ser utilizada: Epistroma (`https://diagnosticpathology.biomedcentral.com/articles/10.1186/1746-1596-7-22`).

Observações:

- O arquivo de características deve ser gravado em modo texto. Use "," para separar os valores. Para o exemplo ilustrado na Figura 1, temos:

  2, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 0, 0, 0, 0, 0, 1, 4, 0, 0, 0, 0, 0, 0, 0, 3, 7, 1, 0, 0, 0 ... **stroma**

  onde o valor em negrito representa o rótulo.

- A quantidade de níveis para quantização deve ser fornecida por linha de comando. Identifique no nome do arquivo de características a quantidade de níveis de quantização utilizado.

3 Experimentos

A equipe deve apresentar, juntamente com códigos, os resultados obtidos a partir de um experimento de classificação utilizando o Weka.

As perguntas a serem respondidas são as seguintes: *dado a base de imagens utilizada neste trabalho, com quantos níveis de quantização N (8 ou 16) a SCM apresenta os melhores resultados? E qual o impacto de se utilizar o filtro da média com janela 3x3, 5x5 e 7x7?*

Nas seções seguintes deste documento estão descritas algumas informações necessárias para o correto desenvolvimento deste trabalho.

4 Formato PGM

Neste trabalho, deve-se utilizar o formato PGM (*portable graymap*) para armazenar imagens em arquivos. Este formato tem duas variações, uma binária (o PGM "normal" ou *raw*) e outra textual (o PGM ASCII ou *plain*). Em ambos os casos, o arquivo deve conter um cabeçalho e a matriz correspondente à imagem. O exemplo a seguir mostra um arquivo PGM textual:

```
P2
5 4
16
9 4 5 0 8
10 3 2 1 7
9 1 6 3 15
1 16 9 12 7
```

A primeira linha do arquivo contém obrigatoriamente uma palavra-chave, que deve ser "P2" no caso de um arquivo PGM textual e "P5" no caso de um arquivo PGM binário. A segunda linha contém dois números inteiros que indicam o número de colunas e o número de linhas da matriz, respectivamente. A terceira linha contém um número inteiro positivo *maxval*, que deve ser igual ao maior elemento da matriz. Na definição do formato PGM, *maxval* não pode ser maior que 65535. Para fins deste trabalho, entretanto, *maxval* é no máximo 255. Os demais números do arquivo são os elementos de uma matriz de inteiros com os tons de cinza de cada ponto da imagem. Cada tom de cinza é um número entre 0 e *maxval*, com 0 indicando "negro" e *maxval* indicando "branco".

O formato PGM também permite colocar comentários. Todo o texto que vai desde um caractere '#' até (e inclusive) o próximo fim de linha é um comentário e deve ser ignorado. Este é um exemplo de arquivo PGM textual com um comentário:

```
P2
# feep.pgm
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 3 3 3 0 0 7 7 7 7 0 0 11 11 11 0 0 15 15 15 0 0 0
0 0 0 3 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 0 0 3 0 0 0 7 7 7 0 0 0 11 0 0 0 0 0 15 15 15 0 0
0 0 0 3 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 0 3 3 3 0 0 7 0 0 0 0 0 11 11 11 0 0 15 15 15 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

O formato PGM binário tem cabeçalho análogo ao do PGM textual, usando a palavra chave "P5" em vez da "P2". O que muda é o modo como é armazenada a matriz de tons de cinza. No formato PGM textual, essa matriz é guardada como uma sequência de caracteres ASCII contendo as representações decimais das tonalidades de cinza. No formato PGM binário, a matriz é guardada como uma sequência de *bytes*, sendo o valor de cada *byte* (de 0 a 255) uma tonalidade de cinza. O número de *bytes* da sequência é exatamente igual ao número de elementos da matriz[1]. Este é um exemplo de arquivo PGM binário:

```
P5
# feep.pgm
24 7
15
```

... (sequência de 24 x 7 bytes com as tonalidades de cinza)

Existem dois outros formatos de arquivo muito semelhantes ao PGM: o PBM (*portable bitmap*), para imagens monocromáticas (só preto e branco, sem tons de cinza), e o PPM (*portable pixmap*), para imagens coloridas. No primeiro, os elementos da matriz podem assumir apenas os valores 0 e 1. No segundo, os elementos da matriz são triplas de números inteiros positivos correspondentes às intensidades das cores vermelha, verde e azul nos pontos da imagem. Quem quiser saber mais detalhes sobre esses formatos, visite as seguintes páginas:

- http://en.wikipedia.org/wiki/Portable pixmap
- http://netpbm.sourceforge.net/doc/pbm.html
- http://netpbm.sourceforge.net/doc/pgm.html
- http://netpbm.sourceforge.net/doc/ppm.html

[1] Na verdade, essa descrição se aplica apenas a arquivos PGM com $maxval \leq 255$, que são considerados neste trabalho. No caso $256 \leq maxval \leq 65535$, a matriz é guardada como um sequência de pares de *bytes* e o número de *bytes* da sequência é o dobro do número de elementos da matriz.

5 Sobre a organização dos códigos fontes

ATENÇÃO:

1. A nota máxima deste trabalho é 10.0. A distribuição dos pontos segue como definido a seguir:

   - **Entrada e saída com arquivos texto ou binário:** Pontuação: 1.0
   - **Implementação do filtro da média (janela 3x3 e 5x5):** Pontuação: 3.0
   - **Implementação da Quantização com N níveis:** Pontuação: 2.0
   - **Computação da Matriz SCM:** Pontuação: 3.0
   - **Organização do código:** Pontuação: 1.0
   - **Discussão dos resultados a partir do experimento de classificação proposto:** Pontuação Extra: 1.0

   Critérios para cada tópico acima:

   - Apresentação: 20%
   - Domínio da solução adotada: 50%
   - Análise Técnica: 30%

2. Todos os arquivos fontes do seu programa devem conter um cabeçalho como o seguinte:

```
/*************************************************************/
/* Aluno: Joao de Souza
/* Matricula: 12345
/* Avaliacao 04: Trabalho Final                              */
/* 04.505.23 - 2023.2 - Prof. Daniel Ferreira                */
/* Compilador:...(DevC++ ou gcc) versao ...                  */
/*************************************************************/
```

   A organização do programa deve obedecer a construção de arquivos de cabeçalhos e permitir a compilação separada. Deve-se utilizar processo de *linkedição*. Procure dividir seus códigos em arquivos sempre que necessário. Enfim, procure construir bibliotecas genéricas que proporcione posterior reuso.

3. O trabalho deverá ser realizado em **equipes com no máximo 4 (quatro) participantes**. **Nota individual.** Sugiro, portanto, que marquem seminários entre a equipe para que todos possam atingir o mesmo nível de conhecimento.

4. Códigos fontes copiados (com ou sem eventuais disfarces) receberão nota **ZERO**.

5. Programas que não executam receberão nota **ZERO**.

6. Trabalhos atrasados serão penalizados. A regra é a seguinte: para cada dia de atraso o aluno perde **25%** da nota.

7. É muito importante que seu programa tenha comentários e esteja bem indentado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos dos programas (conforme estudado em aula). O não obediência a este item poderá resultar em perda de até 15% da nota obtida.

8. O professor poderá executar o programa tantas vezes quantas forem necessárias para testar os vários casos possíveis para as entradas.

9. Os códigos devem ser disponibilizados ao professor por meio de um link no GitHub. Deve ser construído um README com detalhes para a execução.

10. O relatório com os resultados do experimento deve ser submetido na respectiva atividade o Classroom.

11. No dia da apresentação é **indispensável** o comparecimento de toda a equipe. Faltas devem ser devidamente justificadas para projeto de segunda-chamada.

12. Datas para as apresentações:

    - **Dia 1: 11 de Dezembro de 2023 a partir das 07h40.**
    - **Dia 2: 13 de Dezembro de 2023 a partir das 07h40**.

    A ordem de apresentação será definida posteriormente por sorteio.

13. Para fins de revisão do professor, salve em um arquivo a ordem de processamento das imagens. Organize a sequência de processamento colocando o nome de cada imagem por linha (inclua a extensão).