BCM0505-22 – Processamento da Informação

Matrizes - Parte 2

Carla Negri Lintzmayer
carla.negri@ufabc.edu.br
http://professor.ufabc.edu.br/~carla.negri/

Outline

Multiplicação de matrizes

Quadrado mágico

Imagens

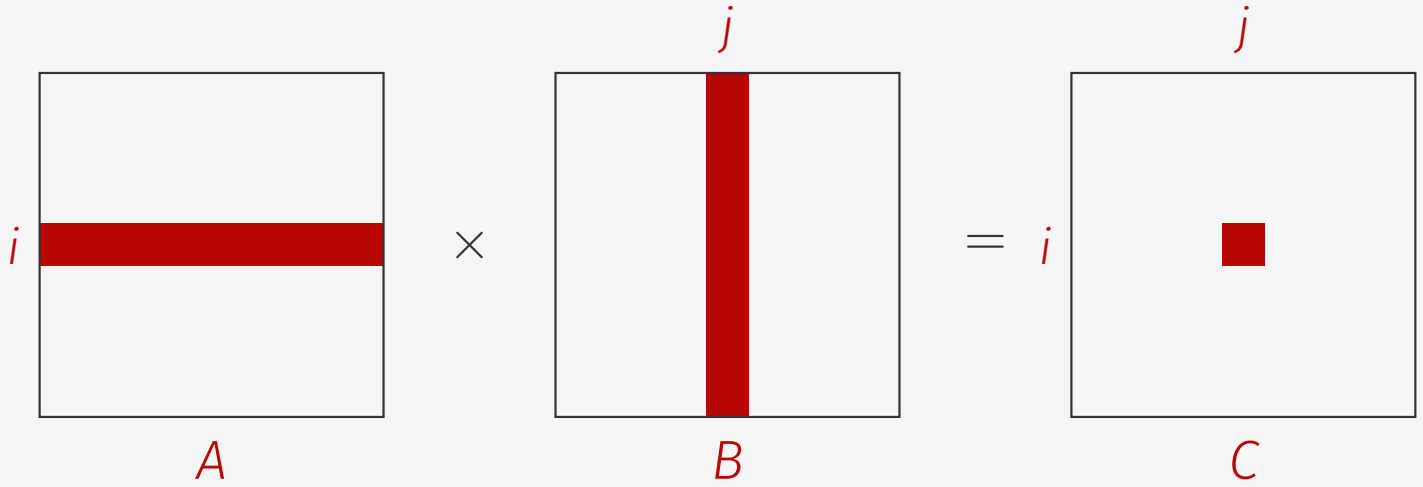
Exercícios

Multiplicação de matrizes

Multiplicação de Matrizes Quadradas

Dadas duas matrizes A e B em $\mathbb{R}^{n \times n}$, calcular $C = A \times B$

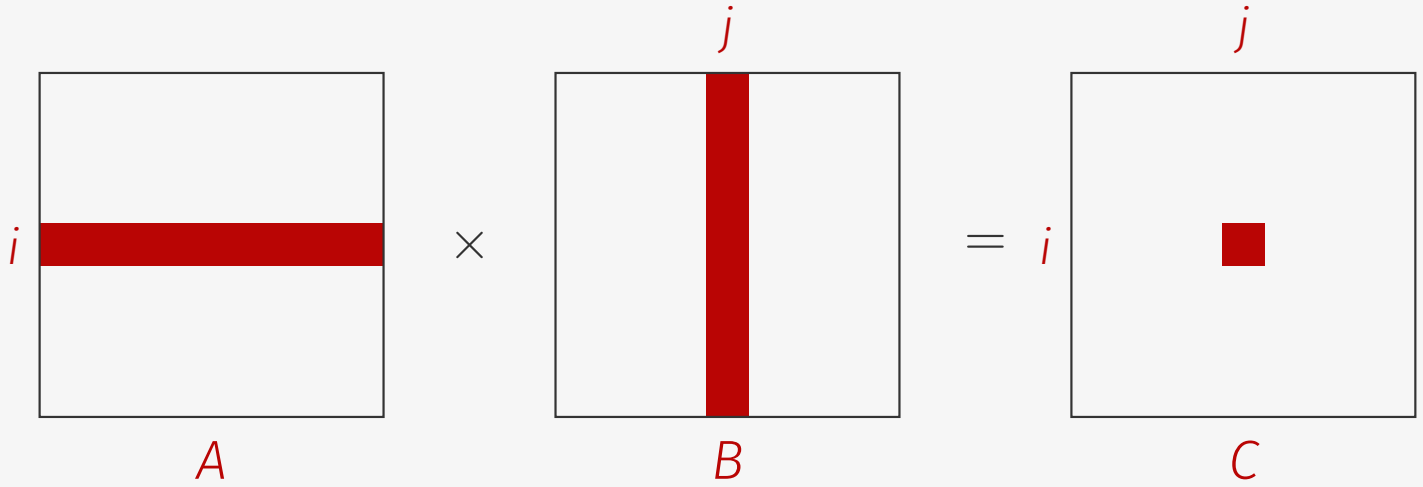
Relembrando...



Multiplicação de Matrizes Quadradas

Dadas duas matrizes A e B em $\mathbb{R}^{n \times n}$, calcular $C = A \times B$

Relembrando...



C_{ij} é o produto escalar da linha i de A com a coluna j de B

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

Python: Multiplicação de Matrizes

Dadas duas matrizes A e B em $\mathbb{R}^{n \times n}$, calcular $C = A \times B$

Python: Multiplicação de Matrizes

Dadas duas matrizes A e B em $\mathbb{R}^{n \times n}$, calcular $C = A \times B$

```python
def matriz_produto(A, B):
    n = len(A)
    C = cria_matriz_nula(n, n)
    for i in range(n):
        for j in range(n):
            for k in range(n):
                C[i][j] += A[i][k] * B[k][j]
    return C
```

Quadrado mágico

Quadrado mágico

Dizemos que uma matriz quadrada de ordem n é um quadrado mágico se todos os números de 1 até n^2 aparecem exatamente uma vez e se a soma das linhas, colunas e diagonais é a mesma.

Por exemplo, se $n = 3$, a seguinte matriz é um quadrado mágico:

```
8  1  6
3  5  7
4  9  2
```

Faça um programa que determine se uma dada matriz é um quadrado mágico.

Quadrado mágico

Comecemos verificando se a matriz só possui números de 1 até n^2. Uma forma é usar um vetor cuja i-ésima posição armazena `True` quando o número i aparece na matriz.

```
1  def verifica_conteudo(A):
2      n = len(A)
3      numeros_encontrados = [False] * n**2
4      for lin in range(n):
5          for col in range(n):
6              if A[lin][col] >= 1 and A[lin][col] <= n**2:
7                  numeros_encontrados[A[lin][col]-1] = True
8
9      for elem in numeros_encontrados:
10         # elem é True ou False
11         if not elem:
12             return False
13
14     return True
```

Quadrado mágico

Outra forma de verificar se a matriz só possui números de 1 até n^2 é procurando por cada um desses números nela.

```
1   def verifica_conteudo(A):
2       n = len(A)
3       for i in range(1, n**2+1):
4           # procura por i em alguma posição da matriz
5           existe = False
6           for lin in range(n):
7               for col in range(n):
8                   if A[lin][col] == i:
9                       existe = True
10          if not existe:
11              return False
12      return True
```

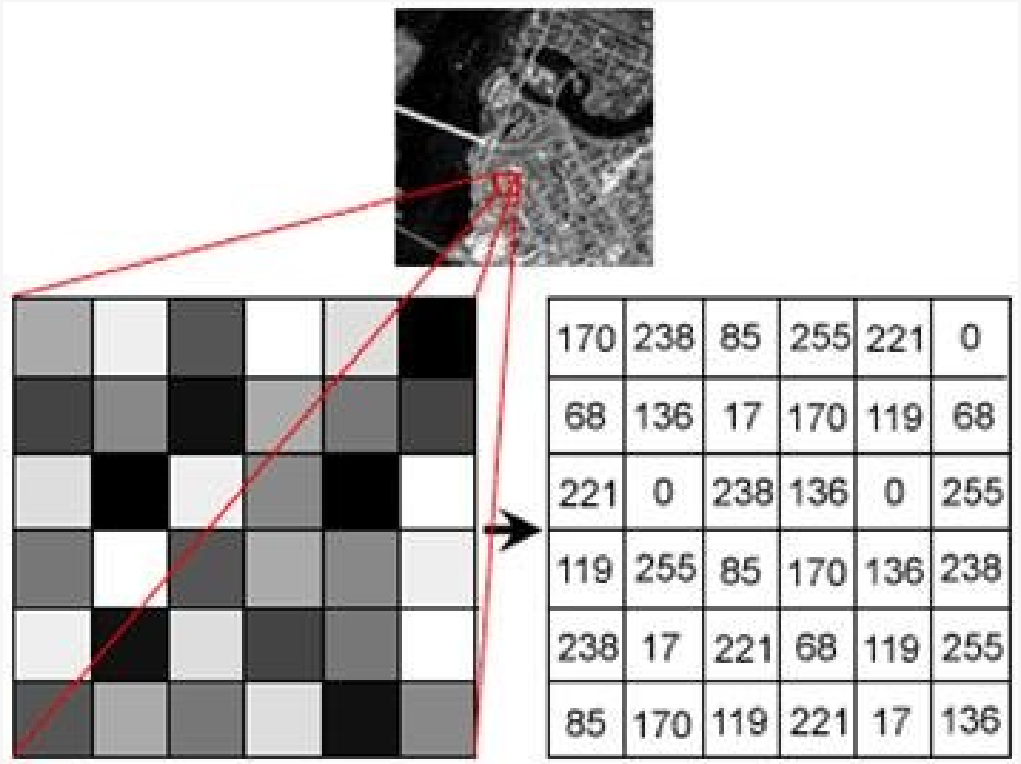
Quadrado mágico

```python
def eh_quadrado_magico(A):
    assert(len(A) == len(A[0]))
    n = len(A)

    # Primeiro verificar se os números de 1 a n² estão
    if verifica_conteudo(A) == False:
        return False

    soma_base = 0           # soma da primeira linha
    for col in range(n):
        soma_base += A[0][col]

    for lin in range(1, n):         # Todas as outras linhas devem ter essa soma
        soma = 0
        for col in range(n):
            soma += A[lin][col]
        if soma != soma_base:
            return False

    for col in range(n):            # Todas as colunas devem ter essa soma
        soma = 0
        for lin in range(n):
            soma += A[lin][col]
        if soma != soma_base:
            return False

    soma_princ = 0
    soma_sec = 0
    for i in range(n):              # As diagonais devem ter essa soma
        soma_princ += A[i][i]
        soma_sec += A[i][n-i-1]
    if soma_princ != soma_base or soma_sec != soma_base:
        return False

    return True
```

Imagens

Imagens

Um uso importante de matrizes é na representação de imagens. Uma imagem em escala de cinza, por exemplo, pode ser representada como uma matriz de inteiros, em que cada número representa o nível de brilho de um pixel, variando de 0 (preto) a 255 (branco).



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

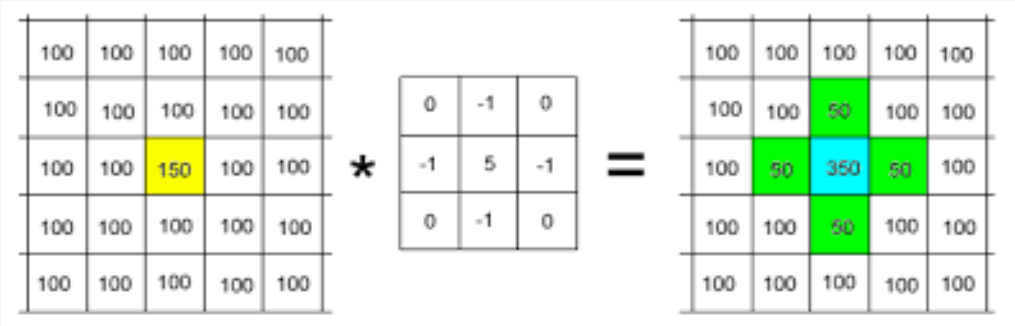
Tratando imagens

A aplicação de um **filtro** em uma imagem é uma técnica usada para manipular ou realçar características da imagem (como suavizar, destacar bordas, borrar, etc). Filtros são base para operações em visão computacional, aprendizado de máquina, processamento de imagens, etc.

O filtro é uma pequena matriz chamada **máscara** (ou **kernel**).

Para aplicar uma máscara a uma imagem, fazemos um processo chamado **convolução**:

- Centralize a máscara em um pixel da imagem
- Multiplique cada valor da máscara pelo valor do pixel correspondente da imagem
- Some os resultados (e divida pela soma dos pesos da máscara, se necessário)
- O resultado substitui o pixel original na nova imagem



Tratando imagens - Principais filtros

Filtro de média: suaviza a imagem, reduzindo ruído e detalhes.
O valor final precisa ser dividido por 9.

```
1  1  1
1  1  1
1  1  1
```

Filtro Gaussiano: suaviza a imagem, mas com pesos que valorizam o centro.
O valor final precisa ser dividido por 16.

```
1  2  1
2  4  2
1  2  1
```

Filtro de detecção de bordas: destaca áreas com transições bruscas de intensidade.
Cuidado com valores fora dos limites da imagem!

```
 0  -1   0
-1   4  -1
 0  -1   0
```

Filtro de realce: contrasta regiões com pouco relevo.
Cuidado com valores fora dos limites da imagem!

```
 0  -1   0
-1   5  -1
 0  -1   0
```

Os filtros podem causar (d)efeitos nas bordas da imagem. Por isso, bordas geralmente são tratadas com cuidados especiais.

Imagens

O Portable Graymap Format (PGM) é um formato simples de arquivo de imagem que representa imagens em escala de cinza.

```
P2
# exemplo simples
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 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 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 0
```

- P2 indica que é um arquivo PGM em texto (ASCII)
- A linha com # é um comentário (opcional)
- 4 3 são as dimensões da imagem: 4 colunas e 3 linhas
- 255 é o valor máximo de cinza (0 = preto, 255 = branco)
- Os valores seguintes são os tons de cinza de cada pixel (escritos linha por linha)

Imagens

O Portable Graymap Format (PGM) é um formato simples de arquivo de imagem que representa imagens em escala de cinza.

```
P2
# exemplo simples
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 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 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 0
```

- P2 indica que é um arquivo PGM em texto (ASCII)
- A linha com # é um comentário (opcional)
- 4 3 são as dimensões da imagem: 4 colunas e 3 linhas
- 255 é o valor máximo de cinza (0 = preto, 255 = branco)
- Os valores seguintes são os tons de cinza de cada pixel (escritos linha por linha)

Faça um programa que recebe uma imagem PGM e aplica um filtro de realce, produzindo uma imagem nova.

Filtro de realce

```python
def aplica_filtro_realce(I):

    mascara = [[0, -1, 0], [-1,  5, -1], [0, -1, 0]]

    novaI = cria_matriz_nula(len(I), len(I[0]))

    for i in range(1, len(I)-1):
        for j in range(1, len(I[i])-1):
            # centralizar máscara em [i][j] e aplicar ao redor
            pixel = 0
            for k in range(-1, 2):
                for l in range(-1, 2):
                    pixel += mascara[k+1][l+1] * I[i+k][j+k]

            if pixel < 0:
                pixel = 0
            elif pixel > 255:
                pixel = 255
            novaI[i][j] = pixel

    return novaI
```

Filtro de realce

Restante do programa:

```python
def carrega_imagem_pgm(arquivo):
    f = open(arquivo, 'r')
    header = f.readline()
    header = f.readline()
    largura, altura = list(map(int, f.readline().split()))
    maior = f.readline()

    I = []
    for i in range(altura):
        linha = list(map(int, f.readline().split()))
        I.append(linha)

    return I

def salva_imagem_pgm(arquivo, imagem):
    altura = len(imagem)
    largura = len(imagem[0])

    f = open(arquivo, 'w')
    f.write("P2\n")
    f.write("# comentário\n")
    f.write(f"{largura} {altura}\n")
    f.write("255\n")
    for i in range(altura):
        for j in range(largura):
            f.write(f"{imagem[i][j]} ")
        f.write("\n")
    f.close()

nome = "imagem.pgm"
imagem = carrega_imagem_pgm(nome)
imagem_realce = aplica_filtro_realce(imagem)
novo_nome = nome.split(".")[0] + "_realce.pgm"
salva_imagem_pgm(novo_nome, imagem_realce)

```

Exercícios

Multiplicação de quaisquer matrizes

Dadas uma matriz A em $\mathbb{R}^{n \times m}$ e uma matriz B em $\mathbb{R}^{m \times p}$, calcular $C = A \times B$.
Continua valendo que C_{ij} é o produto escalar da linha i de A com a coluna j de B

$$C_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}$$

Preencher uma Matriz em Espiral

Escreva um programa que leia um número inteiro positivo n e preencha uma matriz quadrada de tamanho $n \times n$ com os números de 1 até n^2 em ordem crescente, seguindo o formato espiral, no sentido horário, começando do canto superior esquerdo.

Assim, se $n = 3$, o resultado deve ser

```
1  2  3
8  9  4
7  6  5
```

E se $n = 4$, o resultado deve ser

```
1   2   3   4
12  13  14  5
11  16  15  6
10   9   8  7
```

Filtro de redução de ruído

Faça um programa que aplica um filtro de redução de ruído a uma imagem.
(Existem imagens PGM no Moodle.)

Detectar pessoas famosas

Em um conjunto de n pessoas, nomeadas de 0 a $n-1$, sabemos quem conhece quem.

Essa informação é dada por meio de uma matriz de conhecimento `conhece`, $n \times n$, em que `conhece[i][j]` é igual a `1` se a pessoa `i` conhece a pessoa `j`, e `0` caso contrário.

Faça um programa que identifique se existe alguma pessoa famosa no grupo. Uma pessoa famosa é aquela que é conhecida por todos os outros e não conhece ninguém.