Aula 02 - Parte 01 - Sistemas Lineares e Matrizes

Álgebra Linear e Teoria da Informação

Prof. Tiago Tavares

Quanto custa um brigadeiro?

Uma doceria quer calcular o custo para produzir uma fornada de brigadeiros. Para isso, precisa saber a quantidade usada de cada ingrediente e o custo de suas embalagens.

Ingrediente	Custo (Embalagem)	Quantidade Usada
Leite Condensado	R$10,00 (lata 400g)	1 lata
Manteiga	R$15,00 (pote 200g)	1/5 pote (40g)
Chocolate em Pó	R$18,00 (caixa 200g)	1/10 caixa (20g)

A tarefa é multiplicar a quantidade de cada embalagem usada pelo seu custo e somar tudo.

Solução 1: O Laço Explícito

A forma mais direta de resolver isso em programação é usar um laço que percorre cada ingrediente, calcula seu custo parcial e o acumula em uma variável total.

1. Custo inicial é zero: $C_T \leftarrow 0$
2. Para cada ingrediente i com custo c_i e quantidade q_i:
    i. $C_T \leftarrow C_T + c_i q_i$

Matematicamente, estamos implementando a soma:

$$C_T = \sum_{i=0}^{N-1} c_i q_i$$

Solução 2: Representação vetorial

O `numpy` nos permite tratar as listas de custos e quantidades como **vetores**. Isso simplifica o código, eliminando o laço. A multiplicação `*` entre dois vetores `numpy` realiza uma operação "ponto a ponto".

```
import numpy as np

custo_por_ingrediente = np.array([10, 15, 18])
quantidades = np.array([1, 1/5, 1/10])

# Multiplica cada custo pela sua quantidade
custo_parcial = custo_por_ingrediente * quantidades
# Resultado: [10 * 1, 15 * 0.2, 18 * 0.1] = [10, 3, 1.8]

# Soma tudo para obter o custo total
custo_total = np.sum(custo_parcial)
```

Solução 3: O Produto Interno

A operação de multiplicar ponto a ponto e depois somar é tão fundamental que tem um nome: **produto interno**. Ele mede o quanto dois vetores "apontam na mesma direção", ponderando seus elementos.

O `numpy` possui uma função otimizada para isso: `np.inner()`.

```
custo_por_ingrediente = np.array([10, 15, 18])
quantidades = np.array([1, 1/5, 1/10])

# A operação inteira em uma única linha!
custo_total = np.inner(custo_por_ingrediente, quantidades)
# Resultado: 14.8
```

A notação matemática é: $C_T =< \boldsymbol{C}, \boldsymbol{Q} >$

Expandindo o Cardápio

E se a doceria agora vende brigadeiro, ganache, doce de leite e doce de coco?

Podemos ter um vetor de custos com todos os ingredientes possíveis e um vetor de quantidades para cada receita. Para representar um ingrediente que não é usado, simplesmente colocamos **zero** em sua posição.

```
# Vetor com TODOS os ingredientes da loja
custo_por_ingrediente = np.array([10, 15, 18, 2.8, 35])
#                                 [LC, Man, Cho, CL, Coco]

# Vetores de quantidade para cada receita
quant_brigadeiro = np.array([1, 0.2, 0.1, 0, 0])
quant_ganache = np.array([0, 0.2, 0.2, 1, 0])
```

Calculamos o custo de cada um usando o produto interno com o vetor de custos.

Matriz: um "vetor de vetores"

Em vez de uma lista de vetores para as receitas, podemos empilhá-los para formar uma **matriz**: uma estrutura retangular onde cada **linha** é uma receita e cada **coluna** é um ingrediente.

Isso nos dá uma visão completa e organizada do nosso "sistema" de produção.

```
# Cada linha é uma receita, cada coluna um ingrediente
X = np.array([ [1, 0.2, 0.1, 0, 0],    # Brigadeiro
               [0, 0.2, 0.2, 1, 0],    # Ganache
             ])
```

Essa é a nossa **Matriz de Receitas (X)**.

Multiplicação Matricial

A **multiplicação matricial** combina uma matriz com outra, calculando o produto interno de cada linha da primeira com cada coluna da segunda:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 \times 5 + 2 \times 7 & 1 \times 6 + 2 \times 8 \\ 3 \times 5 + 4 \times 7 & 3 \times 6 + 4 \times 8 \end{bmatrix}$$

Importante:

Se $C = AB$, $A \in \mathbb{R}^{N \times K_a}$, $B \in \mathbb{R}^{K_b \times M}$, então:

- Se $K_a = K_b$, então $C \in \mathbb{R}^{N \times M}$
- Se $K_a \neq K_b$, então a multiplicação não pode ser realizada.

Como interpretar a multiplicação matricial

Se tivermos:

- **X**: Matriz de Receitas (4 receitas x 5 ingredientes)
- **Y**: Matriz de Custos (5 ingredientes x 1 coluna de custo)

A operação XY nos dará uma nova matriz Z (4 receitas x 1 coluna de custo), onde cada elemento é o custo total daquela receita.

O Modelo Compacto

Todo o nosso problema de cálculo de custos para múltiplas receitas se resume a uma única linha de código:

```
# Matriz de Receitas (Receitas x Ingredientes)
X = np.array([ [1, 0.2, 0.1, 0, 0],
               [0, 0.2, 0.2, 1, 0],
               ... ])

# Matriz (vetor-coluna) de Custos (Ingredientes x Custo)
Y = np.array([ [10], [15], [18], [2.8], [35] ])

# Calcula o custo de TODAS as receitas de uma só vez
Z = X @ Y
```

O resultado `Z` contém o custo de cada receita, na ordem.

Multiplicação matricial é uma função (com domínio e contradomínio!)

Em nosso exemplo, se temos:

- **X**: Matriz de Receitas (4 receitas x 5 ingredientes)
- **Y**: Matriz de Custos (5 ingredientes x 1 coluna de custo *da receita*)

então $Z = XY \in \mathbb{R}^{4\times 1}$, associando receitas ao custo da receita.

Mas, se tivermos uma matriz de kits $K \in \mathbb{R}^{3\times 4}$ que associa kits (que podem ser vendidos) às receitas que o compõem, então $KZ \in \mathbb{R}^{3\times 1}$ associa os kits ao seu custo total.

... e assim podemos encadear várias transformações!

Exercício: calculando minhas notas

Uma disciplina tem 4 provas. O aluno a_1 tirou 5, 5, 6 e 2. O aluno a_2 tirou 2, 3, 6 e 8. Os pesos das provas são 2, 2, 3 e 3.

1. Usando o produto interno, faça um modelo que permita calcular a nota final do aluno a_1.
2. Usando matrizes, faça um modelo que permita calcular as notas finais dos dois alunos.