Programação Dinâmica

Juliana Kaizer Vizzotto

Universidade Federal de Santa Maria

Disciplina de Inteligência Artificial

Roteiro

- Programação Dinâmica: Revisão
- Multiplicação de Cadeias de Matrizes com Programação Dinâmica

Programação Dinâmica: introdução

- Resolve problemas combinando as soluções de subproblemas.
- A programação dinâmica é aplicável quando os subproblemas não são indenpendentes, i.e., quando os subproblemas compartilham subproblemas.
- O termo "programação" se refere ao método tabular (montar uma tabela dinamicamente)
- A programação dinâmica em geral é aplicada a **problemas de otimização**.
- Em problemas de otimização pode haver muitas soluções possíveis. Cada solução tem um valor e desejamos encontrar uma solução com um valor ótimo.

Programação Dinâmica: etapas

1. Caracterizar a estrutura de uma solução ótima.
2. Definir recursivamente o valor de uma solução ótima.
3. Calcular o valor de uma solução ótima em um processo de baixo pra cima (botton-up)
4. Construir uma solução ótima a partir de informações calculadas.

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- **Problema:** recebemos uma cadeia de entrada $\langle A_1, A_2, A_3, ..., A_n$ de n matrizes a serem multiplicadas e desejamos calcular o produto:

$$A_1 A_2 \ldots A_3$$

- Podemos avaliar a expressão acima usando o algoritmo padrão para multiplicação de pares de matrizes como uma subrotina, uma vez que o tenhamos colocado entre parênteses para solucionar as ambiguidades.
- Um produto de matrizes é **completamente colocado entre parênteses** se ele for uma única matriz ou o produto de dois produtos de matrizes completamente colocado entre parênteses.

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- A multiplicação de matrizes é associativa, e assim todas as colocações entre parentêses resultam no mesmo produto.
- Por exemplo, se a cadeia de matrizes é $\langle A_1, A_2, A_3, A_4 \rangle$, o produto $A_1 A_2 A_3 A_4$ pode ser completamente colocado entre parênteses de cinco modos distintos:

$$(A_1(A_2(A_3A_4)))$$
$$(A_1((A_2A_3)A_4))$$
$$((A_1A_2)(A_3A_4))$$
$$((A_1(A_2A_3))A_4)$$
$$(((A_1A_2)A_3)A_4)$$

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- O modo como uma cadeia de matrizes é colocada entre parênteses pode ter um impacto dramático sobre o custo de avaliação do produto.

- Considere, primeiro o custo de multiplicar duas matrizes:

```
MATRIX-MULT(A,B)
1 if colunas[A] != linhas[B]
```

Programação Dinâmica: Multiplicação de cadeias de Matrizes

```
MATRIX-MULT(A,B)
1 if colunas[A] != linhas[B]
2 then error "dimensoes incompativeis"
3 else for i <- 1 to linhas[A]
4         for j<- 1 to linhas[B]
5          C[i,j] <- 0
6          for k <- 1 to colunas[A]
7            C[i,j] <- [i,j]+A[i,k]*B[k,j]
8 return C
```

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- Podemos multiplicar duas matrizes A e B somente se elas forem compatíveis, i.e., o número de colunas do A deve ser igual ao número de linhas do B.
- Se A é uma matriz $p \times q$ e B é uma matriz $q \times r$, a matriz resultante C tem dimensão $p \times r$
- O tempo para calcular C é determinado pelo número de multiplicações escalares na linha 7 do código acima, que é $p * q * r$.
- Agora vamos ver como realmente a maneira de colocação dos parênteses realmente afeta o tempo do cálculo do produto.

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- Considere o problema de uma cadeia $\langle A_1, A_2, A_3 \rangle$.
- Suponha que as dimensões das matrizes sejam $10 \times 100, 100 \times 5$ e 5×50, respectivamente.
- Se multiplicarmos as matrizes de acordo com a colocação de parênteses $((A_1 A_2)A_3)$, executaremos $10 * 100 * 5 = 5000$ multiplicações escalares para calcular o produto $A_1 A_2$ de dimensão 10×5.
- E mais outras $10 * 5 * 50 = 2500$ multiplicações para multiplicar essa matriz por A_3, produzindo um total de 7500 multiplicações.

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- Se ao invés disso de acordo com a colocação de parênteses $(A_1(A_2A_3))$, executamos $100 * 5 * 50 = 25000$ multiplicações escalares para calcular o produto de matrizes A_2A_3 de dimensão 100×50, e mais outras $10 * 100 * 50 = 50000$, dando um total de 75000 multiplicações escalares.

- Desse modo, o cálculo do produto de acordo com a primeira colocação dos parênteses é 10 vezes mais rápido.

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- **O problema da multiplicação de cadeias de matrizes** pode ser enunciado como segue: dada uma cadeia $\langle A_1, A_2, A_3, \ldots A_n \rangle$ de n matrizes, na qual, para $i = 1, 2, 3, \ldots n$, a matriz A_i tem dimensão $p_{i-1} p_i$, coloque completamente entres parênteses o produto $A_1 A_2 ... A_n$ de um modo que minimize o número de multiplicações escalares.
- Observe que nesse problema de multiplicação de cadeias de matrizes, não estamos realmente multiplicando matrizes.
- Nossa meta é apenas determinar uma ordem para multiplicar matrizes que tenha o custo mais baixo.
- Em geral, o tempo investido para determinar essa ordem ótima é mais que compensado pelo tempo economizado mais tarde.

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- Lembre-se de antes de resolver o problema de multiplicação de matrizes por programação dinâmica, você deve se convencer de que a verificação exaustiva de todas as possíveis colocações entre parênteses não resulta em um algoritmo eficiente.

- Um exercício interessante é calcular o número de alternativas possíveis para uma cadeia de tamanho n. (Dica: você deve lembrar como resolver recorrências).

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- **Etapa 1: a estrutura de uma colocação ótima de parênteses**.
- Lembre-se que a primeira etapa da programação dinâmica é encontrar a subestrutura ótima, e depois usá-la para construir uma solução ótima para o problema a partir de soluções ótimas para subproblemas.
- Vamos adotar a notação $A_{i..j}$, onde $i \leq j$ para a matriz que resulta da avaliação do produto $A_i A_{i+1} ... A_j$.
- Obserser que se o problema é não trivial, i.e., se $i < j$, qualquer colocação ótima dos parênteses do produto $A_i A_{i+1} ... A_j$ deve dividir o produto entre A_k e A_{k+1} para algum k no intervalo $i \leq k < j$.

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- Ou seja, para algum valor de k, primeiro calculamos as matrizes $A_{i..k}$ e $A_{k+1..j}$ e depois multiplicamos os dois para gerar o produto final $A_{i..j}$.
- O custo dessa colocação dos parênteses é portanto o custo de calcular a matriz $A_{i..k}$, mais o custo de calcular $A_{k+1..j}$, mais o custo de multiplicá-las uma pela outra.
- Por exemplo, para $i = 1$ e $j = 4$, temos as seguintes possibilidades $k = 1, 2, 3$:

$$(A_1(A_2(A_3A_4)))$$
$$(A_1((A_2A_3)A_4))$$
$$((A_1A_2)(A_3A_4))$$
$$((A_1(A_2A_3))A_4)$$
$$(((A_1A_2)A_3)A_4)$$

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- **Etapa 2: Uma solução Recursiva**.
- Definimos recursivamente o custo de uma solução ótima em termos das soluções ótimas para subproblemas.
- Seja $m[i, j]$ o número mínimo de multiplicações escalares necessárias para calcular a matriz $A_{i..j}$.
- Para o problema completo, o custo de um caminho mais econômico para calcular $A_{1..n}$ seria portanto $m[1, n]$.
- Podemos definir $m[i, j]$ recursivamente:

$$m[i, j] = m[i, k] + m[K + 1, j] + p_{i-1} * p_k * p_j$$

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- Essa equação recursiva pressupõe que conhecemos o valor de k, o que não ocorre.
- Porém, sabemos que $k = i, i+1, ..., j-1$. Como a colocação ótima deve usar um desses valores para k, precisamos apenas verificar todos eles para encontrar o melhor.
- Assim, temos:

$$m[i,j] = 0 \text{ se } i = j$$
$$m[i,j] = min_{i \leq k < j}[m[i,k] + m[k+1,j] + p_{i-1} * p_k * p_j] \text{ se } i < j$$

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- Os valores de $m[i, j]$ fornecem os custos de soluções ótimas para subproblemas.

- Para nos ajudar o modo de construir uma solução ótima, vamos definir $s[i, j]$ como um valor de k no qual podemos dividir o produto $A_i A_{i+1} ... A_j$ para obter uma solução ótima dos parênteses.

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- **Etapa 3: Como calcular os custos ótimos**.
- Nesse ponto é simples escrever um algoritmo recursivo baseado na recorrência acima para calcular o custo mínimo $m[1, n]$ para multiplicar $A_1 A_2 ... A_n$.
- Entretanto, esse algoritmo demora um tempo exponencial - ele não é nada melhor que o método de força bruta.
- Um algoritmo recursivo pode encontrar cada subproblema muitas vezes em diferentes ramificações dessa árvore de recursão.
- Essa propriedade de superpor subproblemas é a segunda indicação da aplicabilidade da programação dinâmica.
- Temos que executar a terceira etapa para calcular o custo ótimo usando uma abordagem tabular de baixo pra cima.

Programação Dinâmica: Multiplicação de cadeias de Matrizes

- O pseudocódigo a seguir pressupõe que a matriz A_i tem dimensões $p_{i-1} \times p_i$ para $i = 1, 2, ..., n$.
- A entrada é uma sequência $p = \langle p_0, p_1, ..., p_n \rangle$, onde $length[p] = n + 1$.
- O procedimento utiliza uma tabela auxiliar $m[1..n, 1..n]$ para armazenar os custos de $m[i, j]$ e uma tabela auxiliar $s[1..n, 1..n]$ que registra qual ínidice de k alcançou o custo ótimo no cálculo de $m[i, j]$.
- Usaremos a tabela s para construir uma solução ótima.

Programação Dinâmica: Multiplicação de cadeias de Matrizes