M342 Álgebra Computacional

Christian Lomp

FCUP

17 de Outubro de 2011

3 Álgebra linear

Determinante

Para calcular a determinante o mais rápido é usar o algoritmo de eliminação de Gauss para transformar uma matriz numa matriz em escada.

Johann Carl Friedrich Gauss (1777-1855)

3 Álgebra linear

Operações de linhas

Seja $A = (a_{ij})_{1 \leq i,j \leq n}$ cujas linhas denotamos por $L_1, \ldots, L_n$.

3 Álgebra linear

Operações de linhas

Seja $A = (a_{ij})_{1 \leq i,j \leq n}$ cujas linhas denotamos por $L_1, \ldots, L_n$.
Dado um escalar $\lambda \in K$ e $1 \leq i \neq j \leq n$ a operação de linha

$$L_i \leftarrow L_i - \lambda L_j$$

substitue a i-esmia linha $L_i = (a_{i1}, \ldots, a_{in})$ de A pela linha

$$L_i - \lambda L_j = (a_{i1} - \lambda a_{j1}, \ldots, a_{in} - \lambda a_{jn}).$$

3 Álgebra linear

Operações de linhas

Seja $A = (a_{ij})_{1 \leq i,j \leq n}$ cujas linhas denotamos por $L_1, \ldots, L_n$.
Dado um escalar $\lambda \in K$ e $1 \leq i \neq j \leq n$ a operação de linha

$$L_i \leftarrow L_i - \lambda L_j$$

substitue a i-esmia linha $L_i = (a_{i1}, \ldots, a_{in})$ de A pela linha

$$L_i - \lambda L_j = (a_{i1} - \lambda a_{j1}, \ldots, a_{in} - \lambda a_{jn}).$$

3 Álgebra linear

Operações de linhas

Se $E_{ij} = (\delta_{ij})_{1 \leq i,j \leq n}$ é a matriz que tem entrada 1 na componente (i, j) e 0 nos restantes componentes, então a multiplicação $E_{ij}A$ consiste de matriz cuja i-ésima linha é a j-ésima linha de A e cujas outras linhas são nulos.

3 Álgebra linear

Operações de linhas

Se $E_{ij} = (\delta_{ij})_{1 \leq i,j \leq n}$ é a matriz que tem entrada 1 na componente (i, j) e 0 nos restantes componentes, então a multiplicação $E_{ij}A$ consiste de matriz cuja i-ésima linha é a j-ésima linha de A e cujas outras linhas são nulos.
O resultado da operação da linha $L_i \leftarrow L_i - \lambda L_j$ na matriz A é

$$(Id_n - \lambda E_{ij})A.$$

3 Álgebra linear

Eliminação de Gauss

Se $E_{ij} = (\delta_{ij})_{1 \leq i,j \leq n}$ é a matriz que tem entrada 1 na componente (i, j) e 0 nos restantes componentes, então a multiplicação $E_{ij}A$ consiste de matriz cuja i-ésima linha é a j-ésima linha de A e cujas outras linhas são nulos.

3 Álgebra linear

Eliminação de Gauss

Se $E_{ij} = (\delta_{ij})_{1 \leq i,j \leq n}$ é a matriz que tem entrada 1 na componente (i, j) e 0 nos restantes componentes, então a multiplicação $E_{ij}A$ consiste de matriz cuja i-ésima linha é a j-ésima linha de A e cujas outras linhas são nulos.
O resultado da operação da linha $L_i \leftarrow L_i - \lambda L_j$ na matriz A é

$$(Id_n - \lambda E_{ij})A.$$

3 Álgebra linear

Input: matriz A
Output: matriz em escada A'

for $1 \leq j < n$ **do**
  if $\exists k : a_{kj} \neq 0$ **then**
    trocar L_k e L_j
    for $j < k \leq n$ **do**
      $L_k \rightarrow a_{kj} L_k - a_{jj} L_k$
    end for
  end if
end for

$$\sum_{j=1}^{n-1} (n-j) = \frac{n(n-1)}{2} \quad \text{iterações.}$$

3 Álgebra linear

```
MATRIZ MATRIZ::OpLinha(int i, int j, Tipo lambda, Tipo mu)
{
  vector<VETOR> output(data);
  for (int k=0; k<data.size(); k++)

    output[i][k] = mu*output[i][k] - lambda*output[j][k];

  return MATRIZ(output);
};
```

3 Álgebra linear

```
MATRIZ MATRIZ::Trocar(int i, int j)
{
  vector<VETOR> output(data);
  for (int k=0; k<data.size(); k++)
  {
    Tipo aux=output[i][k];
    output[i][k]=output[j][k];
    output[j][k]=aux;
  }
  return MATRIZ(output);
};
```

3 Álgebra linear

```
MATRIZ MATRIZ::TransformarEscadas()
{
  int n=data.size();
  MATRIZ output(data);
  for (int j=0; j<n; j++)
  {
    int k=j;
    while(k<n && output.data[k][j] == 0) k++;
    if (k<n)
    {
       output=output.trocar(j,k);
       for (k=j+1; k<n; k++)
         output=output.OpLinha(k,j,output.data[k][j],output.data[j][j]);
    }
  };
  return output;
};
```

3 Álgebra linear

```
Tipo MATRIZ::Determinante()
{
  MATRIZ escadas=TransformarEscadas();
  Tipo det=escadas.data[0][0];
  for(int i=1: i<escadas.data.size(); i++)
   det=det*escadas.data[i][i];
  return det;
};
```

3.3 Algoritmo de Strassen

Multiplicação rápida de matrizes.

Volker Strassen (1936-)

3.3 Algoritmo de Strassen

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in M_{2^n}(K) \qquad N = \begin{pmatrix} E & F \\ G & H \end{pmatrix} \in M_{2^n}(K)$$

$A, B, C, D, E, F, G, H \in M_{2^{n-1}}(K)$. Então

$$MN = \begin{pmatrix} R & S \\ T & U \end{pmatrix} \qquad \text{tal que} \qquad \begin{cases} R &= AE + BG \\ S &= AF + BH \\ T &= CE + DG \\ U &= CF + DH \end{cases}$$

8 Multiplicações 4 Adições

3.3 Algoritmo de Strassen

$$\begin{cases} M_1 &= (A+D)\cdot(D+H) \\ M_2 &= (C+D)\cdot E \\ M_3 &= A\cdot(F-H) \\ M_4 &= D\cdot(G-E) \\ M_5 &= (A+B)\cdot H \\ M_6 &= (C-A)\cdot(E+F) \\ M_7 &= (B-D)\cdot(G+H) \end{cases}$$

$$\begin{cases} R &= M_1 + M_4 - M_5 + M_7 \\ S &= M_3 + M_5 \\ T &= M_2 + M_4 \\ U &= M_1 - M_2 + M_3 + M_6 \end{cases}$$

7 Multiplicações 18 Adições

3.3 Algoritmo de Strassen

Caso 1 do "Master Theorem"

Dado uma função recursiva que divide os dados da entrada com tamanho n em b subconjuntos e que aplica recursivamente essa função a vezes nestes subconjuntos e que tem custos adicionais dado por uma função $f(n)$ satisfaz a formula de custos:

$$T(n) = aT(n/b) + f(n)$$

Se $f(n) \in O(n^{\log_b(a)-\epsilon})$ onde $\epsilon > 0$ então $T(n) \in \Theta(n^{\log_b(a)})$.

Notação O

$$f(x) \in O(g(x)) \Leftrightarrow \exists C, n_0 : \forall n > n_0 : f(n) \leq Cg(n).$$

$$f(x) \in \Theta(g(x)) \Leftrightarrow \exists C_1, C_2, n_0 : \forall n > n_0 : C_1 g(n) \leq f(n) \leq C_2 g(n).$$

3.3 Algoritmo de Strassen

Multiplicação de matrizes usual

Tamanho das matrizes $n \times n$:

$$T(n) = 8T(n/2) + 4n^2$$

Como $f(n) = 4n^2 \in O(n^{\log_2(8)-1})$ então $T(n) \in \Theta(n^{\log_2(8)}) = \Theta(n^3)$.

Multiplicação de matrizes usando o Algoritmo de Strassen

Tamanho das matrizes $n \times n$:

$$T(n) = 7T(n/2) + 18n^2$$

Como $f(n) = 18n^2 \in O(n^{\log_2(7)-0.8})$ então $T(n) \in \Theta(n^{2.807})$, onde $\log_2(7) \simeq 2.807$.

3.3 Algoritmo de Strassen

$n = 2^k$	Classico 8^k	Strassen 7^k	percentagem
2	8	7	$\sim 87\%$
4	64	49	$\sim 76\%$
8	512	343	$\sim 67\%$
16	4096	2401	$\sim 58\%$
32	32768	16807	$\sim 51\%$
64	262144	117649	$\sim 49\%$
128	2097152	823543	$\sim 39\%$
256	16777216	5764801	$\sim 34\%$
512	134217728	40353607	$\sim 30\%$
1024	1073741824	282475249	$\sim 26\%$