Algorithme de V. Strassen pour la multiplication rapide de matrices

Gérard Sookahet (Août 1997)

Le produit de matrices est une opération qui intervient souvent dans les calculs informatiques et numériques. C'est typiquement le genre d'opération assez lourd à mener. Ce n'est pas sa complexité qui est embarassante mais son aspect répétitif dès que la taille de la matrice devient importante(disons à partir de 6x6 pour un humain et 100000x100000 pour un ordinateur). Pour économiser une peu de temps de calcul Volker Strassen a imaginé l'algorithme que nous allons voir par la suite.

1) La méthode classique.

Soient A et B des matrices NxN. Le produit est une matrice C=AB de taille NxN:

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & \dots & \dots & c_{1N} \\ c_{21} & c_{22} & c_{23} & \dots & \dots & \vdots \\ c_{31} & c_{32} & c_{33} & \dots & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ c_{N1} & \dots & \dots & \dots & \dots & c_{NN} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots. & a_{1N} \\ a_{21} & a_{22} & \dots. & \vdots \\ \vdots & \vdots & \dots. & \vdots \\ \vdots & \vdots & \dots. & \vdots \\ \vdots & \vdots & \dots. & \vdots \\ a_{N1} & \dots. & \dots. & a_{NN} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \dots. & b_{1N} \\ b_{21} & b_{22} & \dots. & \vdots \\ \vdots & \vdots & \dots. & \vdots \\ \vdots & \vdots & \dots. & \vdots \\ \vdots & \vdots & \dots. & \vdots \\ b_{N1} & \dots. & \dots. & b_{NN} \end{pmatrix}$$

Je vous laisse le soin de compléter les pointillé ;-)

Les élément de la matrice C s'écrivent:

$$c_{ij} = \sum_{n=1}^{N} a_{in} b_{nj}$$

(i et j varient de 1 à N). Donc pour calculer le produit de deux matrices NxN cela revient à faire N^3 multiplications et $N^{2(N-1)}$ additions.

2) L'Algorithme de Volker Strassen.

En 1969, Volker Strassen [1] imagine un algorithme permettant de descendre le nombre de multiplications en dessous de N^3.
Pour se fixer les idées, prenons le produit de matrices 2x2:

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

Cela nécessite 8 multiplications et 4 additions. L'idée de Volker Strassen est de trouver un moyen de multiplier les matrices A et B en utilisant seulement 7 multiplications au lieu de 8 par des combinaisons d'opérations astucieuses entre les éléments de A et B.
Posons:

$$
\begin{aligned}
E_1 &= (a_{12} - a_{22})(b_{21} + b_{22}) \\
E_2 &= (a_{11} - a_{22})(b_{11} + b_{22}) \\
E_3 &= (a_{11} - a_{21})(b_{11} + b_{12}) \\
E_4 &= (a_{11} + a_{12})b_{22} \\
E_5 &= a_{11}(b_{12} - b_{22}) \\
E_6 &= a_{22}(b_{21} - b_{11}) \\
E_7 &= (a_{21} + a_{22})b_{11}
\end{aligned}
$$

Les éléments de C s'obtiennent grâce à:

$$
\begin{aligned}
c_{11} &= E_1 + E_2 - E_4 + E_6 \\
c_{12} &= E_4 + E_5 \\
c_{21} &= E_6 + E_7 \\
c_{22} &= E_2 - E_3 + E_5 - E_7
\end{aligned}
$$

Si vous faites le compte, cela fait bien 7 multiplications. L'économie d'une multiplication a toujours des conséquences. Dans notre cas, on se retrouve avec 18 additions (*diantre !*). A ce stade, on est en droit de se demander si l'on réalise réellement une économie. Nous allons voir par la suite que c'est le cas. Pour le moment, il faut juste savoir que la multiplication des matrices va être faite de manière *récursive*. Ce dernier terme a beaucoup d'importance.
Tenant compte des formules de l'algorithme de Volker Strassen, comment faire pour l'appliquer aux produits de matrices NxN?
Deux notions essentielles vont intervenir. La première est la multiplication de matrices par blocs [2]. En effet, il suffit de partitionner notre matrice NxN en 4 sous-matrices de N/2xN/2 chacune afin de pouvoir appliquer les formules vues plus haut de manière récursive.

$$
\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}
$$

Avec:

$$
C_{11} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1\frac{N}{2}} \\ c_{21} & c_{22} & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ c_{\frac{N}{2}1} & \cdots & \cdots & c_{\frac{N}{2}\frac{N}{2}} \end{pmatrix} \quad C_{12} = \begin{pmatrix} c_{1\frac{N}{2}+1} & \cdots & \cdots & c_{1N} \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ c_{\frac{N}{2}\frac{N}{2}+1} & \cdots & \cdots & c_{\frac{N}{2}N} \end{pmatrix}
$$

$$
C_{21} = \begin{pmatrix} c_{\frac{N}{2}+1,1} & \cdots & \cdots & c_{\frac{N}{2}+1,\frac{N}{2}} \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ c_{N1} & \cdots & \cdots & c_{N\frac{N}{2}} \end{pmatrix} \quad C_{22} = \begin{pmatrix} c_{\frac{N}{2}+1,\frac{N}{2}+1} & \cdots & \cdots & c_{\frac{N}{2}+1,N} \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ c_{N\frac{N}{2}+1} & \cdots & \cdots & c_{NN} \end{pmatrix}
$$

(les sous-matrices A_{ij} et B_{ij} se déduisent des C_{ij} *mutatis mutandis*).
Cependant, un problème se pose: pour utiliser le partitionnement des matrices à bon escient, et de manère récursive, il convient que N soit une puissance de 2 ($N=2^n$). C'est là qu'intervient la seconde notion: il suffit d'augmenter artificiellement la taille de la matrice en ajoutant les colonnes et les

lignes de 0 nécessaires afin d'arriver à la puissance de 2 la plus proche supérieurement. Cette puissance de 2 est quelquechose comme:

$$n = INT(\frac{\ln N}{\ln 2}) + 1$$

(INT() désigne la fonction entière).
Ainsi, si nous sommes partis de matrices de taille $N=2^n$, les sous-matrices A_{ij}, B_{ij} et C_{ij} seront de taille 2^{n-1}. Quand N est très grand pour ne pas dire très très grand, l'économie d'une multiplication est salutaire.
Si l'on implémente cet algorithme en utilisant une fonction récursive, cela nous donne en pseudo-code quelque chose comme:

```
Fonction Strassen(A,B: matrices; N: entier): matrices;

Si N n'est pas une puissance de 2
  Alors on ajoute des lignes et des colonnes de 0 afin d'acceder
        a la puissance de 2 la plus proche superieurement.
Si N=1 Alors Strassen=A*B;
Sinon {
       On partitionne A et B en blocs de taille N/2;

       E1=Strassen(A12 - A22,B21 +  B22,N/2);
       E2=Strassen(A11 + A22,B11 +  B22,N/2);
       E3=Strassen(A11 - A21,B11 +  B12,N/2);
       E4=Strassen(A11 + A12,B22,N/2);
       E5=Strassen(A11,B12 - B22,N/2);
       E6=Strassen(A22,B21 - B11,N/2);
       E7=Strassen(A21 + A22,B11,N/2);

       C11=E1 + E2 - E4 + E6;
       C12=E4 + E5;
       C21=E6 + E7;
       C22=E2 - E3 + E5 - E7;
      }
(* Fin de la fonction Strassen *)
```

Finalement, la méthode de V. Strassen a un coût en calcul de l'ordre de $O(N^{2.81})$ ou plus précisément $O(N^{\ln 7/\ln 2})$. Par rapport à $O(N^3)$ c'est un gain significatif du moins en algorithmique.

3) Peut-on mieux faire?

Tout porte à croire que oui. Cependant il faut modérer notre enthousiasme. En effet, il paraît plus que difficile de passer de $O(N^{2.81})$ à $O(N^2)$.
En 1970, S. Winograd a apporté une légère amélioration [3] à l'algorithme de V. Strassen en réduisant le nombre d'addition à 15. En utilisant A,B et C comme précédemment, cela nous donne:

$$
\begin{array}{lcllcllcl}
Q_1 & = & A_{21} - A_{11} & P_1 & = & A_{21}B_{11} & Q_9 & = & P_1 + P_7 \\
Q_2 & = & A_{11} + A_{12} & P_2 & = & A_{22}B_{21} & Q_{10} & = & Q_9 + P_3 \\
Q_3 & = & A_{12} - Q_1 & P_3 & = & Q_1 Q_5 & Q_{11} & = & P_4 + P_5 \\
Q_4 & = & A_{22} - Q_3 & P_4 & = & Q_2 Q_6 & & & \\
Q_5 & = & B_{22} - B_{12} & P_5 & = & Q_4 B_{22} & & & \\
Q_6 & = & B_{12} - B_{11} & P_6 & = & A_{12} Q_8 & & & \\
Q_7 & = & B_{11} + Q_5 & P_7 & = & Q_3 Q_7 & & & \\
Q_8 & = & B_{21} - Q_7 & & & & & &
\end{array}
$$

$$
\begin{array}{lcl}
C_{11} & = & Q_{10} + P_6 \\
C_{12} & = & Q_{10} + P_4 \\
C_{21} & = & P_1 + P_2 \\
C_{22} & = & Q_9 + Q_{11}
\end{array}
$$

Donc 7 multiplications et 15 additions.

Ce serait très intéressant de passer à 6 multiplications. Malheureusement, S. Winograd [3] et Hopcroft et Kerr [4] en 1971, ont achevé de montrer que l'on ne pouvait pas multiplier des matrices 2x2 avec moins de 7 multiplications. Dans le même registre technique, certains spécialistes explorent des pistes telles que l'amélioration des produits de matrices 3x3, 5x5, etc .... A titre indicatif on parvient à multiplier des matrices 3x3 avec 22 mutiplications.
Dernière chose, en 1980, S. Winograd et D. Coppersmith ont réussi à concevoir une méthode [5] dont le coût est de l'ordre de $O(N^{2.5})$. En 1987, ils parvinrent à descendre jusqu'à $O(N^{2.376})$ [6]. C'est à l'heure actuelle l'algorithme le plus rapide que l'on connaisse.

Voilà c'est tout pour cette fois.

4) Bibliographie.

- [1] V. Strassen, *Gaussian elimination is not optimal*, Numerische Math., 13(1969), pp354-356.
- [2] E. Lehman, *Mathématique pour l'étudiant de première année*, Tome 1, Edition Belin, pp168-169.
- [3] S. Winograd, *On the multiplication of 2x2 matrices*, IBM Research Report, RC267, January 1970.
- [4] J.E. Hopcroft, L.R. Kerr, *On minimizing the number of multiplications necessary for matrix multiplication*, SIAM Journal Appl. Math., 20:1(1971), pp30-36.
- [5] D. Coppersmith, S. Winograd, *On the asymptotic complexity of matrix multiplication*, SIAM Journal Comp., 11(1980), pp472-492.
- [6] D. Coppersmith, S. Winograd, *Matrix multiplication via arithmetic progressions*, 19th Annual ACM Symposium on Theory of Computing, 1-6, 1987.
- Bibliographie complémentaire:
  Victor Pan, *How to multiply matrices faster*, Lectures Notes in Computer Science No179, Edition Springer-Verlag, 1984.

Gérard Sookahet (Août 1997)