ミニチュア 11

行列のかけ算を検算する

二つの $n \times n$ 行列のかけ算は、とても大事な演算である。素朴に計算すると約 n^3 回の演算が必要だが、ミニチュア 10 でも述べたように漸近的にはずっと速い巧妙なアルゴリズムが見つかっている。現在の記録は $O(n^{2.376})$ のアルゴリズムだ。しかしながら係数の定数が天文学的に大きいので、このアルゴリズムの面白さは理論面に限られる。実際、この方式が素朴に計算するより速かったとしたら、そこで使われる行列は、現在あるいは将来のどんなコンピュータにも収まらない[1]だろう。

しかし進歩は止められないから、そのうちソフトウェア会社は行列の達人といったプログラムを売り出すかもしれない。想像するに、それを使うと行列のかけ算が本当に速くできるのだ。計算を間違っては大変だから、行列の達人に単純な検算プログラムを付け加えたい。これで結果の行列 C が入力した行列 A と B の積に本当になっているのか、毎回チェックするのだ。

もちろん、検算プログラムが実際に A と B をかけ算してその結果を C と比べるというのでは意味がない。行列の達人と同じくらい高速に行列のかけ算をする方法が我々にはわからないからだ。しかし検算を間違う可能性をほんの少しだけ許せば、行列の積の検算は、とても単純に、かつ高速に行える。

[1] (訳注) どんなコンピュータにも入りきれないほど大きな行列になってしまうということ。

話を具体的にするため、行列の成分は有理数としよう。もっともどんな体でも以下の議論はそのまま成り立つ。検算プログラムは入力として $n \times n$ 行列 A, B, C を受け取る。乱数を生成して各成分が 0 と 1 からなる n 次元ベクトル $\mathbf{x}$ をつくる。もっと正確にいうと、$\{0,1\}^n$ から各ベクトルを等確率 2^{-n} で選ぶ。このアルゴリズムは積 $C\mathbf{x}$ を ($O(n^2)$ 回の演算で) 計算し、次に $AB\mathbf{x}$ を (再び $O(n^2)$ 回の演算で) 計算する。(後者の正しい括弧付けはもちろん $A(B\mathbf{x})$ である。) 両者の結果があえば、アルゴリズムは YES を、そうでなければ NO を返す。

もし $C = AB$ ならば、アルゴリズムはいつでも YES を返し、これは正しい。しかし $C \neq AB$ ならば、YES を返すことも NO を返すこともありうる。後者で YES を返すのは間違いだが、これから示すのは、これが起きる確率は高々 $\frac{1}{2}$ であること、したがって、このアルゴリズムは行列の積の誤りを少なくとも確率 $\frac{1}{2}$ で検出できることだ。

そこで $D := C - AB$ とおく。もし D が $n \times n$ の非零行列で、$\mathbf{x} \in \{0,1\}^n$ がランダムベクトルならば、ベクトル $\mathbf{y} := D\mathbf{x}$ が零ベクトルになる確率は高々 $\frac{1}{2}$ であることを示せばよい。

$D \neq 0$ と仮定し[2]、$d_{k\ell} \neq 0$ となる k と ℓ を固定する[3]。このとき $y_k = 0$ となる確率は高々 $\frac{1}{2}$ であることを示そう。

ここで

$$y_k = d_{k1}x_1 + d_{k2}x_2 + \cdots + d_{kn}x_n = d_{k\ell}x_\ell + S$$

である。ただし

$$S = \sum_{\substack{j=1,2,\ldots,n \\ j \neq \ell}} d_{kj}x_j$$

とおいた。

$\mathbf{x}$ の n 個の成分を、連続 n 回のコイン投げで決めると考えてみよう。x_ℓ の値は最後のコイン投げで決めても構わない (というのは、コイン投げの結果は独立だから)。最後のコイン投げの前に、すでに S は決まっている。なぜなら、それは x_ℓ には依存しないから。最後のコイン投げのあとで、S をそのまま変えない ($x_\ell = 0$ のとき) か、S に非零の $d_{k\ell}$ を加える ($x_\ell = 1$ のとき)。この二つの場合の少なくとも片方では零でない数が得られるから、$D\mathbf{x} \neq \mathbf{0}$ となる確率は少なくとも $\frac{1}{2}$ であり、これが示したいことだった。

---

[2](訳注) 右辺の 0 は零行列。

[3](訳注) $d_{k\ell}$ は D の (k,ℓ) 成分。記法の項を見よ。

ここで述べた検算のアルゴリズムは速いが、あまり信頼できない。計算間違いの検出に失敗する確率は $\frac{1}{2}$ の高さだ。しかし、もしこの検算を繰り返し行ったら、例えば一組の入力 A, B, C に対して 50 回検算を行うと、計算間違いの検出に失敗する確率は高々 $2^{-50} < 10^{-15}$ となり、この確率は実用的見地からは完全に無視できるほど小さい。

注意. 計算の確率的な検証という考え方について、ここでは単純な形で提示してみたが、これはとても有効なものだとわかった。計算量理論のいわゆる PCP 定理[4]によれば、効率的に解けるどんな計算問題についても確率的な検証が高速にできる。原理的には、遅いパソコンでも最強のスーパーコンピュータの計算を検算できる。さらにこれらの結果と近似アルゴリズムの間の驚くべき関係が見つかってきている。

情報源 R. Freivalds, *Probabilistic machines can use less running time*, in Information Processing 77, IFIP Congr. Ser. **7**, North-Holland, Amsterdam, 1977, 839–842.

PCP および計算量に関する手始めとして、例えば次の本がある。

O. Goldreich, *Computational complexity: A conceptual perspective*, Cambridge University Press, Cambridge, 2008.

[4](訳注) PCP は Probabilistically checkable proofs (確率的検証可能証明) の略。