The permanent and the determinant

Uri Feige

December 30, 2009

1 Introduction

Given an order n matrix A, its *permanent* is

$$per(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

where σ ranges over all permutations on n elements. Recall that the determinant of a matrix is

$$det(A) = \sum_{\sigma} (-1)^{\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

where $(-1)^{\sigma}$ is $+1$ for even permutations and -1 for odd permutations. (For those more familiar with the inductive definition of the determinant, obtained by developing the determinant by the first row of the matrix, observe that the inductive definition if spelled out leads exactly to the formula above. The same inductive definition applies to the permanent, but without the alternating sign rule.)

The determinant can be computed in polynomial time by gaussian elimination, and in time n^{ω} by fast matrix multiplication. On the other hand, there is no polynomial time algorithm known for computing the permanent. In fact, Valiant showed that the permanent is complete for the complexity class $\#P$, which makes computing it as difficult as computing the number of solutions of NP-complete problems (such as SAT, Valiant's reduction was from Hamiltonicity).

For 0/1 matrices, the matrix A can be thought of as the adjacency matrix of a bipartite graph (technically, A is an of-diagonal block of the adjacency matrix), and then the permanent counts the number of perfect matchings. Computing the permanent of integer matrices can be reduced in polynomial time to that of computing the permanent of 0/1 matrices.

The permanent and the determinant can both be viewed as multilinear polynomials, when the entries of the respective matrices are variables. Hence computing the permanent or determinant can be viewed as evaluating a multinomial whose coefficients are given by the respective matrix. Valiant showed how the evaluation of any explicit multinomial (one that may have exponentially many terms, though there is a polynomial time algorithm that given a term computes its coefficient) can be reduced in polynomial time to computing the permanent of some matrix.

2 Computing the permanent

The natural algorithm takes time roughly $n!$. Ryser showed a quicker way to compute the permanent, using the exclusion-inclusion formula.

For a nonempty set S of columns, let $R_i(S)$ denote the sum of items in columns S of row i. Then:

$$per(A) = \sum_S (-1)^{n-|S|} \prod_{i=1}^{n} R_i(S)$$

Computing the permanent using the above formula takes time roughly 2^n.

To see that Ryser's formula is correct, consider the representation of the permanent as a multinomial, and observe that Ryser's formula gives the correct coefficient for every monomial. For monomials that correspond to permutations, this coefficient is 1. For monomials that involve variables from c columns,the number of times that they appear in Ryser's formula is

$$\sum_{j=0}^{n-c} (-1)^{n-c+j} \binom{n-c}{j} = 0$$

(Equality with 0 can be seen by the equivalence with having $n-c$ coin tosses, requiring that an even number of them come up heads. It all depends on the last coin toss.)

3 Relations with matchings

To see whether a bipartite graph has an even or odd number of perfect matching, compute the permanent modulo 2. This is equivalent to computing the determinant modulo 2, and hence can be done in time $O(n^\omega)$.

To see whether the number of perfect matching is divisible by 3, compute the permanent modulo 3. However, this problem is difficult (unless P=NP).

To see whether a bipartite graph has a perfect matching, Lovasz suggests the following randomized algorithm that works in time roughly n^ω. Observe that there is a perfect matching iff the determinant, with formal variables replacing the 1 entries in the matrix, is not the 0-multinomial. Let $p > n^2$ be prime, replace the 1 entries by random entries in $\{0, \ldots p-1\}$ and compute the determinant modulo p. If there is no perfect matching, the answer is 0. If there is a perfect matching, the answer is nonzero with probability at least $(p-n^2)/p$. This last fact follows from the Schwartz-Zippel lemma.

Lemma 1 *Let p be prime, and assume that all computations are performed in Z_p (namely, all coefficients are in $\{0, \ldots p-1\}$ and all computations are done modulo p). Let $P(x_1, \ldots, x_k)$ be a multilinear polynomial over k variables that is not identically 0. Let x be a random assignment to all variables, where the the value of each variable is chosen independently at random from $\{0, \ldots p-1\}$. Then the probability that $P(x) \neq 0$ is at least $(1-1/p)^k \geq (p-k)/p$.*

Proof: The proof is by induction on k. For the base case $k = 1$, $P(x)$ is a linear polynomial $ax + b$. If $a = 0$ then $b \neq 0$, and the polynomial is never 0. If $a \neq 0$, then $P(x) = 0$ only for $x = -b/a$ modulo p, and there is a unique such x in $\{0, \ldots p-1\}$.

For the inductive step, assume the lemma for k and prove for $k+1$. Write $A(x_1, \ldots x_{k+1}) = x_{k+1}A_1(x_1, \ldots x_k) + A_2(x_1, \ldots x_k)$. Pick random values for $x_1, \ldots, x_k$ first. By induction, with probability at least $(1-p)^k$, $A_1(x_1, \ldots x_k) \neq 0$. If so, then when picking x_{k+1} at random, with probability $(p-1)/p$, $x_{k+1} \neq A_2(x_1, \ldots x_k)/A_1(x_1, \ldots x_k)$. Hence $P(x_1, \ldots, x_{k+1}) \neq 0$ with probability at least $(1-1/p)^{k+1}$. $\square$

The approach described above can be extended to testing whether a non-bipartite graph has a perfect matching, and also to actually find a perfect matching, essentially in time n^ω. See [1] for more information.

4 Counting spanning trees

As we have seen, counting matchings is NP-hard. Here we sketch the famous tree-matrix theorem of Kirchoff, that shows how to efficiently count spanning trees. A clear exposition with more details can be found at [2]. The same exposition also explains Kasteleyn's polynomial time algorithm for counting the number of perfect matchings in a planar graph, though this algorithm will not be shown in class.

Direct edges arbitrarily. Consider the incidence matrix M of the directed graph with vertices as rows, edges as columns, and $+1$ and -1 entries for incoming and outgoing edges.

Restrict attentions to $n-1$ columns (edges). The rank of this submatrix is precisely n minus the number of connected components in this subgraph. To see this, transpose this matrix, and see that every eigenvector must be constant on every connected component. Hence the rank is $n-1$ iff these edges form a spanning tree. Remove one row from this matrix to get a matrix B. The rank does not change, because all rows sum up to 0, and hence the removed row was spanned by the other rows. Hence the matrix has full rank and nonzero determinant iff the edges form a spanning tree. Being a matrix with at most one $+1$ and at most one -1 entry in each column, the matrix is totally unimodular, and hence its determinant is ± 1. It follows that $det(BB^T) = det(B)det(B^T) = 1$ if the columns form a spanning tree, and 0 otherwise.

Hence removing one row from from M and having S range over all blocks of $n-1$ columns, we have proved that the number of spanning trees is exactly $\sum_B det(BB^T)$.

Observe that MM^T is exactly the Laplacian of the graph (degrees along the diagonal, -1 in entry $L_{i,j}$ if there is an edge (i,j).

Now we can use the Binet-Cauchy expansion of the determinant. For r by n matrices A and B (we shall take A to be M without one row and $B = A$, and hence in our application $r = n-1$ and $m = n$) we have:

$$det(AB^T) = \sum_S det(A_S)det(B_S)$$

where A_S is the block of S columns from A, with $|S| = r$.

This shows that removing an arbitrary row and same column from the Laplacian, the determinant counts the number of spanning trees.

We now prove the Binet-Cauchy formula. Note that we may assume that $n > r$. For $n = r$ the Binet-Cauchy formula is the known equality $det(AB^T) = det(A)det(B)$, and for $n < r$ we have that the rank of AB^T is at most $n < r$, and hence $det(AB^T) = 0$. (The

right hand side of the Binet-Cauchy formula is undefined in this case.) Let Δ be an order n diagonal matrix with formal variables x_i along the diagonal. We show that

$$det(A\Delta B^T) = \sum_S det(A_S)det(B_S) \prod_{i \in S} x_i$$

are identical as formal polynomials, and the Binet-Cauchy formula follows by setting $x_i = 1$ for all i.

Observe that $A\Delta B^T$ is an order r matrix with entries that are linear forms in the x_i variables. Hence $det(A\Delta B^T)$ is a homogeneous polynomial of degree r. For monomials with fewer than r distinct variables, their coefficient must be 0. This can be seen by substituting arbitrary values in these variables and 0 in the rest. The rank of $A\Delta B^T$ in this case becomes smaller than r, and hence the polynomial restricted only to these variables is the 0 polynomial. The coefficients of other monomials (say, defined over a set S of r variables) can be determined by substituting 1 for the S variables and 0 for other variables. Then $A\Delta B^T = A_S(B_S)^T$ and the coefficient of $\prod_{i \in S} x_i$ is indeed $det(A_S)det(B_S)$.

References

[1] Nicholas J. A. Harvey. Algebraic Algorithms for Matching and Matroid Problems. SIAM Journal on Computing, 39(2):679-702, 2009.

[2] Mark Jerrum. Chapter 1 in the lecture notes in http://homepages.inf.ed.ac.uk/mrj/pubs.html.