Lecture 16: Permanent and Interactive Proofs

Anup Rao

November 29, 2018

Determinant vs Permanent

The determinant is very similar to another polynomial, the permanent:

$$\mathsf{perm}(M) = \sum_{\pi \in S_n} \prod_{i=1}^{n} M_{i\pi(i)}.$$

It's the same polynomial as the determinant, except that the coefficients are all 1. Surprisingly, the permanent seems much harder to compute. Indeed, there is a good reason for this. One can reduce 3SAT to computing the permanent!

Let us define the complexity class **#P** as follows. We say that a function f is in **#P** if and only if there is a polynomial time turing machine M and a polynomial p such that $f(x) = |\{y \in \{0,1\}^{p(|x|)} : M(x,y) = 1\}|$. Thus if one thinks of M as the verifier to an **NP** problem, f counts the number of witnesses to x. Examples of problems in **#P** include #SAT, where $\#\mathsf{SAT}(\phi)$ is the number of satisfying assignments to the boolean formulat ϕ.

We shall not prove the following theorem, but it shows that an efficient algorithm for the permanent would prove that $\mathbf{P} = \mathbf{NP}$.

Theorem 1. *Every f in* **#P** *can be reduced to* $\#\mathsf{SAT}(\phi)$ *in polynomial time. Every f in* **#P** *can be reduced to* perm *in polynomial time.*

A randomized algorithm for estimating the permanent

Given the matrix M, let us define the matrix A as follows:

$$A_{ij} = \begin{cases} -\sqrt{M_{ij}} & \text{with probability } 1/2, \\ \sqrt{M_{ij}} & \text{with probability } 1/2. \end{cases}$$

All entries are sampled independently. (Note that A_{ij} may be a complex number).

The algorithm is to just output $\det(A)^2$.

Claim 2. $\mathbb{E}\left[\det(A)^2\right] = \mathsf{perm}(M)$.

Proof We have

$$\begin{aligned} \mathbb{E}\left[\det(A)^2\right] &= \mathbb{E}\left[\left(\sum_{\sigma \in S_n} \mathsf{sign}(\sigma) \prod_{i=1}^{n} A_{i\sigma(i)}\right)^2\right] \\ &= \mathbb{E}\left[\sum_{\sigma,\sigma' \in S_n} \mathsf{sign}(\sigma)\cdot\mathsf{sign}(\sigma') \prod_{i=1}^{n} A_{i\sigma(i)} A_{i\sigma'(i)}\right] \end{aligned}$$

Now we separate out the terms where $\sigma = \sigma'$ to get

$$\begin{aligned}
\mathbb{E}\left[\det(A)^2\right] &= \mathbb{E}\left[\sum_{\sigma\in S_n} \mathsf{sign}(\sigma)^2 \prod_{i=1}^{n} A_{i\sigma(i)}^2 + \sum_{\sigma\neq\sigma'\in S_n} \mathsf{sign}(\sigma)\cdot\mathsf{sign}(\sigma') \prod_{i=1}^{n} A_{i\sigma(i)}A_{i\sigma'(i)}\right] \\
&= \mathbb{E}\left[\sum_{\sigma\in S_n} \prod_{i=1}^{n} M_{i\sigma(i)} + \sum_{\sigma\neq\sigma'\in S_n} \mathsf{sign}(\sigma)\cdot\mathsf{sign}(\sigma') \prod_{i=1}^{n} A_{i\sigma(i)}A_{i\sigma'(i)}\right] \\
&= \mathsf{perm}(M) + \mathbb{E}\left[\sum_{\sigma\neq\sigma'\in S_n} \mathsf{sign}(\sigma)\cdot\mathsf{sign}(\sigma') \prod_{i=1}^{n} A_{i\sigma(i)}A_{i\sigma'(i)}\right] \\
&= \mathsf{perm}(M) + \sum_{\sigma\neq\sigma'\in S_n} \mathsf{sign}(\sigma)\cdot\mathsf{sign}(\sigma')\,\mathbb{E}\left[\prod_{i=1}^{n} A_{i\sigma(i)}A_{i\sigma'(i)}\right]
\end{aligned}$$

Whenever $\sigma \neq \sigma'$, we have that there is some i for which $\sigma(i) \neq \sigma'(i)$, and so the variable $A_{i\sigma(i)}$ is independent of all other variables in $\mathbb{E}\left[\prod_{i=1}^{n} A_{i\sigma(i)}A_{i\sigma'(i)}\right]$. Thus we get that $\mathbb{E}\left[\prod_{i=1}^{n} A_{i\sigma(i)}A_{i\sigma'(i)}\right] = \mathbb{E}\left[A_{i\sigma(i)}\right]\cdot\mathbb{E}\left[A_{i\sigma'(i)}\right]\cdot\mathbb{E}\left[\prod_{j\neq i} A_{j\sigma(j)}A_{j\sigma'(j)}\right] = 0$, as required. ■

Interactive proofs

One way to define **NP** is via the idea of a proof system. **NP** is the set of functions f for which there is a polynomial time verifier algorithm V such that given any x with $f(x) = 1$, there exists a prover P that can prove to the verifier that $f(x) = 1$ by providing a polynomial sized witness w for which $V(x, w) = 1$, yet if $f(x) = 0$, no such prover exists.

What happens if we allow the verifier to have a longer *interactive* conversation? Presumably, giving the verifier the ability to adaptively ask the prover questions based on his previous responses should give the verifier more power, and so allow the verifier to verify the correctness of the value for a larger set of functions. In fact, this does *not* give the verifier additional power: for if there is such an interactive verifier V^I for verifying that $f(x) = 1$, we can design a non-interactive verifier that does the same job. The new verifier will demand that the prover provide the entire transcript of interactions between V^I and a convincing prover. The new verifier can then verify that the transcript is correct, and would have convinced V^I. Thus, if f has an interactive verifier, then $f \in$ **NP**.

The story is more interesting if we allow the verifier to be randomized. We say that $f \in$ **IP** if there is a polynomial time randomized verifier V such that

Completeness For all x, if $f(x) = 1$, there is an oracle P such that $\Pr_r[V^P(x,r) = 1] \geq 2/3$.

Soundness For all x, if $f(x) = 0$, for every oracle P, $\Pr_r[V^P(x,r) = 1] \leq 1/3$.

Since any prover can be simulated in polynomial space, if $f \in \mathbf{IP}$, then $f \in \mathbf{PSPACE}$. The algorithm for f can just try all possible sequences of messages from the prover until it finds a sequence of messages that convinces the verifier, if such a sequence exists.

Theorem 3. $\mathbf{IP} \subseteq \mathbf{PSPACE}$.

It is easy to check that allowing the prover to be randomized does not change the model.

We shall eventually prove that $\mathbf{IP} = \mathbf{PSPACE}$ (and so **IP** is potentially much more powerful than **NP**).

Example: Graph non-Isomorphism

Two graphs on n vertices are said to be *isomorphic* if the vertices of one of the graphs can be permuted to make the two equal.

Consider the problem of testing whether two graphs are *not* isomorphic: the boolean function f such that $f(G_1, G_2)$ is 1 if and only if G_1 is not isomorphic to G_2. $f \in co\mathbf{NP}$, since the prover can just send the verifier the permutation that proves that they are isomorphic. We do not know if $f \in \mathbf{NP}$, but it is easy to prove that $f \in \mathbf{IP}$.

Here is the simple interactive protocol:

1. The verifier picks a random $i \in \{1, 2\}$.
2. The verifier randomly permutes the vertices of G_i and sends the resulting graph to the prover.
3. The prover responds with $b \in \{1, 2\}$.
4. The verifier accepts if $i = b$.

If G_1, G_2 are not isomorphic, then any permutation of G_i determines i, so the prover can determine i and send it back. However, if G_1, G_2 are isomorphic, then the graph that the prover receives has the same distribution whether $i = 1$ or $i = 2$, thus the prover can guess the value of i with probability at most $1/2$. Repeating the protocol several times, the verifier can make the probability of being duped by a lying prover exponentially small.