Robots take over the world

Chris Calabro

June 1, 2007

1 Introduction

It is often said that if P = NP then we could many solve many important problems efficiently, such as automatically finding bugs in circuits. Of course this is not automatic since it may be that the fastest we can solve SAT is n^{100} time.

This essay, inspired by Russell Impagliazzo's "robots take over the world" lectures, which in turn were inspired by discussions with Steven Rudich, will explore other more ridiculous things that can be done if P = NP and we somewhat implausibly take tractable to mean "solvable in polynomial time". This 2nd assumption could also be replaced by "and SAT can be solved *very* efficiently."

If P = NP then we claim that computers can learn to do the mental jobs that humans currently do, at least as well as we do them, and without us having to spend any effort programming them to do each particular task, except possibly initially programming the very first universal problem solver. The only real cost would be to train the computer by exposing it to lots of (tagged) examples of what humans like and don't like.

E.g. if we want computers to automatically generate great songs, we could expose a machine to lots of songs and say which ones we like and don't, and then it could automatically generate novel songs that we would like. This is not impressive yet: just take a good song and change one bit of its sound file - the result is a novel song that is also probably pretty good. But we could demand of the machine that the new song be uniformly random from the space of all songs that we'd like. This would make it seem "creative", and more so than any human is likely to be, given that nearly all human works are extremely derivative.

This last condition is what really makes the following into a "if pigs can whistle, then horses can fly" kind of result: P = NP may seem a little absurd, but automatic efficient generation of maximally creative and pleasing intellectual work should sound much harder to believe.

Among the things that would be useful to be done automatically (and could be if SAT could be solved efficiently) are mathematical theorem proving, poetry, hollywood movie generation, singing, court trial evaluation, optimizing computer hardware and software, matchmaking, conference paper generation - basically anything that currently seems to require human thought. It should also be noted that we don't need to specify so precisely what the problem domain is to the machine. We could give it a fairly vague goal, such as, "do something that pleases us."

2 Outline

Assume that the human mind uses some circuit J to judge whether a particular input is "good", e.g. a song may seem to us good or not, and that J has fanin $O(1)$, s gates, n input bits, and a single output bit.

A *mistake-bounded learner* is a randomized algorithm A s.t. for any J, at step t, J takes as input (x_i, j_i, x'_i, j'_i) for $i = 1, \ldots, t-1$, such that $J(x_i) = j_i$ and $J(x'_i) = j'_i$, and outputs some $x_t \in X_t = \{0,1\}^n - \{x_1, \ldots, x_{t-1}, x'_1, \ldots, x'_{t-1}\}$, provided this set is $\neq \emptyset$. The number of mistakes is

$$M = |\{t \mid x'_t \in X_t \wedge J(x_t) < J(x'_t)\}|.$$

We will show how to construct a poly-time mistake-bounded learner that makes fewer than $O(s \lg s)$ mistakes.

Notice that once the entire set $J^{-1}(1)$ is used up, implicitly the learner is no longer required to perform, since there can be no witness x'_t to future mistakes. We also do not require x'_t to be in X_t in general, only when counting mistakes. This allows us to point out the flaws with the machine when it is convenient to us, we need not provide new training data at each step.

Notice that it *is* essential that mistakes are pointed out to the machine in order for it to learn quickly. If we got rid of the x'_t entirely, then the learner would be trying to satisfy a black-box circuit which could have just one solution, in which case the learner would output a satisfying solution in expected time no less than 2^{n-1}.

Supposing that the human mind uses several billion neurons to judge something, s will be quite large, but some things are judged very quickly by people, such as visual information, and so probably use a much smaller value of s.

Also many generalizations are possible, such as requiring that the hamming distance between successive outputs be large, or that J has $O(1)$ outputs, but we will try to keep the presentation simple.

We need 2 supporting theorems: 1st we show how to efficiently sample witnesses to NP relations uniformly at random, then we show that BPP $\subseteq \Sigma_2$, which will allow us to derandomize algorithms. Both are useful in their own right.

3 Uniform sampling of NP witnesses

The following proof is adapted from Bellare, et al, [BGP00].

Let $R \in \mathrm{P}$ be an NP relation, so that

$$\exists p \in \text{poly } \forall x, y \in \Sigma^* \ (R(x,y) \rightarrow |y| \leq p(|x|)),$$

and let $L = \operatorname{dom} R$ be its language. We show how to sample from the set of witnesses $R(x) = \{y \mid R(x,y)\}$ of x uniformly at random and in expected poly-time given an NP oracle. From now on, fix $x \in L$ and let $S = R(x)$. Wlog assume $\forall y \in S\ |y| = n$.

Let $h : 2^n \rightarrow 2^m$ be a random k-wise independent hash function so that $\forall x_1, \ldots, x_k, y_1, \ldots, y_k$ with the x_i distinct,

$$Pr(\forall i \in [k]\ h(x_i) = y_i) = 2^{-mk}.$$

Such functions can be polynomially represented, sampled, and evaluated, e.g. by taking h to be the low-order m bits of the evaluation of a random degree k polynomial over $\mathrm{GF}(2^{\max\{m,n\}})$.

We can think of the elements of the codomain of h as *buckets*. The weight of a bucket is the number of elements of $\operatorname{dom} h$ that map to it under h. If each bucket is only polynomially heavy, say each has weight $\leq w$, then we can pick a bucket b at random, and use the NP oracle to enumerate the elements of b. We then pick a number i between 1 and w and output the ith element of b, or reject and try again (with the same h but new b and i) if $i >$ the number of elements of b.

We can think of this procedure as artificially filling the unfull buckets with nonwitnesses until each has weight exactly w. Then the probability that a particular witness y is output on a given round, given that a witness is output in that round, is the probability that y's bucket is chosen times the probability that y is chosen within the bucket. But this is exactly 1 over the number of witnesses. If the number of nonwitnesses needed to completely fill each bucket summed over all buckets is only a constant factor more than $|S|$, then only $O(1)$ rounds are required to output a witness with $\Omega(1)$ probability.

We can decide whether a given h actually makes each bucket have weight $\leq w$ by asking the NP oracle whether there is a bucket with $> w$ witnesses. It remains to be shown that w.h.p. a random h will have the property that each bucket has weight $\leq w$.

To do this requires a limited-independence tail inequality. Let X be a sum N of Bernoulli trials. If they are fully independent, then we can bound the tail of the distribution of X by a Chernoff bound, which is exponentially small in N. If the Bernoulli trials are only pair-wise independent, then we can use Chebyshev's inequality, which gives only a polynomial bound. So it should not be too surprising that with k-wise independence we get something in between.

In particular, [BGP00] show that if the trials are k-wise independent and $k \geq 4$ is an even integer, $\mu = E(X)$, and $a > 0$, then

$$Pr(|X - \mu| \geq a) \leq 8\Big(\frac{k\mu + k^2}{a^2}\Big)^{\frac{k}{2}}. \tag{1}$$

The proof there is elegant but elementary, using the CDF formulation of expectation, the gamma function, and Stirling's approximation.

The number of buckets is 2^m. The expected weight of a bucket is $|S|2^{-m}$. By the union bound, the probability that there is a bucket of weight more than a fixed polynomial w is $\leq 2^m$ times the probability that a fixed bucket has weight $> w$. The Bernoulli trials are then whether each element hashes to our fixed bucket. We can use (1) with $k = n$, $m = \lg |S| - \Theta(1)$, and a a polynomial in n to make the probability that there is an overfull bucket exponentially small in n. [BGP00] choose the parameters much more carefully for optimality, but we are only shooting for a philosophical point here. This completes the proof that NP witnesses can be sampled uniformly given an NP oracle.

4 BPP $\subseteq \Sigma_2$

Again we will make use of hashing but this time we only need a pair-wise independent hash. An affine map $h(r) = Ar + b$, where A, b are a random 0-1 matrix and vector, will suffice.

Consider a BPP algorithm for a language L. By using a polynomial amount of repetition, we can drive the error probability down to be exponentially small. So the number of witnesses in the case of $x \in L$ is large. If we choose the codomain of h small enough, then no matter how h is chosen, there will be collisions among the witnesses. On the other hand, if we choose the codomain large enough, then in the case that $x \notin L$, then there are few enough witnesses that they can all hash without collision. So $x \in L$ iff for all hashes of the form $Ar + b$ there is a collision between witnesses r, which would show that $L \in \Sigma_2$.

It remains to be shown that we can choose the size of the codomain of h as claimed. Unfortunately, this idea doesn't quite work. The problem is that even with the Chernoff bound we can't reduce the error fast enough through repetition to keep up with the extra randomness needed.

The following idea from Sipser [Sip83] comes to the rescue. [1] First use standard error reduction so that the witness set W satisfies

$$\begin{aligned} |W| &> \frac{1}{n}2^n && \text{when } x \in L \\ |W| &\leq \frac{1}{2n^2}2^n && \text{when } x \notin L \end{aligned}$$

Instead of concentrating on whether h is injective, let us select n pairwise independent hashes $h_1, \ldots, h_n$ independently and say that a witness r is *isolated* iff $\exists i \in [n]\ \forall r' \in W - \{r\}\ h_i(r) \neq h_i(r')$. Also set m so that $2^m = \frac{1}{n^2}2^n$. Then we claim that the event

$$B : \exists r \in W\ \forall i \in [n]\ \exists r' \in W - \{r\}\ h_i(r) = h_i(r')$$

[1] It should be mentioned that a fairly different-looking proof can be found in the lecture notes from the UCSD Spring 2003 CSE201A Advanced Complexity notes, lecture 5.

that there is a non-isolated witness has probability 1 when $x \in L$ and < 1 when $x \notin L$.

To see this, if $x \in L$, then each h_i can isolate at most 2^m witnesses, and so $\leq n2^m = \frac{1}{n}2^n < |W|$ witnesses are isolated. So some witness is not isolated. If $x \notin L$, then from the union bound and independence,

$$Pr(B) \leq |W|(|W|2^{-m})^n = \frac{1}{2n^2}2^n\left(\frac{\frac{1}{2n^2}2^n}{\frac{1}{n^2}2^n}\right)^n = \frac{1}{2n^2} < 1.$$

So $x \in L$ iff there are n hash functions such that B fails. Notice that this predicate has 4 quantifiers: $\exists h_1, \ldots, h_n \ \forall r \ \exists i \ \forall r'$. But the middle $\exists$ is over a poly-sized domain and so can be transformed into a poly-sized disjunction to make the above into a Σ_2 predicate.

5 Constructing the learner

We are now ready to describe the learning algorithm. Let S_{t-1} be the set of s-gate, $O(1)$ fanin circuits that agree with the data $(x_i, j_i), (x_i', j_i')$ for $i = 1, \ldots, t-1$. Then given that data and an NP oracle, we can sample uniformly at random from S_{t-1}. By repeating and taking an average, in poly time we can estimate

$$E_{C \in S_{t-1}}(C(x)) \tag{2}$$

for a given x to within an accuracy of $\frac{1}{k}$. We are not saying that we can compute (2) efficiently, but there is some function f computable within randomized poly-time approximating (2) to within an additive error of $\frac{1}{k}$. Since $\mathrm{BPP} \subseteq \Sigma_2 = \mathrm{NP}^{\mathrm{NP}} = \mathrm{NP}^{\mathrm{P}} = \mathrm{NP} = \mathrm{P}$, we can compute f in deterministic poly-time. Furthermore, we can do so given only the data points as a description of f.

Using our NP oracle, in poly time we can find the maximum value of f and uniformly sample from those x that match this maximum. This is our output x_t.

Notice that x_t is not guaranteed to be uniform over the witnesses of J, but rather over those x that maximize f. But we claim that if the learner makes enough mistakes, then for some t, all the circuits in S_t will agree and then x will be uniform over the witnesses of J. But even if the learner does not make enough mistakes - perhaps because the humans got too lazy to give more training data - when it gives a witness of J, it will be uniformly random, and when it does not, we can become outraged and give more training data, which can only happen a bounded number of times, or shrug our shoulders and live with it.

6 Bounding the number of mistakes

First note that the number of circuits with s-gates, and $O(1)$ fanin is $2^{O(s \lg s)}$. By a *circuit*, we mean a dag - with no restriction on the number of inputs or outputs. To see this, note that each of the s interior nodes must select $O(1)$ children from a list of size $O(s)$. There are at most $(O(s)^{O(1)})^s$ such sequences of choices. So $|S_0| = 2^{O(s \lg s)}$.

We claim that each time the learner makes a mistake, $|S_t|$ is reduced by a factor of $\frac{2}{3}$, and hence that the number of mistakes is $\leq O(s \lg s)$. To see this, suppose indirectly that $x'_t \in X_t \wedge J(x_t) < J(x'_t)$ and

$$|S_t| > \frac{2}{3}|S_{t-1}|.$$

Notice that $\forall C \in S_t \ C(x_t) = 0, C(x'_t) = 1$. So

$$\begin{aligned} &E_{C \in S_{t-1}}(C(x_t)) \geq f(x_t) - \frac{1}{k} \geq f(x'_t) - \frac{1}{k} \\ \geq &E_{C \in S_{t-1}}(C(x'_t)) - \frac{2}{k} \\ \geq &\frac{1}{|S_{t-1}|}\Big(\sum_{C \in S_{t-1}} C(x_t) + |S_t| - (|S_{t-1}| - |S_t|)\Big) - \frac{2}{k} \\ > &E_{C \in S_{t-1}}(C(x_t)) + \frac{1}{3} - \frac{2}{k}, \end{aligned}$$

which gives a contradiction if we choose $k = 6$.

7 Conclusion

In homage to this result, a special robot fund has been established at the UCSD CSE theory lab to prepare for the day when we may need to run from our new robot masters.

References

[BGP00] M. Bellare, O. Goldreich and E. Petrank. *Uniform generation of NP-witnesses using an NP-oracle.* Information and Computation, Vol. 163, 2000, pp. 510–526.

[Sip83] M. Sipser. *A complexity theoretic approach to randomness* STOC, 1983, pp. 330–335.