Undergraduate Complexity Theory **CMU 15-455, Spring 2017**

Homework 10

Due: 5:00pm, Thursday April 27

Note the later due date.

Feature: As before, if your homework is typeset (as opposed to handwritten), you will receive 1 bonus point.

---

1. **(Two-sided error versus one-sided error.)** As mentioned in class, we don't really have any "natural" problem that we know to be in BPP (efficiently solvable with two-sided error) but don't know to be in either RP or coRP (efficiently solvable with one-sided error). However, in this problem you will see that an "unnatural" problem may fit the bill. First:

   (a) (2 points.) Show that if $\mathsf{BPP} = \mathsf{RP}$, then $\mathsf{RP} = \mathsf{ZPP}$.

   The contrapositive of the above is $\mathsf{RP} \neq \mathsf{ZPP} \implies \mathsf{BPP} \neq \mathsf{RP}$. In other words, if you can find a language $L \in \mathsf{RP}$ that is not in ZPP — i.e., a problem which genuinely needs the full power of RP — then you can find some language $L' \in \mathsf{BPP}$ that is not in RP. (L' also wouldn't be in coRP.) Such an L' would be an example of a problem efficiently solvable with two-sided error that's not efficiently solvable with one-sided error.

   Of course, we probably can't literally do this, since it is somewhat commonly believed that $\mathsf{BPP} = \mathsf{RP} = \mathsf{ZPP} = \mathsf{P}$. However, we might be able to do it "de facto", using a language $L \in \mathsf{RP}$ that we currently don't know to be in ZPP.[1] The following problem shows how to convert such an L into a language $L' \in \mathsf{BPP}$ that isn't obviously in RP or coRP. (However this L' is kind of unnatural.)

   (b) (8 points.) Assume $L \in \mathsf{RP} \setminus \mathsf{ZPP}$. Define

   $$L' = \{(x, y) : \text{either } x \in L \text{ and } y \notin L, \text{ or vice versa}\}.$$

   Prove that $L' \in \mathsf{BPP}$ and also that $L' \notin \mathsf{RP} \cup \mathsf{coRP}$ (4 points each).

2. **(VC Dimension.)** The "VC (Vapnik–Červonenkis) Dimension" is an extremely important concept in learning theory. Let $\mathcal{X}$ be a "universe" (set) of "instances". A "concept" H is a "classifier" that labels each instance as positive or negative; more precisely, it is a function $H : \mathcal{X} \to \{0, 1\}$. A "concept class" $\mathcal{H}$ is a set of concepts.

   Suppose $S = (x^1, \ldots, x^m)$ is a "sample of size m", meaning a sequence of m distinct instances in $\mathcal{X}$. Now for any concept H, if you apply it to the instances of S you get a sequence of labels, $H(S) \in \{0, 1\}^m$. We say that sample S is "shattered by $\mathcal{H}$" if for *every* string $y \in \{0, 1\}^m$, there is some $H \in \mathcal{H}$ with $H(S) = y$. For example, if $m = 1$ and $S = (x)$, then S is shattered by $\mathcal{H}$ as long as there is some $H \in \mathcal{H}$ with $H(x) = 0$ and also some other $H' \in \mathcal{H}$ with $H(x) = 1$. For another example ($m = 2$), we say $S = (w, x)$ is shattered by $\mathcal{H}$ if it's possible to find four concepts $H_{00}, H_{01}, H_{10}, H_{11} \in \mathcal{H}$ such that $H_{01}(w) = 0, H_{01}(x) = 1$ and similarly for H_{00}, H_{10}, H_{11}.

[1] An example of such an L is the language of all arithmetic formulas (with symbols for variables, $\cdot$, $+$, $-$, and 1) that compute a nonzero polynomial. E.g., this L contains $(x+y)\cdot(x+y)+(x-y)\cdot(x-y)$ but does not contain $(x+y)\cdot(x+y)-(x-y)\cdot(x-y)-(1+1)\cdot(1+1)\cdot x\cdot y$. We know this language is in RP — the algorithm is basically "plug in a few random n-bit integers, evaluate the formula modulo a random n-bit prime, and accept if you ever get a nonzero answer". However we don't know if this problem is in ZPP.

(a) (3 points.) (This problem has nothing to do with complexity theory, it's to help you understand the definitions.) Suppose $\mathcal{X} = \mathbb{R}^2$, so instances are points in the 2-d plane. And suppose $\mathcal{H}$ is the set of all "halfspaces"; here we say $H : \mathbb{R}^2 \to \{0,1\}$ is a "halfspace" if there is some (infinite) straight line in the plane such that H labels all points on one side of the line 0 and all points on the other side of the line 1 (say that points exactly on the line are also labeled 1). For this problem, do three things: (i) Find a sample of size 3 that is shattered. (ii) Find a sample of size 3 that is not shattered. (iii) Show that a size-4 sample consisting of the corners of a square is not shattered.

Actually, something stronger than (iii) is true: there is *no* size-4 sample that is shattered. You can probably convince yourself this is true, although you're not asked to prove it.

Here is the key definition for this problem: given a concept class $\mathcal{H}$, its *VC Dimension* $\mathit{VC}(\mathcal{H})$ is the largest possible size of a set that is shattered. Thus in the example where $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{H} = \{\text{halfspaces}\}$, we have $\mathit{VC}(\mathcal{H}) = 3$, because there *is* a shattered size-3 sample, and there *isn't* a shattered size-4 sample.

(b) (1 point.) Assume $\mathcal{H}$ is finite. Show that $\mathit{VC}(\mathcal{H}) \leq \log_2 |\mathcal{H}|$.

In a typical learning theory scenario, there is a "dimension" d (the number of "features"), the universe is $\mathcal{X} = \{0,1\}^d$, and the concept class $\mathcal{H}$ contains an exponential-in-d number of concepts. However, each $H \in \mathcal{H}$ has a relatively short descriptor h (like, the equation of the line, in the case of halfspaces), and given this descriptor and an instance $x \in \mathcal{X}$, it is easy to tell if $H(x)$ is 0 or 1. More formally, let C be a Boolean circuit that takes as input two strings $x \in \{0,1\}^d$ and $h \in \{0,1\}^e$. We say that C "implicitly defines" a concept class $\mathcal{H}_C$ of cardinality 2^e, consisting of a concept H_h for each $h \in \{0,1\}^e$ defined by $H_h(x) = C(x,h)$. In other words, C takes as input the name of an instance and the name of a concept and outputs the label that concept gives to that instance.

(c) (6 points.) Let $\text{VCD} = \{\langle C, k\rangle : \mathit{VC}(\mathcal{H}_C) \geq k\}$. Here C is the encoding of a two-input circuit as before, and k is a natural number encoded in binary. In words, the VCD problem is to decide if the VC Dimension of an (implicitly defined) concept class is at least k.

Prove that $\text{VCD} \in \Sigma_3$.

(Slight hint: You might find it easier to prove that it's in Σ_4; you may need part (b) to get it down to Σ_3.)

In fact, VCD is Σ_3-complete, but this is noticeably harder to prove.

3. **(Σ_2P in NP with a SAT oracle.)** A *nondeterministic* SAT-*oracle TM* N is the nondeterministic analogue of a SAT-oracle TM, as defined in Lecture 24. Specifically, N is a nondeterministic Turing Machine with an extra power. It has an extra read/write tape, called the "oracle tape", and it has an new "ORACLE" instruction with the following behavior: when ORACLE is called with string ϕ on the oracle tape, the oracle tape's contents are replaced with "1" if $\phi \in$ SAT and with "0" if $\phi \notin$ SAT. Note that the ORACLE instruction is completely deterministic (albeit unrealistic). The *nondeterministic* aspect is as usual: besides the ORACLE instruction, N has the usual "goto-both"/nondeterministic-branching feature, and we say that $N(x)$ overall accepts if there *exists* a computation branch on which ends in an accepting state.

We write $\mathsf{NP}^{\mathrm{SAT}}$ for the class of all languages L that are accepted by a polynomial-time nondeterministic SAT-oracle TM.

More generally, given any language B, we can analogously define a *nondeterministic B-oracle TM*, and NP^{B}.

(a) (2 points.) Prove that $\mathsf{NP}^{B} = \mathsf{NP}^{\overline{B}}$, where $\overline{B}$ is the complement of language B.

(b) (2 points.) Prove that if $A \leq_m^P B$, then $\mathsf{NP}^{A} \subseteq \mathsf{NP}^{B}$.

(c) (3 points.) Prove that if B is NP-complete *or* if B is coNP-complete, then $\mathsf{NP}^{\mathrm{SAT}} = \mathsf{NP}^{B}$. (For this reason, $\mathsf{NP}^{\mathrm{SAT}}$ is often denoted $\mathsf{NP}^{\mathsf{NP}}$.)

(Hint: Parts (a), (b), (c) are all equally true about deterministic oracle-computation; is there any difference for nondeterministic oracle-computation?)

(d) (3 points.) Prove that $\Sigma_2\mathsf{P} \subseteq \mathsf{NP}^{\mathrm{SAT}}$. (Hint: your SAT-oracle NTM will probably only need to use the ORACLE once, but possibly not in the very last step.)

4. **(And the other way around.)** (10 points.) Prove that $\mathsf{NP}^{\mathrm{SAT}} \subseteq \Sigma_2\mathsf{P}$. (Thus $\Sigma_2\mathsf{P}$ is the same as $\mathsf{NP}^{\mathrm{SAT}} = \mathsf{NP}^{\mathsf{NP}}$, and indeed it is sometimes defined this way.)

(Hint: Mimic, fill in details for, and extend the proof from Lecture 24 that $\mathsf{P}^{\mathrm{SAT}} \subseteq \Sigma_2\mathsf{P}$.)