6.S976 and 18.S996: Problem Set 1

Due: 23:59 April 3, 2026

Total number of points: 100.

Collaboration Policy. You are welcome to collaborate with other students or AI models, but you should write up your solutions fully on your own *without any external assistance whatsoever*; and you should *acknowledge all of your collaborators* (both students and AI models). Solutions will be graded not only for correctness but also for (human) legibility and clarity.

1. (a) (10 points) Let $\varepsilon, \delta \in (0,1)$, let $\mathcal{X}$ be a finite set, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a set of functions mapping $\mathcal{X}$ to $\{0,1\}$. Consider an algorithm A for the following task: For an arbitrary distribution $\mathcal{D}$ over $\mathcal{X}$ and arbitrary $h \in \mathcal{H}$, given m labeled samples $Z = \{(x_i, h(x_i))\}_{i\in[m]}$ where $x_i \sim_{\text{i.i.d.}} \mathcal{D}$, output $\hat{h} : \mathcal{X} \to \{0,1\}$ so that

$$\mathop{\mathbf{Pr}}_{\hat{h}\leftarrow A(Z)}\left[\hat{h} \in \mathcal{H} \text{ and } \mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}\left[\hat{h}(x) \neq h(x)\right] \leq \varepsilon\right] \geq 1-\delta.$$

   Provide such an algorithm A and prove that solves it the above task with sample complexity

$$m = O\left(\frac{\log|\mathcal{H}| + \log(1/\delta)}{\varepsilon}\right).$$

   We emphasize that this is a *proper* learning task, as it is required that $\hat{h} \in \mathcal{H}$. There are no constraints on the computational complexity of A.

   Note: You may *not* invoke existing results about PAC learning or VC dimension; please provide a self-contained proof.

   (b) (5 points) Show that the $\log|\mathcal{H}|$ dependence in 1a is tight in the following sense. For $\mathcal{X} = \{0,1\}^n$, construct a hypothesis class $\mathcal{H}$ and (family of) distribution(s) $\mathcal{D}$ such that any algorithm A with $m < \log_2|\mathcal{H}|$ must have

$$\mathop{\mathbf{Pr}}_{\hat{h}\leftarrow A(Z)}\left[\hat{h} \in \mathcal{H} \text{ and } \mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}\left[\hat{h}(x) \neq h(x)\right] \leq 1/4\right] \leq 3/4,$$

   where $Z = \{(x_i, h(x_i))\}_{i\in[m]}$ for $x_i \sim_{\text{i.i.d.}} \mathcal{D}$.
   (Hint: Consider linear functions $\mathbb{F}_2^n \to \mathbb{F}_2$.)

   (c) (10 points) From a learning theory perspective, it would be wonderful if the algorithm you provided in 1a could be made to run in $\mathsf{poly}(m)$ time. Prove that this is not possible, assuming the existence of a computationally secure secret-key encryption scheme.

   Specifically, prove that the existence of a computationally secure secret-key encryption scheme implies that an algorithm as in 1a cannot be made to run in time $\mathsf{poly}(\log|\mathcal{H}|)$,

even when $m = \mathsf{poly}(\log |\mathcal{H}|)$, $\varepsilon = \delta = 1/4$, and the requirement that $\hat{h} \in \mathcal{H}$ is removed (i.e., A can be *improper*). Note that 1a shows that $m = O(\log |\mathcal{H}|)$ samples suffice when computational complexity is not a constraint. Therefore, basic cryptographic primitives imply *superpolynomial* gaps between the sample complexity and computational complexity of PAC learning.

(Hint: Consider the possible decryption functions.)

Note: If helpful, here is a standalone definition of computationally secure secret-key encryption for message space $\mathcal{M} = \{0,1\}$ you can use.

- The secret key sk is uniformly sampled from $\{0,1\}^n$.
- $\mathsf{Enc}_{\mathsf{sk}} : \{0,1\} \to \{0,1\}^n$ is a randomized (polynomial-time) encryption function. (If you want to be more formal about the randomness, you can write the syntax of $\mathsf{Enc}_{\mathsf{sk}}$ as $\mathsf{Enc}_{\mathsf{sk}} : \{0,1\} \times R \to \{0,1\}^n$, where sampling from R is implicit.)
- $\mathsf{Dec}_{\mathsf{sk}} : \{0,1\}^n \to \{0,1\}$ is a (polynomial-time) decryption function.

Correctness says that for $b \in \{0,1\}$, $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{sk}}(b)) = b$ with probability 1. Computational security says that given $\mathsf{poly}(n)$ samples of $(b, \mathsf{Enc}_{\mathsf{sk}}(b))$ for any polynomial function $\mathsf{poly}(\cdot)$ where b is sampled uniformly from $\{0,1\}$, no polynomial time algorithm can predict b given $\mathsf{Enc}_{\mathsf{sk}}(b)$ (where b is a uniformly random bit) with probability greater than $1/2 + \mathsf{negl}(n)$.

2. You learned about pseudorandom functions in class, families of functions $\mathcal{F} = \{F_k : [N] \to \{0,1\}\}_{k \in \{0,1\}^K}$, specified by a secret key k of length $K = O(\log N)$. The key requirement is that oracle access to a random function chosen from $\mathcal{F}$ is indistinguishable to an adversary running in $\mathsf{poly}(\log N)$ time from oracle access to a uniformly random function $G : [N] \to \{0,1\}$. This may not remain true when the adversary obtains *enhanced access* to the function. For this problem, enhanced access will mean that the adversary can, in a single query (a, b) where $a, b \in [N]$ and $a \leq b$, obtain

$$\sum_{a \leq i \leq b} F_k(i) \pmod 2 \in \{0,1\}.$$

   (a) (10 points) Suppose PRFs exist. Construct a PRF that is secure with enhanced query access (and prove it).

   (b) (10 points) Show that PRFs are not necessarily secure given enhanced query access, by demonstrating that if PRFs exist at all, then there are PRFs that are secure with query access but insecure with enhanced query access. As a consequence, there are concept classes that are hard to learn with query access but easy to learn given enhanced query access.

   Note: For full credit, you need to start from the assumption that PRFs exist, construct a modified PRF, prove that the modified PRF is secure, and prove that the modified PRF is insecure when given enhanced access.

   (c) (5 points) Consider functions of the form $F_{x^*}(y) = 1$ if and only if $y = x^*$. These functions are not learnable with query access (you do not need to show this). Show that these functions *are* learnable given enhanced access. (For this problem, to learn a function F_{x^*}, it suffices to output x^*.)

3. In this problem, we will work with a simple version of generative models. Formally, by a generative model, we mean a tuple $G = (Q, q_0, T, M)$, where Q is a finite state space, $q_0 \in Q$

is an initial state, $T \in \mathbb{N}$ is a time horizon, and $M \colon Q \times [T] \to \Delta(Q \times \{0,1\})$ is a transition map (here, $\Delta(S)$ denotes the set of probability distributions over a finite set S). The model generatively samples a sequence $(\sigma_1, \ldots, \sigma_T) \in \{0,1\}^T$ as follows:

1. Set $q \leftarrow q_0$.
2. For each $t = 1, \ldots, T$:
   i. Sample $(q', \sigma_t) \sim M(q, t)$.
   ii. Update $q \leftarrow q'$.
3. Output $(\sigma_1, \ldots, \sigma_T) \in \{0,1\}^T$.

One can think of this as a very simple autoregressive generative model, where there is no prompt and the model does not keep track of its previous outputs (except via its internal state $q \in Q$). The number of states of the generative model is $|Q|$.

By *learning* a generative model in polynomial time, we mean outputting a next-token sampler in polynomial time. Specifically, given $\mathsf{poly}(T)$ samples from the generative model G, in $\mathsf{poly}(T)$ time, output a $\mathsf{poly}(T)$-time computable function $f : \{0,1\}^* \to [0,1]$ with probability at least $9/10$ such that for all $t \in [T]$,

$$\mathop{\mathbf{E}}_{(\sigma_1,\ldots,\sigma_T)\sim G}\left[\left| f(\sigma_1\sigma_2\ldots\sigma_{t-1}) - \mathop{\mathbf{Pr}}_{(\sigma'_1,\ldots,\sigma'_T)\sim G}[\sigma'_t = 1 \mid \sigma'_1 = \sigma_1, \sigma'_2 = \sigma_2, \ldots, \sigma'_{t-1} = \sigma_{t-1}] \right|\right] \leq \frac{1}{100}.$$

In words, this means that for all token indices $t \in [T]$, in expectation over samples from G, f approximately computes the correct conditional distribution of the tth token. For an algorithm to learn a *family* of generative models, the algorithm must be able to learn all generative models in the family in polynomial time.

(a) (10 points) Prove that the family of all 1-state generative models can be learned in polynomial time.

(b) (15 points) Suppose the Learning Parity with Noise (LPN) assumption is true. Prove that there is a family of generative models with 2 states that cannot be learned in polynomial time.

Note: If helpful, here is a standalone definition of the LPN assumption you can use. For noise rate $\eta = 1/10$ and sample complexity $m = \mathsf{poly}(n)$ for any polynomial function $\mathsf{poly}(\cdot)$,

$$\left(\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top\right) \approx_c \left(\mathbf{A}, \mathbf{u}^\top\right),$$

where $\mathbf{A} \leftarrow \mathbb{F}_2^{n \times m}, \mathbf{s} \leftarrow \mathbb{F}_2^n$, $\mathbf{e} \leftarrow \mathrm{Bern}(\eta)^m$, and $\mathbf{u} \leftarrow \mathbb{F}_2^m$. Here, the computational indistinguishability holds with respect to all $\mathsf{poly}(n)$-time distinguishers.

4. (25 points) This problem is about watermarks that are unremovable by any polynomial-time strategy. This is a tricky notion, one that we do not even know how to even define well, let alone achieve, at this point. However, in this problem, you will show how to watermark a particular, admittedly somewhat contrived, model.

   The model M_K is defined by a key $K \in \{0,1\}^n$ of a pseudorandom function $F_K : \{0,1\}^{2n} \to \{0,1\}^{100n}$; on input a prompt $\pi \in \{0,1\}^n$, it samples a random string $r \in \{0,1\}^n$, and outputs $r || F_K(\pi || r)$ (where $||$ denotes concatenation of strings).

Assuming the existence of pseudorandom functions, construct a watermarked version of M (namely efficient Mark and Detect algorithms) that satisfies completeness, soundness, undetectability and unremovability in the following sense:

- *Completeness*: For any $K \in \{0,1\}^n$ and any prompt $\pi \in \{0,1\}^n$,

$$\mathop{\mathbf{Pr}}_{\mathsf{sk} \leftarrow \{0,1\}^n} [\mathsf{Detect}_{\mathsf{sk}}(\mathsf{Mark}_{K,\mathsf{sk}}(\pi)) = 1] = 1,$$

  where the probability is also taken over the randomness of $\mathsf{Mark}_{K,\mathsf{sk}}$.

- *Soundness*: For any $K \in \{0,1\}^n$ and any fixed string $y \in \{0,1\}^{101n}$,

$$\mathop{\mathbf{Pr}}_{\mathsf{sk} \leftarrow \{0,1\}^n} [\mathsf{Detect}_{\mathsf{sk}}(y) = 1] \leq \mathsf{negl}(n).$$

- *Undetectability*: For any $K \in \{0,1\}^n$ and for all PPT distinguishers D,

$$\left| \mathbf{Pr}\left[D^{M_K}(K) = 1\right] - \mathbf{Pr}\left[D^{\mathsf{Mark}_{K,\mathsf{sk}}}(K) = 1\right] \right| \leq \mathsf{negl}(n).$$

  The probability is taken over $\mathsf{sk} \leftarrow \{0,1\}^n$, the randomness of $\mathsf{Mark}_{K,\mathsf{sk}}$, and the randomness used by the distinguisher D. We emphasize that *the distinguisher D has access to the key K defining the model*.

- *Unremovability*: For all PPT A,

$$\mathop{\mathbf{Pr}}_{\substack{K \leftarrow \{0,1\}^n \\ \mathsf{sk} \leftarrow \{0,1\}^n \\ (\pi,y) \leftarrow A^{\mathsf{Mark}_{K,\mathsf{sk}}}(1^n)}} [(\mathsf{Detect}_{\mathsf{sk}}(y) = 0) \text{ and } (y \text{ in the support of } M_K(\pi))] \leq \mathsf{negl}(n).$$

  In words, given only oracle access to Mark, it is computationally hard to find an output y that both (a) evades watermark detection and (b) is a valid output from the model M_K. We emphasize that A does not have direct access to the (unwatermarked) model M_K (i.e., no access to K).

The key difference between M_K and real-world models is that M_K is designed to be *not* learnable while real-world models typically are. Defining a (potentially) achievable notion of unremovable watermarks for *learnable models* is a much trickier proposition; and good fodder for a course project.