10-716: Advanced Machine Learning **Spring 2019**

Lecture 18: April 2

Lecturer: Pradeep Ravikumar *Scribes: Xuejian Wang, Zijun Shi, Mengxiong Liu*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

18.1 Oracle Complexity

18.1.1 Statistical Query Model

Suppose $x \sim P$, which we don't have access to. Instead, we have access to an oracle that takes a query $q : x \to R^d$ and returns $Z_q \in R^d$, which is a noisy evaluation of $E_{x\sim P}[q(x)]$, with a guarantee that

$$P\Big(\sup_{q\in Q}\big|Z_Q - E_{x\sim P}[q(x)]\big| \geq \tau_Q\Big) \leq \epsilon$$

i.e., Z_Q concentrated around $E_{x\sim P}[q(x)]$. We aim to minimize the empirical risk by only querying the oracle. We are interested in establishing lower bounds on the sample complexity under a computational budget.

Definition 18.1 *Let $\mathcal{P}$ be the problem space, Φ_T be the set of algorithms that makes at most T queries to the oracle, ρ be some distance measure between two solutions, $\bar{P}$ be the distribution over oracle answers. Then the minimax error is defined as*

$$\inf_{\hat{\phi}\in\Phi_T}\sup_{P\in\mathcal{P}} E_{\hat{P}}\Big[\rho(\hat{\phi}, \theta(P))\Big]$$

18.1.2 Hypothesis Testing

Consider the case of hypothesis testing, $H_0 \equiv \theta \in G_0, H_1 \equiv \theta \in G_1, \phi : Z \to \{0, 1\}$, where Z is the oracle output. Then the minimax error can be written as the following

$$\inf_{\phi}\Big[\sup_{\theta\in G_0} P_\theta(\phi(Z) = 1) + \sup_{\theta\in G_1} P_\theta(\phi(Z) = 0)\Big]$$

E.g. Gaussian Mixture Model $H_0 \equiv X \sim \mathcal{N}(\mu_0, \Sigma), H_1 \equiv X \sim v\mathcal{N}(\mu_1, \Sigma) + (1 - v)\mathcal{N}(\mu_2, \Sigma)$ with $\mu_0, \mu_1, \mu_2 \in R^p$ and v fixed. Let $s = ||\mu_1 - \mu_2||_0$, we have the following information theoretic lower bound.

$$\text{testing error} > c' > 0 \text{ if } \|\mu_1 - \mu_2\|_2 \leq c\sqrt{\frac{s\log p}{n}}$$

$$\text{testing error} \xrightarrow{n\to\infty} 0 \text{ if } \|\mu_1 - \mu_2\|_2 > c\sqrt{\frac{s\log p}{n}}$$

Let Φ_T be the set of algorithms that makes at most T queries to the oracle with $T = O(d^c)$, using oracle complexity we have the following lower bound

$$\text{testing error} \xrightarrow{n\to\infty} 0 \text{ if } \|\mu_1 - \mu_2\|_2 \geq \sqrt{\frac{s^2 \log p}{n}}$$
$$\text{testing error} > c' > 0 \text{ if } \|\mu_1 - \mu_2\|_2 = O\left(\sqrt{\frac{s^2 \log p}{n}}\right)$$

18.1.3 Stochastic Gradient Descent

Let $f : x \to R$ be the function to be minimized, Φ_T be the set of algorithms that makes at most T queries to the oracle, which returns a noisy gradient g given any input x. Furthermore, assume $\forall x$, we have

$$E[g] = \nabla f(x)$$
$$E\Big[(g - \nabla f(x))^2\Big] \leq \sigma^2$$

Then the minimax error has the following lower bound

$$\inf_{\phi\in\Phi_T} \sup_{f\in\mathcal{F}} E_Z\Big[\rho(\phi(Z), \inf_x f(x))\Big] \geq \begin{cases} \frac{c}{\sqrt{T}} & \mathcal{F} \text{ is a set of convex Lipschitz functions} \\ \frac{c}{T} & \mathcal{F} \text{ is a set of strongly convex functions} \end{cases}$$

where the expectation is taken over oracle answers.

18.2 Decision Problems

Define $\{0,1\}^* = \bigcup_{n\geq 0}\{0,1\}^n$ where $\{0,1\}^n$ be n bits binary string. Define $f : \{0,1\}^* \to \{0,1\}$. Define language associated with $f, L_f = \{x \in \{0,1\}^* | f(x) = 1\}$. Then $x \in L_f \Leftrightarrow f(x) = 1$.

18.2.1 Deterministic time and the class P

Definition 18.2 (Computing a function and running time) *Let $f : \{0,1\} \to \{0,1\}$ and let $T : \mathbb{N} \to \mathbb{N}$ be some functions, and let M be a Turing machine. We say that M computes f in $T(n)$-time if for every $x \in \{0,1\}$, if M is initialized to the start configuration on input x, then after at most $T(|x|)$ steps it halts with $f(x)$ written on its output tape. We say that M computes f if it computes f in $T(n)$ time for some function $T : \mathbb{N} \to \mathbb{N}$.*

Definition 18.3 (The class DTIME) *Let $T : \mathbb{N} \to \mathbb{N}$ be some function. We let $DTIME(T(n))$ be the set of all Boolean (one bit output) functions that are computable in $c \cdot T(n)$-time for some constant $c > 0$.*

Definition 18.4 (The class P) $P = \bigcup_{c\geq 1} \mathbf{DTIME}(n^c)$

The class **P** contains all decision problems that could be solved by Turing machines in polynomial times. Normally, we say the problems in **P** are efficiently solvable.

18.2.2 Criticisms of P

- Requires the amount of computation to be bounded for all input. However, not all possible inputs arise in practice.
- Formulates computation in terms of decision problems, which are too limited.
- Does there exist other physically realizable computation device, such as an alien computer, faster than Turing Machine?
  No, according to Church-Turing (CT) thesis, which states that every physically realizable computation device whether its silicon-based, DNA-based, neuron-based or using some alien technology can be simulated by a Turing machine. (The CT thesis is not a theorem, merely a belief about the nature of the world.) However, there are several objections made to the strong form of CT thesis.

18.2.3 The class NP

Definition 18.5 (The class NP) *A language $L \subseteq \{0,1\}^*$ is in **NP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial time Turing machine M such that $\forall x \in \{0,1\}^*$:*

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \text{ s.t. } M(x,u) = 1$$

Here u is called a certificate for x w.r.t. language L and machine M.

For example, given an inequality $Ax \leq b$, we would like to know whether there is a rational solution x to the problem. This might not be easy to answer. However, if a possible solution u is also given, we can check if $Au \leq b$.

Theorem 18.6 $P \subseteq NP \subseteq \cup_{c>1} DTIME(2^{nc})$

18.2.4 Reducibility, NP-hard and NP-completeness

Definition 18.7 (Reduction) *$A \subseteq \{0,1\}^*$ is polynomial-time reducible to some other language $B \subseteq \{0,1\}^*$, if there exists polynomial-time computation $f\{0,1\}^* \to \{0,1\}^*$, s.t. $\forall x \in \{0,1\}^*$, $x \in A$, if and only if $f(x) \in B$.*

Definition 18.8 (NP-hard) *B is NP-hard, if $\forall A \in NP$, A is reduced in B.*

Definition 18.9 (NP-complete) *B is NP-complete, if B is NP-hard and $B \in NP$.*

18.2.5 Decision vs. Search

Given an inequality $Ax \leq b$, a decision problem is asking whether there exists a rational solution, while a search problem is asking what exactly the solution is.

Apparently, the search problem is harder than the decision problem. However, if the problem is **NP**-complete, they are equivalent in the sense that if the decision problem can be solved(hence $\mathbf{P} = \mathbf{NP}$) then the search problem can also be solved in polynomial time.

Theorem 18.10 *There exists polynomial time TM M where $x \in L$ iff $\exists u \in \{0,1\}^{p(|x|)}$ such that $M(x,u) = 1$. There exists TM B that makes polynomial queries to NP-decision oracle, such that $M(x, B(x)) = 1$, $B(x) \in \{0,1\}^{p(|x|)}$.*

18.3 Computation Complexity of Learning

Next, we study how size of training data can be leveraged to reduce computational complexity. We'll give a toy example first and delve deeper in the next lecture. We follow the notation of [SST] in the following sections. Let $x \in \mathcal{X}, y \in \mathcal{Y}$, a prediction rule $h : x \to y$. A learning algorithm, A, receives a training set of m examples, $S_m = ((x_1, y_1), ..., (x_m, y_m))$ and produces a prediction rule $A(m)$. The goal of the learner is to find a prediction rule with low risk, defined as

$$err(h) = E_{(x,y)\sim\mathcal{D}}\Big[l(h(x), y)\Big]$$

The learner is required to produce a predictor whose risk will be close to $\inf_{h\in\mathcal{H}} err(h)$. Denote by $time(A, m)$ the upper bound on the run time of the algorithm A when running on any training set of m examples. The main object that we will be studying is the following:

$$T_{\mathcal{H},\epsilon}(m) = \min\{\exists A \text{ s.t. } \forall \mathcal{D}, time(A, m) \le t \wedge err(A(m)) \le \inf_{h\in\mathcal{H}} err(h) + \epsilon\}$$

18.3.1 Learning 3-DNF

The following example is adopted from [SST]. Consider the problem of learning the class of 3-term disjunctive normal form (DNF) formulas in the realizable case. A 3-DNF is a boolean function, $h : \{0,1\}^d \to \{0,1\}$, of the form $h(x) = T_1(x) \wedge T_2(x) \wedge T_3(x)$, where $T_i(x)$ is a conjunction of an arbitrary number of literals, e.g. $Ti(x) = x_1 \wedge \neg x_3 \wedge x_5 \wedge \neg x_7$. Since the number of 3-DNF formulas is at most 3^{3d}, it follows that the information theoretic sample complexity is $O(\frac{d}{\epsilon})$. However, unless RP=NP, the search problem of finding a 3-DNF formula which is (approximately) consistent with a given training set cannot be performed in $poly(d)$ time. On the other hand, if number of samples $m = \theta(\frac{d^3}{\epsilon})$ then $T_{\mathcal{H},\epsilon}(m) = poly(\frac{d}{\epsilon})$. Note that there is no contradiction between the last two sentences, since the former establishes hardness of proper learning while the latter claims feasibility of improper learning. The reduction of computational complexity comes with the fact that with more samples we can search in a large hypothesis space to achieve the same generalization error.

References

[SST] Shalev-Shwartz, Shamir, Tromer, "Using More Data to Speed-up Training Time" *Proceedings of the 15th International Con- ference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[AB] Arora, Sanjeev, Barak, Boaz, "Computational complexity: a modern approach" *Cambridge University Press*, 2009.