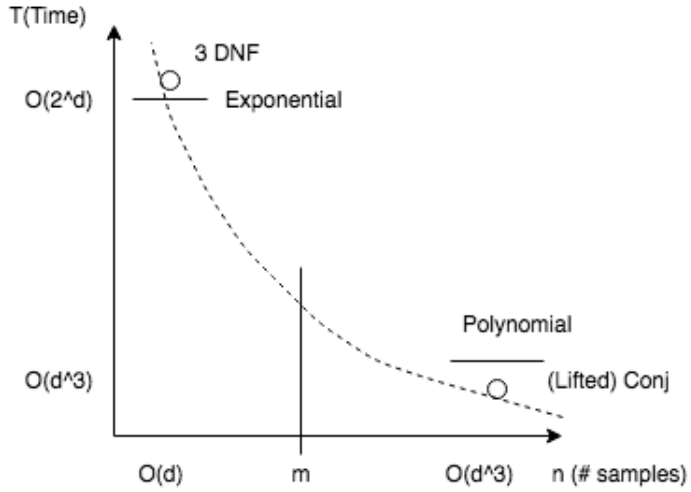
10-716: Advanced Machine Learning **Spring 2019**

Lecture 19: March 28

Lecturer: Pradeep Ravikumar *Scribes: Xinze Wang, Zhiqi Wang, Fan Fan*

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



We can see that if given more samples, computation complexity decreases. However, we don't have very good ways to verify the existence of the dotted curve. In other words, given m samples, we couldn't lower bound the computation complexity and given a certain computation complexity, we couldn't lower bound the number of samples.

Let $\{0,1\}^*$ denote the set of all finite bits strings and Unif $\{0,1\}^*$ denotes the uniform distribution over $\{0,1\}^*$, the we have the following definition:

Definition 19.1 *One-way permutation: A one-way permutation $P : \{0,1\}^* \mapsto \{0,1\}^*$ is a boolean function which for any n, maps $\{0,1\}^*$ to itself; there exists an algorithm for computing P(x) with polynomial runtime. And for $\forall$ polynomial-time algorithm A, $\forall$ x, $\mathop{\mathbb{P}}_{x \in unif\{0,1\}^*}[A(P(x)) = x] < \frac{1}{poly(n)}$ for sufficient large n.*

It's widely conjectured that such one-way permutation exist. One concrete candidate is the RSA permutation function. P(x) $\equiv x^3$ mod N, which treats $x \in \{0,1\}^n$ as a number in $\{0, ..., 2^n - 1\}$. Here, N is a product of two random primes of length n such that (p - 1)(q - 1) does not divide 3. Since the existence of such a one-way permutation would imply P $\neq$ NP, there is no formal proof that such functions exist.

Let P be a one-way permutation, and consider a classification problem. Let $\mathcal{X} = \{0,1\}^n$ and treat it as a pair (r, s). Let $f^*(x) =< r, p^{-1}(s) > \in \{0,1\}$, where $< r, r^{'} >= \sum_{i=1}^{n} r_i r_i^{'}$ mod 2. The hypothesis class $\mathcal{H}$ consists of randomized functions, parameterized by $\{0,1\}^n$, and defines as follow:

$$h_\alpha(r,s) = \begin{cases} < r, \alpha >, & \text{if } \alpha = p^{-1}(s) \\ \text{Unif}\{0,1\}, & \text{otherwise} \end{cases}$$

$\mathcal{H} = \{h_\alpha\}_{\alpha \in \{0,1\}^n}$. We define $\mathcal{D}_\alpha$ as a distribution such that:

$$D_\alpha = \{\underbrace{< r, P(\alpha) >}_{\text{x}}, \underbrace{< r, \alpha >}_{\text{y}}\}$$

Note that $r \in \{0,1\}^n$. Also, note that for any such distribution D_α, $\inf_{h \in \mathcal{H}} err(h) = 0$, and this is achieved with the hypothesis h_α.

19.1 Learning Algorithms

Theorem 19.2 *There exists an agonostic binary classification learning problem over $\mathcal{X} = \{0,1\}^{2n}$ and $\mathcal{Y} = \{0,1\}$ with the following properties:*

1. *It is inefficiently learnable with sample size $m = O(1/\epsilon^2)$, and running time $O(2^n + m)$.*
2. *Assuming one-way permutations exist, there exist no polynomial-time learning algorithm based on a sample of size $O(\log(n))$.*
3. *It is efficient learnable with a sample of size $m = O(n/\epsilon^2)$. Specifically, the trainging time is O(m), resulting in an improper predictor whose runtime is $O(m^3)$.*

For the theorem 19.2.2 it implies that the learning algorithm based on a sample of size $O(\log(n))$ will fail to learn a classifier with low error.

Proof: We consider the following "hard" set of distributions D_α, parameterized by $\alpha \in \{0,1\}^n$: each D_α is a uniform distribution over all $((r, P(\alpha)), \langle r, \alpha \rangle)$. Note that there are exactly 2^n such examples, one for each choice of $r \in \{0,1\}^n$. Also, note that for any such distribution D_α, $\inf_{h \in \mathcal{H}} err(h) = 0$, and this is achieved with the hypothesis h_α.

First, we will prove that with a sample size $m = O(\log(n))$, any efficient learner fails on at least one of the distributions D_α. Suppose on the contrary that we have an efficient distribution-free learner A, that works on all D_α, in the sense of seeing $m = O(\log(n))$ examples and then outputting some hypothesis h such that $h(r, P(\alpha)) = \langle r, \alpha \rangle$ with even some non-trivial probability (e.g. at least $1/2 + 1/poly(n)$). However, by the Goldreich-Levin Theorem, such an algorithm can be used to efficiently invert P, violating the assumption that P is a one-way permutation.

Thus, we just need to show how given $P(x), r$, we can efficiently compute $\langle x, r \rangle$ with probability at least $1/poly(n)$. The procedure works as follows: we pick $m = O(\log(n))$ vectors $r_1, ..., r_m$ uniformly at random from $\{0,1\}^n$, and pick uniformly at random bits $b_1, ..., b_m$, $b_i \sim Uniform\{0,1\}^n$. We then apply our learning algorithm A over the examples $D_m \equiv \{(r_i, P(\alpha)), b_i\}_{i=1}^m$, getting us some predictor h'. We then attempt to predict $\langle x, r \rangle$ by computing $h'((x, P(x)))$.

To see why this procedures works, we note that with probability of $1/2^m = 1/poly(n)$, we picked values for $b_1, ..., b_m$ such that $b_i = \langle r_i, \alpha \rangle$ for all i. If this event happened, then the training set we get is distributed like m *i.i.d.* examples from D_m. By our assumption on A, and the fact that $\inf_h err(h) = 0$, it follows that with probability at least $1/poly(n)$, A will return a hypothesis which predicts correctly with probability at least $1/2 + 1/poly(n)$ as required. ■

Lemma 19.3 *Let $D_\alpha \equiv \{\underbrace{(r_i, P(\alpha))}_{x_i}, \underbrace{\langle r_i, alpha \rangle}_{y_i}\}$, for a test point $(r, P(\alpha))$, suppose $r \in span(\{r_1, ..., r_m\})$, which is $r = \sum_{i=1}^m c_i r_i$; $\langle r, \alpha \rangle = \sum_{i=1}^n c_i \langle r_i, \alpha \rangle = \sum_{i=1}^n c_i y_i$, then*

$$P(r \notin span((\{r_i\}_{i=1}^m))) \leq \frac{n}{m} \leq \epsilon$$

$$m \geq \frac{n}{\epsilon}$$

Proof: For a set of random variables $r_1, ..., r_m$, let B_i is indicator random variable that indicate whether $r_i \notin span(\{r_1, ..., r_{i-1}\})$, so $B_i \in \{0, 1\}$,

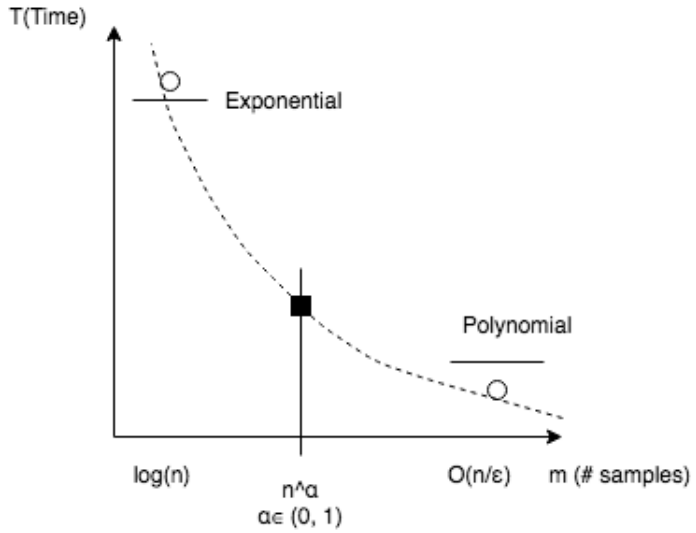
$$\sum_{i=1}^{m} B_i \leq n$$

$$n \geq \sum_{i=1}^{n} \underbrace{E[B_i]}_{p_i} \geq mP_m$$

where $1 = P_1 \geq P_2 ... \geq P_m$

$$P_m \geq \frac{n}{m}$$

B_i decreases with the increase of r_i.



In the setting of this classification problem, we still couldn't figure out the time needed given n^α samples.

We cannot bound the time token between $O(\log(n))$ and $O(\frac{n}{\epsilon})$ for sample size. ■

19.2 Sparse PCA

For *i.i.d.* samples of vectors X drawn from R^d with $E[X] = 0$, the first principle component is a direction v, such that

$$\arg \sup_{\substack{v:\|v\|_2=1 \\ v:\|v\|_0=k}} E[(v^T X)^2]$$

which means the variance along direction v is larger than in any other direction. If no such v exists, the distribution of X is said to be isotropic. The goal of sparse PCA is to test whether X follows an isotropic

distribution P_0 or a P_v, which exists a sparse v, where $v : \|v\|_0 = k < d$, along with the variance.

Without loss of generality, we define null hypothesis H_0 and H_1^θ:

$$H_0 \equiv \sup_{v:\|v\|_2=1} E[(v^T x)^2] = 1$$

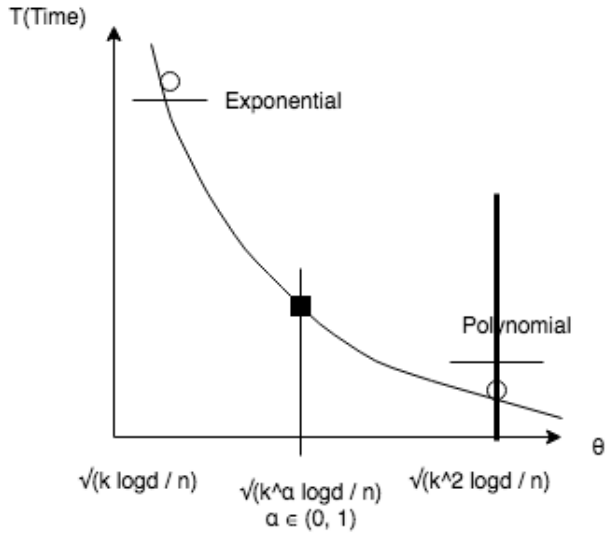
$$H_1^\theta \equiv \sup_{v:\|v\|_2=1 v:\|v\|_0=k} E[(v^T X)^2] = 1 + \theta$$

Under H_0, we assume that X is under the isotropic distribution, all directions have unit variance. Under H_1^θ, we assume the variance along v is equal to $1 + \theta$, where θ is much larger than 1. Note that since v has unit norm, θ captures the signal strength.

Similar to the previous GGM example, when θ is much larger than 1, it's easy to perform the hypothesis testing. From information theoretical limit, when

$$\theta \ll \sqrt{\frac{k \log d}{n}}$$

no algorithm can control the hypothesis testing error to 0 as n goes to infinity.



There exists no polynomial time algorithm for $\alpha < 2$

19.3 Planted Clique Problem

For random graphs, the number of vertices $|V| = m$ fix an integer $m \geq 2$ and let G_m denote the set of undirected graphs on m vertices. For all pairs of vertices, add edges with probability P. Denote by $\mathcal{G}(m, p = 1/2)$ the distribution over G_m generated by choosing to connect every part of vertices by an edge independently with probability 1/2. For any $\kappa \in 2, ..., m$, the distribution $\mathcal{G}(m, 1/2, \kappa)$ is constructed by two steps:

- pick a random subset of κ vertices and place a clique between them

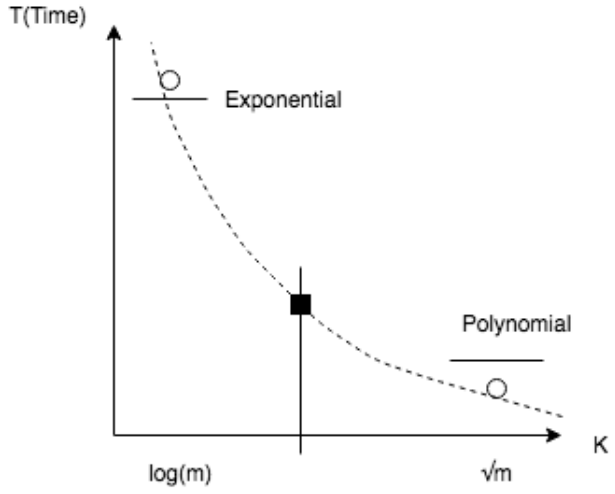
- for all remain pairs of vertices, add edges with probability $P = 1/2$

Theorem 19.4 *Fix $m \geq \kappa \geq 2$, let **Planted Clique** denote the following statistical hypothesis testing problem:*

$$H_0^{PC} : G \sim \mathcal{G}(m, 1/2) = P_0^{(G)}$$
$$[H_1^{PC} : G \sim \mathcal{G}(m, 1/2, \kappa) = P_1^{(G)}$$

A test for the planted clique problem is a family $\xi = \{\xi_{m,k}\}$, where $\xi_{m,k} : G_m \to \{0, 1\}$.

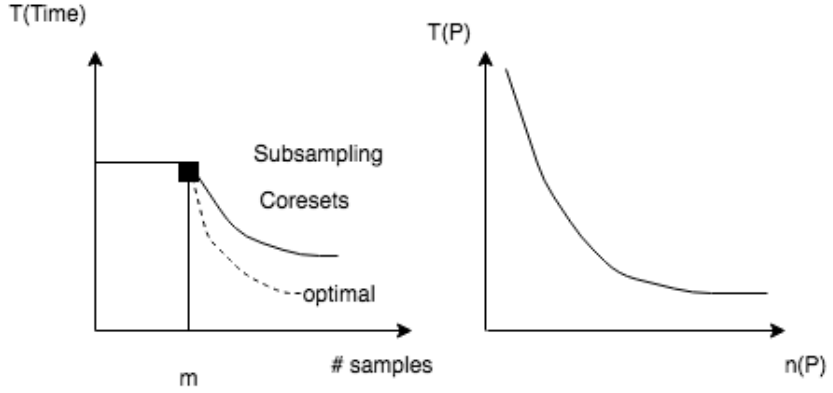
For κ that is extremely large, say $\kappa = n$, the problem is easy since there will be a point mass over a fully connected graph. For κ that is small, say, $\kappa = 2$, the first step is just to pick 2 vertices and add an edge. There's no way to distinguish between the two hypothesis.



Planted Clique Conjecture: $\nexists$ polynomial time algorithm with vanishing test error when $\kappa \leq O(\sqrt{m})$

19.4 Algorithmic Weakening

Question: is there a practical way to reduce computation complexity given more samples?



$$P \equiv ProblemSize$$
$$\epsilon^A(P) \equiv Error$$
$$T^A(P) \equiv Time$$
$$n^A(P) \equiv \#\text{ Samples used}$$

If we fix $\epsilon(P) = 0.1$, see if we could get a curve similar to the right side.

19.4.1 Denoising

Given the following, where X^* is the target signal

$$Y_i = X^* + \sigma Z_i$$
$$Z_i \sim N(0, I)$$

Our goal is to recover X^* given $\{Y_i\}_{i=1}^n$ and we have prior info that $X^* \in S \subseteq \mathbb{R}^d$. A natural estimator is that:

$$\min_{X \ inS} \sum_{i=1}^{n} ||X - Y_i||_2^2 \equiv \min_{X \in S} ||X - \frac{\sum_{i=1}^n Y_i}{n}||$$

References

[W] M. Wainwright, "High Dimensional Statistics," *Prerelease*, 2019

[CJY18] Yuansi Chen, Chi Jin, Bin Yu, "Stability and Convergence Trade-off of Iterative Optimization Algorithms," 2018

[SPR] Arun Sai Suggala, Adarsh Prasad, Pradeep Ravikumar, "Connecting Optimization and Regularization Paths," NeurIPS 2018