Statistics 291: Lecture 18 (March 28, 2024)

OGP for Inference (Sparse PCA)

Instructor: Mark Sellke

Scribe: Neil Shah

1 Final Project Reminders

The rough expectation for the final project is that there will be:

- a written report: approximately 8 pages if solo and 12 if duo,
- and a presentation: approximately 20 minutes if solo and 30 if duo.

The available presentation dates will be April 16, April 18, and April 23. At most four presentations will take place per day.

2 Defining the Problem

Say we have a sparse (Wigner) PCA with $x \in \{-1,0,1\}^n$, $\|x\|_1 = k << n$, and write $x \in T_k$. Observe that

$$Y = \frac{\lambda}{k} x x^\top + W$$

and $W \sim \mathrm{GOE}(n)$.

There are two natural goals that we might have here:

(a) Detection: Create a statistical test that distinguishes Y versus $W \sim \mathrm{GOE}$.

(b) Recovery: Devise an algorithm such that $\mathscr{A}(Y) = \pm x$ (the sign won't be distinguishable).

For this setup, think $k = n^\alpha$, $\alpha \in (0,1)$, $\lambda = n^{-\theta}$, and $\theta \in (0,1)$.

One example of a similar problem is community detection. In community detection, we have that

$$\text{Adjacency Matrix} = [\text{Low rank signal}] + [\text{Noise}],$$

as is the case in this problem. Planted cliques, sparse (Wishart) PCA, and Gaussian mixtures are also similar to this problem.

To solve questions like this, we want to know: what are some efficient approaches for sparse PCA?

- Top eigenvalue: $\lambda_1(Y)$, $\lambda_1(W) \approx 2$, and $\lambda_1(Y) > 2 + \delta$ when $\lambda > 1 + \epsilon$.

- Look at largest diagonal entries: $\frac{\lambda}{k}xx^\top$ shifts by $\frac{\lambda}{k}$. So, if

  $$\frac{\lambda}{k} \geq C\sqrt{\frac{\log n}{n}},$$

  then we can just take the k largest diagonal entries. For now, we're working with $\lambda \gg \frac{k}{\sqrt{n}}$.

It is generally believed that these are the only polynomial time algorithm for these problems (in the sense that if these algorithms fail, then no other polynomial time algorithm exists).

3 Analysis

Proposition 3.1. *Suppose $\lambda \geq c\sqrt{\frac{k\log n}{n}}$ for some constant c, $k \leq n^{1-\delta}$. Then, since we have maximum likelihood estimate (MLE)*

$$\hat{x}_{MLE} = \operatorname*{arg\,max}_{v\in T_k}\langle Y, v^{\otimes 2}\rangle,$$

we have, with high probability, that

(a) $\max_{v\in T_k}\langle Y, v^{\otimes 2}\rangle \geq \frac{\lambda k}{2}$.

(b) $\max_{v\in T_k}\langle W, v^{\otimes 2}\rangle \leq \frac{\lambda k}{3}$.

Hence, thresholding, max value suffices to distinguish Y versus W.

Proof. We'll start with part (a). Take $v = x$:

$$\langle Y, x^{\otimes 2}\rangle = \frac{\frac{\lambda}{k}\langle x, x\rangle^2}{\lambda k} + \frac{\langle W, x^{\otimes 2}\rangle}{\text{Normal}(0, \frac{k^2}{n})}.$$

We need $\frac{k}{\sqrt{n}} \ll \lambda k$, which would require that $\lambda \gg \frac{1}{\sqrt{n}}$. The theorem's assumption for λ then finishes this part.

Next, we address part (b). Start with $\langle W, v^{\otimes 2}\rangle \sim \text{Normal}(0, \frac{k^2}{n})$. Taking a union bound, we find that with high probability (since we know $|T_k| \sim n^k = \exp(k\log n)$),

$$\max_{v\in T_k}\langle W, v^{\otimes 2}\rangle \leq \frac{k}{\sqrt{n}}\cdot\sqrt{C\log|T_k|} \leq \frac{k^{\frac{3}{2}}\sqrt{\log n}}{\sqrt{n}} \ll \lambda k.$$

□

Lemma 3.2. *If $\lambda \gg \sqrt{\frac{k}{n}}$, then with high probability,*

$$|\langle \hat{x}_{MLE}, x\rangle| \geq \frac{k}{10}.$$

Proof. Using a union bound, with high probability,

$$\langle Y, v^{\otimes 2}\rangle < \frac{\lambda k}{2}.$$

for all v such that $|\langle v, x\rangle| \leq \frac{k}{10}$.

Then, note that $\#v \leq |T_k|$. We have

$$\langle Y, v^{\otimes 2}\rangle = \frac{\lambda}{k}\langle x, v\rangle^2 + \text{Normal}(\frac{k^2}{n}) \implies \max_{v\in T_k, |\langle v,x\rangle|\leq\frac{k}{10}}\langle Y, v^{\otimes 2}\rangle < \frac{\lambda k}{2}.$$

Naturally, one might boost by taking large coordinates of $Y\hat{x}_{\text{MLE}}$. However, this is hard to analyze because $\hat{x}_{\text{MLE}}$ already depends on Y. Instead, suppose $Y' = \frac{\lambda}{k}x^{\otimes 2} + W'$. We wishfully analyze $Y'\hat{x}_{\text{MLE}}$. Then,

$$Y'\hat{x}_{\text{MLE}} = \frac{\lambda x}{k}\langle x, \hat{x}_{\text{MLE}}\rangle + W'\hat{x}_{\text{MLE}}.$$

Note that $\langle x, \hat{x}_{\text{MLE}}\rangle$ is roughly order k by construction and the entries in the second term are roughly Normal$(0, \frac{k}{n})$. Then,

$$\text{Entry j} \approx \frac{\lambda x_j}{10} \pm O(\sqrt{\frac{k}{n}}.$$

So, if $\lambda \gg \sqrt{\frac{k}{n}}$, take the largest coordinates (with signs).

A trick we can use is to turn Y into 2 independent observations with slightly worse SNR. Take $\tilde{W} \sim \text{GOE}(n)$. Then, take

$$Y_1 = Y + \tilde{W} = \frac{\lambda x^{\otimes 2}}{k} + (W + \tilde{W})$$

and

$$Y_2 = Y - \tilde{W} = \frac{\lambda x^{\otimes 2}}{k} + (W - \tilde{W}).$$

Then,

$$\frac{W + \tilde{W}}{\sqrt{2}}, \frac{W - \tilde{W}}{\sqrt{2}} \overset{\text{i.i.d.}}{\sim} \text{GOE}(n).$$

So, we have the inefficient recovery algorithm as follows:

(a) Compute $\hat{x}_{\text{MLE}}(Y_1)$.

(b) Take large entries of $Y_2 \cdot \hat{x}_{\text{MLE}}$. □

It is natural to expect the MLE might be the fastest algorithm possible when polynomial time algorithms fail. The next result shows this is not true: one can instead use a modified MLE on a sparser set of coordinates. In particular it runs in time $e^{n^{o(1)}}$ when on the boundary of the polynomial time region, and degrades gracefully with problem parameters. a

Theorem 3.3 (Ding-Kunisky-Wein-Bandeira 19)**.** *In hard regimes, there is a recovery algorithm that uses time*

$$e^{\tilde{\Theta}\left(\frac{k^2}{n\lambda^2}\right)}.$$

The idea is to take $\frac{k^2}{\lambda^2 n} \ll k^1 \ll \min(k, \lambda^2 n)$. Then, compute $\hat{x}^{k'}_{\text{MLE}}(Y_1) \in T_{k'}$ and take large entries of $Y_2\hat{x}^{k'}_{\text{MLE}}$.

This paper gave this theorem as well as evidence of optimality. Based on low degree polynomials. The idea is that degree $\le D$ polynomial functions of the input serve as a proxy for algorithms with running time $n^{O(D)}$. Moreover there is a natural optimal hypothesis test within the linear space of degree $\le D$ polynomials, given by an orthogonal projection of the likelihood ratio between the null and alternative hypotheses, which can be analyzed explicitly.
Now, if we believe $\exp(\tilde{\Theta}(\frac{k^2}{n\lambda^2}))$ is the optimal running time how could we validate this belief via the landscape of $\hat{H}(v) = \langle Y, v^{\otimes 2}\rangle$?

One natural guess is that the Gibbs measure $\mu^{\hat{H}}_{\beta}$ with $\beta = \beta_* = \frac{\lambda n}{2k}$ (which gives the posterior distribution from a uniform signal prior) should exhibit free energy barriers. Thus consider the function

$$F_j = \log \sum_{v \in T_k, |\langle x, v\rangle| = j} \exp(\beta \hat{H}(v)). \tag{1}$$

In the "MLE works but polynomial time algorithms fail" regime, this function should be maximized for large values of $\langle x, v\rangle$ (unlike shattering!). If F_j were monotone, we might expect a simple Markov chain sampler such as Glauber dynamics to converge fast, although this would require much more to prove. On the other hand, suppose F_j is D-non-monotone, i.e. $F_j \le F_0 - D$ for some $D > 0$. Then we would expect such a Markov chain to mix in time at least e^D. In fact it is not hard to prove this from a *worst-case* initialization v with small inner product (though ideally one could understand a uniformly random initialization).

However we just saw that optimal algorithms require fiddling with the sparsity level. So the right conjecture should be that for **any** (β, k') in (1), there is a $\tilde{\Theta}(\frac{k^2}{n\lambda^2})$ non-monotonicity in F_j, and this might be minimized at the value k' used in the faster algorithm above. This is essentially what [Ben Arous-Wein-Zadik] were able to show!

Theorem 3.4 (Ben Arous-Wein-Zadik)**.** *(Roughly): for any β, k', there is a free energy barrier of depth $D \ge \frac{k^2}{n\lambda^2} \approx$ equality for $\beta = \beta_*$ Bayesian.*

While this can be considered an OGP result, it is in a different style from the ones we've seen previously. First, the problem setting is quite different: instead of optimizing a "pure noise" function, we are trying to recover a random signal. Second, it gives a fine-grained prediction for the running time of algorithms. On the other hand, the OGPs we saw before rigorously prove failure of all algorithms in a natural family. The result above implies slow mixing for certain Markov chains from a worst-case initialization, but doesn't prove failure of other natural algorithms.