Lecture 5: Filtering from spectral signatures

October 13, 2025

1 Recap: spectral signatures

Recall from last lecture we proved the following lemma:

Lemma 1.1 (Spectral signatures with bounded second moment)**.** *Let $\varepsilon < 1$. Let $S = S_{\text{good}} \cup S_{\text{bad}} \setminus S_r$ be of size n so that $|S_{\text{good}}| = n$, $|S_{\text{bad}}| = \varepsilon n$, and $|S_r| = \varepsilon n$. Let $\mu_g = \frac{1}{n}\sum_{i \in S_{\text{good}}} X_i$, and assume that*

$$\frac{1}{n}\sum_{i \in S_{\text{good}}} (X_i - \mu_g)(X_i - \mu_g)^\top \preceq 2I \ . \tag{1}$$

Then, for all $w \in \mathscr{W}_{S,\varepsilon}$, we have

$$\|\mu_g - \mu(w)\|_2 \le \frac{1}{1-\varepsilon}\left(2\sqrt{\varepsilon} + \sqrt{\varepsilon \, \|\Sigma(w)\|_2}\right) \ . \tag{2}$$

In this class, we will use this fact to develop a very simple randomized algorithm for robust mean estimation in this setting which achieves error $O(\sqrt{\varepsilon})$.

2 Univariate filtering

Before we give our full algorithm, we first require a simple primitive. At a high level, the spectral signatures we obtain will allow us to obtain univariate scores, assigned to each point. These scores will have the property that the outliers will in aggregate contribute much more to these scores than the uncorrupted points will. To effectively use this in our algorithm, we first give a method which is guaranteed (with high probability) to, given a set of scores with this property, to remove more mass from outliers than inliers. For any vector $v \in \mathbb{R}^n$, let $\text{nnz}(v)$ denote the number of nonzeros of v. Then, we have the following lemma:

Lemma 2.1. *Let $\tau \in \mathbb{R}^n$ be entrywise nonnegative, let $w \in \Gamma_{[n]}$, and assume $[n] = A \sqcup B$, so that*

$$\sum_{i \in A} w_i \tau_i < \sum_{i \in B} w_i \tau_i \ . \tag{3}$$

Then there is an algorithm 1DFilter *which takes τ and w, runs in time $O(n)$, and outputs $w' \le w$ so that:*

- $\text{nnz}(w') < \text{nnz}(w)$*, and*
- $\sum_{i \in A} w_i - w'_i < \sum_{i \in B} w_i - w'_i$.

The first condition should be thought of as a progress condition: it says that we are making progress, specifically, by decreasing the size of the support of the weight vector. The second condition is a safety condition: this says that we remove more mass from the bad points than from the good points, i.e., we are downweighting the bad points more aggressively.

The algorithm 1DFilter is very easy to describe. The update to w'_i is given by

$$w'_i = \left(1 - \frac{\tau_i}{\tau_{\max}}\right) w_i \ , \tag{4}$$

where $\tau_{\max} = \max_{i : w_i > 0} \tau_i$. We now prove correctness of this algorithm.

Proof of Lemma 2.1. The liveness condition follows immediately since the support of w' is clearly contained within the support of w, and moreover, $w'_i = 0$ for the i which attains $\tau_{\max}$. We now prove the safety condition. We directly compute:

$$\sum_{i \in A} w_i - w'_i = \frac{1}{\tau_{\max}} \sum_{i \in A} \tau_i w_i \ ,$$
$$\sum_{i \in B} w_i - w'_i = \frac{1}{\tau_{\max}} \sum_{i \in B} \tau_i w_i \ ,$$

from which the safety condition follows immediately from our assumption on τ_i and w. □

3 Robust mean estimation via filtering

We now have all the tools we need to describe our first polynomial-time algorithm for robust mean estimation that will achieve optimal rates. The algorithm is very simple to describe at this point: given a set of samples S, initialize $w^{(0)} = w(S)$ to be the uniform weights over S. Given $w^{(t)}$, form $\Sigma^{(t)} = \Sigma(w^{(t)})$. If $\left\|\Sigma^{(t)}\right\|_2 \leq C$ for some universal constant C ($C > 11$ suffices for this choice of parameters), then terminate and output $\mu(w^{(t)})$. Otherwise, let $v^{(t)}$ be the top eigenvector of $\Sigma^{(t)}$. Then, define scores

$$\tau_i^{(t)} = \left\langle v^{(t)}, X_i - \mu\left(w^{(t)}\right)\right\rangle^2 \ , \tag{5}$$

and let $w^{(t+1)} = 1\text{DFilter}(\tau^{(t)}, w^{(t)})$, and repeat. The formal pseudocode for this algorithm is given in Algorithm 1. Our main technical result for this algorithm is:

Algorithm 1 Filter

procedure Filter(S)
  Let $w^{(0)} = w(S)$.
  Let $C \geq 11$ be a universal constant.
  Let $\Sigma^{(0)} = \Sigma(w^{(0)})$.
  Let $t = 0$.
  while $\left\|\Sigma^{(t)}\right\|_2 > C$ **do**
    Let $\tau^{(t)}$ be as defined in (5).
    Let $w^{(t+1)} = 1\text{DFilter}(\tau^{(t)}, w^{(t)})$ and $\Sigma^{(t+1)} = \Sigma(w^{(t)})$.
    Let $t \leftarrow t + 1$.
  end while
end procedure

Theorem 3.1. *Let S, μ_g be as in Lemma 1.1, and suppose additionally that $\varepsilon \leq 0.1$. Then* Filter(S) *terminates after at most $2\varepsilon n + 1$ iterations, and outputs $\widehat{\mu}$ so that*

$$\|\widehat{\mu} - \mu_g\|_2 \lesssim \sqrt{\varepsilon} \ .$$

We will prove Theorem 3.1 by establishing the following invariant: for all iterations t before termination, we have:

$$\sum_{i \in S_{\text{good}} \cap S} \frac{1}{n} - w_i^{(t)} < \sum_{i \in S_{\text{bad}}} \frac{1}{n} - w_i^{(t)} \ . \tag{6}$$

Specifically, we will show:

Lemma 3.2. *Let $s \geq 0$, and suppose that in iteration $t = s$, we have that* (6) *is satisfied and furthermore $\Sigma(w^{(t)}) > C$. Then* (6) *is satisfied at iteration $t = s + 1$.*

The proof of Lemma 3.2 will be most of the technical work, but first let us see how Lemma 3.2 implies Theorem 3.1. We first observe:

Lemma 3.3. *Suppose $w^{(t)}$ satisfies* (6)*. Then $w^{(t)} \in \mathscr{W}_{S,2\varepsilon}$.*

Proof. Observe that $\sum_{i \in S_{\text{bad}}} \frac{1}{n} - w_i^{(t)} \leq \varepsilon$ as $w^{(t)}$ is entrywise nonnegative. Then (6) immediately implies the desired conclusion. □

Proof of Theorem 3.1 given Lemma 3.2. By the properties of 1DFilter, the support of $w^{(t)}$ decreases by one in every iteration. Suppose we ran for $2\varepsilon n + 1$ iterations. Then we must have removed at least εn points from $S_{\text{good}} \cap S$ from the support of w. However, then we will have decreased $\sum_{i \in S_{\text{good}} \cap S} \frac{1}{n} - w_i^{(t)}$ by at least ε, which by the invariant implies that $\sum_{i \in S_{\text{bad}}} w_i = 0$, and in particular, we must terminate at the next iteration, as we cannot decrease anymore.

It suffices to prove correctness. By Lemma 3.2, we know that whenever we terminate, (6) is satisfied. Then Lemma 1.1 and Lemma 3.3 together directly imply that $\|\widehat{\mu} - \mu_g\|_2 \lesssim \sqrt{\varepsilon}$, as claimed. □

We now turn to the proof of the main lemma.

Proof of Lemma 3.2. We will show that

$$\sum_{i \in S_{\text{good}} \cap S} w_i^{(s)} - w_i^{(s+1)} < \sum_{i \in S_{\text{bad}}} w_i^{(s)} - w_i^{(s+1)} \, .$$

By our inductive hypothesis, this will immediately imply (6) holds at iteration $t = s+1$. To simplify notation slightly, and since we will only deal with iterations s and $s+1$, let us drop the superscripts and denote $w^{(s+1)}$ by w'. By Lemma 2.1, to demonstrate the above inequality, it suffices to demonstrate that

$$\sum_{i \in S_{\text{good}} \cap S} w_i \tau_i < \sum_{i \in S_{\text{bad}}} w_i \tau_i \, . \tag{7}$$

We first observe that

$$\sum_{i \in S} w_i \tau_i = \sum_{i \in S} w_i \left\langle v, X_i - \mu(w) \right\rangle^2 = v^\top \Sigma(w) v = \|\Sigma(w)\|_2 \geq C \, , \tag{8}$$

where we have used that v is by definition the top eigenvector of $\Sigma(w)$. To establish (7), it therefore suffices to show that

$$\sum_{i \in S_{\text{good}} \cap S} w_i \tau_i < \frac{\|\Sigma(w)\|_2}{2} \, . \tag{9}$$

Recall we take the convention that $w_i = 0$ for all $i \in S_{\text{good}} \setminus S$. We then have:

$$\begin{aligned} \sum_{i \in S_{\text{good}} \cap S} w_i \tau_i &= \sum_{i \in S_{\text{good}} \cap S} w_i \left\langle v, X_i - \mu(w) \right\rangle^2 \\ &\leq \sum_{i \in S_{\text{good}}} \frac{1}{n} \left\langle v, X_i - \mu(w) \right\rangle^2 \, , \end{aligned}$$

since the summands are nonnegative and $w_i \leq \frac{1}{n}$ for all i. We now proceed as follows:

$$\begin{aligned} \sum_{i \in S_{\text{good}}} \frac{1}{n} \left\langle v, X_i - \mu(w) \right\rangle^2 &= \sum_{i \in S_{\text{good}}} \frac{1}{n} \left\langle v, X_i - \mu_g \right\rangle^2 + \|\mu_g - \mu(w)\|_2^2 \\ &\leq 2 + \frac{1}{(1-2\varepsilon)^2} \left(2\sqrt{2\varepsilon} + \sqrt{2\varepsilon \|\Sigma(w)\|_2} \right)^2 && (10) \\ &= 2 + \frac{1}{(1-2\varepsilon)^2} \left(8\varepsilon + 8\varepsilon \sqrt{\|\Sigma(w)\|_2} + 2\varepsilon \|\Sigma(w)\|_2 \right) && (11) \\ &\leq 2 + \frac{1}{0.64} \left(0.8 + 0.8 \sqrt{\|\Sigma(w)\|_2} + 0.2 \|\Sigma(w)\|_2 \right) && (12) \end{aligned}$$

where in (11) we have invoked Lemma 1.1 since $w \in \mathscr{W}_{S,2\varepsilon}$ by Lemma 3.3. We are actually done: one can verify that the quadratic inequality

$$2 + \frac{1}{0.64}\left(0.8 + 0.8x + 0.2x^2\right) \le \frac{x^2}{2}$$

holds for all $x > 3.2$, which immediately implies (11) under our choice of parameters. □

3.1 Breakdown point

An interesting question is for how large of an ε does this proof yield any meaningful guarantees. If we stare at the proof of Lemma 3.2, we will observe that the main barrier is in (11). Specifically, the highest order term in this expression is $2\varepsilon \left\|\Sigma\right\|_2 / (1-2\varepsilon)^2$, and we need ε to be small enough so that this is less than $\left\|\Sigma\right\|_2 / 2$. All other terms can be made negligible by taking C to be sufficiently large. Solving for ε yields that this proof requires that

$$\varepsilon \le \frac{4 - \sqrt{12}}{4} \approx 0.134 \, .$$

As you'll show in the homework, by optimizing the proof further, you can improve this to $\varepsilon \approx 1.14$. An interesting open question is whether or not one can go beyond this ratio with efficient algorithms.

3.2 Optimizing the runtime

As described, the algorithm is clearly polynomial time, as each iteration requires only taking a top eigenvector, and we run for at most n iterations. Naively, exact computation of the top eigenvector requires time $O(d^\omega + nd^2)$ where $\omega < 2.373$ is the matrix multiplication constant, so this yields a runtime of $O((\varepsilon n + 1)(d^\omega + nd^2))$ overall. We can already improve this runtime dramatically by using two observations.

Approximate eigenvector computation First, observe that we only require a relatively crude multiplicative approximation to the top eigenvector. Specifically, if we are willing to tolerate some small loss in C, and how large ε can be, it suffices for the proof of Lemma 3.2 to find any unit vector $u \in \mathbb{R}^d$ so that

$$u^\top \Sigma(w) u \ge (1-\delta) \left\|\Sigma(w)\right\|_2 \, , \tag{13}$$

for any $\delta < 1$. We leave the details of the proof of Lemma 3.2 with this approximate eigenvector to the reader, however, it is relatively straightforwardly checked. Luckily, finding a unit vector satisfying (13) can be done much faster than a full SVD:

Theorem 3.4. *Let $\delta \in (0,1)$, and let $Z_1, \ldots, Z_n \in \mathbb{R}^d$ be arbitrary. There is a randomized algorithm* PowerMethod$(Z_1, \ldots, Z_n)$ *which runs in time $O(nd \log d/\delta)$ and with probability at least $1 - \exp(-\Omega(d))$ outputs a unit vector $u \in \mathbb{R}^d$ satisfying*

$$u^\top \left(\sum_{i=1}^n Z_i Z_i^\top \right) u \ge (1-\delta) \left\| \sum_{i=1}^n Z_i Z_i^\top \right\|_2 \, .$$

Using this method to compute an approximate top eigenvector of our $\Sigma(w)$ immediately decreases our per iteration runtime complexity to $O(nd \log d/\delta)$. If $\delta = \Omega(1)$, this runtime is now just $O(nd \log d)$. This then yields a total runtime of at most $O((\varepsilon n + 1) nd \log d)$. However, we can improve this further:

Naive truncation As currently described, it is not hard to come up with problem instances where indeed the filter will run for $2\varepsilon n + 1$ iterations. A careful look at this bad case yields that the main bottleneck to making more progress at every iteration is when $\tau_{\max}$ is very large at every iteration.

To improve this runtime, one can combine this filtering routine with a simple truncation step. As you'll show in the HW, there are simple techniques which throw away at most $\leq \frac{1}{10}\varepsilon n$ points from S_{good} with high probability, and which guarantees that the remaining set of points satisfies that $\|X_i - \mu_g\|_2^2 \lesssim d/\varepsilon$ for all $i \in S_{\text{good}}$. If we do this procedure before we run FILTER, we can guarantee that $\tau_{\max} \lesssim d/\varepsilon$ for all iterations. As a result, the weight that we remove in iteration t, assuming that $\left\|\Sigma(w^{(t)})\right\|_2 > C$ can be lower bounded by

$$\sum_{i \in S_{\text{good}}} w_i^{(t)} - w_i^{(t+1)} = \frac{1}{\tau_{\max}} \left\|\Sigma(w^{(t)})\right\|_2 \gtrsim \frac{\varepsilon}{d} .$$

This implies that after T iterations, the total mass removed is at least

$$\sum_{i \in S_{\text{good}}} \frac{1}{n} - w_i^{(t+1)} \gtrsim \frac{\varepsilon T}{d} .$$

The safety condition still guarantees that the total mass that can be removed is at most $O(\varepsilon)$, and so this implies that we will terminate after at most $O(d)$ iterations. By combining this with a linear-time truncation algorithm (which you'll give in the HW), this yields an overall runtime of $\widetilde{O}(nd^2)$ for the full algorithm.

Further remarks We make a number of remarks about this final runtime. First, notice that when $n = \Theta(d/\varepsilon)$, this trick doesn't actually buy us anything in the iteration count, as $\varepsilon n = \Theta(d)$. Indeed, if we are happy with losing in the constants we obtain in the accuracy and in logarithms in the runtime, we can avoid this additional step by simply subsampling my samples down to $\widetilde{O}(d/\varepsilon)$ samples, and running the filter without truncation, since this number of samples suffices to get error $O(\sqrt{\varepsilon})$.

Secondly, it is not hard to see that $d^{O(1)}$ iterations is unavoidable for the filter presented here, even with additional preprocessing. At a high level, this is because the algorithm only looks in one direction at a time, namely, the top eigenvector of the current covariance. If the adversary then puts the points in many orthogonal directions, the filter will have to look in each individual direction to remove the points in that direction. In a future lecture, we'll see how to use ideas from fast semi-definite programming to improve this runtime.

References