NYU Mini-Theory Symposium Scribe: Kiran Vodrahalli
NYU Courant 02/02/2018

Contents

1 Sublinear Time Low Rank Approximation of PSD Matrices (David Woodruff)

1.1 Introduction: General matrices

We will think of $A \in \mathbb{R}^{n \times d}$, n points in $\mathbb{R}^d$. Often, A can be approximated by a low-rank matrix: Typically, the high rank is because of the noise. There are many benefits to storing a low-rank matrix: It's easier to store, quicker to multiply, and makes the data more interpretable.

What is a good low-rank approximation? Well, truncated SVD gives a good solution. We write $A = U_k\Sigma_k V_k + E$, where E is noise. This is the optimal solution under any rotationally invariant norm. The issue is that computing this solution exactly is expensive. We want to get much faster algorithms for computing low-rank approximations.

One natural relaxation is the following: $\|A - A'\|_F \leq (1+\epsilon)\|A - A_k\|_F$. You can now allow for randomized approximate algorithms. With this relaxation, you can solve this problem much faster. A few years ago, we showed you can get this guarantee in time $\mathcal{O}(nnz(A) + (n+d)\text{poly}(k/\epsilon))$. The current state of the art is k^2/ϵ for the polynomial term. The proof is based on using sparse JL, I won't get into that here.

So that's low-rank approximation for general matrices. For general matrices A, there is a $nnz(A)$ time lower bound for relative error approximation - you do need to read all non-zero entries. This lower bound also holds to estimate the norm of this matrix (i.e., consider putting ∞ in one value).

1.2 PSD matrices: Approximate norm

In this talk, we'll consider restricted classes of matrices for which we can do better than this lower bound. In particular, we consider PSD matrices. Lots of matrices are PSD. Now we might ask, is there a $nnz(A)$ lower bound for low rank approximation? For the norm? Well, what makes PSD matrices easier to handle? The diagonal has a lot of information. We can write

$$\|A\|_F^2 = \|BB^T\|_F^2 = \sum_{i,j} \langle B_i, B_j \rangle^2$$

where $A = BB^T$. We have this diagonal dominance property: By Cauchy-Schwarz, $\langle B_i, B_j \rangle^2 \leq \|B_i\|_2^2 \|B_j\|_2^2$, where the RHS are the diagonal entries of the matrix. Imagining that all diagonals are 1, then Cauchy-Schwarz says that all off-diagonal elements are at most 1 in absolute value. Now, the off-diagonal entries are at least ϵ times the diagonal. Then, $\sum_{i=j} \langle B_i, B_j \rangle^2 \geq \epsilon n$. In this case, uniform sampling $n \cdot \text{poly}(1/\epsilon^2)$ off-diagonal entries allows us to estimate the off-diagonal contribution to the norm. We have at least ϵn mass off-diagonal with 1s on the diagonal. Thus, one can sample a very sublinear number of entries. More generally, you just use importance sampling to estimate the norm. When, $\|B_i\|_2^2 \neq 1$ for all i, just sample offdiagonal entry B_{ij} with probability $p_{i,j} = \|B_i\|_2^2 \cdot \|B_j\|_2^2 / \|B\|_F^2$. Let $x = \langle B_i, B_j \rangle^2 / p_{i,j}$ if entry i, j is sampled. Then, $\mathbb{E}[x] = \|A\|_F^2$ and $\text{Var}(x) \leq n\|A\|_F^2$.

1.3 PSD matrices: low-rank approximation

Now what about low-rank approximations?

Theorem 1.1. *Given an $n \times n$ PSD matrix A, in $n \cdot k^2 \cdot \text{poly}(1/\epsilon)$ time we can output a factorization of a rank k matrix A' for which w.h.p.*

$$\|A - A'\|_F \leq (1 + \epsilon)\|A - A_k\|_F$$

The number of entries read is $n \cdot k \cdot \text{poly}(1/\epsilon)$. Also, any algorithm requires reading $\Omega(n \cdot k \cdot 1/\epsilon)$ entries.

In typical applications, you don't need the actual matrix: You just need the factors. You might want to multiply by a vector for instance. You usually just need to do matrix-vector multiplication. Just knowing the eigenvectors is interesting. This avoids the issue having to use the time to look at the whole $n \times n$ factorized matrix.

1.3.1 The Proof

How can we get this result?

Proof. We start with an *adaptive sampling* algorithm for general matrices. The way it works is as follows: We sample columns of A one at a time. It looks at the squared norms of all the columns in your matrix, and samples proportional to the squared norm. Now it looks at remaining columns, and the distance to the column you already sampled. It samples the next column according to that distance. And so on, sampling with respect to the span of the columns you've chosen so far. You need k^2/ϵ columns. This is a result by Vempala ($DV06$). Then, there is some k-dimensional subspace in the span, called V, such that if you projected columns onto that space ($\|A - P_V A\|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2$, you'd get a rank k matrix which has cost at most $1 + \epsilon$.

Now what is the knowledge you need to carry out the algorithm? In the beginning, you need to know the squared norm of each of the columns. For all the remaining columns, you need to know the distance to the span of columns already chosen. So you need pairwise distance information. You have potentially $\binom{n}{2}$ pairs, but how many distances does the algorithm actually use? Just $n \cdot k^2/\epsilon << n^2$ of these inner products. So a very sublinear amount of information here. Since A is PSD, we can write it $A = B^T B$: Thus, A is like a truth table of inner products which have been pre-computed for you! So why don't we run this adaptive sampling time in $n \cdot k^2/\epsilon$ time on B using A, to find P_V? Then by guarantee from algorithm before, we can find a good low-rank approximation to B using A. Since $B = \sqrt{A}$, setting $\epsilon = 1/\sqrt{n}$, then $B^T P_V B$ can be shown to be a $(1+\epsilon)$ approximation to A. This matrix can also be calculated only from A. This is giving an $n^{3/2} \cdot \text{poly}(k)$ algorithm. What's a good intuition for the $\epsilon = 1/\sqrt{n}$? You're relating the spectrum of B to the square root of A. You kind of need to translate the $1+\epsilon$ approximation of B to a $1+\epsilon$ approximation of A.

Then, we show how to compute sampling probabilities of columns and rows of A to reduce $A \to C \in \mathbb{R}^{n \times \sqrt{nk}}$, and then reduce that to a matrix $R \in \mathbb{R}^{\sqrt{nk} \times \sqrt{nk}}$. This takes time $\tilde{O}(nk)\text{poly}(1/\epsilon)$. The sampling probabilities are the "ridge leverage scores" of B where $A = B^T B$. These can be computed in time $\tilde{O}(nk)$ given A (see previous work [MM16]). Then, you can spend $nnz(R)$ time to find its top k principal components. We use those to sample k left-singular vectors of C, and those are the ones we use in the output of our low-rank approximation to A.

Thus, we get sublinear time algorithm for relative error of low rank approximation of PSD matrices, bypassing the $nnz(A)$ lower bound for general matrices. We get tight $\tilde{\Theta}(nk)$ bounds for constant ϵ. It turns out that spectral norm error is impossible to approximate in sublinear time, but can find a rank k A' with $\|A - A'\|_2^2 \leq (1+\epsilon)\|A - A_k\|_2^2 + \frac{\epsilon}{k}\|A - A_k\|_F^2$ in $n \cdot \text{poly}(k/\epsilon)$ time. We can output PSD rank$-k$ matrix A' in $n \cdot \text{poly}(k/\epsilon)$ time.

□

This work kind of opens the possibility for looking at sublinear time algorithms for more classes of matrices that people might be interested in.

2 Talk 2: An Alon-Boppana Type Bound for Weighted Graphs and Lowerbounds for Spectral Sparsification (Luca Trevisan and Nikhil Srivastava)

2.1 What is the Alon-Boppana Theorem?

We want to find graphs which are good spectral expanders. Look at the adjacency matrix of undirected graph, it is real-valued. The top eigenvalue will be d for a d-regular graph. The smaller the range of the other eigenvalues, the better the graph is as an expander. So you can express the goodness of an expander in terms of the second eigenvalue. You can think of this as saying how well our graph approximates a clique.

The Alon-Boppana theorem says that if we have a d-regular graph, there is some limit to how good the expansion will be:

Theorem 2.1. *$\lambda_2 \geq 2\sqrt{d-1} - o(1)$ and $|\lambda_n| \geq 2\sqrt{d-1} - o(1)$. Sarnak and others proved that can get $\lambda_2, |\lambda_n| \leq 2\sqrt{d-1}$ for infinitely many n, d (Ramanujan graphs). Markus-Spielman-Srivastava proved that you can get $\lambda_2 \leq 2\sqrt{d-1}$ for all n, d. Cohen proved you can construct in polynomial time. It's still open for Ramanujan graphs.*

There is a lot of interest in seeing whether similar stories can be told about other classes of graphs, not just d-regular graphs. So we need to focus on some important parameter of the graph. We would like an Alon-Boppana type bound and a Ramanujan type construction showing such graphs actually exist.

2.2 Cut-sparsifiers

The other seemingly unrelated notion I want to introduce is a cut-sparsifier. Suppose I have some dense graph. A sparser graph is a good cut-sparsifier of the original graph if for every partition of the set of vertices in the original graph, the number of edges crossed in each partition of the original graph and the same quantity in the sparser graph is roughly the same. The idea is that the sparser graph mimicks the sizes of cuts. Thus, if you're interested in solving a cut-type algorithm in the dense graph, you can first calculate the sparser graph (note: this is a kind of "dimension reduction") and run the cut algorithm on the sparser graph.

Definition 2.2. Cut-sparsifier.

$$1 \leq \frac{E_H(S, V-S)}{E_G(S, V-S)} \leq 1 + \epsilon$$

Another way (stronger) to define is as follows:

Definition 2.3. Spectral sparsifier.

$$1 \leq \frac{\sum_{u,v} w_H(u,v)(x_u - x_v)^2}{\sum_{u,v} w_G(u,v)(x_u - x_v)^2} \leq 1 + \epsilon$$

Spielman-Srivastava proved its possible to construct sparsifiers in the second sense (implying the first kind as well) such that every graph has a spectral sparsifier of average degree d and error

$$\epsilon \leq \frac{4\sqrt{2} + o(1)}{\sqrt{d}}$$

and every sparsifier of a clique of average degree d has error

$$\epsilon \geq \frac{2 - o(1)}{\sqrt{d}}$$

Also, every *cut sparsifier* of clique average degree d has error

$$\epsilon \geq \frac{\Omega(1)}{\sqrt{d}}$$

It is thus not clear what is the optimal constant: We want to close these gaps.

A way of thinking about spectral expansion is saying a graph is a good spectral-expander if it is a good *spectral sparsifier of the clique.* Thus,

Theorem 2.4. *An unweighted regular graph is a spectral sparsifier of the clique with error ϵ iff it is a spectral expander with $\frac{1+\lambda_2}{1+\lambda_n} = 1 + \epsilon$. So every spectral sparsification of the clique for any unweighted regular graph is*

$$\epsilon \geq \frac{4 - o(1)}{\sqrt{d}}$$

Can we do better with graphs which are weighted and not regular?

Let H be a graph, then $L_H = D_H - A_H$ be the Laplacian matrix and $0 = \lambda_1(L) \leq \lambda_2(L) \leq \cdots \leq \lambda_n(L)$. Then, H is a spectral sparsifier of the clique with error ϵ iff $\frac{\lambda_n(L)}{\lambda_2(L)} \geq \epsilon$. Our result is as follows:

Theorem 2.5. *If a graph H is of average degree d, maximum degree $n^{o(1)}$ and girth $\omega(1)$, we have that*

$$\epsilon \geq \frac{4 - o(1)}{\sqrt{d}}$$

We also discovered a bottleneck in improving the upper bound, which is more of a technical issue. The Srivastava result is a corollary of an even more general result. Suppose we have a sum of rank 1 matrix, after a change of basis we can say it's the identity. We want to find

$$I \succeq \sum_e h_e v_e v_e^T \succeq (1 + \epsilon) \cdot I$$

and if we allow $dn/2$ non-zero coefficients, we can have $\epsilon \leq \frac{4\sqrt{2}+o(1)}{\sqrt{d}}$. Furthermore, the algorithm choosing the coefficients is "online". Basically, you pick one term of the sum at a time. Our result shows that $\frac{4\sqrt{2}+o(1)}{\sqrt{d}}$ is the best possible in this setup.

So, if the right constant is $4/\sqrt{d}$, then we need to get away from the sum of matrices which adapt to the identity, or we need to get away from the online setting.

2.3 Alon-Boppana Proof with Large Girth

The proof is much easier if you have large girth. We need to prove a lower bound on λ_2. We can apply the variational definition of eigenvalues to say that we need to come up with some vector x such that the variational ratio is as small as possible. We can construct x by assigning a number to every vertex. We pick an arbitrary start vertex, assign its value as 1. We give the labels of that vertex label $1/\sqrt{d-1}$, and its neighbors $1/d-1$, and so on, keeping dividing by $\sqrt{d-1}$. We do this k times where k is smaller than the girth of the graph. What can we say about x? We have $\|x\|_2^2 \leq k+2$. Each time you square, you have $1/(d-1)$ times the number of vertices, but the weight is $1/(d-1)$ smaller. So, each layer is giving 1. Then, we have

$$\lambda_2 = \max_x \frac{x^T A x}{\|x\|^2}$$

Note that $x^T A x = 2k\sqrt{d-1}+2$. So each of the k layers is giving $2\sqrt{d-1}$. Now we want to do something similar for graphs which are weighted, and for which degree is not regular.

We can interpret this construction in a similar way. Choose randomly root vertex r. Then if a vertex was distance l for the root, then the weight we gave to that vertex v was

$$\sqrt{\mathbb{P}\left\{r \to v \text{ in l-step nonbacktracking random walk}\right\}}$$

Now we have a quantity that we can define for arbitrary weighted non-regular graphs. Then, by convexity, the worst that can happen is that the degree has a degree to average degree and all the weights are the same. So the regular case is the worst case.

2.4 Open problems

- Remove the large girth assumption.
- Tighter bounds (possibly better upper bound is needed)
- Are there different constants for best spectral sparsifier versus best cut sparsifier of the clique? For instance, is the constant strictly less than 4 for cut sparsifiers?
- Only loose bounds are known for cut sparsifiers of *hypergraphs*. We don't even understand the right dependence on n.

2.5 Sparsification as a Data Structure

We can think of cut-sparsifiers as data structures, or compressions of the original graphs (i.e. dimension reduction view). You can think of this which takes in $\Omega(\epsilon^{-2}n)$ bits which can answer queries of the form "how many edges cross" which give ϵ error. So you can think of it as a sketching problem now. It is known as of very recently that you can, given a graph, come up with a data structure of size n/ϵ which will answer cut queries with multiplicative error $1+\epsilon$, but only polynomially many times.

Suppose you want a data structure that can answer all 2^n possible queries: Then Andoni et. al. proved that $\Omega(\epsilon^{-2} n \log n)$ bits are necessary. This proof used communication complexity. We show that this is the right answer: We find $\exp(\Omega(\epsilon^{-2} n \log n))$ graphs that are not 3ϵ-cut sparsifiers of one another. This means that you can't use the same data structure for all these graphs. Consider all d-regular graphs where $d = 1/(100\epsilon^2)$. There are about $N = \exp(\frac{1}{50\epsilon^2} n \log n)$ such graphs. Each can be a 3ϵ-cut sparsifier for $<< \sqrt{N}$ graphs.

3 Talk 3: A General Characterization of Statistical Query Complexity (Vitaly Feldman)

3.1 Introduction: Motivation and the result

Suppose you would like to solve a statistical problem. You think of your data as being drawn i.i.d. from some unknown distribution D. Say you can't find an efficient algorithm, so you believe this is a computationally hard problem. You'd like a lower bound for this problem. It is difficult to prove meaningful lower bounds on computation. Instead, you do things by reduction: reduce to a known "hard" problem. In statistics, it is harder to find problems to reduce to.

Here we will look at a different approach specific to statistical models. Consider the statistical query oracle. Instead of access to distribution via samples: We will be able to ask it questions about the distribution D. Many of the most powerful techniques in ML can be expressed with this model. The only things we cannot solve in this model are things that require solving linear equations over a finite field, and those are not things you'd use to solve in a statistical problem anyways.

Theorem 3.1. *For any problem its SQ (statistical query) complexity can be nearly tightly characterized by a "simple" dimension. Complexity is parametrized by a parameter of the oracle, and is the number of queries you need to ask.*

This is relatively simple, doesn't involve interaction or computation, and involves a bit of simple algebra and a couple of quantifiers. So you'll have a parameter you can analyze and give evidence of computational hardness without having to prove $Pvs.NP$. It has been influential and useful in learning theory, the goal is to extend the framework to all statistical problems.

3.2 Statistical problems

You get set of n samples drawn i.i.d. from distribution D. Here, we look at $STAT_D(\tau)$, the oracle. A query is described as $\phi_1 : X \to [-1, 1]$. The oracle will provide an answer $|v_1 - \mathbb{E}_{x\sim D}[\phi_1(x)]| \le \tau$. τ is the tolerance of the query, think of it as $\tau = 1/\sqrt{n}$. For instance, you can ask for the indicator of an event, you'll get an estimate of the probability of the event. For a loss function, you'll get an estimate of the true loss of the predictor. Note that this is *less powerful* than getting the actual samples information theoretically, though it might depend on the amount of computational power you get. Also note that you repeat many queries and responses. These have been known as counting queries, linear queries, linear statistical functional estimator, etc. People care about this model for applications: They come from the idea that once you abstract data to some oracle and implement some algorithm with this oracle model, then you can implement it in many other contexts while satisfying additional constraints. The model was originally defined to get noise-tolerant algorithms from regular algorithms. You can also use this to get differential privacy (local differential privacy). You can also get algorithms in distributed, low communication, and streaming settings. It has also been used to understand the framework of adaptive data analysis.

Definition 3.2. SQ complexity.
For a problem Z, the SQ complexity with oracle STAT(τ) $\min q$, there exists a randomized SQ algorithm that solves Z using q queries with *any* $STAT_D(\tau)$ oracle. We have

$$SQC(Z) : \min q \text{ s.t. } SQC(Z, STAT(1/q)) \leq q$$

This is a proxy for computational complexity of Z given $1/\tau^2$ samples. For many known Z solvable in time T using n samples we have

$$SQC(Z, STAT(1/poly(n))) = poly(T)$$

It is possible to analyze and prove lower bounds for this notion.

This idea was first investigated in Blum et al 94. Here they gave the SQ dimension for weak PAC learning of class C relative to distribution P. D is $(v, c(v))$ where $v \sim P$ and $c \in C$ is unknown. The goal is to output f such that $\mathbb{P}\{f(v) \neq c(v)\} = 1/2 - \gamma$. The Blum et al result gives that $SQDim(C, P, \gamma)$ is the max d for which exist $f_1, \cdots, f_d \in C$ s.t. forall $i \neq j$,

$$|\mathbb{E}_{v \in P}[f_i(v) \cdot f_j(v)]| \leq \gamma$$

Parity functions are uncorrelated relative to U over hypercube, these give lower bounds for SQ learning DNFs, decision trees, juntas over U.

Extending these ideas to strong PAC learning is also possible if you know the distribution. For a while it wasn't known if it would be possible to find a parameter characterizing distribution. It was also not known whether the worst case distribution is equivalent to every distribution.

3.3 Main result

Let $Z = (D, F, R)$ where D is the possible set of input distributions, F is a set of solutions (outputs of the algorithm), and R is a relation on $D \times F$ defining valid outputs. For input distribution $d \in D$, the output should be in $\{f : (d, f) \in R\}$. This general setting was first considered with several co-authors. We showed techniques to prove lower bounds for this general setting. None of the parameters we defined were strong enough though. Is there a parameter which is stronger?

Theorem 3.3. *Describe $SQdim(Z, \tau)$:*

$$SQC(Z, STAT(\tau)) \geq SQDim(Z, \tau)$$

$$SQC(Z, STAT(\tau)) \leq \frac{SQDim(Z, \tau) \cdot R_{KL}(D)}{\tau^2}$$

Here, we have $\inf_{d_0} \sup_{d \in D} KL(d \| d_0) \leq \tau$ as our parameter.

For one-vs-many decision problems, the goal is given some distibution D to decide which of many potential distributions is the correct one (think property testing and so on). Every query function is a measurement on the set of points of distributions (projection effectively). Given that we know the values within τ, we can distinguish points from d_0: so we can eliminate a fraction of the distribution.