COS 598C: Detecting overlapping communities, and theoretical frameworks for learning deep nets and dictionaries

Lecturer: Sanjeev Arora
Scribe: Max Simchowitz

April 8, 2015

Today we present some ideas for provable learning of deep nets and dictionaries, two important (and related) models. The common thread is a simple algorithm for detecting overlapping communities in networks. While community detection is typically thought of as a way to discover structure in, say, large social networks, here we use as a general purpose algorithmic tool to understand structure of latent variable models. The algorithm for learning deep nets and dictionaries starts by identifying correlations among variables, and represent these pairwise correlations using a graph. Then it uses community-finding to uncover the underlying connection structure.

1 Detecting Overlapping Communities in Networks

Community detection has been well studied in planted settings where the communities are disjoint. We discussed the *stochastic block model* in an earlier lecture. The concrete setting was that we are given $G = (V, E)$, where the vertices of V are partitioned into two sets S and S^c, and edges within S and S^c are drawn with probability p, and between S and S^c are drawn with probability q, such that $p - q = \Omega(1)$. Then, as long as $\min(S, S^c) = \Omega(\sqrt{|V|})$, we can easily recover S and S^c using an SVD or semi-definite programming [6].

However, when communities overlap, this problem does not seem doable via SVD. To recap the notation, let $G = (V, E)$ be our graph, and lets assign users to (perhaps more than one) communities $C_1, \ldots, C_m$. In the simplest setting - such as the one that arises in the dictionary learning problem - $(v_1, v_2) \in E$ if and only if there is a community C_j such that $v_1 \in C_j$ and $v_2 \in C_j$. In this case, we can identity C_j are subgraphs of G, all of which are cliques, and G is precisely the union of these cliques. I don't know of an algorithm to find the cliques given the graph, if the graph is a union of arbitrary cliques.

Luckily, in the dictionary learning setting, the structure of G is not determined adversarially. Instead, we assume that the vertices $v \in V$ are distributed across the communities fairly evenly, and that each v doesnt belong to too many communities (say, there are no hubs). We formalize the generative process for G as follows:

Definition 1.1 (Planted Problem Corresponding to Overlaping Communities)**.** Let $G = (V, E)$, where $|V| = N$. Suppose that there are m communities $C_1, \ldots, C_m$, and each vertex is assigned to k communities uniformly at random. Finally, if u, v belong to the same community, then $Pr[(u, v) \text{ is an edge }] = p \geq 0.9$. If they do not have any overlapping community, then there are no edges.

Remark. If $p = 1$, then $C_1, \ldots, C_m$ are cliques, and we are back in the clustering setting for dictionary learning.

The nice thing about our generative process is that it admits for a local search heuristic, as described in [2]. First, set $T = kN/m$, which is roughly the expected size of each community. Now, if (u, v) are in a community C_i, then the expected number of edges shared between them is about pT, so by a Chernoff bound, there are at least $.9pT$ vertices w connected to u and v with high probability.

On the otherhand, suppose (u, v) are not in the same community. Then, while they are not necessarily joined by an edge, there may be vertices w such that (u, w) are in one community, say C_i, and (u, v) are in another community, say C_j, giving rise to edges from both u and v to w.

Now, how many such spurious edges are there? That, is, what is the probability that edges occur between any two vertices, neglecting shared community structure? Well, there are $\binom{N}{2}$ ways to pick pairs of vertices, and the number of edges in the graph is no more than the sum of the number of edges in one community, which concentrates around $p\binom{T}{2}$ using a standard Chernoff argument. Taking the union over all m communities, we see that the probability of an edge between two vertices is no more than

$$p_0 := pm \binom{T}{2} \binom{N}{2}^{-1} \tag{1.1}$$

Hence, the probability of a spurious edge between w and u and w and v is about p_0^2, and thus the number of spurious edges between (u, w) concentrats around $p_0^2 N$. Hence, to distinguish between u and v sharing only spurious edges and sharing non-trivial edges due to common community memberships, we want to ensure

$$p_0^2 N \ll pT \tag{1.2}$$

This amounts to imposing the requirement that

$$\frac{N \cdot T^4 m^2}{N^4} \ll T \iff m \ll (N/T)^{3/2} \iff m \ll (m/k)^{3/2} \tag{1.3}$$

that most of the edges between u and v will be because they are in the same community. Hence, we can greedily assign vertices to communities by considering the number of common edges.

1.1 Notation

Given a vector $x \in \mathbb{R}^n$, we will denote its i-th entry by $x(i)$. We will denote the inner product between two vectors $x, y \in \mathbb{R}^n$ by $\langle x, y \rangle$, or $x^T y$ interchangably. Given a matrix $A \in \mathbb{R}^{n \times m}$, we denote its i-th column by A_i.

2 Neural Netowrks

Before expounding on the applications of the community finding algorithms described in [2], we will take a brief detour into the world of neural networks: perhaps one of the most popular tools in contemporary machine learning. At a very basic expert, neural networks mimic the structure of physical brains. One abstraction for emulating a brain is to view a network of biological neurons as large graphs, whose vertices are neurons and whose edges are synapses (or other forms of connections). The state of such a neural network is described by the potential with which each neuron is activited (and by other factors like current), and the synapses determine how much potential is transferred from one neuron-node to the next.

Motivated by both common implementation practices and theoretical feasibility, we will consider study artificial neural networks which decompose in L-layers. We can therefore describe the state of this network at a given time by an L-tuple of vectors $x^{(1)}, \dots, x^{(L)}$, where the entries of the vector $x^{(l)} \in R^{N_l}$ record the potentials of a corresponding neuron in the l-th layer. For example $x^{(2)}(1)$ is the potential of the first neuron in layer two. We refer to $x^{(1)}$ as the top layer and $x^{(L)}$ as the bottom layer.

What makes neural networks fascinating is the way the potential vector $x^{(l)}$ in different layers relate to one another. In biological neural tissue, electrical potential and chemical signals are being exchanged continuously. In our setting, we instead imagine that, at discrete interviews $t = 1, \dots, T$, nature draws top layer - potential vectors $x_t^{(1)}$. Then, the potentials in each succesive layer $x^{(l)}$ is given by a noisy objection of a deterministic function of the potentials $x^{(l-1)}$ in layer $x_t^{(1)}$.

We model this transfer of potentials as

$$x^{(l+1)} = h(A^{(l)} x^{(l)}) \tag{2.4}$$

where h is an (often nonlinear) function which which operates entrywise and identically on each entry, and $A^{(l)} \in \mathbb{R}^{N^{(l)} \times N^{(l+1)}}$ is a matrix specifying how the potentials in one layer feed into the next. Equivalently, we can think of $A^{(l)}$ as the adjacency matrix of a bipartite graph $G^{(l)}$, whose edges represent the connection between neurals. In what follows, we will interchange between identifying the vertices of $G^{(l)}$ with the entries of $x^{(l)}$, both of which semantically correspond to the neurons in the l-th layer.

To lighten up the notation and facillitate exposition, the majority of these notes will focus on learning networks with only two layers: one encoded by a sparse vector x, hidden to the observer and drawn from a suitably behaved generative process, and a dense layer encoded by a dense vector y, which can be observed. We will use G and A to refer to the connection graph between x and y, and its adjacency matrix, respectively.

3 Dictionary Learning, Neural Nets, and Community Finding

We can imagine that even the two layer problem is rather difficult for arbitrary, nonlinear h. Thus, it makes sense to start off by considering the simpler case where h is just the identity; that is $Ax = y$. This problem is known as *Dictionary Learning*, and the adjacency matrix A is called the dictionary.

In the Dictionary Learning problem, we are given samples $y_1, \ldots, y_N$ samples of observed potentials, and our goal is to reconstruct both A, and the hidden samples $x_1, \ldots, x_N$ so as to minimize the error

$$\min_{A,\{x_i\}} \|Ax_i - y_i\|_2^2 \tag{3.5}$$

In general, this problem is extremeley over-determined. Indeed, if the x's have dimension greater than the y's, then it is trivial to reconstruct A and x_i for which $Ax_i = y_i$ exactly. In order to make the problem both meaningful and tractable, we need to posit that the x_i have some additional structure. Here, we will assume that the samples x are sparse.

There are two motivations for recovering sparse x_i. The first is empirical - biological neurons tend to show sparse activation patterns. More broadly, sparsity is a rather intuitive assumption for capturing a sense of "latent simplicity" or "hidden structure" in otherwise very high dimensional data. The second motivation is that, assuming sparsity, we can leverage insights from sparse recovery and compressed sensing, under certain conditions on the dictionary matrix A. Recall that matrices that have low column inner products are called incoherent:

Definition 3.1. Let A be a matrix with columns A_i, such that $\|A_i\| = 1$. We call A $\mu/\sqrt{n}$ incoherent if $|A_i^T A_j| \le \frac{\mu}{\sqrt{n}}$

Now, if we knew A exactly, and A is sufficiently incoherent, then we have the following result

Theorem 3.1 (Compressed Sensing, Stated Loosely)**.** *Let A be a matrix with unit norm columns, such that $|\langle A_i, A_j\rangle| \le \frac{1}{2k}$. Suppose given $y = Ax$ where x is k-sparse. Then, x is the unique k-sparse vector for which $y = Ax$. Hence, x can be recovered in polynomial time.*

The guiding insight is that, for $\mu/\sqrt{n}$ incoherent dictionaries, $A^T A \approx I$, since the of diagonals are bounded above by $\mu/\sqrt{n}$. Note that this approximation is not necessarily a great one in the spectral sense, since $A^T A - I$ can have $n^2 - n$ entries of size $\Omega(\mu/\sqrt{n})$, and thus $\|A^T A - I\|$ might be $\Omega(\mu\sqrt{n})$.

But looking only at the spectral norm *does not take advantage of sparsity*: Indeed, $\|A^T A - I\| = \max_{\|z\|:1} z^T(A^T A - I)z$, and if $A^T A - I$ has entries all around $\mu/\sqrt{n}$, this maximimum will be attained for $z^* \approx \frac{1}{\sqrt{n}}(1, \ldots, 1)$. However, if we impose that z^* is k-sparse, things are a bit different. Define the seminorm $\|z\|_0 := \sum_i I(z_i \neq 0\}$, and let $B_0(k) := \{z \in \mathbb{R}^n : \|z\| \le 1, \|z\|_0 \le k\}$. It is rather easy to show that

$$\sup_{z \in B_0(k)} z^T(A^T A - I)z \le k\mu/\sqrt{n} \tag{3.6}$$

This restriction to the subset of k=sparse vectors gives rise to the notion of the "Restricted Isometry Property" in the compressed sensing literature [4]. Indeed, if $k\mu/\sqrt{n} < 1/2$, then A is effictively "invertible" for all $2k$ sparse vectors z, and if $k\mu/\sqrt{n} = o(1)$, then $\langle Az, Az\rangle = \|z\|^2 + z^T(A^T A - I)z \approx \|z\|^2$ for all k-sparse z. More precisely, we can prove the following lemma:

Lemma 3.2. *Let z_1 and z_2 be two k-sparse vectors, and let A have unit norm columns, Then Then $\langle Az_1, Az_2\rangle = \langle z_1, z_2\rangle \pm \frac{2k\mu}{\sqrt{n}}\|z_1\|\|z_2\|$.*

Proof. By relableing the columns of A and then entries of z_1 and z_2, we can imagine that z_1 and z_2 are both supported on the indices $[2k] := \{1, \dots, 2k\}$. Hence,

$$\begin{aligned} \langle Az_1, Az_2 \rangle &= \sum_{i \in [2k]} \|A_i\|^2 z_1(i) z_2(i) + \sum_{i \in [2k]} \sum_{j \neq 1 \in [2k]} z_1(i) z_2(j) \langle A_i, A_j \rangle && (3.7) \\ &= \langle z_1, z_2 \rangle + E && (3.8) \end{aligned}$$

where $E := \sum_{i \in [2k]} \sum_{j \in [2k]} z_1(i) z_2(j) \langle A_i, A_j \rangle$.

$$\begin{aligned} |E| &\leq \sum_{i \in 2k} \sum_{j \neq i \in [2k]} |z_1(i)||z_2(j)| \cdot |\langle A_i, A_j \rangle| && (3.9) \\ &\leq \sum_{i \in 2k} \sum_{j \in [2k]} |z_1(i)||z_2(j)| \cdot |\langle A_i, A_j \rangle| && (3.10) \\ &\leq \frac{\mu}{\sqrt{n}} \sum_{i \in [2k]} \sum_{j \in} |z_1(i)||z_2(j)| \leq \frac{\mu}{\sqrt{n}} \|w_1 w_2^T\|_F && (3.11) \end{aligned}$$

where $w_1 \in \mathbb{R}^{2k}$ has $w_1(i) = |z_1(i)|$ for all $i \in [2k]$, and w_2 is defined similarly for z_2, and $\|\cdot\|_F$ denotes the Frobenius norm. Because $w_1 w_2^T$ is a $2k \times 2k$ matrix, we have $\|w_1 w_2^T\|_F \leq 2k \|w_1 w_2^T\|$, where $\|\cdot\|$ denotes the spectral norm. But $\|w_1 w_2^T\| = \|w_1\|\|w_2\| = \|z_1\|\|z_2\|$, whence

$$|E| \leq \frac{2k\mu}{\sqrt{n}} \|z_1\|\|z_2\| \tag{3.12}$$

□

3.1 Formal Models for Dictionary Learning

To encourage sparsity, Olshausen and Field [5] designed an alternating gradient descent algorithm to minimize the following objective:

$$\min \sum_{i=1}^{N} |y_i - Ax_i|^2 + \sum_{i=1}^{N} \text{penalty}_K(x) \tag{3.13}$$

As we remarked about, an unpenalized Dictionary learning is highly underdetermined. Hense, Olshausen and Field introduced the penalties - for example, l_1-regularization - to encourage sparsity and ensure (or at least promote) model indentifiability [5]. In [3], Arora, Ge, et al. describe an alternating minimization algorithm based on Olshausen and Field to learning the objective in Equation 3.13. In these notes, we will restrict our attention to the "overlapping community methods" to be described shortly. In either case, both the alternating minimization algorithms in [3] and the overlapping community detection methods from [2] will make use of roughly the same assumptions, which we formalize as follows:

1. The dictionary $A \in \mathbb{R}^{n \times m}$ has unit norm columns, and has $\frac{\mu}{\sqrt{n}}$-incoherent columns, that is: $|\langle A_i, A_j \rangle| \leq \frac{\mu}{\sqrt{n}}$.

2. We are concerned with the regime $m \geq n$, and we require that $\|A\| = O(\sqrt{m}/\sqrt{n})$.
3. Each x has exactly k nonzero coordinates, drawn uniformly from $\{1, \ldots, m\}$ (this can be relaxed somewhat, as in [3])
4. x each coordinate is independent conditioned on its support, and $x_i | x_i \neq 0$ is subgaussian with $O(1)$ variance proxy, and there is a constant C - universal across all $i \in [m]$ - such that $|x_i||x_i \neq 0 \geq C$ almost surely. For example, we can think of $x_i | x_i \neq 0$ as being drawn uniformly from $[1, 10]$, or from $[-10, -1] \cup [1, 10]$.
5. We will start off by assumping that $x_i \geq 0$ almost surely. An adpatation of the arguments in this paper will also hold for the case where $\mathbb{E}[x_i] = 0$

Given samples $y_1 = Ax_1$ and $y_2 = Ax_2$, it follows from Lemma 3.2 that

$$\langle y_1, y_2 \rangle = \langle x_1, x_2 \rangle \pm \|x_1\|\|x_2\| \frac{k\mu}{\sqrt{n}} \tag{3.14}$$

By subgaussian concentration, it holds with high probability that $\|x_1\|\|x_\| = \tilde{O}(k)$, so as long as $k^2\mu/\sqrt{n}$ is roughly $o(1)$, then

$$\langle y_1, y_2 \rangle = \langle x_1, x_2 \rangle \pm o(1) \tag{3.15}$$

If we assume that x_1 and x_2 are entrywise non-negative, then

$$\begin{aligned} \langle x_1, x_2 \rangle &= \sum_{i \in \mathrm{Supp}(x_1) \cap \mathrm{Supp}(x_2)} x_1(i) x_2(i) && (3.16) \\ &\geq C|\mathrm{Supp}(x_1) \cap \mathrm{Supp}(x_2)| && (3.17) \\ &\geq CI(\mathrm{Supp}(x_1) \cap \mathrm{Supp}(x_2) \neq \emptyset) && (3.18) \end{aligned}$$

Hence, with high probability, it holds that $\langle y_1, y_2 \rangle \geq C/2$ if and only iff x_1 and x_2 have a nonzero entry in common. We will state this in an informal lemma:

Lemma 3.3. *If $k^2\mu/\sqrt{n}$ is roughly $o(\log n)$, then with very high probability $\langle x_1, x_2 \rangle \geq C/2$ if and only x_1 and x_2 share a nonzero entry.*

This observation allows us to transform the problem from an analytic one to a combinatorial one. Indeed, given N observations $y_1, \ldots, y_N$, let each observation y_i correspond to a vertex i in a graph $G = (V, E)$. We draw an edge between the vertices i and j if and only if $\langle y_i, y_j \rangle \geq C/2$. By the above discussion, it holds high probability that edges are drawn between i and j if and only if x_i and x_j share a common non-zero entry. Taking a union bound, the following claim holds:

Lemma 3.4. *With high probability that $G \simeq \tilde{G}$, where $\tilde{G}$ is the graph over the vertices $i \in [N]$ with edges connected all indices i, j for which x_i and x_j have non-disjoint support*

We now give a more intuitive way to characterize $\tilde{G}$: Let $C_1, \ldots, C_m$ be sets defined so that $C_j := \{i \in [N] : x_i(j) \neq 0\}$. We will call the sets "communities", in the sense that all $i \in C_j$ share a nonzero entry in common. By the assumption that there are exactly k nonzero entries of each sample x_i selected uniformly at random, each vertex i is assigned to exactly k communities $C_{j_1}, \ldots, C_{j_k}$. Moreover, it follows directly from the definition of the sets C_j that x_i and x_j share a common nonzero if and only if they both lie in the same community: $\tilde{G}$ is precisely the graph constructed by drawing edges between vertices which belong to at least one of the same community. Hence, to recover the sparsity patterns of the x_i with high probability, Lemma 3.4 tells us that the graph G, whose edges are constructed from the inner products of samples y_i and y_j, is precisely generated by the community assignments of its vertices.

3.1.1 Mean Zero Case

If the entries of x_i are mean zero, then the argument is a little different: indeed,

$$\sum_{i \in \mathrm{Supp}(x_1) \cap \mathrm{Supp}(x_2)} x_1(i) x_2(i) \tag{3.19}$$

can have absolute value much smaller than C due to cancellations from the entries of x_1 and x_2 having cancelling signs. However, with probability $\Omega(k^2/m^2)$, $\mathrm{Supp}(x_1)$ and $\mathrm{Supp}(x_2)$ will overlap at at most one entry, so we can neglect these correlations if we are willing to accept a small (but not as small as $n^{-\omega(1)}$) probability of missing an edge (which will also not be independent of the common support of and x_2 x_1). On the other hand, we can improve the bound in Lemma 3.2 due to cancellations. Indeed, we have

$$\langle y_1, y_2 \rangle - \langle x_1, x_2 \rangle = \sum_{i \neq j} \langle A_i, A_j \rangle x_1(i) x_2(j) \tag{3.20}$$

Using the bound $|\langle A_i, A_j \rangle| \leq \mu/\sqrt{n}$, this term has mean zero and moment roughly $O(\sqrt{k}\mu/\sqrt{n})$ due to cancellations. Hence, $\langle y_1, y_2 \rangle = \langle x_1, x_2 \rangle + \tilde{O}(\sqrt{k}\mu/\sqrt{n})$, so our error drops by roughly a factor of $\sqrt{k}$.

3.2 Reduction of Dictionary Learning to Community Detection

Given our community detection algorithm, we have given a sktech of how to efficiently recover the sparsity patterns of the latent samples x_i. We show how to use this technique to recover the dictionary A, folllowing [2]. Note that, once A has been retrieved, we can use more standard techniques from sparse recovery to (approximately) recover the latent signal vectors x.

The basic idea is that the j-th column of A, A_j, should roughly resembly the average of all sampes y_i for which the j-th entry is active: that is, $y_i = Ax_i$ where $x_i(j) \geq 0$. Thus, a first first attempt at recovering A would be simply to compute the following average:

$$A_j := \frac{1}{|C_j|} \sum_{i : y_i \in C_j} y_i \tag{3.21}$$

Unfortunately, in the case were the x_i have mean zero, we have $\mathbb{E}[y_i] = \mathbb{E}[Ax_i] = A\mathbb{E}[x_i] = 0$, In the case where the x_i have do not have mean zero, then we get a lot of spurious contributions from the nonzero entries of the samples at indices not equal to j: that is, $y_i = A_j x_i(j) + \sum_{j' \neq j} A_{j'} x_i(j')$.

A better idea is to instead look at the best rank 1 approximation to $E[yy^T] : y \in C_j$. For simplicity, we will first handle the mean zero case. Note first that, because the problem is invariant to permutation of the columns of A, it suffices to prove an algorithm which recovers A_1, the column of A which corresponds to community C_1. Our strategy will be to compute the best rank-one approximation to the empirical covariance matrix of all samples y_i which have an active first column, that is $y_i \in C_1$. Let

$$M_1 := \frac{1}{\#y : y \in C} \sum_{y \in C} [yy^T] \tag{3.22}$$

that is, the empirical average of all yy^T for $y \in C$. First, lets show that M_1 is a good approximate of $A_1 A_1^T$ up to a constant factor:

$$\begin{aligned} M_1 &= \mathbb{E}[x(1)^2 A_1 A_1^T] + \mathbb{E}[\sum_{i \geq 2} x(i)^2 A_i A_i^T] \\ &+ \mathbb{E}[\sum_{i \geq 2} x(i)x(j)(A_1 A_i + A_i A_1)] + E[\sum_{i,j \geq 2} x(i)x(j) A_i A_j] + \text{ statistical error} \\ &\approx \Theta(A_1 A_1^T) + O(\frac{k}{m} \sum_{i \geq 1} A_i A_i^T) + \tilde{O}(k^2/\sqrt{N}) \end{aligned}$$

Here N is the number of samples used, the $O(f)$ notation is means a quantity whose spectral norm is bounded by Cf for some $C > 0$, and $O(M)$ (resp $\Theta(M)$) means a quantity which is less than CM (resp less than CM and greater than cM) according to the cannonical ordering $\preceq$ of the semidefinte cone . The first term comes from the fact that $\mathbb{E}[x(1)^2 | y \in C] = \Theta(1)$, the second term comes from the fact that $\mathbb{E}[x(i)^2 | y \in C] = O(k/m)$. Note that the second error term is systemic - it does not depend on the number of samples used by the algorithm.

The remaining error term $\tilde{O}(k^2/\sqrt{N})$ is statistical in nature, and comes from the deviation of all the terms from their exptation. It is easy to establish the $\tilde{O}(k^2/\sqrt{n})$ by conditioning on the very high probability even that all the x_i are small, and then using Chernoff bounds to finish up. The bound can be improved to $\tilde{O}(k/\sqrt{N})$, but this improved bounded affects the sample complexity of the algorithm. On the other hand, the systemic error of $O(\frac{k}{m} \sum_i A_i A_i^T)$ determines the conditions on k and m under which the best-rank-one approximation algorithm accurately retrives the underlying dictionary.

First, we will use a standard assumption in the dictionary learning literature that $\|A\| = O(\sqrt{m}/\sqrt{n})$. Under this condition, it holds that $\|(\frac{k}{m} \sum_{i \geq 2} A_i A_i^T)\| = O(k/n)$. We will assume that the number of samples is large enough that the statistical error is also dominated by $O(k/n)$. Thus, $M_1 \propto A_1 A_1^T + E$, where E has norm $O(k/n)$. N

Now let $\hat{A}_1$ be the top eigenvector of M_1. We can show that $\hat{A}_1$ is a good estimate of A_1 by appealing to Wedin's Theorem, an elementary result from linear algebra, which bounds the distance of the top eigenvector of a PSD matrix A to that of $A + E$, where E is a small perturbation. Because A_1 is the top eigenvecotr of $A_1 A_1^T$, Wedin's Theorem will help us show that the top eigenvector of M_1 should be close to A_1 as well:

Theorem 3.5. *Let v_1 be the top eigenvector of PSD matrix A and let v_2 be the top eigenvector of $A + E$. Let θ be angle between v_1 and v_2. Then $\sin\theta \leq \frac{2\|E\|}{\sigma_1(A) - \sigma_2(A)}$.*

As a corrolary, we get a clean bounded on the Euclidean distance between the (normalized) top eigenvectors of A and $A+E$

Corollary. *Let A be a rank one matrix of norm 1 with top eigenvector v_1, let v_2 be the top eigenvector of $A+E$. Then as long as $\|E\| = o(1)$ $\|v_1 - v_2\| \le \sqrt{1/2}$, $\|v_1 - v_2\| \le 2\|E\|$*

Proof. Because $\sigma_1(A_1 A_1^T) = 1$, and $\sigma_2(A_1 A_1^T) = 0$, we have that $\sin\theta(v_1, v_2) \le 2\|E\|$. Because $v_1^T v_2 = 1 - \|v_1 - v_2\|^2$, we have

$$\sin\theta(v_1, v_2) = \sin\arccos(v_1^T v_2) = \sqrt{1 - (v_1^T v_2)^2} = \sqrt{2\|v_1 - v_2\|^2 - \|v_1 - v_2\|^4} \quad (3.23)$$

$$= \sqrt{2}\|v_1 - v_2\|\sqrt{1 - \|v_1 - v_2\|^2} \quad (3.24)$$

Because $E = o(1)$, it follows that $\sin\theta(v_1, v_2)$, and hence $\|v_1 - v_2\|$, must be $o(1)$ as well. Hence, $\|v_1 - v_2\| \le \sqrt{2}\|E\| / \sqrt{1 - \|v_1 - v_2\|^2} \le 2\|E\|$. □

From this corrolary, it follows immediately that $\|\hat{A}_1 - A_1\| \le O(k/n)$. Hence, given enough (but still polynomially many) samples, we can recover easy the columns of A up to an error of k/n.

4 Unsupervised learning of Deep Nets

Let's return from the restricted setting of dictionary learning to the more general setting of neural nets. Aside from their success, one of the major reasons for the popularity of deep nets is that the last layer seems to capture "meaningful features". For example, in vision problems, the pixel-representations of an object learned by neural nets often represents the shape of that object very closely. And, in many applications, one can train very effective clasifiers (using, say, an SVM or Logistic Regression) on the features learning by the last layer. In fact, if we train a multilayer neural network for a classification task - say, distinguishing between cats and dogs - and then retrain the last layer to learn a new task - say, distinguish between birds and bees - *without retraining the parameters of most of the hidden layers*, the retrained network is still remarkably succesful at its new classification task. This suggest that the representations learned in the deeper layers of the neural network capture most of the relevant information, or at least enough information to build effective classifiers.

This suggests that deep nets are capturing some inherent structure in the images themselves, raising hope that the hidden layers correspond to natural "features" that could be learnt from just unlabeled data. (By contrast, the recent successes involve leveraging large amounts of labeled images.) Unsupervised training of deep nets is a holy grail of this area, and major researchers in this area have tried to define a generative model corresponding to deep nets. This quest is very much in the spirit of the discriminative-generative pairs we discussed in an earlier lecture (eg naive bayes classifier is a generative analog of logistic regression).

If we move from the descriminative perspective to the generative perspective, we might wonder - what is the structure that neural nets can extract structure from the data? Let's now consider a two layer neural net whose top layer is encoded in a vector x and bottom layer is encoded in a vector y.

Rather than imagining a linear map A which takes a sparse input x and maps it to a dense y, we now imagine an *encoding function* $E(\cdot)$ which *encodes* a dense y as a sparse x. In the linear case, we had that $y = Ax$, so that $x \approx A^T y$. In the general case, we model $x = E(y) = h(A'y + b)$, where b is an offset function and $h(\cdot)$ is a nonlinear map which acts identically and independently on each coordinate, for example, $h(\cdot)$ can be the function which returns the sign of the entries of its arguments. Again, we can imagine that eancy entry of x and y are treated as vertices in a bipartite graph, and that A is the adjacency matrix which captures the edge weights between the entries of x and y.

The hope is that we can now invert the encoding function $E(\cdot)$, and in fact perform the inversion in the presence of noise. This motivates the following definition:

Definition 4.1 (Denoising Auto-Encoder)**.** Give an adjacency matrix A, an *autoencoder* consists of a pair of an encoding function of the form $E(y) = h(A'y + b)$ and a decoding function $D(x) = h(Ax + b')$. The autoencoder is called *denoising* for a noise model $\xi \sim \mathcal{D}$ if the the decoding robust to noise in the sense that:

$$E(D(h) + \xi) = h \quad \text{with high probability} \tag{4.25}$$

and said to be weight tying if $A' = A^T$. Here $D(h) + \xi$ is shorthand for D corrupted with the noise vector ξ. This corruption might not necessarily be additive.

The following theorem states that, if the entrywise nonlinear function $h(\cdot)$ is the sign function, and A is sufficiently sparse, then [1] show that the two layer neural network is in fact a denoising autoencoder:

Theorem 4.1 (stated loosely)**.** *Consider a two layer neural network with sparse bi-partite graph G with adjancey matrix A with edge weights drawn uniformly in $[-1, 1]$. Suppose that the latent sample x are binary with support S. Finally, suppose that $y = sign(Ax)$. Then there is a b' for which the pair $E(\cdot)$ and $D(\cdot)$ form an denoising autoencoder, where $E(\cdot) = sign(A^T y + b')$.*

In fact, we can learning the encoding/decoding function with high probability:

Theorem 4.2. *Under some regularity assumptions, there is a polynomial time algorithm to learn the encoding and decoding functions for a two layer neural with sparse edge weights drawn uniformly in* $[-1, 1]$

Proof. To preserve intuition, we assume that we have an unweighted bipartite graph which is drawn unifromly from all d-regular bipartite graphs on vertex set given by the entries of x and y. We assume that thare $E(\cdot)$ and $D(\cdot)$ are chosen with no thresholding function, so that $b = b' = 0$. We also assume that the x_i are uniformly drawn, k-sparse binary vectors, where $x = \rho n$ for some small ρ.

Let's begin by learning the adjacency matrix A, or equivalently, the graph G. What are the communities? They are subsets of nodes with a common neighbor. So what happens when two nodes have a common neighbor. If u, v have a common neighbor, then $Pr[u, vare1] \geq \rho$. So $Pr[u, v$ are both 1$] \leq (pd)^2$. So if $\rho \gg (\rho d)^2$, we can recover the communities with high probability.

Now lets describe how to recover the entries of samples x. The key intuitition is that, if an entry of x, say x_1 is active, then some number of its neighbors y_i will be active as well. Hence, we can recover x_1 by determining if above a certain threshold of its neighboring indicies y are 1. The guarantees behind these algorithm come from the following observation: that uniformly drawn sparse bipartite graphs are expanders with high probability. Let's be more specific:

Let U denote the vertex set corresponding to the entries of x, and V the vertex set corresponding to the entries if y. Given $u \in U$, let $F(u)$ denote all of its neighbors in V. Fiallly for some set $S \subset U$, let $UF(u, S)$ be the set of unique neighbors of u with respect to S: that is

$$UF(u, S) := \{v \in V : v \in F(u), v \notin F(S - \{u\})\} \tag{4.26}$$

It turns out that, for a randomly generated bipartite graph and sufficiently small set S, then for every $u \in U$, the total number of u's neighbors in $UF(u, S)$is at least $9d/10$ of its total number of neighbors. Hence, if an entry x_i is not active, we expect no more than, say $2d/10$ of its neighbors in V to be active. Hence, we can recover the vector x with high probability by setting

$$x_i = \text{threshold}_{2d/10} (\#\text{neighbors of } x_i \text{ active }) \tag{4.27}$$

□

Perhaps even more surprisingly, [1] show that one can learn the connect graphs $G^{(l)}$ in a multilayer neural network by learning the bottom-most layers first, and then moving upward thhrough the graph:

Theorem 4.3 (Generalization to Deep Nets)**.** *Given a deep neural network with layers $x^{(l)}$ and weighted connection graphs $G^{(l)}$ drawn with expected degree $d^{(l)}$, and edge weights uniformly in $[-1, 1]$, and where the samples in the top layer are binary vectors with uniform sparse support of size ρn. Then, if ρ is sufficiently small, and the degrees $d^{(l)}$ do not grow too quickly, then the ground truth graphs $G^{(l)}$ and corresponding samples x can be learned with high probability. In fact, they can be learned my infering the second to bottom layer from the bottom layer, and then moving up layerwise through the network.*

References

[1] Sanjeev Arora, Rong Ge, Aditya Bhaskara, Tengyu Ma. "Provable Bounds for Learning Some Deep Representations." *Journal of Machine Learning*, volume 32, 2014.

[2] Sanjeev Arora, Rong Ge, Ankur Moitra. "New Algorithms for Learning Incoherent and Overcomplete Dictionaries" *Conference on Learning Theory*, 2014.

[3] Arora, Sanjeev, et al. "Simple, Efficient, and Neural Algorithms for Sparse Coding" *arXiv preprint arXiv:1503.00778*, 2015

[4] Candes, Emmanuel J. "The restricted isometry property and its implications for compressed sensing." *Comptes Rendus Mathematique* 346.9 (2008): 589-592.

[5] Bruno A. Olshausen and David J. Field. "Sparse coding with an overcomplete basis set: a strategy employed by v1." *Vision Research*, 37:3311–3325, 1997a.

[6]