Mathematical Aspects of Deep Learning

Mathematics of Deep Learning: Lecture 5 – Random Sparse Nets etc.

Transcribed by Jiaoyang Huang (edited by Asad Lodhia, Elchanan Mossel and Matthew Brennan)

For today's lecture we will present two papers. The first one is *Provable Bounds for Learning Some Deep Representations* by Arora, Bhaskara, Ge and Ma. They proved that random sparse neural networks are learnable. The second paper is *Train faster, generalize better: Stability of stochastic gradient descent* by Hardt, Recht and Singer. Their main results state that any model trained with stochastic gradient method in a reasonable amount of time attains small generalization error.

Provable Bounds for Learning Some Deep Representations

The sparse random network model has ℓ layers of vectors of binary variables $h^{(\ell)}, h^{(\ell-1)}, \cdots, h^{(1)}$ (where $h^{(\ell)}$ is the top layer) and an observed layer $h^{(0)}$ at bottom. Each layer has n nodes.

$$\text{Depth} = \ell, \qquad \text{Width} = n. \tag{1}$$

The activation function is the sgn function, i.e.

$$\sigma(x) = \begin{cases} 1 & x > 0, \\ 0 & x = 0, \\ -1 & x < 0. \end{cases} \tag{2}$$

The network between two layers is a random graph with average degree $d = n^\gamma$ with $0 < \gamma \leq 0.2$. More precisely, each node at level k is connected to each node at level $k-1$ with probability $n^{\gamma-1}$ independently. The edge weights are random in $\{\pm 1\}$ with probability $1/2$ each. For any vertex i in the k-th layer, and vertex j in the $(k-1)$-th layer, the weight $W_{ji}^{k-1} \in \{-1, 0, 1\}$ is such that

$$W_{ji}^{k-1} = \begin{cases} 0 & \text{there is no edge between } i, j. \\ -1 & \text{there is an edge with weight } -1 \text{ between } i, j. \\ 1 & \text{there is an edge with weight } 1 \text{ between } i, j. \end{cases} \tag{3}$$

We remark that W^{k-1} is the weighted adjacency matrix between $(k-1)$-th and k-th layers. The samples are generated from top to bottom. Given $h^{(k)}$, the vector in the k-th layer, the vector in the $(k-1)$-th layer is $h^{(k-1)} = \mathrm{sgn}(W^{k-1} h^{(k)})$,

$$h_j^{(k-1)} = \mathrm{sgn}\left(\sum_{i=1}^{n} W_{ji}^{k-1} h_i^{(k)} \right), \quad j = 1, 2, \cdots, n. \tag{4}$$

We sample the top layer $h^{(\ell)}$, where the set of nodes that are 1 is picked uniformly among all sets of size $\rho_\ell n$. ρ_ℓ is called the density of the top layer, and we assume that $\rho_\ell (d/2)^\ell = O(1)$. Since averagely, each node has d children nodes, the density at level k is roughly of size $\rho_\ell d^{\ell-k} \ll 1$, and $h^{(k)}$ is also a sparse vector. The main theorem we will prove is the following:

Theorem. *When degree $d = n^\gamma$, for $0 < \gamma \leq 0.2$, with density $\rho_\ell (d/2)^\ell = O(1)$, the network model can be learned with high probability using $O(\log n / \rho_\ell^2)$ samples and $O(n^2 \ell)$ time.*

Thanks to the sparsity, the net should approximately preserve information, and allows easily going back/forth between representations in two adjacent layers. In the following we first focus on learning a single-layer network. It has a top layer with vector $h^{(1)}$ and a bottom layer with vector $h^{(0)}$. We assume that $h^{(1)}$ is sampled among sparse vectors, and then $h^{(0)}$ is generated using $h^{(1)}$. To learn a single-layer network, we need first to recover the network structure. Then for each sample, we need to recover $h^{(1)}$ from $h^{(0)}$.

Given a single-layer network (with known graph structure), the property that we can recover $h^{(1)}$ from $h^{(0)}$ is characterized by the following definition:

Definition. Fix an activation function. An autoencoder consists of a decoding function $D(h) = s(Wh + b)$ and an encoding function $E(h) = s(W'h + b')$ where W, W' are linear transformations, b, b' are fixed vectors and s is an activation function, which acts on each coordinate. The autoencoder is said to be well denoising if $E(D(h) \oplus \xi) = h$ with high probability with respect to the noise ξ and the randomness of h, where $D(h) \oplus \xi$ means that $D(h)$ is corrupted by noise ξ. The autoencoder is called weight tying if $W' = W^T$.

We will show that a single-layer random network is a denoising autoencoder if the input layer and noise are both sparse.

Lemma. *A one layer network with density ρ satisfying $\rho d < 0.1$ is a denoising autoencoder with high probability, with respect to noise that flips every output bit independently with probability < 0.1. Furthermore this autoencoder is weight tying.*

Proof. We denote W the weighted adjacency matrix between the top layer and the bottom layer. By definition, the one layer network implements the encoder $E(y) = \mathrm{sgn}(W^T y)$. We will show that the one layer network is an autoencoder with decoder $D(h) = \mathrm{sgn}(Wh)$ with high probability. Note that this will also imply it is weight tying.

Let S be the support of the top layer $h^{(1)}$, and S' be the support of the bottom layer $h^{(0)}$. The key property for the sparse network is that most neighbors of an activated node are unique. More precisely, fix i in the top layer, and $j \in \mathcal{N}(i)$ in the bottom layer, where $\mathcal{N}(i)$ is the set of neighbor vertices of i, i.e. $W_{ji} \neq 0$. Thanks to the sparsity assumption, with high probability, j has no non-zero neighbor other than i, i.e.

$$\mathbb{P}(|\mathcal{N}(j) \cap S \setminus \{i\}| \geq 1) \leq \rho d < 0.1. \tag{5}$$

If $\mathcal{N}(j) \cap S \setminus \{i\} = \emptyset$, and $h_j^{(0)}$ was not flipped by the noise, then $h_i^{(1)} = W_{ji} h_j^{(0)}$. Moreover,

$$\mathbb{P}(\mathcal{N}(j) \cap S \setminus \{i\} = \emptyset \text{ and } h_j^{(0)} \text{ was not flipped by the noise}) < 0.2. \tag{6}$$

Motivated by these observations, we can recover $h^{(1)}$ using the following quantity,

$$I_i = \sum_{j \in \mathcal{N}(i)} W_{ji} h_j^{(0)}. \tag{7}$$

If $|I_i| \geq d/2$, then we set $h_i^{(1)} = sgn(I_i)$. Otherwise, if $|I_i| < d/2$, we set $h_i^{(1)} = 0$.

In the following we focus on learning the structure of a single-layer network. ■ We recall that $h^{(1)}$ is sampled among sparse vectors (with density ρ), and then $h^{(0)}$ is generated using $h^{(1)}$. The graph structure can be recovered using $\text{poly}(n)$ samples.

Lemma. *Given* $\text{poly}(n)$ *samples, one can recover* W *the weighted adjacency matrix (between the top layer* 0 *and* 1*).*

Proof. We define a $\sim$ relation among vertices in the bottom layer: call two nodes i_1, i_2 at the bottom layer related, denoted by $i_1 \sim i_2$, if they are connected to the same node at the top layer. We can identify related nodes by their correlations. If $i_1 \sim i_2$, then they have common neighbors. With probability ρ, one of their common neighbors is non-zero, so

$$\mathbb{E}\left[h_{i_1}^{(0)} h_{i_2}^{(0)}\right] \approx \rho. \tag{8}$$

If $i_1 \nsim i_2$, then they do not have common neighbors. With good probability, at least one of them is zero

$$\mathbb{E}\left[h_{i_1}^{(0)} h_{i_2}^{(0)}\right] \approx 0. \tag{9}$$

Once we recovered the $\sim$ relations, the recovering of the graph structure is straightforward. In fact we can name vertices in the top layer by largest clusters in the bottom layer. More precisely, each vertex in the top layer corresponds to a largest cluster A in the bottom layer such that $|A| \geq 0.1d$ and for any $i, j \in A$, $i \sim j$. This approach can be strengthened to include some errors by identifying clusters that are different in less than $0.1d$ nodes, i.e. we identify A and A' if $|A \Delta A'| \leq 0.1d$. ■

The high level network inference algorithm in the paper learns the network layer by layer starting from the bottom. For the network between i-th and $(i+1)$-th layers, the algorithm first recover the network structure between levels i and $i+1$. Then it recovers $h^{(i+1)}$ from $h^{(i)}$ for each sample as above.

The [illegible] issue. In the original model, we sample $h^{(\ell)}$ uniformly among all the sparse vectors, then sequentially generate $h^{(\ell-1)}, h^{(\ell-2)}, \cdots, h^{(0)}$.

The bits of $h^{(i+1)}$ are not independent! We can not directly use the Lemma above to recover the network structure between levels i and $i+1$. However, this problem can be resolved by showing that the bits of $h^{(i+1)}$ cannot be too correlated, unless they have a common ancestor. Moreover, the correlation decays exponentially, i.e.

$$\text{grandparents correlation } \leq 1/d \text{ parent correlation.} \tag{10}$$

Train Faster, Generalize Better: Stability of Stochastic Gradient Descent

Let's consider the following general setting of supervised learning. There is an unknown distribution $\mathcal{D}$. We receive a sample $S = (z_1, z_2, \cdots, z_n)$ of n i.i.d. examples drawn from $\mathcal{D}$. Let $f(w, z)$ denote the a loss function where w specifies the model and z denotes an example. Our goal is to minimize the population risk:

$$R[w] := \mathbb{E}_{z\sim\mathcal{D}}[f(w, z)], \tag{11}$$

over all possible models w. Since the distribution $\mathcal{D}$ is unknown, we cannot measure the quantity $R[w]$ directly, we instead minimize the empirical risk:

$$R_S[w] := \frac{1}{n}\sum_{i=1}^{n} f(w, z_i). \tag{12}$$

The generalization error of a model w is the difference

$$R_S[w] - R[w]. \tag{13}$$

If $w = A(S)$ is chosen as a function of the data by a possibly random algorithm A, we need to estimate the expected generalization error

$$\varepsilon_{gen} := \mathbb{E}_{S,A}[R_S[A(S)] - R[A(S)]]. \tag{14}$$

In order to bound the generalization error, we recall the following definition of

Definition. The algorithm A is ε-uniform stable if for all data sets S, S' that differ at most one example:

$$\sup_z \mathbb{E}_A[f(A(S), z) - f(A(S'), z)] \leq \varepsilon. \tag{15}$$

We define $\varepsilon_{Stab}(A, n)$ the smallest such ε.

Claim. *The generalization error can be controlled by* $\varepsilon_{Stab}(A, n)$*:*

$$|\mathbb{E}_{S,A}[R_S[A(S)] - R[A(S)]]| \leq \varepsilon_{Stab}(A, n). \tag{16}$$

Proof. Denote by $S = (z_1, z_2, \cdots, z_n)$ and $S' = ({z_1}', {z_2}', \cdots, z_n')$ two independent random samples and let $S^{(i)} = (z_1, \cdots, z_{i-1}, z_i', z_{i+1}, \cdots, z_n)$. With these notations we have

$$\begin{aligned}
\mathbb{E}_S\mathbb{E}_A[R_S[A(S)]] &= \mathbb{E}_S\mathbb{E}_A\left[\frac{1}{n}\sum_{i=1}^n f(A(S), z_i)\right] \\
&= \mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_A\left[\frac{1}{n}\sum_{i=1}^n f(A(S^{(i)}), z_i')\right] \\
&= \mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_A\left[\frac{1}{n}\sum_{i=1}^n f(A(S), z_i')\right] + \delta \\
&= \mathbb{E}_S\mathbb{E}_A\left[R[A(S)]\right] + \delta,
\end{aligned} \tag{17}$$

where

$$|\delta| = \left|\mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_A\left[\frac{1}{n}\sum_{i=1}^n \left(f(A(S^{(i)}), z_i') - f(A(S), z_i')\right)\right]\right| \tag{18}$$

$$\leq \varepsilon_{Stab}(A, n). \tag{19}$$

This finishes the proof. ■

In the following we show that for stochastic gradient iterative algorithms, we can control their uniform stability, and obtain some upper bound on the generalization error.

Given the loss function $f(w, z)$, the stochastic gradient update with learning rate $\alpha_t > 0$ is given by

$$w_{t+1} = w_t - \alpha_t \nabla_w f(w_t, z_{i_t}) \tag{20}$$

where the indices i_t are either picked uniformly randomly in $\{1, 2, \cdots, n\}$, or picked according to a random permutation.

In the rest of the lecture, we analyze the stochastic gradient descent with convex minimization. Let me first discuss a little bit of convexity,

Claim. *If f is convex and ∇f is β-Lipschitz then*

$$\langle \nabla f(v) - \nabla f(w), v - w \rangle \geq \frac{1}{\beta} \|\nabla f(v) - \nabla f(w)\|_2^2. \tag{21}$$

Proof. First we prove that

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{1}{2\beta} \|\nabla f(y) - \nabla f(x)\|_2^2. \tag{22}$$

Then by symmetry, we have

$$f(x) \geq f(y) + \nabla f(y)^T (x - y) + \frac{1}{2\beta} \|\nabla f(y) - \nabla f(x)\|_2^2. \tag{23}$$

Our claim (21) follows by taking sum of (22) and (23).

For the proof of (22), we introduce a new function

$$g(z) = f(z) - \nabla f(x)^T z. \tag{24}$$

Then $g(z)$ is convex and ∇g is β-Lipschitz. Moreover, thanks to the convexity of f, i.e.

$$f(z) \geq f(x) + \nabla f(x)^T (z - x), \tag{25}$$

g achieves its minimum at $z = x$.

Since ∇g is β-Lipschitz, by the Taylor expansion, for any z,

$$g(z) \leq g(y) + \nabla g(y)^T (z - y) + \frac{\beta}{2}\|z - y\|_2^2. \tag{26}$$

By taking the minimum on both sides,

$$\begin{aligned} g(x) &= \min_z \{g(z)\} \\ &\leq \min_z \left\{ g(y) + \nabla g(y)^T (z - y) + \frac{\beta}{2}\|z - y\|_2^2 \right\} \\ &= g(y) - \frac{1}{2\beta}\|\nabla g(y)\|_2^2, \end{aligned} \tag{27}$$

which is (22).

We prove the stability bound for convex loss minimization via stochastic gradient method. ■

Theorem. *Assume that for all z, $f(\cdot, z)$ is convex and L-Lipschitz, and ∇f is β-Lipschitz. Suppose that we run the stochastic gradient method for T steps with step sizes $\alpha_t \leq 2/\beta$, then*

$$\varepsilon_{Stab} \leq \frac{2L^2}{n} \sum_{t=1}^{T} \alpha_t. \tag{28}$$

Proof. Let S and S' be two samples of size n differing in only a single example. Let $\{w_t\}$ and $\{w'_t\}$ be two runs of the algorithm with data sets S and S', and $\delta_t = \|w_t - w'_t\|_2$.

We couple the two algorithms with data sets S and S'. At step t, there are two cases: with probability $1 - 1/n$, the algorithms choose the same data example; with probability $1/n$, the algorithms choose different data examples.

If the two algorithms choose the same data example z at step t, then

$$\begin{aligned} \delta_{t+1}^2 =& \|w_t - w'_t - \alpha_t(\nabla f(w_t, z) - \nabla f(w'_t, z))\|_2^2 \\ =& \|w_t - w'_t\|^2 + \alpha_t^2 \|\nabla f(w_t, z) - \nabla f(w'_t, z))\|_2^2 \\ & - 2\alpha_t \langle w_t - w'_t, \nabla f(w_t, z) - \nabla f(w'_t, z)\rangle \\ \leq& \|w_t - w'_t\|^2 = \delta_t^2, \end{aligned} \tag{29}$$

provided that $\alpha_t \leq 2/\beta$, where we used (21).

If the two algorithms choose different data examples z and z', then

$$\begin{aligned}\delta_{t+1} &= \|w_t - w'_t - \alpha_t(\nabla f(w_t, z) - \nabla f(w'_t, z))\|_2 \\ &\leq \|w_t - w'_t\|_2 + \alpha_t \left(\|\nabla f(w_t, z)\|_2 + \|\nabla f(w'_t, z))\|_2\right) \\ &\leq \delta_t + 2\alpha_t L.\end{aligned} \tag{30}$$

Since the first case happens with probability $1 - 1/n$, and the second case happens with probability $1/n$. In average, we have

$$\begin{aligned}\mathbb{E}[\delta_{t+1}] &\leq \left(1 - \frac{1}{n}\right)\mathbb{E}[\delta_t] + \frac{1}{n}\mathbb{E}[\delta_t + 2\alpha_t L] \\ &= \mathbb{E}[\delta_t] + \frac{2\alpha_t L}{n}.\end{aligned} \tag{31}$$

Overall, we have

$$\mathbb{E}[\delta_T] \leq \frac{2L}{n}\sum_{i=1}^{T}\alpha_t, \tag{32}$$

and

$$|f(w_T, z) - f(W'_T, z)| \leq L\|w_T - w'_T\|_2 \leq \frac{2L^2}{n}\sum_{i=1}^{T}\alpha_t. \tag{33}$$

This finishes the proof.

■

elmos / May 28, 2017