CS 224 Fall 2023 Scribe: Naomi Bashkansky
October 23, 2023

Supervised learning II: noise sensitivity, one-hidden layer MLPs, Hermite analysis, CSQ algorithms

1 Noise Sensitivity and Fourier Concentration

1.1 Recap: Low-Degree Approximation Implies Learnability

Last time, we discussed that Boolean functions $f : \{\pm 1\}^n \to \{\pm 1\}$ admits a nice decomposition, called the Fourier expansion:

$$f(x) = \sum_{S \subseteq [n]} \hat{f}[S] \cdot x_S$$

Note that $x_S = \prod_{i \in S} x_i$. This is a multilinear polynomial - that is, no variable x_i appears squared, cubed, etc. We call $\hat{f}[S]$ the Fourier coefficient and x_S the Parity/Fourier character/basis function.

Suppose that f is well-approximated by its low-degree truncation:

$$f^{\leq t}(x) = \sum_{S : |S| \leq t} \hat{f}[S] \cdot x_S.$$

That is, the error we get by approximating using the low-degree truncation is small, such that $\sum_{S:|S|>t} \hat{f}[S]^2 \leq \epsilon$. Then: given $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ for $\mathbf{x}_i \sim \{\pm 1\}$, solving the "L1 polynomial regression",

$$\min_{p:\deg(p) \leq t} \frac{1}{n} \sum_{i=1}^{n} |y_i - p(\mathbf{x}_i)|,$$

results in an estimator p that achieves test loss $\mathrm{OPT} + O(\epsilon)$. Thus, Fourier Concentration is all you need for agnostically learning Boolean Functions.

1.2 Noise-Stability Implies a Low-Degree Approximation

Noise sensitivity can help us show that a given concept class is well-approximated by a low-degree polynomial (and thus, that functions in that class are agnostically learnable).

Definition 1. *Given $0 < \eta < \frac{1}{2}$, the noise sensitivity $NS_\eta(f)$ is given by*

$$NS_\eta(f) := P(f(\mathbf{x}) \neq f(\mathbf{x}')),$$

where $\mathbf{x} \sim \{\pm 1\}^n$ and $\mathbf{x}'$ is given by flipping each bit of $\mathbf{x}$ independently with probability η.

Intuitively, high-degree functions (e.g. $f = Parity$) are very noise sensitive, since flipping even one bit changes the output dramatically. On the flip side, if a function is not noise-sensitive, then it should be well-approximated by a low-degree polynomial. Formally, we can say the following:

Theorem 1. *[KOS04] Suppose $NS_\eta(f) \leq m(\eta)$. Let $t \asymp 1/m^{-1}\left(\Theta(\epsilon)\right)$. Then $\sum_{S:|S|>t} \hat{f}[S]^2 \leq \epsilon$.*

Proof. First, we will prove the following equation:

$$NS_\eta(f) = \frac{1}{2} - \frac{1}{2} \sum_{S \subseteq [n]} (1-2)^{|S|} \hat{f}[S]^2 \tag{1}$$

The following is the proof for equation 1.

We use the Fourier transformation.

$$\begin{aligned} 1 - 2NS_\eta(f) &= (1 - NS_\eta(f)) - (NS_\eta) \\ &= P(f(\mathbf{x}) = f(\mathbf{x}')) - P(f(\mathbf{x}) \neq f(\mathbf{x}')) \\ &= E[f(\mathbf{x})f(\mathbf{x}')] \\ &= \sum_{S,T \subseteq [n]} \hat{f}[S]\hat{f}[T]E[x_S x'_T] \end{aligned}$$

Simplifying for $E[x_S x'_T]$, we get:

$$\begin{aligned} E[x_S x'_T] &= E\left[\left(\prod_{i \in S \setminus T} x_i\right)\left(\prod_{i \in S \cap T} x_i x'_i\right)\left(\prod_{i \in T \setminus S} x'_i\right)\right] \\ &= \prod_{i \in S \setminus T} E[x_i] \prod_{i \in S \cap T} E[x_i x'_i] \prod_{i \in T \setminus S} E[x'_i] \\ &= \begin{cases} 0 & \text{if } S \neq T \\ (1-2\eta)^{|S|} & \text{if } S = T \end{cases} \end{aligned}$$

Thus, we're left with:

$$\begin{aligned} 1 - 2NS_\eta &= \sum_{S \subseteq [n]} (1-2\eta)^{|S|} \hat{f}[S]^2 \\ \implies NS_\eta(f) &= \frac{1}{2} - \frac{1}{2} \sum_{S \subseteq [n]} (1-2\eta)^{|S|} \hat{f}[S]^2 \end{aligned}$$

Second, to complete our proof, we will show that:

$$\sum_{S\subseteq[n]:|S|\geq 1/\eta} \hat{f}[S]^2 \lesssim NS_\eta(f). \tag{2}$$

To show equation 2, we will use the fact that $(1-x)^a \leq e^{-ax}$, which implies that, if $|S| \geq 1/\eta$, then $(1-2\eta)^{|S|} \leq e^{-2\eta|S|} \leq e^{-1/2}$. We will also use the fact that $1 = E[f^2] = \sum_{S\subseteq[n]} \hat{f}[S]^2$. Now, we can say:

$$\begin{aligned}
2NS_\eta(f) &= 1 - \sum_{S\subseteq[n]} (1-2\eta)^{|S|}\hat{f}[S]^2 \\
&= \sum_{S\subseteq[n]} \hat{f}[S]^2 \left(1-(1-2\eta)^{|S|}\right) \\
&\geq \sum_{|S|\geq 1/\eta} \hat{f}[S]^2 \left(1-(1-2\eta)^{|S|}\right) \\
&\geq \left(1-e^{1/2}\right) \sum_{|S|\geq 1/\eta} \hat{f}[S]^2
\end{aligned}$$

Finally, notice that $t = 1/\eta$ implies that $O(\epsilon) = m(\eta) \geq NS_\eta(f)$. This completes our proof. □

Furthermore, we can say the following about functions of k halfspaces with low noise sensitivity. Recall that a halfspace is a function of the form $g(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x}\rangle)$.

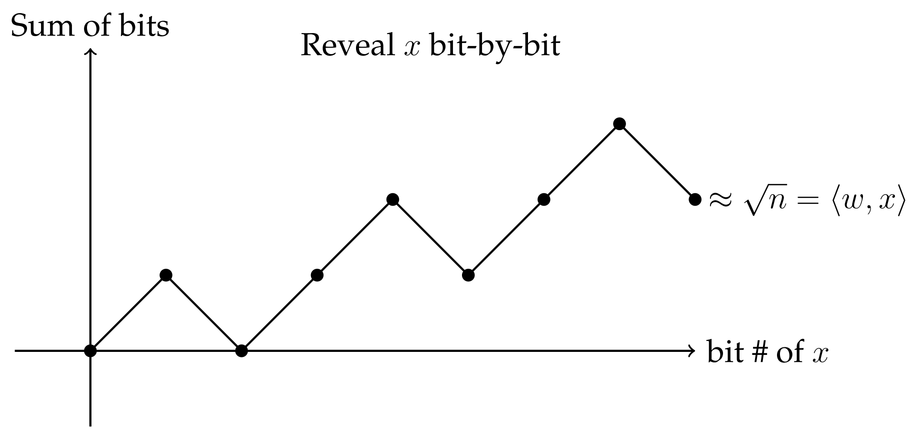
Theorem 2. *[KOS04] Any function f of k halfspaces satisfies $NS_\eta(f) \lesssim k\sqrt{\eta}$. Furthermore, f can be approximately learned in time $n^{O(k^2/\epsilon^2)}$.*

Proof. The proof is involved, so we will not give the full proof. Instead, we will prove a baby version to give intuition. Suppose $\mathbf{w} = [1,\ldots,1]^T$. Then $f(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x}\rangle)$ is simply the majority function: if and only if the majority of the entries in x are $+1$, $f(\mathbf{x})$ will output $+1$.

Suppose we reveal x bit-by-bit, and we keep track of the sum of all bits. This is a random walk that is approximately Gaussian. The magnitude of our final point should be about $\sqrt{n}$.

Now, x' flips each bit of x with probability η. We flip approximately ηn bits. Equivalently, we extend the random walk about ηn steps. We want to know the probability that the displacement in the opposite-to-original direction is more than $\sqrt{n}$. To upper bound this probability, we will note that $E[\text{displacement}] = \sqrt{\eta n}$ and we will use the Markov inequality to say the following:

$$\Pr[\text{displacement} \geq \sqrt{n}] \leq \sqrt{\eta} \tag{3}$$



Now, suppose we have a function f of k halfspaces, i.e.:

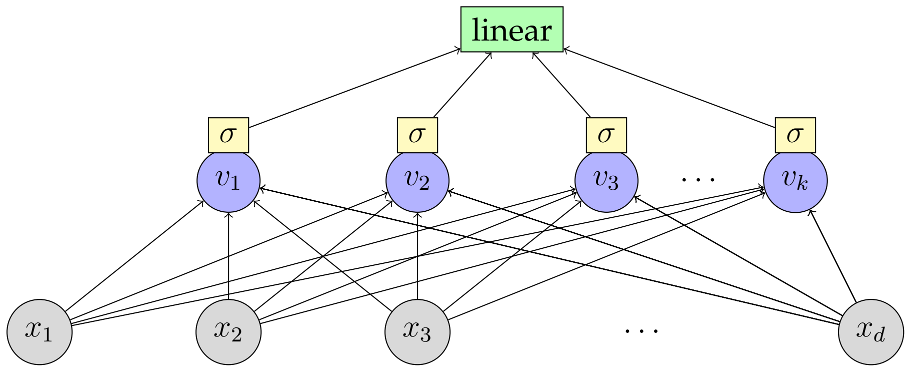
$$f(x) = h(\text{sgn}(\langle \mathbf{w_1}, \mathbf{x}\rangle), \dots, \text{sgn}(\langle \mathbf{w_k}, \mathbf{x}\rangle).$$

Then we want to prove that $NS_\eta(f) \lesssim k\sqrt{n}$. Indeed:

$$\begin{aligned} NS_\eta(f) &= \Pr[f(\mathbf{x}) \neq f(\mathbf{x}')] \\ &\leq \Pr[\exists i \in [k] : \text{sgn}(\langle \mathbf{w_i}, \mathbf{x})\rangle \neq \text{sgn}(\langle \mathbf{w_i}, \mathbf{x}'\rangle)] \\ &\leq \sum_{i=1}^{k} \Pr[\text{sgn}(\langle w_i, x)\rangle \neq \text{sgn}(\langle \mathbf{w_i}, \mathbf{x}'\rangle)] \qquad \text{(Union-Bound)} \\ &\lesssim k\sqrt{\eta} \end{aligned}$$

Let $\eta = \frac{\epsilon^2}{k^2}$. Then $k\sqrt{\eta} = \epsilon$. The degree of our low-degree approximation of f is $1/\eta = O(k^2/\epsilon^2)$. And any degree-$t$ approximation of f can be learned in time n^t, completing our proof. □

2 One-hidden-layer MLPs



We turn to a more realistic scenario for Probably Approximately Correct (PAC) learning: functions that have continuously-valued input and output. A function is a mapping $f : \mathbb{R}^d \to \mathbb{R}$. Our activation function σ can be more complex than the sign, such as ReLU or tanh. We can think of this one-hidden-layer multilayered perceptron as the "model organism" for neural networks: though it is simple, it can serve as a rich testbed for algorithms such as nonconvex optimization, tensor methods, kernel methods, and representation learning.

Succintly, we can write a one-hidden-layer MLP as:

$$f(\mathbf{x}) = \sum_{i=1}^{k} \lambda_i \sigma(\langle \mathbf{v_i}, \mathbf{x} \rangle), \quad \text{for } \|\mathbf{v_i}\| = 1 \tag{4}$$

This is the simplest nontrivial neural network, but it is expressive. We will state but not prove the following theorem:

Theorem 3. *[Kol56] [Arn57] If the $\langle \mathbf{v_i}, \mathbf{x} \rangle$'s are replaced with $\sum_j \phi_{ij}(x_j)$'s, then $f(\mathbf{x})$ can realize any continuous function over a compact support.*

Now, we will begin learning the toolbox for analyzing PAC-learning on continuously-valued inputs.

3 Hermite Polynomial, Low-Degree Approximation

The canonical distribution of inputs in this setting - the analogue of the uniform distribution over the hypercube - is the Gaussian. That is, we let $\mathcal{D}_X \sim \mathcal{N}(\mathbf{0}, \mathrm{Id}_d)$.

There is an analogous toolbox for this setting: in place of the Fourier characters $\{\mathbf{x} \to x_S\}$, we have the Hermite polynomials $\{h_\alpha\}$, which form an orthonormal basis (of all square-integrable functions) with respect to the Gaussian measure, i.e.:

$$\int_{-\infty}^{\infty} h_\alpha(\mathbf{x}) h_\beta(\mathbf{x}) \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} = \begin{cases} 1 & \text{if } \alpha = \beta \\ 0 & \text{if } \alpha \neq \beta \end{cases}$$

Or, equivalently:

$$\mathbb{E}_{\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathrm{Id}_d)}[h_\alpha(\mathbf{x}) h_\beta(\mathbf{x})] = \mathbb{1}[\alpha = \beta]$$

The first three Hermite polynomials in one dimension are $h_1(x) = x, h_2(x) = \frac{1}{\sqrt{2}}(x^2 - 1), h_3(x) = \frac{1}{\sqrt{6}}(x^3 - 3x)$. The exact functional form is not important for us, but interested readers can see Wikipedia for "probabilist's Hermite Polynomials." In higher dimensions, the Hermite polynomials are simply products of one-dimensional Hermite polynomials: for tuple $\alpha = (\alpha_1, \ldots, \alpha_d)$,

$$h_\alpha(\mathbf{x}) = \prod_{i=1}^{d} h_{\alpha_i}(x_i).$$

Instead of a Fourier expansion, we have a Hermite expansion. We can express any square-integrable function as:

$$f(\mathbf{x}) = \sum_{\alpha} \hat{f}_\alpha \cdot h_\alpha(\mathbf{x}),$$

where $\hat{f}_\alpha$ is the Hermite coefficient. As before, we can empirically estimate $\mathbb{E}[f \cdot h_\beta(\mathbf{x})]$ using the training data and then extract the coefficients using orthonormality:

$$\mathbb{E}[f \cdot h_\beta(\mathbf{x})] = \sum_{\alpha} \hat{f}_\alpha \cdot \mathbb{E}[h_\alpha(\mathbf{x}) h_\beta(\mathbf{x})] = \hat{f}_\beta.$$

Furthermore, we have the Gaussian analogue of Plancherel's:

$$\mathbb{E}_{x\sim\mathcal{N}(0,1)}\left[f(x)^2\right] = \sum_{\alpha} \hat{f}_\alpha^2 := \|f\|_2^2$$

Note that the L2 norm of a function f is defined as $\|f\|_2 = \sqrt{\int |f(x)|^2 \mathrm{dx}}$.

There exists analogous notions of low-degree approximations in this Gaussian setting. If you plot the square of the Hermite coefficients as a function of the degree l, you get a nice polynomial decay that's about $1/l^{5/4}$. As the degree gets larger, the Hermite coefficients get smaller. This gives an intuition for the following two theories that we will not prove. In the one-dimensional case:

Theorem 4 (folklore)**.** *There exists a degree-$O(1/\epsilon^{4/3})$ polynomial $p : \mathbb{R} \to \mathbb{R}$ such that:*

$$\|p(\cdot) - \mathrm{ReLU}(\cdot)\|_2 \le \epsilon$$

In the d-dimensional case:

Theorem 5 (folklore)**.** *There exists a degree-$O(1/\epsilon^{4/3})$ polynomial $p : \mathbb{R} \to \mathbb{R}$ such that $\forall \mathbf{v} \in \mathbb{S}^{d-1}$:*

$$\|p(\langle \mathbf{v}, \cdot\rangle) - \mathrm{ReLU}(\langle \mathbf{v}, \cdot\rangle)\|_2 \le \epsilon$$

This implies there exists a $d^{O(1/\epsilon^{4/3})}$-time algorithm (polynomial regression) for agnostically learning $\mathbf{x} \to \mathrm{ReLU}(\langle \mathbf{v}, \mathbf{x}\rangle)$ over Gaussian inputs.

More generally, for one-hidden-layer MLPs $f(x) = \sum_{i=1}^{k} \lambda_i \mathrm{ReLU}(\langle \mathbf{v}_i, \mathbf{x}\rangle)$, the degree-$t = \Theta(1/\epsilon^{4/3})$ Hermite truncation $f^{\le t} = \sum_i \lambda_i p_i$ satisfies:

$$\|f - f^{\le t}\|_2 \le \sum_i |\lambda_i| \cdot \|p_i - \mathrm{ReLU}(\langle \mathbf{v}_i, \cdot\rangle)\|_2 \le \epsilon \|\lambda\|_1.$$

So far, we've found that we can agnostically learn ReLUs in time $d^{\mathrm{poly}(1/\epsilon)}$ and one-hidden-layer MLPs in time $d^{\mathrm{poly}(\|\lambda\|_1/\epsilon)}$.

Now, we ask ourselves, for one-hidden-layer MLPs:

1. Can we remove the ϵ dependence?
2. Can we remove the $\|\lambda\|_1$ dependence?
3. Can we handle more layers?

4 Tensor Methods

In the setting where the one-hidden-layer MLPs are non-degenerate - that is, the v_i's are robustly linearly independent - we can turn this supervised learning problem into a tensor decomposition problem.

As our starting point, we will organize the Hermite polynomials into a tensor, sometimes called the (higher-order) score function (of a Gaussian):

$$S_l(\mathbf{x}) \in \left(\mathbb{R}^d\right)^{\otimes l}, \quad (S_l)_{i_1,\ldots,i_l} = \prod_i \sqrt{\alpha_{i!}} h_{\alpha_i}(\mathbf{x})$$

where $\alpha = (\alpha_1, \ldots, \alpha_d)$ is a tuple where α_i is the number of occurrences of i among the $\{i_1, \ldots, i_l\}$. The first three score functions are: $s_1(\mathbf{x}) = \mathbf{x}, S_2(\mathbf{x}) = \mathbf{x}\mathbf{x}^T - \mathrm{Id}_d, S_3(\mathbf{x}) = \mathbf{x}^{\otimes 3} - x \otimes_3 \mathrm{Id}_d$. We can think of each S_j as the tensor equivalent of the Hermite polynomial h_j. More succinctly, we could write:

$$S_l(\mathbf{x}) = \frac{(-1)^l}{\gamma(x)} \cdot \nabla^l \gamma(x),$$

where γ is the d-dimensional Gaussian PDF, and $\nabla^l \gamma(x)$ is its l-th derivative. The upshot of this is that we want to approximate f, and the correlation of f and the score function S_l gives us a tensor that will be low-rank (and so we can through Jennrich's algorithms or sum-of-squares at it).

Theorem 6. *Stein's identity: If f is sufficiently "regular," then:*

$$\mathbb{E}[f(\mathbf{x}) \cdot S_l(\mathbf{x})] = \mathbb{E}[\nabla^l f(x)].$$

Proof. We will prove a baby version of Stein's identity using Gaussian integration by parts. In particular, we will prove the statement that for $x \sim \mathcal{N}(0, 1)$:

$$\mathbb{E}[f(x) \cdot x] = \mathbb{E}[f'(x)] \tag{5}$$

$$\mathbb{E}[f(x) \cdot (x^2 - 1)] = \mathbb{E}[f''(x)] \tag{6}$$

Let γ denote the Gaussian density. Note that $x\gamma(x) = -\gamma'(x)$, and that $\gamma''(x) = (-x\gamma(x))' = (x^2-1)\gamma(x)$. Then:

$$\begin{aligned}
\mathbb{E}[f(x)x] &= \int_{-\infty}^{\infty} f(x)x\delta(x)\mathrm{dx} \\
&= \left(\sum_{-\infty}^{\infty} f'(x)\gamma(x)\mathrm{dx}\right) + \left(f(x)\delta(x)\Big|_{\infty}^{\infty}\right) \\
&= \mathbb{E}[f'(x)] + 0 \\
&= \mathbb{E}[f'(x)] \\
\mathbb{E}[f(x)(x^2-1)] &= \int_{-\infty}^{\infty} f(x)(x^2-1)\gamma(x)\mathrm{dx} \\
&= \left(\int_{-\infty}^{\infty} f'(x)(x\gamma)\mathrm{dx}\right) + \left(f(x)x\gamma(x)\Big|_{-\infty}^{\infty}\right) \\
&= \int_{-\infty}^{\infty} f''(x)\gamma(x)\mathrm{dx} + \left(f'(x)\gamma(x)\Big|_{-\infty}^{\infty}\right) + \left(f(x)x\gamma(x)\Big|_{-\infty}^{\infty}\right) \\
&= \mathbb{E}[f''(x)] + 0 + 0 \\
&= \mathbb{E}[f''(x)]
\end{aligned}$$

□

Happily, Stein's identity gives us a one-line proof of problem 1a of pset 1! Recall the setup, that we want to construct a tensor which we an run Jennrich's on to get the $\mathbf{v}_i$'s that make up f. That is,

$$f(\mathbf{x}) = \sum_{i=1}^{k} \lambda_i \langle \mathbf{v}_i, \mathbf{x}\rangle^3.$$

Applying Stein's identity, we have:

$$\mathbb{E}[f(\mathbf{x}) \cdot S_3(\mathbf{x})] = \mathbb{E}[\nabla^3 f(\mathbf{x})] = 6\sum_{i=1}^{k} \lambda_i \mathbf{v}_i^{\otimes 3},$$

which we can use Jennrich's on. Note that the last equality is because:

$$(\nabla^3 \langle \mathbf{v}, \mathbf{x}\rangle^3)_{abc} = \frac{\partial^3}{\partial x_a \partial x_b \partial x_c} \langle \mathbf{v}, \mathbf{x}\rangle^3 = 6v_a v_b v_c.$$

More generally, for any smooth activation σ, consider the following function f that we want to learn:

$$f(\mathbf{x}) = \sum_{i=1}^{k} \lambda_i \sigma(\langle \mathbf{v}_i, \mathbf{x}\rangle).$$

Applying Stein's identity, we have:

$$\mathbb{E}[f(\mathbf{x}) \cdot S_l(\mathbf{x})] = \mathbb{E}[\nabla^l f(\mathbf{x})] = \mathbb{E}\left[\sigma^{(l)}(\mathbf{x})\right] \sum_{i=1}^{k} \lambda_i \mathbf{v}_i^{\otimes l},$$

where $\sigma^{(l)}$ is the l-th derivative of the activation. Importantly, we get a low-rank tensor multiplied by some constant factor. Even if σ is not smooth, as long is it is square-integrable, we can show that:

$$\mathbb{E}[f(\mathbf{x}) \cdot S_l(\mathbf{x})] = \sqrt{l!}\hat{\sigma}_l \sum_{i=1}^{k} \lambda_i \mathbf{v}_i^{\otimes l},$$

where $\hat{\sigma}_l$ is the l-th Hermite coefficient of your activation function σ. Thus, as proven in [JSA15], learning one-hidden-layer MLPs reduces to tensor decomposition. When $\mathbf{v}_i$'s are robustly linearly independent, there is an algorithm that runs in $poly(d, k, 1/\epsilon)$ time and is proper. It's proper in that, rather than outputting a low-degree polynomial, we're outputting the parameters of a network.

However, if we make no assumptions about the weights $\mathbf{v}_1, \ldots, \mathbf{v}_k$, we have an issue. We can no longer hope to recover the parameters: for instance, we run into a problem with the two networks $\left[\mathrm{ReLU}(\langle \mathbf{v}, \mathbf{x}\rangle) - \mathrm{ReLU}(\langle \mathbf{v} + \epsilon \cdot \mathbf{w}, \mathbf{x}\rangle)\right]$ vs. the network that is the constant 0. Nonetheless, we still have the following two theorems:

Theorem 7. *[CN23] For learning arbitrary one-hidden-layer MLPs over Gaussians, we have a proper algorithm that runs in time poly*$(d^{k^2}, 1/\epsilon)$.

Theorem 8. *[DK23] For learning arbitrary one-hidden-layer MLPs over Gaussians, we have an improper algorithm that runs in time poly*$(d^k, 1/\epsilon)$.

Intuitively, both algorithms are bottlenecked at d^{k^c} because they work with $\sum_i \lambda_i \mathbf{v}_i^{\otimes l}$ for $l = 2, \ldots, k^c$.

5 Lower bounds

The question is: can we get away with just using lower-degree tensors than k^c? The answer, we will see, is no. Indeed, there is a simple, two-dimensional example for which the corresponding tensor is 0 unless we go up to a high degree.

Theorem 9. *[DKKZ20] There exists a choice of* $\{\sigma_i, \mathbf{v}_i\}_{i=1}^k$ *such that for all* $1 \leq l \leq k/2$, *we have:*

$$\sum_i \lambda_i \mathbf{v}_i^{\otimes l} = 0$$

Specifically, take $\lambda_i - (-1)^i$ *and* $v_i = \left(\cos\left(\frac{2\pi i}{k}\right), \sin\left(\frac{2\pi i}{k}\right)\right)$.

The above theorem suggests that any tensor-based algorithm must incur $d^{\Omega(k)}$ runtime.

What about for non-tensor algorithms, like kernel methods or gradient decent? These approaches all have one major thing in common: they only use the correlations between label y and functions of $\mathbf{x}$.

1. Tensor methods: $\underbrace{\mathbb{E}[yS_l(\mathbf{x})]}_{\text{correlation of x \& y}}$.

2. Kernel methods: $\min_x \mathbb{E}\left[\left(y - \sum_j c_j\phi_j(\mathbf{x})\right)^2\right]$ for a basis of features $\{\phi_j(\mathbf{x})\}$.

   This is equivalent to $\min_x \mathbb{E}\left[\left(\sum_j c_j\phi_j(\mathbf{x})\right)^2\right] - \underbrace{2\mathbb{E}\left[y \cdot \sum_j c_j\phi_j(\mathbf{x})\right]}_{\text{correlation of x \& y}} + \mathbb{E}[y^2]$.

3. Gradient descent: $\nabla_\theta \left\{\mathbb{E}\left[(y - f_\theta(\mathbf{x}))^2\right]\right\} = 2\mathbb{E}[f_\theta(\mathbf{x})\cdot\nabla f_\theta(]boldx)] - \underbrace{2\mathbb{E}[y \cdot \nabla f_\theta(\mathbf{x})]}_{\text{correlation of x \& y}}$.

These are all "correlational statistical query" algorithms. In the Correlational Statistical Query (CSQ) model, we're not allowed to view individual data points. Instead, we only get a population-level statistic. In particular, we feed a function $\psi : \mathbb{R}^d \to \mathbb{R}$ to an Oracle, and the Oracle produces outputs a noisy estimate for the correlation between y and $psi(x)$, i.e. $\mathbb{E}[y \cdot \psi(x)] +$ noise. We say the noise is bounded: $|\text{noise}| \leq \tau$, where the tolerance τ roughly corresponds to $\sqrt{1/\#\text{ samples}}$. Unfortunately, we have strong evidnece that any method that falls under the CSQ model cannot beat d^k:

Theorem 10. *In CSQ, learning one-hidden-layer MLP's over Gaussians, even to constant error, requries* $2^{d^{\Theta(1)}}$ *queries or tolerance* $d^{-\Omega(k)}$.

We will see the proof in the computational complexity unit next week.

In summary, recall our guiding questions. For one-hidden-layer MLP's, we could learn a low-degree approximation in time $d^{\text{poly}(\|\lambda_1\|/\epsilon)}$. We asked:

1. Can we impprove the ϵ dependance? Yes!

2. Can we improve the $\|\lambda\|_1$ dependenace? Not with CSQ, though we'll see how to improve on this by going beyond correlational statistical query models.

3. Can we handle more layers? We'll answer this when we go beyond CSQ.

References

[Arn57] Vladimir Arnold. On functions of three variables. *Proceedings of the USSR Academy of Sciences*, 114:679–681, 1957.

[CN23] Sitan Chen and Shyam Narayanan. A faster and simpler algorithm for learning shallow networks. 2023.

[DK23] Ilias Diakonikolas and Daniel M. Kane. Efficiently learning one-hidden-layer relu networks via schur polynomials. 2023.

[DKKZ20] Ilias Diakonikolas, Daniel M. Kane, Vasilis Kontonis, and Nikos Zarifis. Algorithms and sq lower bounds for pac learning one-hidden-layer relu networks. 2020.

[JSA15] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. 2015.

[Kol56] Andrey Kolmogorov. On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *Proceedings of the USSR Academy of Sciences*, 108:179–182, 1956.

[KOS04] Adam R. Klivans, Ryan O'Donnell, and Rocco A. Servedio. Learning intersections and thresholds of halfspaces. *Journal of Computer and Science System*, 68(4):808–840, 2004.