Feature learning and limitations of kernels

By Berkan and Johannes

Guiding Questions

- Why is **feature learning** important to deep learning?
- What is the connection between feature learning and adaptivity?
- Why is the **NTK** unable to do feature learning?

Any ideas?

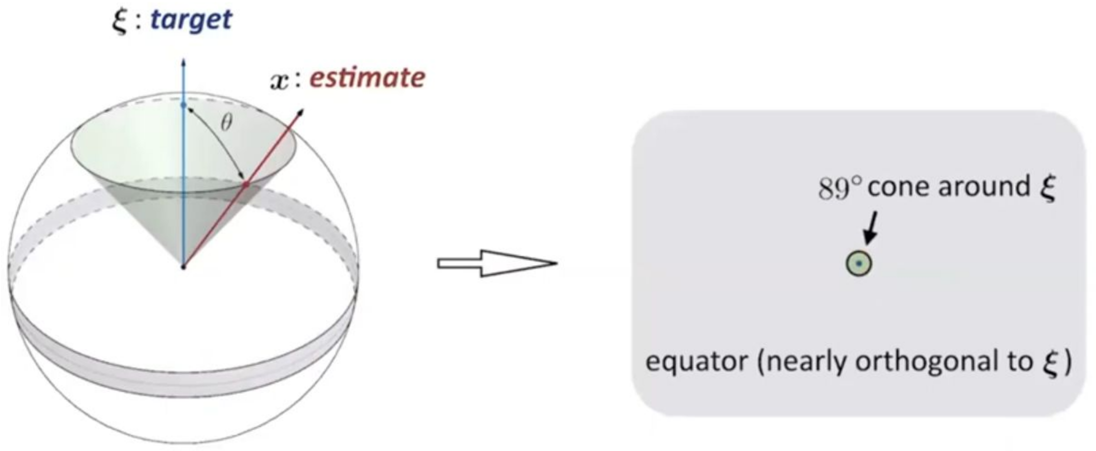
What is Adaptability?

Comparing kernel methods (KM) and neural networks (NN).

Credits: https://francisbach.com/quest-for-adaptivity/

Curse of Dimensionality

- Our intuitions fall apart in higher dimensions
- Exponential decay in the probability of a good guess



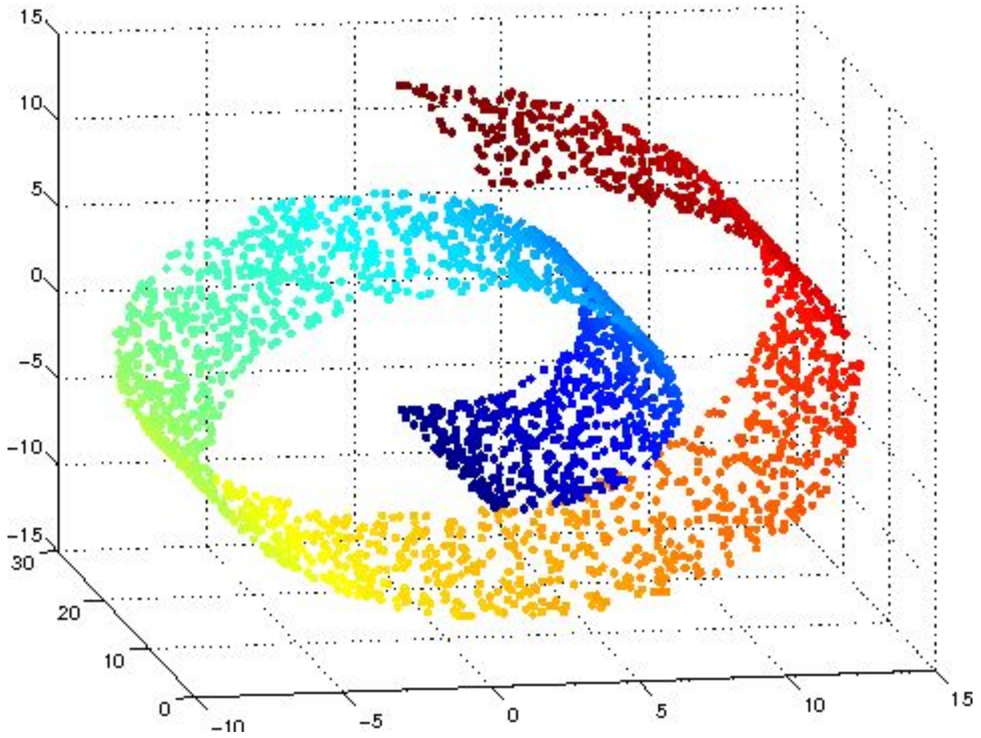
For some desired error $\epsilon < 1$, $n = \Omega((1/\epsilon)^d)$

Credit: Yue Lu | Nov 30, 2021 | Learning by Random Features and Kernel Random Matrices

Adaptability to the rescue!!

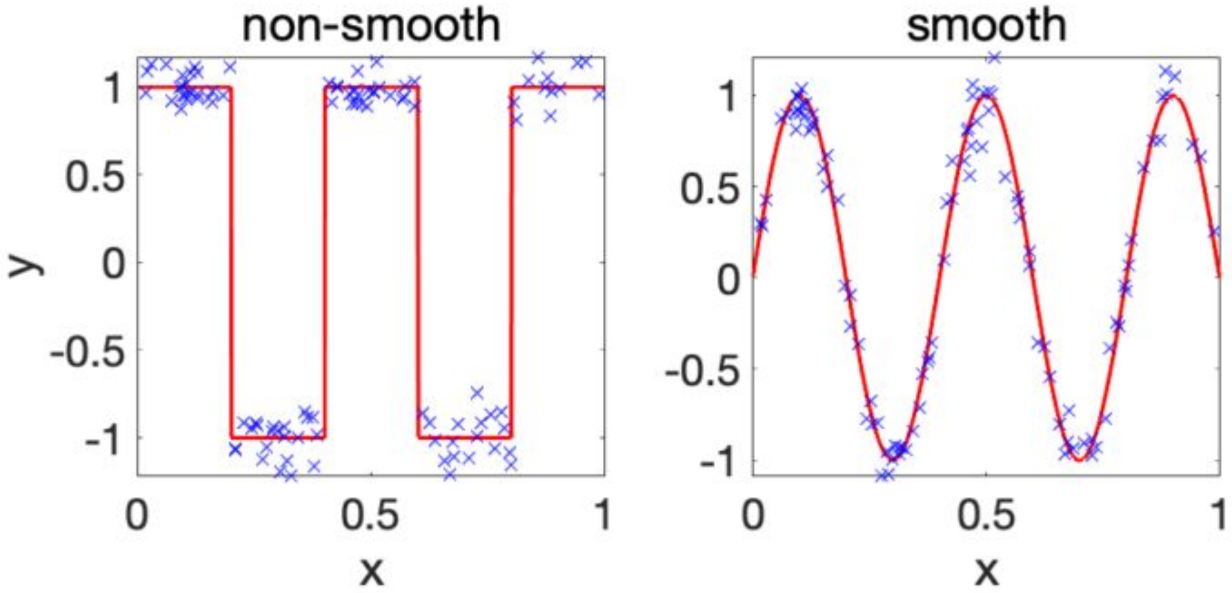
Low Dimensional Support

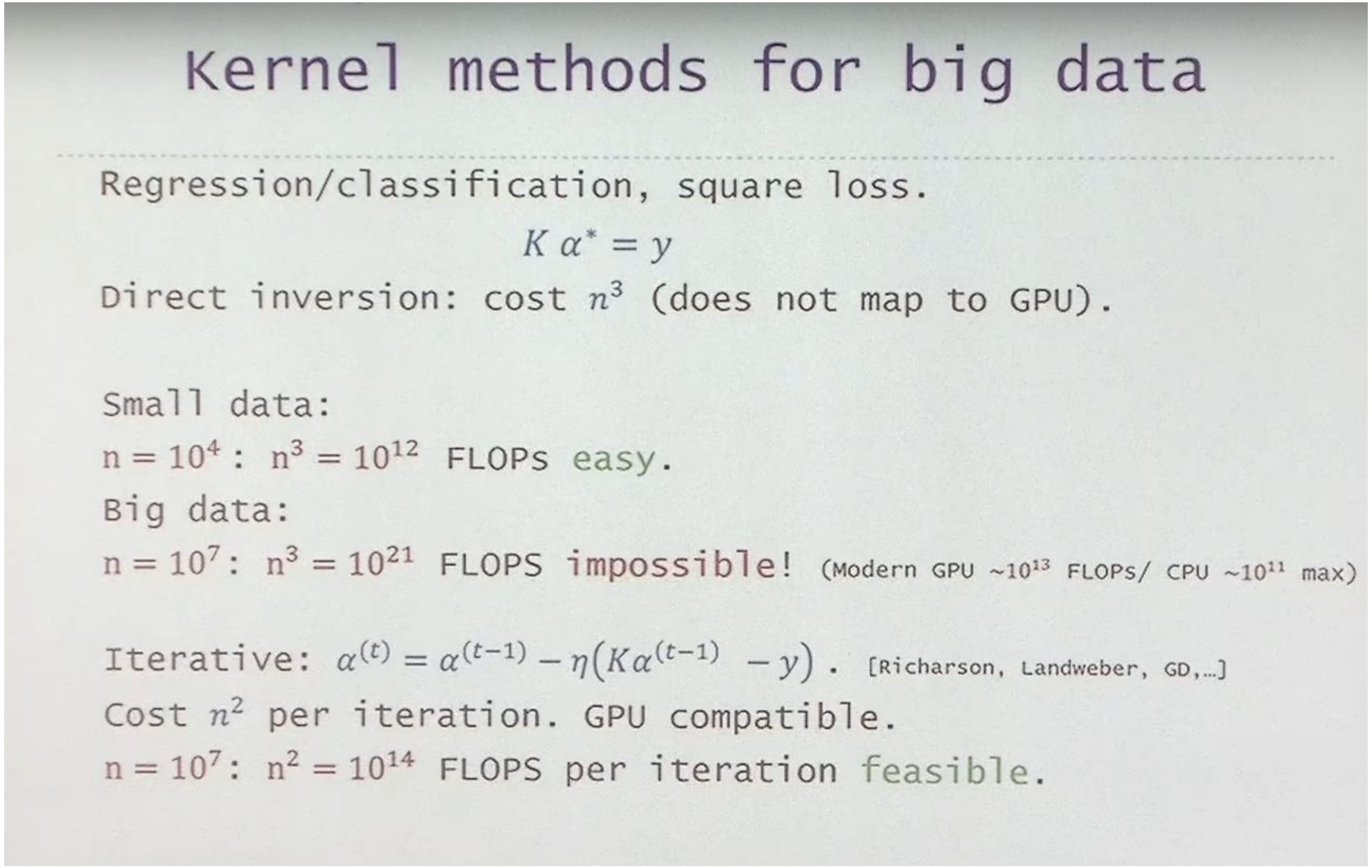
- Data lies on lower dimensional manifold.
- Ambient dimension
    - Total # dimensions
- Intrinsic dimension
    - Number of variables needed to represent the data without loss of information
- Both KMs and NNs are adaptive!



Smoothness

- Both KMs and NNs are adaptive!





Kernel methods for big data

Regression/classification, square loss.

$$K\alpha^* = y$$

Direct inversion: cost n^3 (does not map to GPU).

Small data:

$n = 10^4$: $n^3 = 10^{12}$ FLOPs easy.

Big data:

$n = 10^7$: $n^3 = 10^{21}$ FLOPS impossible! (Modern GPU ~10^{13} FLOPs/ CPU ~10^{11} max)

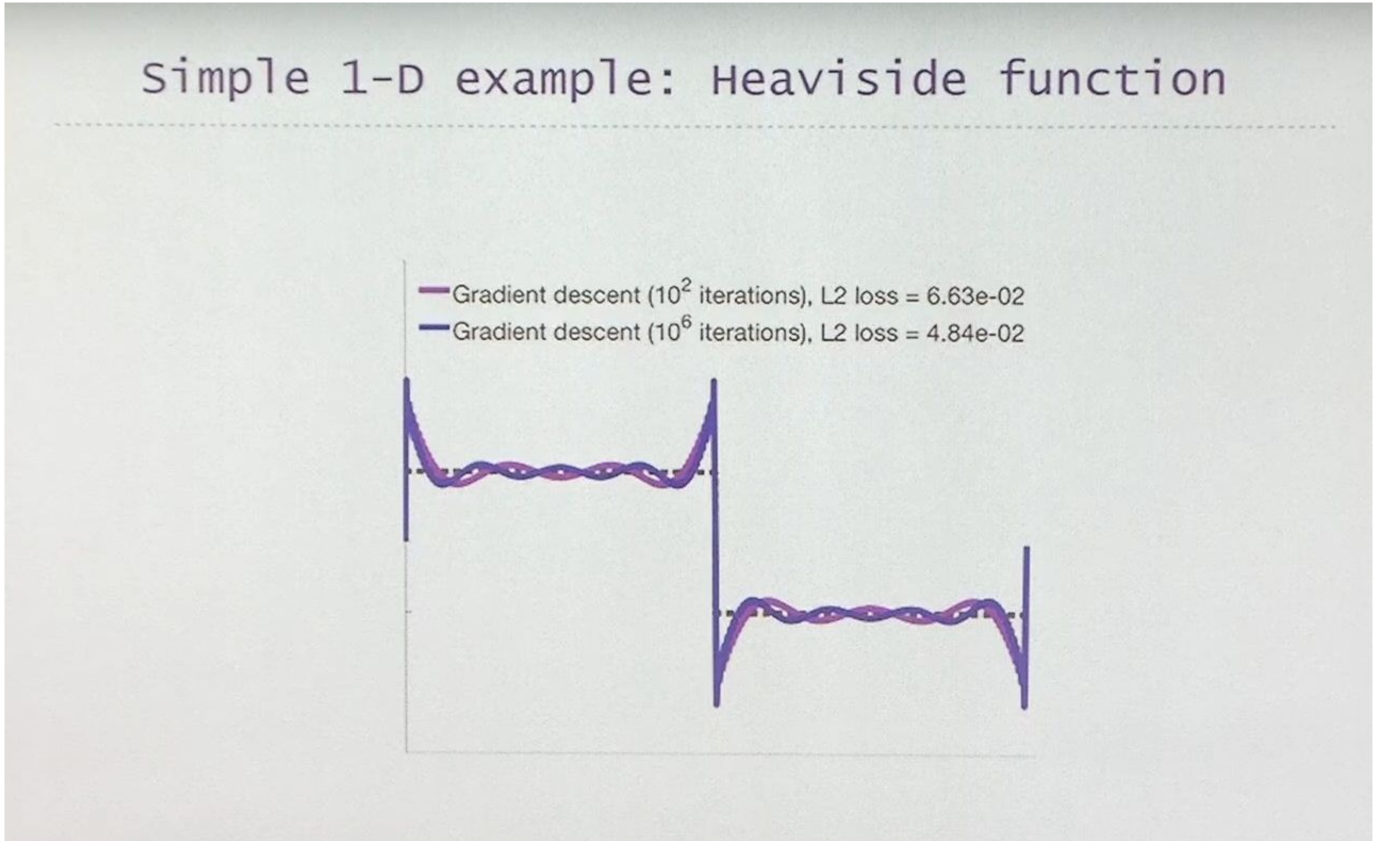
Iterative: $\alpha^{(t)} = \alpha^{(t-1)} - \eta\left(K\alpha^{(t-1)} - y\right)$. [Richarson, Landweber, GD,…]

Cost n^2 per iteration. GPU compatible.

$n = 10^7$: $n^2 = 10^{14}$ FLOPS per iteration feasible.

Credit:
Misha Belkin,
Simons Institute

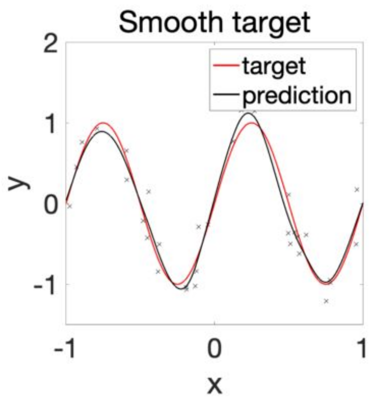
Even taking lots of iterations, Kernel Methods

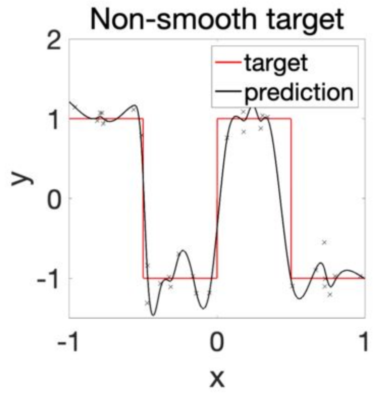


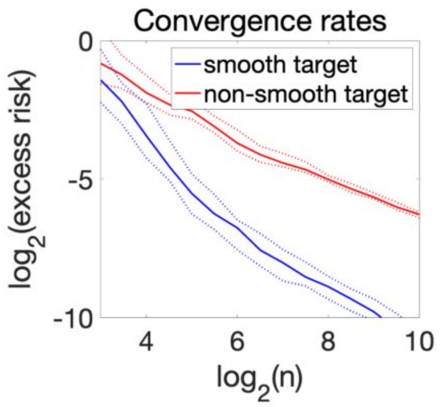
Simple 1-D example: Heaviside function

KMs + NNs and Smoothness

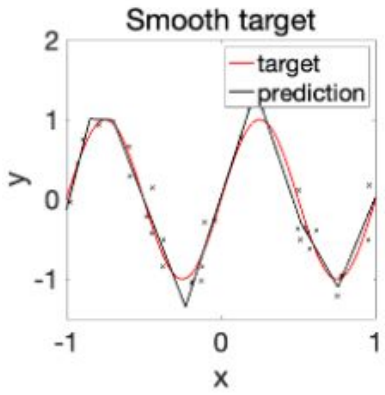
KMs

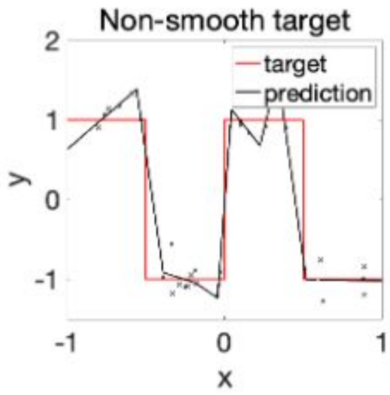


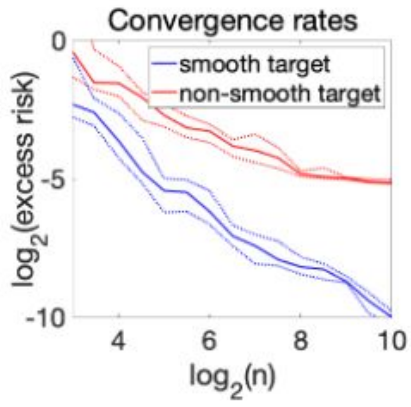




NNs

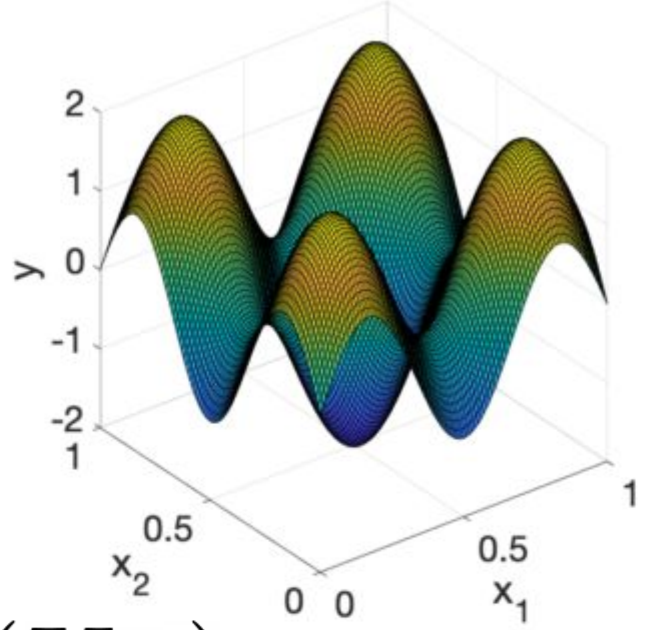


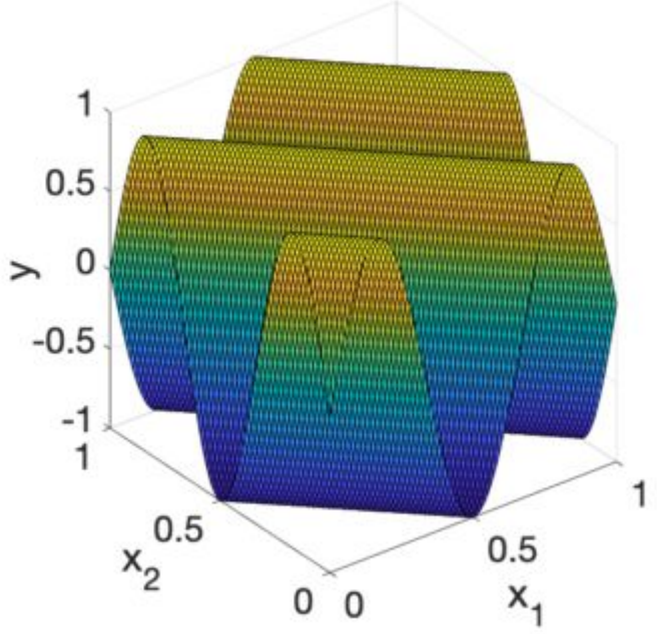




Latent Variables

- NNs are adaptive but KMs aren't!



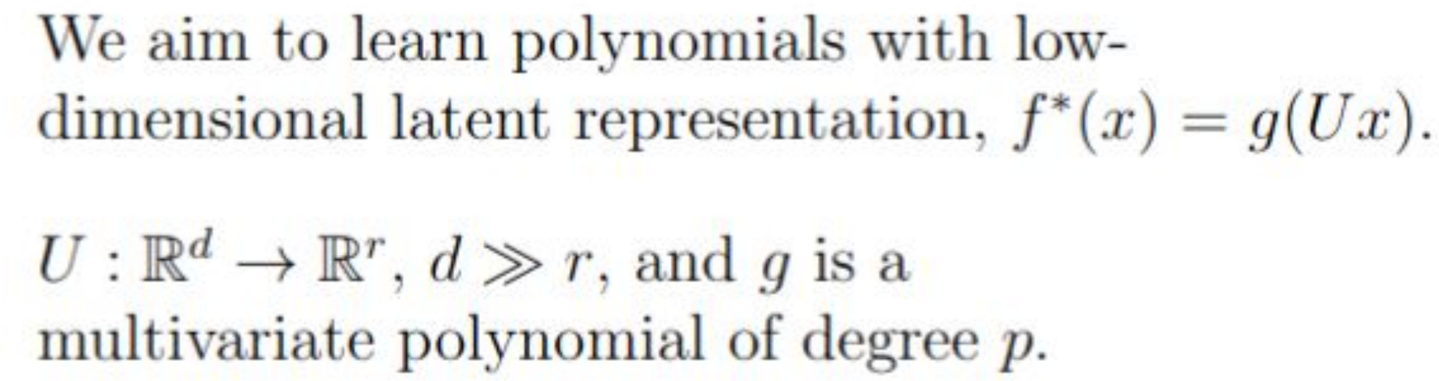


$$f^*(x) = g(Ux)$$

KM vs NN Adaptability On Latent Variables

Credits: https://arxiv.org/pdf/2206.15144.pdf

Objective and Notation



We aim to learn polynomials with low-dimensional latent representation, $f^*(x) = g(Ux)$.

$U : \mathbb{R}^d \to \mathbb{R}^r$, $d \gg r$, and g is a multivariate polynomial of degree p.

Main Result

Objective: Learn a multidimensional polynomial \w low dimensional substruct.

Kernel Regime: $n = \Theta(d^p)$ Ghorbani et. al

A simple 2-layer NN: $n = \Theta(d^2 r + d r^p)$ Damian et. al

Takeaway:

In NNs gradient descent finds low dimensional substructure

Assumptions

Assumption 1. *There exists a function* $g : \mathbb{R}^r \to \mathbb{R}$ *and linearly independent vectors* $u_1, \dots, u_r$ *such that for all* $x \in \mathbb{R}^d$,

$$f^\star(x) = g(\langle x, u_1 \rangle, \dots, \langle x, u_r \rangle).$$

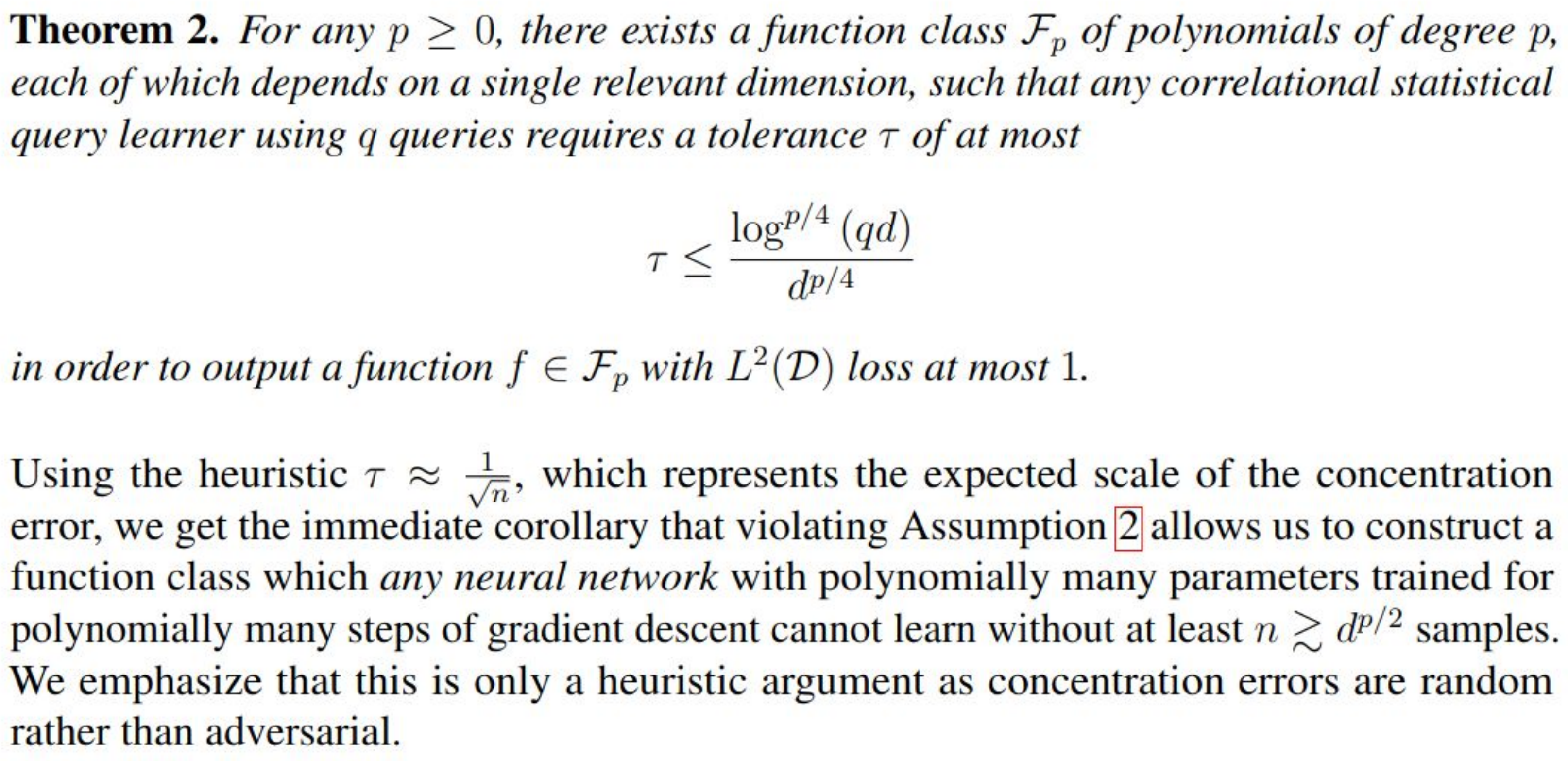
We will call $S^\star := \operatorname{span}(u_1, \dots, u_r)$ *the principal subspace of* $f^\star$. *We will also denote by* $\Pi^\star := \Pi_{S^\star}$ *the orthogonal projection onto* $S^\star$.

Note that Assumption 1 guarantees that for any x, $\operatorname{span}(\nabla^2 f^\star(x)) \subset S^\star$. In particular, if we denote the average Hessian by $H := \mathbb{E}_{x \sim \mathcal{D}}[\nabla^2 f^\star(x)]$, we have that $\operatorname{span}(H) \subset S^\star$ so that H has rank at most r. The following non-degeneracy assumption states that H has rank *exactly* r.

Assumption 2. $H := \mathbb{E}_{x \sim \mathcal{D}}[\nabla^2 f^\star(x)]$ *has rank* r, *i.e.* $\operatorname{span}(H) = S^\star$.

We will also denote the normalized condition number of H by $\kappa := \frac{\|H^\dagger\|}{\sqrt{r}}$.

We need assumption 2



Theorem 2. *For any $p \geq 0$, there exists a function class $\mathcal{F}_p$ of polynomials of degree p, each of which depends on a single relevant dimension, such that any correlational statistical query learner using q queries requires a tolerance τ of at most*

$$\tau \leq \frac{\log^{p/4}(qd)}{d^{p/4}}$$

in order to output a function $f \in \mathcal{F}_p$ with $L^2(\mathcal{D})$ loss at most 1.

Using the heuristic $\tau \approx \frac{1}{\sqrt{n}}$, which represents the expected scale of the concentration error, we get the immediate corollary that violating Assumption 2 allows us to construct a function class which *any neural network* with polynomially many parameters trained for polynomially many steps of gradient descent cannot learn without at least $n \gtrsim d^{p/2}$ samples. We emphasize that this is only a heuristic argument as concentration errors are random rather than adversarial.

Initialization Details

Let $\sigma(x) = \mathrm{ReLU}(x) = \max(0, x)$, let $a \in \mathbb{R}^m$, $W \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$, and let $\theta = (a, W, b)$. We define the neural network f_θ by

$$f_\theta(x) = a^T \sigma(Wx + b) = \sum_{j=1}^{m} a_j \sigma(w_j \cdot x + b_j),$$

where m denotes the width of the network. We use a symmetric initialization, so that $f_{\theta_0}(x) = 0$ [14]. Explicitly, we will assume that m is an even number and that

$$a_j = -a_{m-j}, \quad w_j = w_{m-j} \quad \text{and} \quad b_j = b_{m-j} \qquad \forall j \in [m/2].$$

We will use the following initialization:

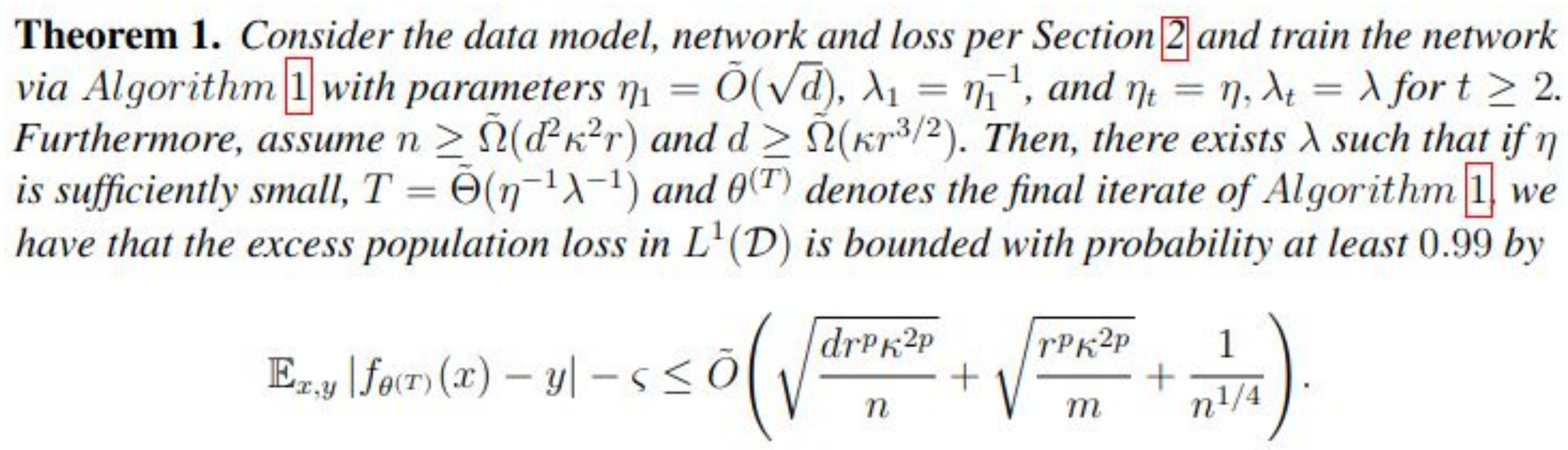
$$a_j \sim \{-1, 1\}, \quad w_j \sim N\left(0, \frac{1}{d} I_d\right) \quad \text{and} \quad b_j = 0.$$

We note that while we focus on such symmetric initialization for clarity of exposition, our results also hold with small random initialization that is not necessarily symmetric. This holds by simple modifications in the proof accounting for the small nonzero output of the network at initialization. We will also denote the empirical and population losses by $\mathcal{L}(\theta)$ and $\mathcal{L}_{\mathcal{D}}(\theta)$ respectively:

Algorithm 1: Gradient-based training

```
Input : Learning rates $\eta_t$, weight decay $\lambda_t$, number of steps $T$
preprocess data
    $\alpha \leftarrow \frac{1}{n}\sum_{i=1}^{n} y_i$, $\beta \leftarrow \frac{1}{n}\sum_{i=1}^{n} y_i x_i$
    $y_i \leftarrow y_i - \alpha - \beta \cdot x_i$ for $i = 1, \ldots, n$
end
$W^{(1)} \leftarrow W^{(0)} - \eta_1[\nabla_W \mathcal{L}(\theta) + \lambda_1 W]$
re-initialize $b_j \sim N(0, 1)$
for $t = 2$ to $T$ do
    $a^{(t)} \leftarrow a^{(t-1)} - \eta_t[\nabla_a \mathcal{L}(\theta^{(t-1)}) + \lambda_t a^{(t-1)}]$
end
return Prediction function $x \rightarrow \alpha + \beta \cdot x + a^T \sigma(Wx + b)$
```

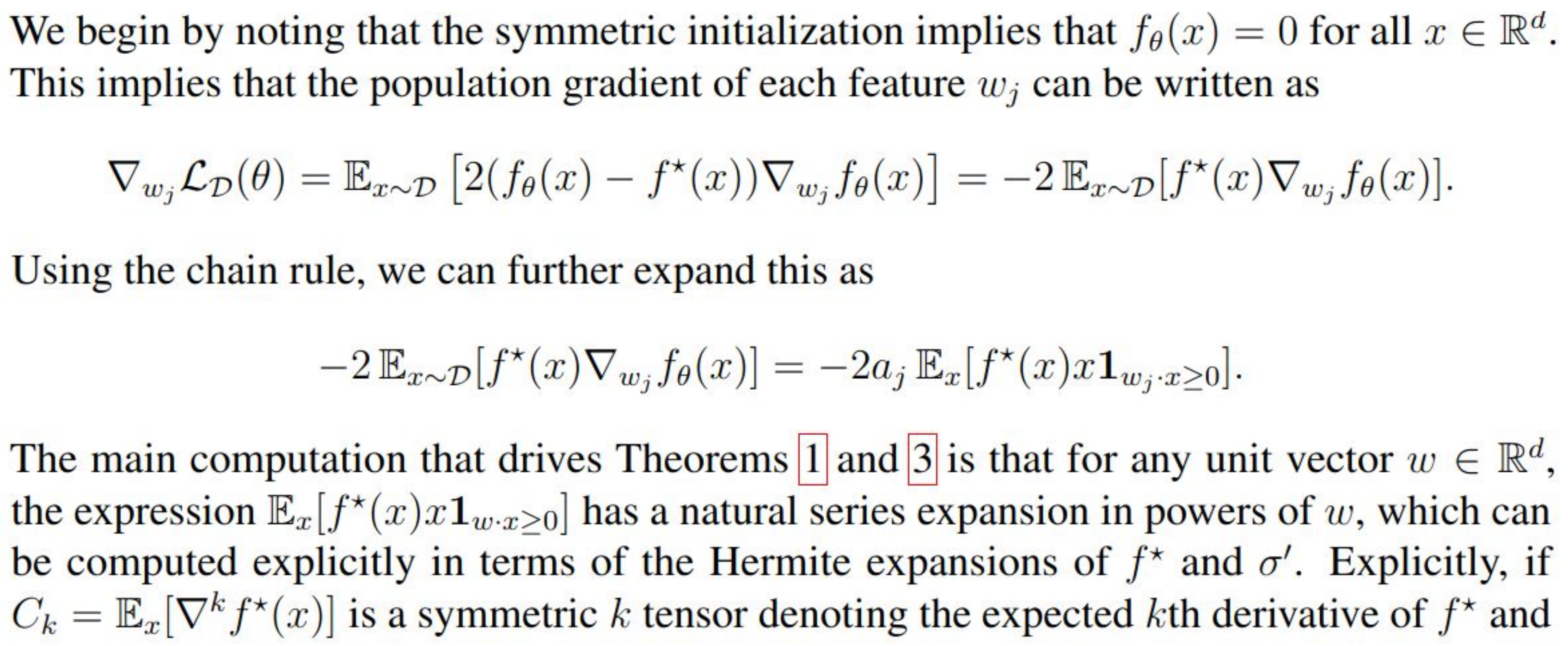
Theorem 1 - Main Result



Theorem 1. *Consider the data model, network and loss per Section 2 and train the network via Algorithm 1 with parameters* $\eta_1 = \tilde{O}(\sqrt{d})$, $\lambda_1 = \eta_1^{-1}$, *and* $\eta_t = \eta, \lambda_t = \lambda$ *for* $t \geq 2$. *Furthermore, assume* $n \geq \tilde{\Omega}(d^2\kappa^2 r)$ *and* $d \geq \tilde{\Omega}(\kappa r^{3/2})$. *Then, there exists* λ *such that if* η *is sufficiently small,* $T = \tilde{\Theta}(\eta^{-1}\lambda^{-1})$ *and* $\theta^{(T)}$ *denotes the final iterate of Algorithm 1, we have that the excess population loss in* $L^1(\mathcal{D})$ *is bounded with probability at least* 0.99 *by*

$$\mathbb{E}_{x,y}\left|f_{\theta^{(T)}}(x) - y\right| - \varsigma \leq \tilde{O}\left(\sqrt{\frac{dr^p\kappa^{2p}}{n}} + \sqrt{\frac{r^p\kappa^{2p}}{m}} + \frac{1}{n^{1/4}}\right).$$

Building Up to Lemma 1



We begin by noting that the symmetric initialization implies that $f_\theta(x) = 0$ for all $x \in \mathbb{R}^d$. This implies that the population gradient of each feature w_j can be written as

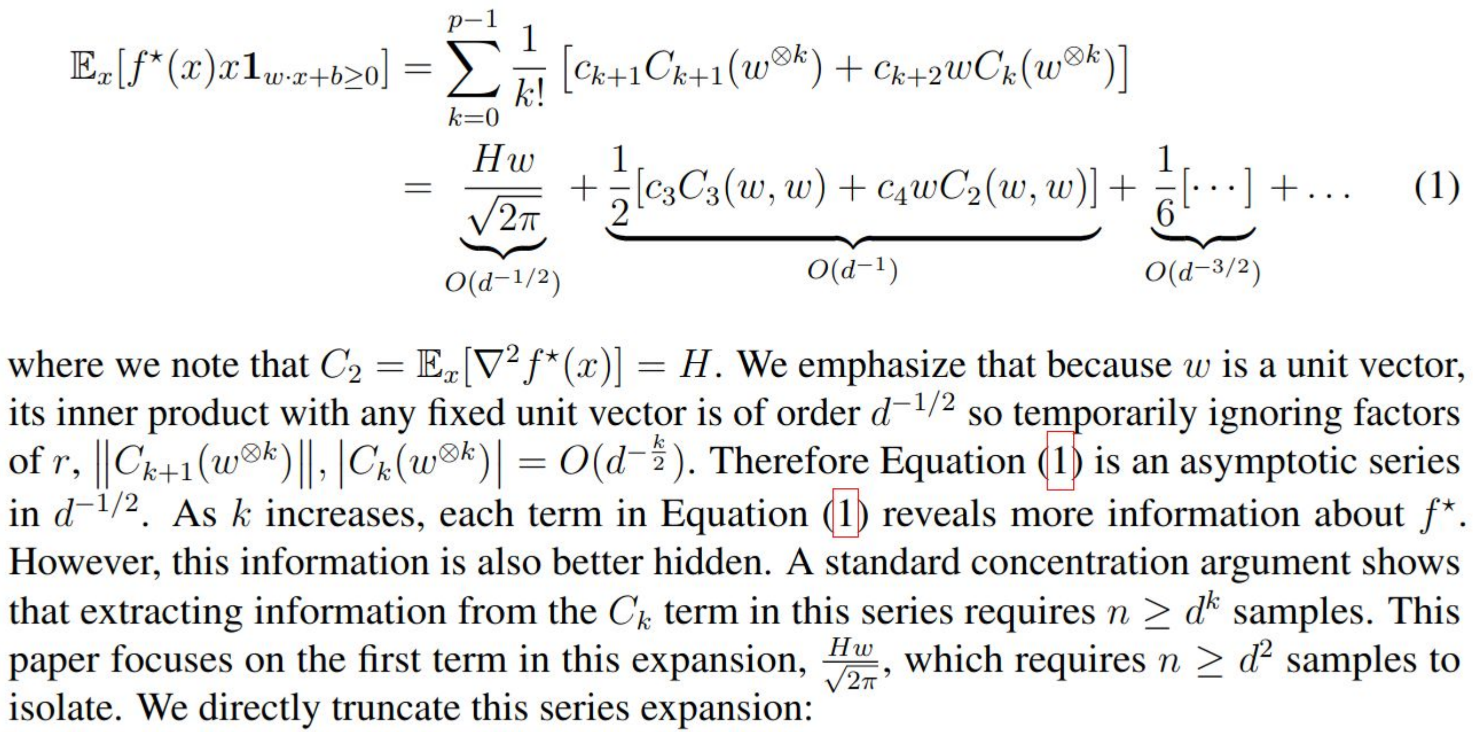
$$\nabla_{w_j} \mathcal{L}_\mathcal{D}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[2(f_\theta(x) - f^\star(x))\nabla_{w_j} f_\theta(x)\right] = -2\,\mathbb{E}_{x\sim\mathcal{D}}[f^\star(x)\nabla_{w_j} f_\theta(x)].$$

Using the chain rule, we can further expand this as

$$-2\,\mathbb{E}_{x\sim\mathcal{D}}[f^\star(x)\nabla_{w_j} f_\theta(x)] = -2a_j\,\mathbb{E}_x[f^\star(x)x\mathbf{1}_{w_j\cdot x\geq 0}].$$

The main computation that drives Theorems 1 and 3 is that for any unit vector $w \in \mathbb{R}^d$, the expression $\mathbb{E}_x[f^\star(x)x\mathbf{1}_{w\cdot x\geq 0}]$ has a natural series expansion in powers of w, which can be computed explicitly in terms of the Hermite expansions of $f^\star$ and σ'. Explicitly, if $C_k = \mathbb{E}_x[\nabla^k f^\star(x)]$ is a symmetric k tensor denoting the expected kth derivative of $f^\star$ and

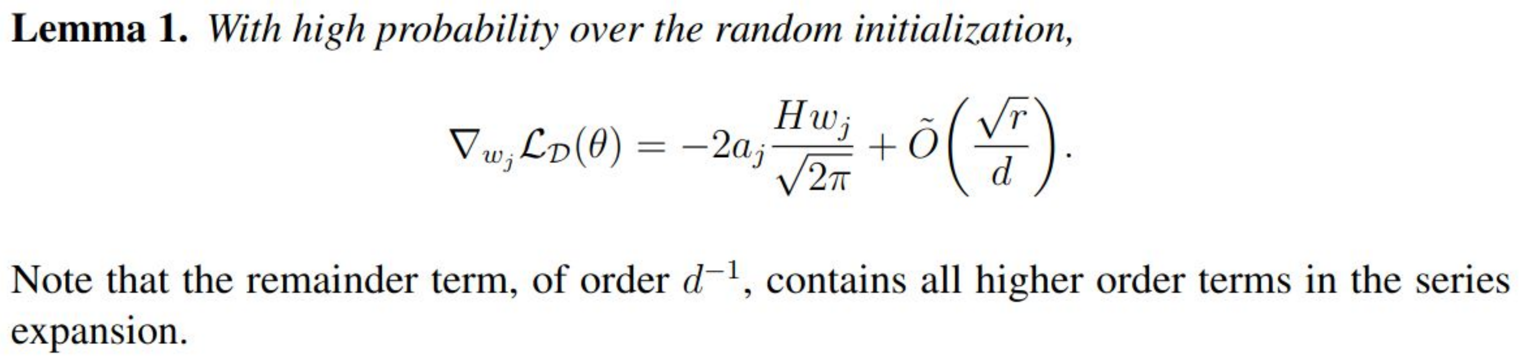
Grad School DLC



$$\begin{aligned}\mathbb{E}_x[f^\star(x)x\mathbf{1}_{w\cdot x+b\geq 0}] &= \sum_{k=0}^{p-1}\frac{1}{k!}\left[c_{k+1}C_{k+1}(w^{\otimes k}) + c_{k+2}wC_k(w^{\otimes k})\right] \\ &= \underbrace{\frac{Hw}{\sqrt{2\pi}}}_{O(d^{-1/2})} + \underbrace{\frac{1}{2}[c_3C_3(w,w) + c_4wC_2(w,w)]}_{O(d^{-1})} + \underbrace{\frac{1}{6}[\cdots]}_{O(d^{-3/2})} + \dots \qquad (1)\end{aligned}$$

where we note that $C_2 = \mathbb{E}_x[\nabla^2 f^\star(x)] = H$. We emphasize that because w is a unit vector, its inner product with any fixed unit vector is of order $d^{-1/2}$ so temporarily ignoring factors of r, $\left\|C_{k+1}(w^{\otimes k})\right\|, \left|C_k(w^{\otimes k})\right| = O(d^{-\frac{k}{2}})$. Therefore Equation (1) is an asymptotic series in $d^{-1/2}$. As k increases, each term in Equation (1) reveals more information about $f^\star$. However, this information is also better hidden. A standard concentration argument shows that extracting information from the C_k term in this series requires $n \geq d^k$ samples. This paper focuses on the first term in this expansion, $\frac{Hw}{\sqrt{2\pi}}$, which requires $n \geq d^2$ samples to isolate. We directly truncate this series expansion:

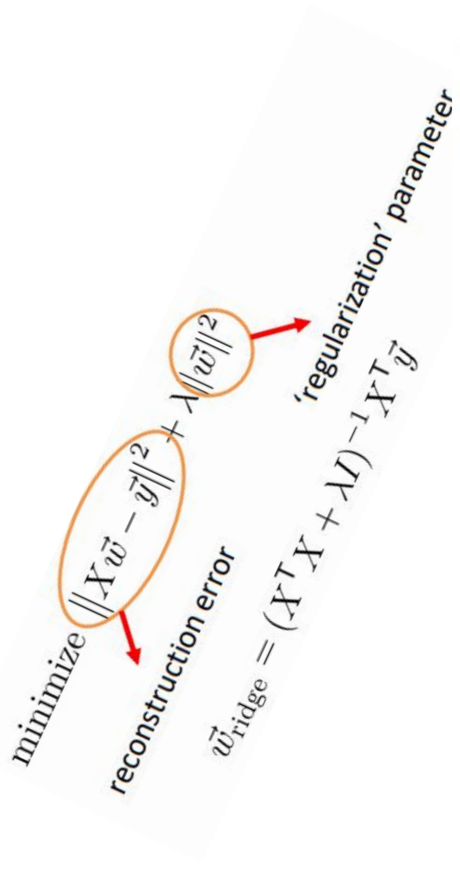
Lemma 1 - the backbone of Theorem 1



Lemma 1. *With high probability over the random initialization,*

$$\nabla_{w_j} \mathcal{L}_{\mathcal{D}}(\theta) = -2a_j \frac{Hw_j}{\sqrt{2\pi}} + \tilde{O}\left(\frac{\sqrt{r}}{d}\right).$$

Note that the remainder term, of order d^{-1}, contains all higher order terms in the series expansion.


minimize $\|X\vec{w}-\vec{y}\|^2 + \lambda\|\vec{w}\|^2$
reconstruction error
'regularization' parameter
$\vec{w}_{\text{ridge}} = (X^TX+\lambda I)^{-1}X^T\vec{y}$

The remainder of Algorithm 1 runs ridge regression on the network head a with fixed features $x \to \sigma(W^{(1)}x + b)$. We can directly analyze the generalization of this algorithm using standard techniques from Rademacher complexity. In particular, a high level sketch of the remainder of the proof goes as follows:

1. (Appendix A.2): We use the features from Lemma 1 to construct a vector $a^\star \in \mathbb{R}^m$ such that

$$\mathcal{L}(a^\star, W^{(1)}, b) \ll 1 \quad \text{and} \quad \|a^\star\| = \tilde{O}\left(\frac{r^p \kappa^{2p}}{\sqrt{m}}\right).$$

2. (Appendix A.3): We show the equivalence between ridge regression and norm constrained linear regression implies the existence of $\lambda > 0$ such that the Tth iterate $a^{(T)}$ satisfies

$$\mathcal{L}(a^{(T)}, W^{(1)}, b) \ll 1 \quad \text{and} \quad \|a^{(T)}\| \le \|a^\star\|.$$

3. (Appendix A.3): A standard Rademacher generalization bound for two layer neural networks bounds the population risk $\mathbb{E}_{x,y}\,|f_{\theta^{(T)}}(x) - y|$ by the empirical risk $\frac{1}{n}\sum_{i=1}^{n} |f_{\theta^{(T)}}(x_i) - y_i|$ and $\|a^{(T)}\|$ which are small from step 2.

Conclusion

- We introduce general low dimensional structure and show that neural networks take advantage of it
- Kernel regime can't explain the expressiveness of neural networks
- Kernel regime is not ideal for massive data
    - Direct inversion costs O(n^3) - intractable for 10 million data points
    - Iterative approach is slow - i.e. 10 million (=n) matrix multiplications (nxn) per prediction

Closing Questions:

- What does this mean for kernel methods?
- Could we use a similar trick on the NTK?
- What do we think about Assumption 2 (non-degeneracy)?