NEURAL NETWORKS CAN LEARN REPRESENTATIONS WITH GRADIENT DESCENT

CS7020 - ADVANCES IN THEORY OF DEEP LEARNING

SHIVARAM
BE20B032

CONTENT

INTRODUCTION

- In specific regimes, neural networks learned via gradient descent behave like kernel methods.
- Existence of large class of function which cannot be efficiently learned by kernel methods but can be easily learned by neural networks with gradient descent.
- Neural networks learn representations relevant to target task.
- Representations help in transfer learning.
- Primary result of the paper, gradient descent learns a representation of the data which depends only on relevant directions.

REPRESENTATION LEARNING

- Success of Deep Learning is the ability of gradient descent to learn good feature representations from training data and learn simple functions on top of it.
- Challenge in understanding this representation learning, highly non convex loss landscape and convergence to global optima.
- In overparamterized nets there are many global optima with poor generalization.

This paper focusses on:

"How do gradient-based methods learn feature representations and why do these representations allow for efficient generalization and transfer learning?"

NTK AND GENERALIZATION

- Dynamics of gradient descent approximated by gradient descent on a linear regression with fixed representation.
- Unrealistic hyper-parameters, does not allow features to evolve, therefore generalization error is not better than kernel methods.
- Lower bounds of NTK show that they don't generalize better than polynomial kernels.
- NTK requires $n \approx d^p$ samples to learn a p degree polynomial.

CONTRIBUTIONS OF THE PAPER

- **Feature Learning:** Target function only depends on the projection of x onto a hidden subspace span(U).

- **Improved Sample Complexity:** Target function $f^* : \mathbb{R}^d \rightarrow \mathbb{R}$ is a polynomial of degree p which depends on r relevant dimensions can be learnt with gradient descent on a two layer neural network with $n \approx d^2 r + dr^p$ samples, while random features model and NTK requires d^p samples.

- **Transfer Learning**: When target task $f^*_{target}(x) = \tilde{g}(Ux)$, can be learnt by retraining the network head with $N \approx r^p$ samples which is independent of dimension d.

- **Lower Bound:** Without non-degeneracy, there is a family of polynomials which depend on single relevant dimension, which cannot be learned with fewer than $n \approx d^{p/2}$ samples with any gradient descent based learner.

INPUT DISTRIBUTION AND TARGET FUNCTION

$f^*: \mathbb{R}^d \to \mathbb{R}$ over $\mathcal{D} := \mathcal{N}(0, I_d)$ and f^* is normalised as $\mathbb{E}_{x\sim\mathcal{D}}[(f^*(x))^2] = 1$
Target function is learned with n i.i.d datapoints, as follows:

$x_i \sim \mathcal{D}, y_i = f^*(x_i) + \epsilon_i$ and $\epsilon_i \sim \{-\zeta, \zeta\}$

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

$$f^*(x) = g(\langle x, u_1\rangle, \ldots \langle x, u_r\rangle)$$

$S^* := span(u_1, ..., u_r)$ principal subspace of f^*

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

NETWORK AND LOSS FUNCTION

Neural network:

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

where $\sigma(x) = ReLU(x), a \in \mathbb{R}^m, W \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^d$ and $\theta = (a, W, b)$

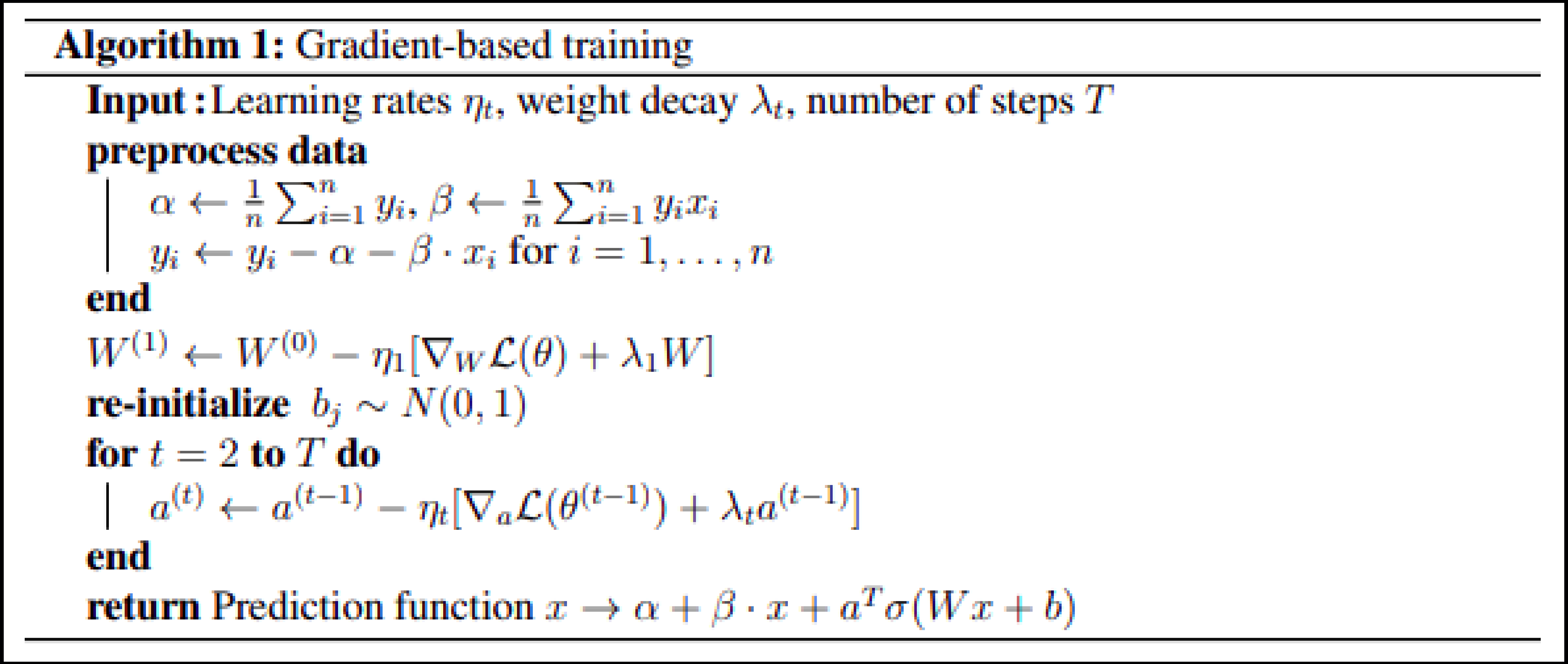
Initialization: We use symmetric initialization such that $f_{\theta_0}(x) = 0$ as follows:

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

Following initialization is used:

$$a_j \sim -1, 1, w_j \sim \mathcal{N}(0, \frac{1}{d} I_d), b_j = 0$$

GRADIENT BASED TRAINING ALGORITHM



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 $i = 1, \dots, n$
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

Training the network via algorithm mentioned with the parameters $\eta_1 = \tilde{O}(\sqrt{d})$ $\lambda_1 = \eta^{-1}$ and $\eta_t = \eta$, $\lambda_t = \lambda$ for $t \geq 2$, Assume $n \geq \tilde{\Omega}(d^2\kappa^2 r)$ and $d \geq \tilde{\Omega}(\kappa r^{\frac{3}{2}})$
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 the excess population loss in $L^1(\mathcal{D})$ is bounded with probability 0.99 by :

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

Learning f^* requires $n \gtrsim dr^p + d^2 r$ samples and requires a very small network $m \gtrsim r^p$

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

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

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

- Violating Assumption 2, allows to construct a functional class, which any neural network cannot learn without at least $n \geq d^{\frac{p}{2}}$ samples.
- In theorem 1, non-degenarcy assumption allows neural network to extract useful features. This helps in transfer learning.

THEOREM 3

Let $g^*(x)$ be a degree p polynomial with $\mathbb{E}_{\mathcal{D}}[g^*(x)^2] = 1$ and $g(x) = g(\Pi^* x)$ for all $x \in \mathbb{R}^d$. Let $\mathcal{D}_N = \{(x_i, y_i)\}_{i\in[N]}$ be a second dataset with $y_i = g(x_i) + \epsilon_i$. We retrain the neural network $f_\theta(x)$ in Theorem 1 with gradient descent with learning rate η and decay rate λ

$$g_a(x) = a^T(W^{(1)}x + b)$$

where $W^{(1)}$ is the second iterate of Algorithm 1. Then there exists λ such that if the network is pretrained on $\boldsymbol{n} \geq \tilde{\Omega}(\boldsymbol{d^2\kappa^2 r})$ (Assume that $d \geq \tilde{\Omega}(\kappa r^{\frac{3}{2}})$) datapoints from f^* and η is sufficiently small, the excess population loss $L^1(\mathcal{D})$ after $T = \tilde{\Theta}(\eta^{-1}\lambda^{-1})$ steps is bounded with probability 0.99

$$\mathbb{E}_{x,y}|g_{a^{(T)}} - y| - \zeta \leq \tilde{O}(\sqrt{\frac{r^p\kappa^{2p}}{min(n,N)}} + \frac{1}{N^{\frac{1}{4}}})$$

EXPERIMENTS

Sample complexity: For $u \in S^{d-1}$, consider the target function

$$f_u^*(x) = g(u \cdot x) \quad \text{where} \quad g(x) = \frac{H_{e_2}(x)}{2} + \frac{H_{e_p}(x)}{\sqrt{2p!}}$$

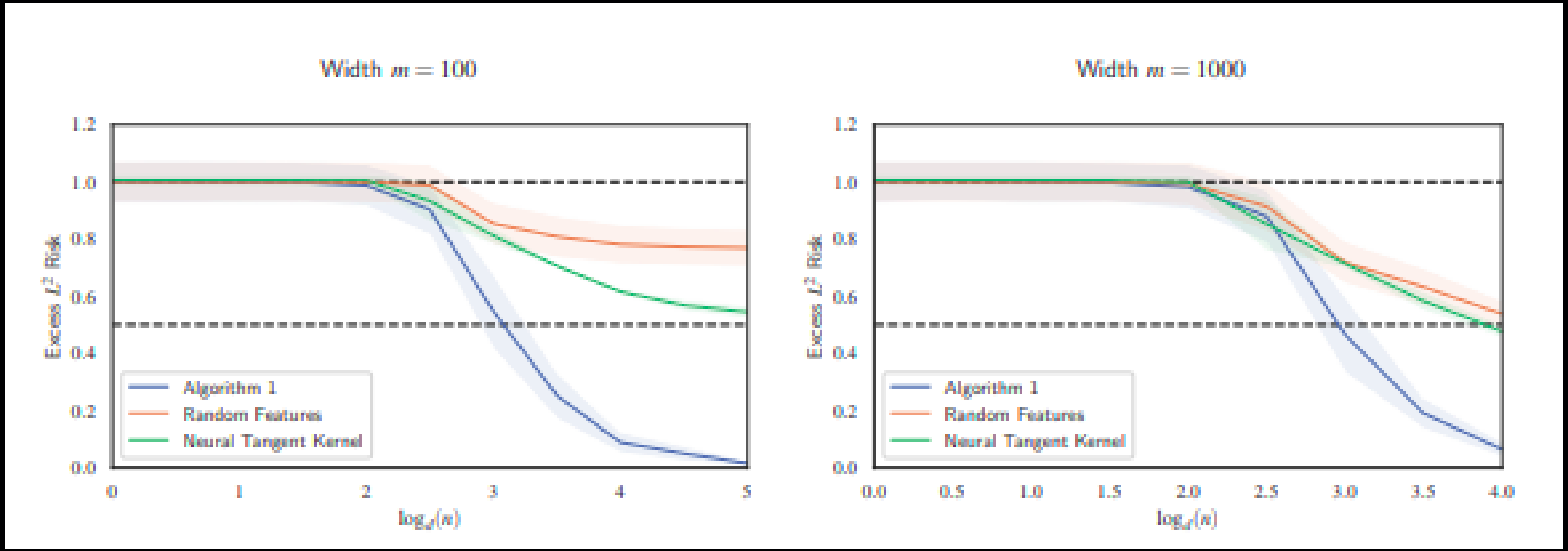
which satisfies $\mathbb{E}_{x \sim \mathcal{D}}[(f^*(x))^2] = 1$

f^* depends only on the single relevant dimension u.

Gradient descent isolates the subspace spanned by u and fits a one dimensional random feature to g which requires $n \approx d^2$ samples.

d^p samples required for NTK and random feature regimes(Ghorbani et. al). d^2 samples, kernel regression returns 0 predictor. $d^2 < n < d^p$ samples, kernel regression returns $\frac{H_{e_2}(u \cdot x)}{2}$ with $L^2(\mathcal{D})$ with loss of 1/2.

SAMPLE COMPLEXITY



Experiment setting: d = 10, p = 4 and learn f^* using Algorithm 1, random features and NTK.

Observations: With algorithm 1, easily converged while random feature model and NTK learned only the quadratic term.
Learning a function f^* is to use the $\frac{H_{e_2}(u \cdot x)}{2}$ component to identify u, afterwards any kernel or random feature model can learn any univariate function on top of it.

EXPERIMENTS

Transfer Learning: Consider the function

$$f^*_{target}(x) = g(u \cdot x) \quad \text{where} \quad g_{target} = \frac{H_{e_p}(x)}{\sqrt{p!}}$$

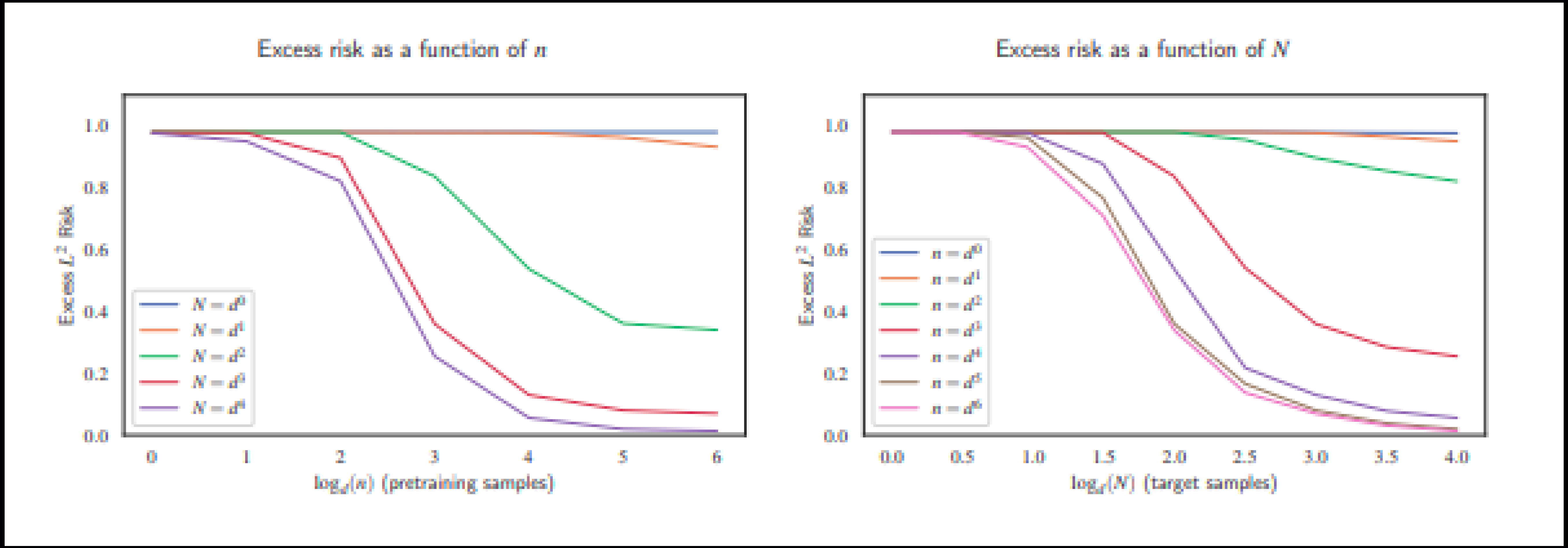
Pretrain f^* with n samples via algorithm 1 and then train the output layer with N samples from $f_{target}(x)$.

p = 3, random feature methods and NTK requires $n \geq d^3$ samples

Observations:

- $n = d^0, d^1$, fine tuning on N training samples gives trivial risk until $N \geq d^3$, which is expected of a kernel method with no prior knowledge.
- For $n \geq d^2$ pertaining samples, we can fine tune on N=O(1) target samples to reach non trivial loss and loss decays rapidly as a function of N.

TRANSFER LEARNING



CONCLUSION AND FUTURE WORK

- There exists family of degree p polynomials which are efficiently learnable by gradient descent.
- Sample complexity and Transfer Learning result implies that gradient descent learns representations of the data.

- **Future Work:** Generalizing the result to neural networks where hidden and output layer are trained together.