INFORMS Optimization Society (IOS) Conference

# Non-Linear Feature Learning with One Gradient Step in Two-Layer Neural Networks

Behrad Moniri

University of Pennsylvania

bemoniri@seas.upenn.edu

March 23rd, 2024

# Collaborators

Donghwan Lee

Hamed Hassani

Edgar Dobriban

# Introduction

# Introduction

Deep Learning is *very* successful.

# Introduction

- It is a huge *engineering* success.

# Introduction

- It is a huge *engineering* success.

- Why is it very successful?

# Introduction

- It is a huge *engineering* success.
- Why is it very successful?
- The problem is notoriously hard.

# Solvable Models

- Is there a *rich enough*, solvable model?

# Solvable Models

- Is there a *rich enough*, solvable model?
- There are **too** many moving parts: architecture, optimization, data, etc.

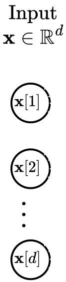
# Architecture

We consider the simplest possible architecture.

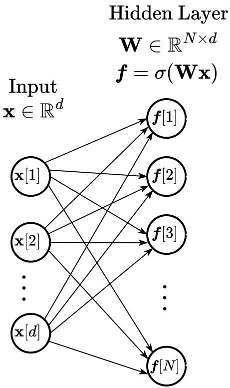
We consider the simplest possible architecture.

A *two-layer fully connected neural network*.

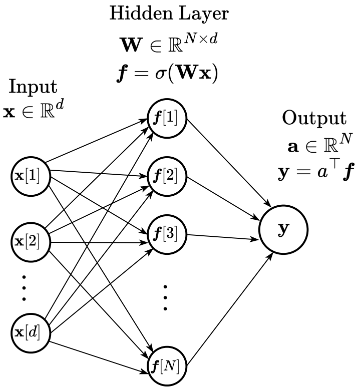
# Two-Layer Neural Networks

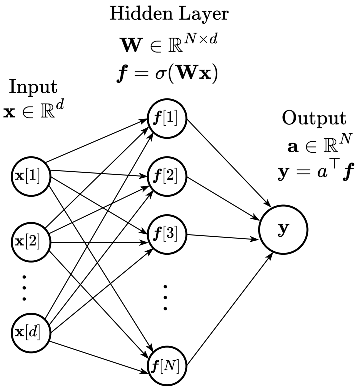


# Two-Layer Neural Networks

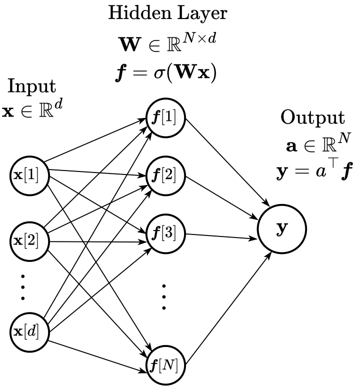


# Two-Layer Neural Networks






**Asymptotic Regime**: $n, d, N \to \infty$ with $d/n \to \phi$ and $d/N \to \psi$.




- **Simplest Model:**

  *Random Features Model.*

  (Rahimi and Recht, 2007)

## Random Features Models

- Random features model is **very popular**:

# Random Features Models

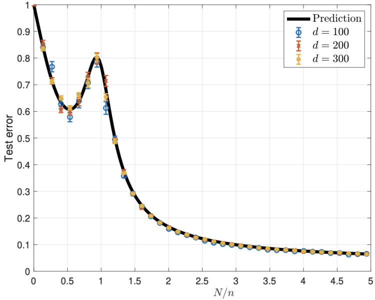
- Random features model is **very popular**:
  - Used to study various aspects of deep learning such as double descent, robustness to adversarial attacks, privacy, fairness, OOD performance, calibration, etc.

See e.g., Mei and Montanari (2022); Lin and Dobriban (2021); Lee et al. (2023); Hassani and Javanmard (2022); Bombari and Mondelli (2023); Bombari et al. (2023); Clarté et al. (2023), etc.

# Random Features Models

- Random features model is **very popular**:
  - Used to study various aspects of deep learning such as double descent, robustness to adversarial attacks, privacy, fairness, OOD performance, calibration, etc.

See e.g., Mei and Montanari (2022); Lin and Dobriban (2021); Lee et al. (2023); Hassani and Javanmard (2022); Bombari and Mondelli (2023); Bombari et al. (2023); Clarté et al. (2023), etc.




Double descent in random feature models (Mei and Montanari, 2022).

# Random Features Models

- Random features models can only learn linear functions.

  Mei and Montanari (2022); Hu and Lu (2023)

# Random Features Models

- Random features models can only learn linear functions.

  Mei and Montanari (2022); Hu and Lu (2023)

- **Gaussian Equivalence**:

# Random Features Models

- Random features models can only learn linear functions.

  Mei and Montanari (2022); Hu and Lu (2023)

- **Gaussian Equivalence**:

$$\begin{aligned}\sigma(\mathbf{W}\boldsymbol{x}) &= c_1\mathbf{W}\boldsymbol{x} + c_2 H_2(\mathbf{W}\boldsymbol{x}) + \cdots \\ &\approx c_1\mathbf{W}\boldsymbol{x} + \boldsymbol{z}\end{aligned}$$

# Random Features Models

- Random features models can only learn linear functions.

  Mei and Montanari (2022); Hu and Lu (2023)

- **Gaussian Equivalence**:

$$\begin{aligned}\sigma(\mathbf{W}\boldsymbol{x}) &= c_1\mathbf{W}\boldsymbol{x} + c_2 H_2(\mathbf{W}\boldsymbol{x}) + \cdots \\ &\approx c_1\mathbf{W}\boldsymbol{x} + \boldsymbol{z}\end{aligned}$$

- This makes analysis *easy* but the model very *limited*.

# Random Features Models

- Random features models can only learn linear functions.

  Mei and Montanari (2022); Hu and Lu (2023)

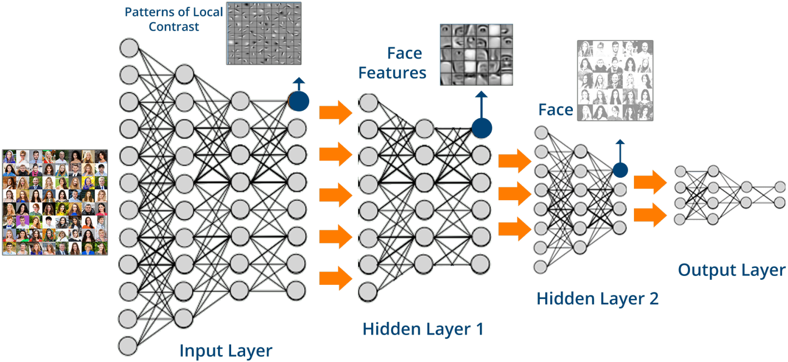
- **Gaussian Equivalence**:

$$\begin{aligned} \sigma(\mathbf{W}\boldsymbol{x}) &= c_1\mathbf{W}\boldsymbol{x} + c_2 H_2(\mathbf{W}\boldsymbol{x}) + \cdots \\ &\approx c_1\mathbf{W}\boldsymbol{x} + \boldsymbol{z} \end{aligned}$$

- This makes analysis *easy* but the model very *limited*.

- What is missing?

# Feature Learning



# Random Features Models

- Feature learning is absent in random feature models.

# Random Features Models

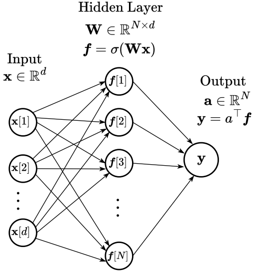
- Feature learning is absent in random feature models.
- How to go beyond random feature models?

## Random Features Models

- Feature learning is absent in random feature models.
- How to go beyond random feature models?
- Let's do one step of gradient descent on first layer weights.

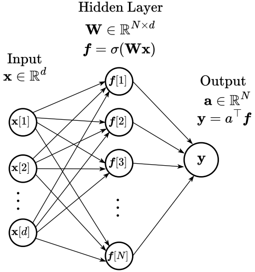
# One Gradient Step

# One Gradient Step Update



# One Gradient Step Update

We train the network as follows:

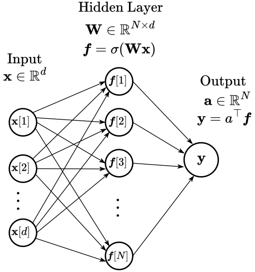


# One Gradient Step Update

We train the network as follows:

1. We first initialize

$$\boldsymbol{a} \sim \mathcal{N}\left(\mathbf{0}_N, \frac{1}{N}\mathbf{I}_N\right), \quad \text{and} \quad [\mathbf{W}_0]_{ij} \sim \mathcal{N}\left(0, \frac{1}{d}\right)$$



# One Gradient Step Update

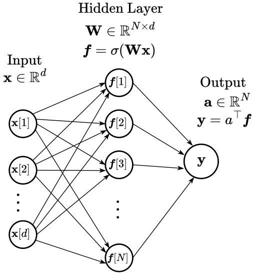
We train the network as follows:

1. We first initialize

$$\boldsymbol{a} \sim \mathcal{N}\left(\mathbf{0}_N, \frac{1}{N}\mathbf{I}_N\right), \quad \text{and} \quad [\mathbf{W}_0]_{ij} \sim \mathcal{N}\left(0, \frac{1}{d}\right)$$

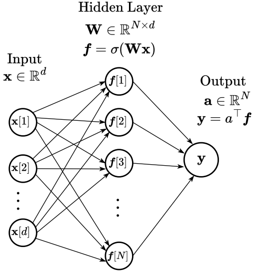
2. We take one gradient step on the empirical MSE loss

$$\mathbf{W}_1 = \mathbf{W}_0 - \eta \frac{\partial}{\partial \mathbf{W}} \left( \|\boldsymbol{y} - \sigma(\mathbf{X}\mathbf{W}^\top)\boldsymbol{a}\|_2^2 \right) \Big|_{\mathbf{W}_0, \boldsymbol{a}}$$



# One Gradient Step Update

We train the network as follows:




1. We first initialize

$$\boldsymbol{a} \sim \mathcal{N}\left(\mathbf{0}_N, \frac{1}{N}\mathbf{I}_N\right), \quad \text{and} \quad [\mathbf{W}_0]_{ij} \sim \mathcal{N}\left(0, \frac{1}{d}\right)$$

2. We take one gradient step on the empirical MSE loss

$$\mathbf{W}_1 = \mathbf{W}_0 - \eta \frac{\partial}{\partial \mathbf{W}} \left( \|\boldsymbol{y} - \sigma(\mathbf{X}\mathbf{W}^\top)\boldsymbol{a}\|_2^2 \right) \Big|_{\mathbf{W}_0, \boldsymbol{a}}$$

3. Fit $\boldsymbol{a}$ via ridge regression:

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}\in\mathbb{R}^N} \frac{1}{n}\|\boldsymbol{y} - \mathbf{F}\boldsymbol{a}\|_2^2 + \lambda\|\boldsymbol{a}\|_2^2, \quad \mathbf{F} = \sigma(\mathbf{X}\mathbf{W}_1^\top) \in \mathbb{R}^{n\times N}.$$

# Data Generation

- **Data generation:**

$$x_i \overset{i.i.d.}{\sim} \mathsf{N}(0, \mathbf{I}_d), \quad y_i = f_\star(\boldsymbol{x}_i) + \varepsilon_i.$$

# Data Generation

- **Data generation:**

$$x_i \overset{i.i.d.}{\sim} \mathsf{N}(0, \mathbf{I}_d), \quad y_i = f_\star(\boldsymbol{x}_i) + \varepsilon_i.$$

- With one-step, only a single-index approximation can be learned. Thus, we let

$$f_\star(\boldsymbol{x}_i) = \sigma_\star(\boldsymbol{\beta}_\star^\top \boldsymbol{x}_i)$$

(see e.g., Dandi et al. (2023), etc.)

# Prior Work

Partial answers in the prior work:

# Prior Work

Partial answers in the prior work:

- Ba et al. (2022) show that if $\eta = O(1)$, still no nonlinear component of the teacher function can be learned.

Partial answers in the prior work:

- Ba et al. (2022) show that if $\eta = O(1)$, still no nonlinear component of the teacher function can be learned.
- Performance is still worse than linear regression.

# Prior Work

## Prior Work

- However, with $\eta = O(\sqrt{n})$ nonlinear functions can be learned.

- However, with $\eta = O(\sqrt{n})$ nonlinear functions can be learned.

- How is this possible? What happens?

# Spectral Analysis

- After the update, the weights are

$$
\begin{aligned}
\mathbf{W}_1 &= \mathbf{W}_0 - \eta \frac{\partial}{\partial \mathbf{W}} \left( \|\boldsymbol{y} - \sigma(\mathbf{X}\mathbf{W}^\top)\boldsymbol{a}\|_2^2 \right) \Big|_{\mathbf{W}_0, \boldsymbol{a}} \\
&= \mathbf{W}_0 + \frac{\eta}{n} \left[ (\boldsymbol{a}\boldsymbol{y}^\top - \boldsymbol{a}\boldsymbol{a}^\top \sigma(\mathbf{W}_0\mathbf{X}^\top)) \circ \sigma'(\mathbf{W}_0\mathbf{X}^\top) \right] \mathbf{X},
\end{aligned}
$$

# Spectrum of the Weights

- After the update, the weights are

$$\begin{aligned}\mathbf{W}_1 &= \mathbf{W}_0 - \eta \frac{\partial}{\partial \mathbf{W}} \left( \|\boldsymbol{y} - \sigma(\mathbf{X}\mathbf{W}^\top)\boldsymbol{a}\|_2^2 \right) \Big|_{\mathbf{W}_0, \boldsymbol{a}} \\ &= \mathbf{W}_0 + \frac{\eta}{n} \left[ (\boldsymbol{a}\boldsymbol{y}^\top - \boldsymbol{a}\boldsymbol{a}^\top \sigma(\mathbf{W}_0\mathbf{X}^\top)) \circ \sigma'(\mathbf{W}_0\mathbf{X}^\top) \right] \mathbf{X},\end{aligned}$$

- Orthogonal decomposition:

$$\sigma'(\mathbf{W}_0\mathbf{X}^\top) = c_1 + \sigma'_\perp(\mathbf{W}_0\mathbf{X}^\top), \quad \text{with} \quad \mathbb{E}\sigma'_\perp(\mathbf{W}_0\mathbf{X}^\top) = \mathbf{0}$$

# Spectrum of the Weights

- After the update, the weights are

$$\mathbf{W}_1 = \mathbf{W}_0 - \eta \frac{\partial}{\partial \mathbf{W}} \left( \|\boldsymbol{y} - \sigma(\mathbf{X}\mathbf{W}^\top)\boldsymbol{a}\|_2^2 \right) \Big|_{\mathbf{W}_0, \boldsymbol{a}}$$
$$= \mathbf{W}_0 + \frac{\eta}{n} \left[ (\boldsymbol{a}\boldsymbol{y}^\top - \boldsymbol{a}\boldsymbol{a}^\top \sigma(\mathbf{W}_0\mathbf{X}^\top)) \circ \sigma'(\mathbf{W}_0\mathbf{X}^\top) \right] \mathbf{X},$$

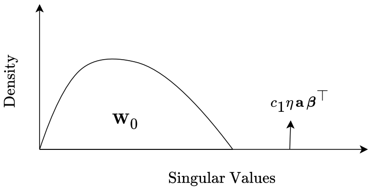
- Orthogonal decomposition:

$$\sigma'(\mathbf{W}_0\mathbf{X}^\top) = c_1 + \sigma'_\perp(\mathbf{W}_0\mathbf{X}^\top), \quad \text{with} \quad \mathbb{E}\sigma'_\perp(\mathbf{W}_0\mathbf{X}^\top) = \mathbf{0}$$

- Rank-1 Approximation:

$$\mathbf{W}_1 = \mathbf{W}_0 + \eta c_1 \boldsymbol{a} \left( \frac{\mathbf{X}^\top \boldsymbol{y}}{n} \right)^\top + \text{small}.$$

$$\mathbf{W}_1 = \mathbf{W}_0 + \eta c_1 \boldsymbol{a} \left( \frac{\mathbf{X}^\top \boldsymbol{y}}{n} \right)^\top + \text{small}.$$

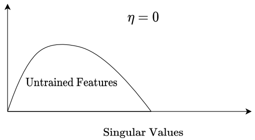



The vector $\boldsymbol{\beta} := \frac{\mathbf{X}^\top \boldsymbol{y}}{n}$ is aligned to $\boldsymbol{\beta}_\star$.

# Spectrum of the Feature Matrix

**Updated Weight Matrix**: $\mathbf{W} = \mathbf{W}_0 + \eta c_1 \boldsymbol{a}\boldsymbol{\beta}^\top$

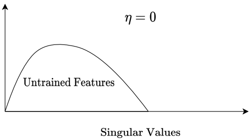
**Feature Matrix**: $\mathbf{F} = \sigma(\mathbf{X}\mathbf{W}^\top) = \sigma(\mathbf{X}\mathbf{W}_0^\top + c_1\eta\mathbf{X}\boldsymbol{\beta}\boldsymbol{a}^\top) \in \mathbb{R}^{n\times N}$

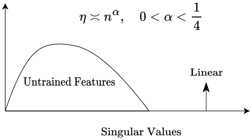


# Spectrum of the Feature Matrix

**Updated Weight Matrix:** $\mathbf{W} = \mathbf{W}_0 + \eta c_1 \boldsymbol{a}\boldsymbol{\beta}^\top$

**Feature Matrix:** $\mathbf{F} = \sigma(\mathbf{X}\mathbf{W}^\top) = \sigma(\mathbf{X}\mathbf{W}_0^\top + c_1\eta\mathbf{X}\boldsymbol{\beta}\boldsymbol{a}^\top) \in \mathbb{R}^{n\times N}$

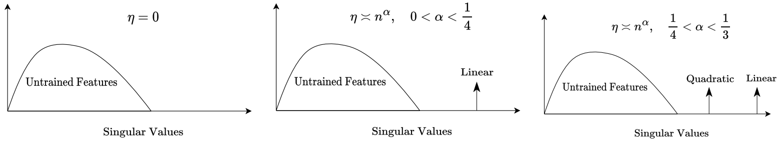


# Spectrum of the Feature Matrix

**Updated Weight Matrix:** $\mathbf{W} = \mathbf{W}_0 + \eta c_1 \boldsymbol{a}\boldsymbol{\beta}^\top$

**Feature Matrix:** $\mathbf{F} = \sigma(\mathbf{X}\mathbf{W}^\top) = \sigma(\mathbf{X}\mathbf{W}_0^\top + c_1 \eta \mathbf{X}\boldsymbol{\beta}\boldsymbol{a}^\top) \in \mathbb{R}^{n \times N}$



# Spectrum of Updated Feature Matrix

**Theorem**

*Let $\eta \asymp n^{\alpha}$ with $\frac{\ell-1}{2\ell} < \alpha < \frac{\ell}{2\ell+2}$ for some $\ell \in \mathbb{N}$. We have*

$$\mathbf{F} = \mathbf{F}_\ell + \mathbf{\Delta}, \text{ with } \mathbf{F}_\ell := \mathbf{F}_0 + \sum_{k=1}^{\ell} c_1^k c_k \eta^k (\mathbf{X}\boldsymbol{\beta})^{\circ k} (\boldsymbol{a}^{\circ k})^\top,$$

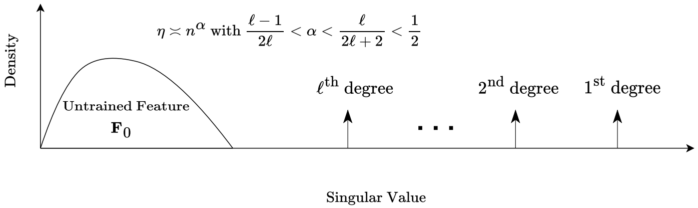
*where $\|\mathbf{\Delta}\|_{\text{op}} = o(\sqrt{n})$ with probability $1 - o(1)$.*

## Theorem

*Let $\eta \asymp n^{\alpha}$ with $\frac{\ell-1}{2\ell} < \alpha < \frac{\ell}{2\ell+2}$ for some $\ell \in \mathbb{N}$. We have*

$$\mathbf{F} = \mathbf{F}_\ell + \boldsymbol{\Delta}, \text{ with } \mathbf{F}_\ell := \mathbf{F}_0 + \sum_{k=1}^{\ell} c_1^k c_k \eta^k (\mathbf{X}\boldsymbol{\beta})^{\circ k} (\boldsymbol{a}^{\circ k})^\top,$$

*where $\|\boldsymbol{\Delta}\|_{\text{op}} = o(\sqrt{n})$ with probability $1 - o(1)$.*



# Analysis of the Training/Test Error

- What error does the trained neural network achieve?

# Universality

- What error does the trained neural network achieve?
- To do this, we show that the limiting behavior of test/train error is <u>unchanged</u> if

- What error does the trained neural network achieve?

- To do this, we show that the limiting behavior of test/train error is unchanged if

  1 we replace $\mathbf{F} = \sigma(\mathbf{X}\mathbf{W}_1^\top)$ with

$$\mathbf{F} = \mathbf{F}_0 + \sum_{k=1}^{\ell} c_1^k c_k \eta^k (\mathbf{X}\boldsymbol{\beta})^{\circ k} (\boldsymbol{a}^{\circ k})^\top.$$

# Universality

- What error does the trained neural network achieve?

- To do this, we show that the limiting behavior of test/train error is <u>unchanged</u> if

  1. we replace $\mathbf{F} = \sigma(\mathbf{X}\mathbf{W}_1^\top)$ with

  $$\mathbf{F} = \mathbf{F}_0 + \sum_{k=1}^{\ell} c_1^k c_k \eta^k (\mathbf{X}\boldsymbol{\beta})^{\circ k} (\boldsymbol{a}^{\circ k})^\top.$$

  2. we replace $\mathbf{F}_0$ with $\mathbf{F}_0 = c_1 \mathbf{X}\mathbf{W}_0^\top + c_{>1}\mathbf{Z}$.

# Analysis of the Training/Test Errors

With these universality results, we can analyze the limiting behaviour of the train/test errors.

# Analysis of the Training/Test Errors

With these universality results, we can analyze the limiting behaviour of the train/test errors.

**Theorem**

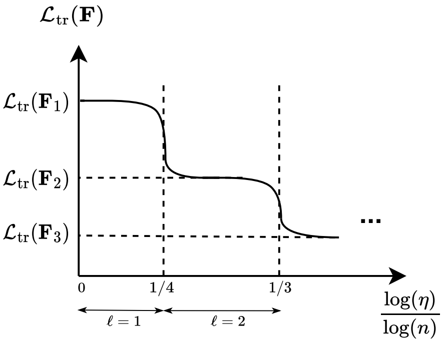
*Let $\ell \in \mathbb{N}$ and $\eta \asymp n^{\alpha}$ with $\frac{\ell-1}{2\ell} < \alpha < \frac{\ell}{2\ell+2}$, then for the learned feature map $\mathbf{F}$ and the untrained feature map $\mathbf{F}_0$, we have*

$$\mathcal{L}_{\text{tr}}(\mathbf{F}_0) - \mathcal{L}_{\text{tr}}(\mathbf{F}) \to_P \Delta_{\text{tr}} > 0$$

$$\mathcal{L}_{\text{te}}(\mathbf{F}_0) - \mathcal{L}_{\text{te}}(\mathbf{F}) \to_P \Delta_{\text{te}} > 0$$

*The expression for $\Delta_{\text{te/tr}}$ can be found in the paper.*

# Analysis of the Training/Test Errors



# Simulations

# Simulation Results

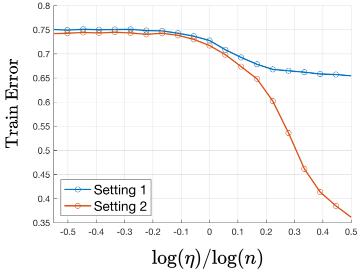
We consider

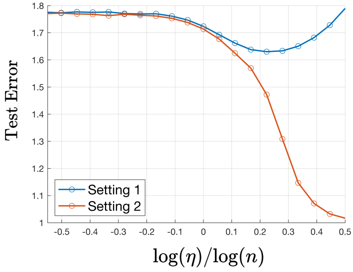
$$\textbf{Setting 1}: y = H_1(\boldsymbol{\beta}_\star^\top \boldsymbol{x}) + \varepsilon, \quad \varepsilon \sim \mathsf{N}(0,1),$$

$$\textbf{Setting 2}: y = H_1(\boldsymbol{\beta}_\star^\top \boldsymbol{x}) + \frac{1}{\sqrt{2}} H_2(\boldsymbol{\beta}_\star^\top \boldsymbol{x}).$$

We consider

$$\textbf{Setting 1}: y = H_1(\boldsymbol{\beta}_\star^\top \boldsymbol{x}) + \varepsilon, \quad \varepsilon \sim \mathsf{N}(0,1),$$
$$\textbf{Setting 2}: y = H_1(\boldsymbol{\beta}_\star^\top \boldsymbol{x}) + \frac{1}{\sqrt{2}} H_2(\boldsymbol{\beta}_\star^\top \boldsymbol{x}).$$



# Conclusion

# Conclusion

- One-step gradient descent with step size $\eta \asymp n^{\alpha}$ can lead to feature learning.

# Conclusion

- One-step gradient descent with step size $\eta \asymp n^{\alpha}$ can lead to feature learning.
- Learned features depend on the range of $\alpha$.

# Conclusion

- One-step gradient descent with step size $\eta \asymp n^{\alpha}$ can lead to feature learning.
- Learned features depend on the range of $\alpha$.
- Unlike random features model, after one gradient update, the model can learn higher order polynomial components of the teacher function.

# References

# References I

Ba, J., Erdogdu, M. A., Suzuki, T., Wang, Z., Wu, D., and Yang, G. (2022). High-dimensional asymptotics of feature learning: How one gradient step improves the representation. In *Advances in Neural Information Processing Systems*.

Bombari, S., Kiyani, S., and Mondelli, M. (2023). Beyond the universal law of robustness: Sharper laws for random features and neural tangent kernels. In *International Conference on Machine Learning*.

Bombari, S. and Mondelli, M. (2023). Stability, generalization and privacy: Precise analysis for random and NTK features. *arXiv preprint arXiv:2305.12100*.

Clarté, L., Loureiro, B., Krzakala, F., and Zdeborová, L. (2023). On double-descent in uncertainty quantification in overparametrized models. In *International Conference on Artificial Intelligence and Statistics*.

Dandi, Y., Krzakala, F., Loureiro, B., Pesce, L., and Stephan, L. (2023). Learning two-layer neural networks, one (giant) step at a time. *arXiv preprint arXiv:2305.18270*.

Hassani, H. and Javanmard, A. (2022). The curse of overparametrization in adversarial training: Precise analysis of robust generalization for random features regression. *arXiv preprint arXiv:2201.05149*.

Hu, H. and Lu, Y. M. (2023). Universality laws for high-dimensional learning with random features. *IEEE Transactions on Information Theory*, 69(3).

Lee, D., Moniri, B., Huang, X., Dobriban, E., and Hassani, H. (2023). Demystifying disagreement-on-the-line in high dimensions. In *International Conference on Machine Learning*.

Lin, L. and Dobriban, E. (2021). What causes the test error? going beyond bias-variance via ANOVA. *Journal of Machine Learning Research*, 22:155–1.

Mei, S. and Montanari, A. (2022). The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766.

Moniri, B., Lee, D., Hassani, H., and Dobriban, E. (2023). A theory of non-linear feature learning with one gradient step in two-layer neural networks. *arXiv preprint arXiv:2310.07891*.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*.

# Questions?