# Lecture 13: Feature Learning

Instructor: Lei Wu [1]

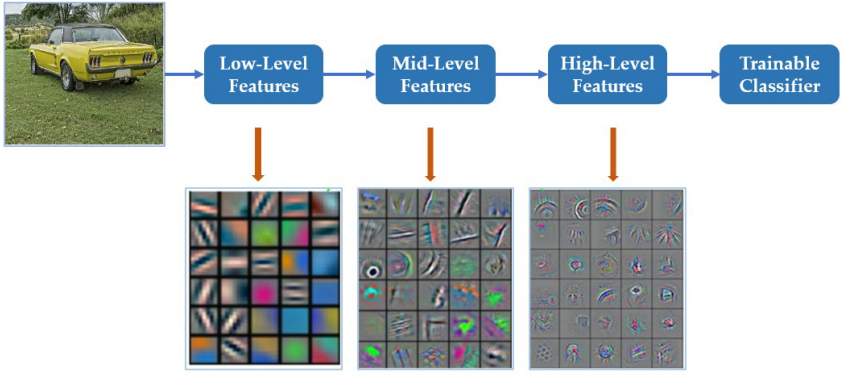
Topics in Deep Learning Theory (Spring 2025)

Acknowledgements: This slide is prepared with the assistance of Zihao Wang.

[1]School of Mathematical Sciences; Center for Machine Learning Research

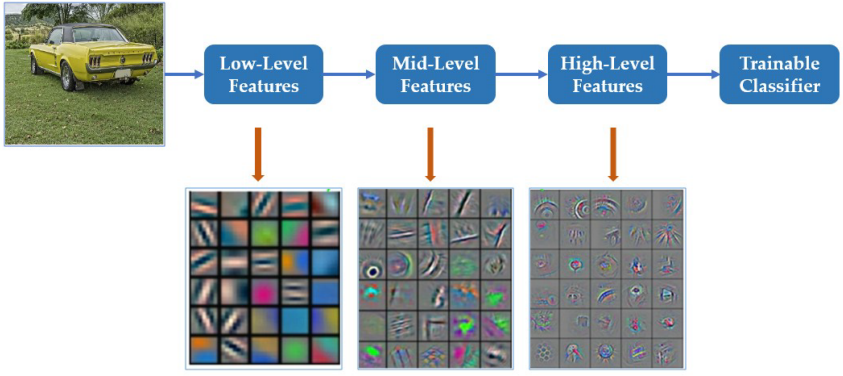
# Feature Learning

- In vison tasks, the result can be predicted using merely a few important **features**.



# Feature Learning

- In vison tasks, the result can be predicted using merely a few important **features**.
- For the CIFAR-10 classification task, the state-of-the-art CNNs can achieve test accuracy 99%, while current kernel methods can only get 87%-92%.



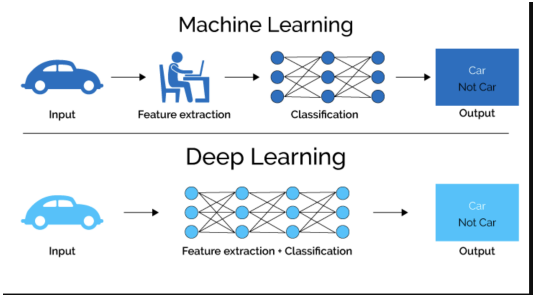
- **The most important conference in deep learning.**


International Conference on Learning Representations 2013

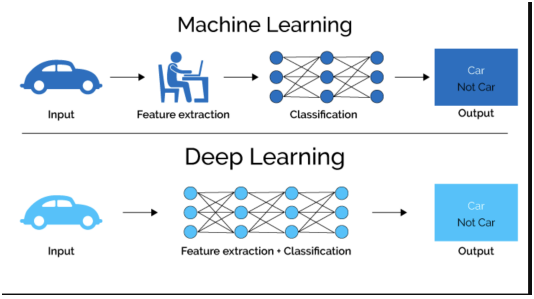
Important information

**Videos of oral presentations are available on the Conference Program page.**

# Feature Learning Changes the Paradigm of Machine Learning

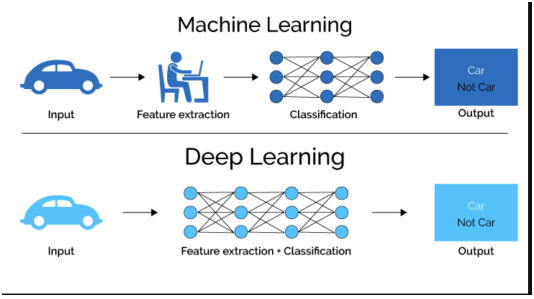


# Feature Learning Changes the Paradigm of Machine Learning



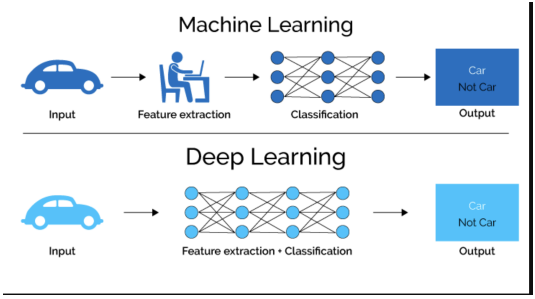

- Transfer learning (e.g., Pretraining + finetuing).
  Deep learning is made popular by the fact that DNNs trained in large-scale ImageNet dataset contains enormous vision-relevant features, which can be used in applying DNNs to other domain with little data.

# Feature Learning Changes the Paradigm of Machine Learning




- Transfer learning (e.g., Pretraining + finetuing).
  Deep learning is made popular by the fact that DNNs trained in large-scale ImageNet dataset contains enormous vision-relevant features, which can be used in applying DNNs to other domain with little data.
- Large language models (LLMs) like ChatGPT contains lots of features, which can be useful for downstream tasks.

# Feature Learning Changes the Paradigm of Machine Learning




- Transfer learning (e.g., Pretraining + finetuing).
  Deep learning is made popular by the fact that DNNs trained in large-scale ImageNet dataset contains enormous vision-relevant features, which can be used in applying DNNs to other domain with little data.
- Large language models (LLMs) like ChatGPT contains lots of features, which can be useful for downstream tasks.
- The feature learning has paved a way towards AGI.

## How can NNs learn features?

We attack this problem by considering the learning of multi-index models.

- Toolbox: Hermite analysis and boolean analysis.

## How can NNs learn features?

We attack this problem by considering the learning of multi-index models.

- Toolbox: Hermite analysis and boolean analysis.
- Introduction of multi-index models.

## How can NNs learn features?

We attack this problem by considering the learning of multi-index models.

- Toolbox: Hermite analysis and boolean analysis.
- Introduction of multi-index models.
- Learnability of multi-index models with modified algorithms.

## How can NNs learn features?

We attack this problem by considering the learning of multi-index models.

- Toolbox: Hermite analysis and boolean analysis.
- Introduction of multi-index models.
- Learnability of multi-index models with modified algorithms.
- Learnability of multi-index models via vanilla SGD: **information exponent** and **leap complexity**.

## How can NNs learn features?

We attack this problem by considering the learning of multi-index models.

- Toolbox: Hermite analysis and boolean analysis.
- Introduction of multi-index models.
- Learnability of multi-index models with modified algorithms.
- Learnability of multi-index models via vanilla SGD: **information exponent** and **leap complexity**.
- Other formulations and open questions.

# Toolbox: Fourier Analysis of Gaussian and Hamming Spaces

- **Hermite Analysis**: An analysis of the **Gaussian space** $L^2(\gamma_d)$ with $\gamma_d = \mathcal{N}(0, I_d)$ by using Hermite polynomials.
  - We shall use $\gamma_d$ to denote both $\mathcal{N}(0, I_d)$ and the corresponding PDF. When $d = 1$, we drop the subscript for brevity.

# Toolbox: Fourier Analysis of Gaussian and Hamming Spaces

- **Hermite Analysis**: An analysis of the **Gaussian space** $L^2(\gamma_d)$ with $\gamma_d = \mathcal{N}(0, I_d)$ by using Hermite polynomials.
  - We shall use $\gamma_d$ to denote both $\mathcal{N}(0, I_d)$ and the corresponding PDF. When $d = 1$, we drop the subscript for brevity.
- **Boolean Analysis**: An analysis of the **Hamming space** $L^2(\pi_d)$ with $\pi_d = \mathrm{Unif}(\{-1, +1\}^d)$ by using monomials.

# 1D Hermite Polynomials (HPs)

## Definition 1 (1D HPs)

Let $\gamma(x) = \exp(-x^2/2)/\sqrt{2\pi}$. The $k$-th **normalized** probabilist's HP [2], $h_k : \mathbb{R} \to \mathbb{R}$, is the degree $k$ polynomial given by

$$h_k(x) = \frac{(-1)^k}{\sqrt{k!}} \frac{\frac{\mathrm{d}^k \gamma}{\mathrm{d}x^k}(x)}{\gamma(x)}. \tag{1}$$

[2] The un-normalized one is $\mathrm{He}_k(x) = \sqrt{k!} h_k(x)$, for which the leading term's coefficient is always 1. The normalization ensures $\|h_k\|_{L^2(\gamma)} = 1$.

# 1D Hermite Polynomials (HPs)

## Definition 1 (1D HPs)

Let $\gamma(x) = \exp(-x^2/2)/\sqrt{2\pi}$. The $k$-th **normalized** probabilist's HP [2], $h_k : \mathbb{R} \to \mathbb{R}$, is the degree $k$ polynomial given by

$$h_k(x) = \frac{(-1)^k}{\sqrt{k!}} \frac{\frac{\mathrm{d}^k \gamma}{\mathrm{d}x^k}(x)}{\gamma(x)}. \tag{1}$$

- The first forth such HPs are

$$h_0(x) = 1, h_1(x) = x, h_2(x) = \frac{x^2 - 1}{\sqrt{2}}, h_3(x) = \frac{x^3 - 3x}{\sqrt{6}}, h_4(x) = \frac{x^4 - 6x^2 + 3}{\sqrt{4!}}.$$

- The physicist's HP is given by $\mathrm{H}_k(x) = (-1)^k e^{-x^2} \frac{\mathrm{d}^k}{\mathrm{d}x^k} e^{-x^2}$.
- We mostly use the normalized probabilist's HPs due to its brevity for $L^2$ analysis.

[2] The un-normalized one is $\mathrm{He}_k(x) = \sqrt{k!} h_k(x)$, for which the leading term's coefficient is always 1. The normalization ensures $\|h_k\|_{L^2(\gamma)} = 1$.

# 1D HPs (cont'd)

- HPs form a complete orthonormal basis of $L^2(\gamma)$:

$$\mathbb{E}_{x\sim\gamma}[h_j(x)h_k(x)] = \langle h_j, h_k\rangle_\gamma = \delta_{j,k}.$$

# 1D HPs (cont'd)

- HPs form a complete orthonormal basis of $L^2(\gamma)$:

$$\mathbb{E}_{x\sim\gamma}[h_j(x)h_k(x)] = \langle h_j, h_k\rangle_\gamma = \delta_{j,k}.$$

- For any $f \in L^2(\gamma)$, the Hermite expansion of $f$ is given by

$$f(x) = \sum_{k=0}^{\infty} \hat{f}_k h_k(x),$$

where $\{\hat{f}\}_{k=0}^{\infty}$ is referred to as the "Hermite coefficient" satisfying

$$\hat{f}_k = \langle f, h_k\rangle = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(z)h_k(z)e^{-\frac{z^2}{2}}\,\mathrm{d}z.$$

# 1D HPs (cont'd)

- HPs form a complete orthonormal basis of $L^2(\gamma)$:

$$\mathbb{E}_{x\sim\gamma}[h_j(x)h_k(x)] = \langle h_j, h_k\rangle_\gamma = \delta_{j,k}.$$

- For any $f \in L^2(\gamma)$, the Hermite expansion of $f$ is given by

$$f(x) = \sum_{k=0}^{\infty} \hat{f}_k h_k(x),$$

where $\{\hat{f}\}_{k=0}^{\infty}$ is referred to as the "Hermite coefficient" satisfying

$$\hat{f}_k = \langle f, h_k\rangle = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(z)h_k(z)e^{-\frac{z^2}{2}}\,\mathrm{d}z.$$

- **Plancherel's Theorem:** For any $f, g \in L^2(\gamma)$, we have

$$\langle f, g\rangle_\gamma = \sum_{k=0}^{\infty} \hat{f}_k \hat{g}_k.$$

# 1D HPs (cont'd)

The following facts will be useful:

- **Recurrence relation**: $\mathrm{He}_k'(x) = x\mathrm{He}_k(x) - \mathrm{He}_{k+1}(x) = k\mathrm{He}_{k-1}(x)$.
- Let $\gamma_d = \mathcal{N}(0, I_d)$. For any $f, g \in L^2(\gamma)$ and $u, v \in \mathbb{S}^{d-1}$, we have

$$\mathbb{E}_{x\sim\gamma_d}\left[f(u^\top x)g(v^\top x)\right] = \sum_{k=0}^{\infty} \hat{f}_k \hat{g}_k (u^\top v)^k$$

(The proof is left as homework.)

We refer to Section 11.2 of [Donnell, 2021] for more details.

# Multidimensional Hermite Polynomials

**Definition 2**

For a multi-index $\alpha \in \mathbb{N}^d$, we define the (normalized) multivariate HP $h_\alpha : \mathbb{R}^d \mapsto \mathbb{R}$ by

$$h_\alpha(x) = \prod_{j=1}^{d} h_{\alpha_j}(x_j).$$

- The total degree of $h_\alpha$ is $|\alpha| := \sum_{j=1}^{d} \alpha_j$.
- $\{h_\alpha\}_{\alpha \in \mathbb{N}^d}$ forms a "Fourier basis" of $L^2(\gamma_d)$.

# Hermite Tensors

We can organize the multidimensional HPs using *Hermite tensors*:

**Definition 3 (Hermite tensors)**

The $k$-th normalized Hermite tensor $\mathrm{H}_k : \mathbb{R}^d \to (\mathbb{R}^d)^{\otimes k}$ is defined as

$$\mathrm{H}_k(x) = \frac{(-1)^k}{\sqrt{k!}} \frac{\nabla^k \gamma_d(x)}{\gamma_d(x)}.$$

# Hermite Tensors

We can organize the multidimensional HPs using *Hermite tensors*:

**Definition 3 (Hermite tensors)**

The $k$-th normalized Hermite tensor $\mathrm{H}_k : \mathbb{R}^d \to (\mathbb{R}^d)^{\otimes k}$ is defined as

$$\mathrm{H}_k(x) = \frac{(-1)^k}{\sqrt{k!}} \frac{\nabla^k \gamma_d(x)}{\gamma_d(x)}.$$

- When $k = 0$, $\mathrm{H}_0(x) = (h_0(x))$.

# Hermite Tensors

We can organize the multidimensional HPs using *Hermite tensors*:

**Definition 3 (Hermite tensors)**

The $k$-th normalized Hermite tensor $\mathrm{H}_k : \mathbb{R}^d \to (\mathbb{R}^d)^{\otimes k}$ is defined as

$$\mathrm{H}_k(x) = \frac{(-1)^k}{\sqrt{k!}} \frac{\nabla^k \gamma_d(x)}{\gamma_d(x)}.$$

- When $k = 0$, $\mathrm{H}_0(x) = (h_0(x))$.
- When $k = 1$, $\mathrm{H}_1(x) = (h_1(x_1), h_1(x_2), \ldots, h_1(x_d)) \in \mathbb{R}^d$.

# Hermite Tensors

We can organize the multidimensional HPs using *Hermite tensors*:

## Definition 3 (Hermite tensors)

The $k$-th normalized Hermite tensor $\mathrm{H}_k : \mathbb{R}^d \to (\mathbb{R}^d)^{\otimes k}$ is defined as

$$\mathrm{H}_k(x) = \frac{(-1)^k}{\sqrt{k!}} \frac{\nabla^k \gamma_d(x)}{\gamma_d(x)}.$$

- When $k = 0$, $\mathrm{H}_0(x) = (h_0(x))$.
- When $k = 1$, $\mathrm{H}_1(x) = (h_1(x_1), h_1(x_2), \ldots, h_1(x_d)) \in \mathbb{R}^d$.
- When $k = 2$,

$$\mathrm{H}_2(x) = \begin{pmatrix} h_2(x_1) & h_1(x_1)h_1(x_2) & \ldots & h_1(x_1)h_1(x_d) \\ h_1(x_2)h_1(x_1) & h_2(x_2) & \ldots & h_1(x_2)h_1(x_d) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(x_d)h_1(x_1) & h_1(x_d)h_1(x_1) & \ldots & h_2(x_d) \end{pmatrix} = \frac{1}{2}(xx^\top - I_d) \in \mathbb{R}^{d \times d}.$$

# Hermite Expansion via Hermite Tensors

- For for any $A, B \in (\mathbb{R}^d)^{\otimes k}$, define $\langle A, B\rangle = \sum_{i_1,i_2,\ldots,i_k\in[d]} A_{i_1,\ldots,i_d} B_{i_1,\ldots,i_d}$.
- Using Hermite tensors, one can write the multidimensional Hermite expansion in the following succinct manner:

$$f(x) = \sum_{k\geq 0} \langle C_k(f), \mathrm{H}_k(x)\rangle \quad \text{where} \quad C_k(f) := \mathbb{E}_{x\sim\gamma_d}[f(x)\mathrm{H}_k(x)],$$

where $C_k(f) \in (\mathbb{R}^d)^{\otimes k}$.

# Hermite Tensors

The following lemmas will be useful.

**Lemma 4**

*If* $\|u\| = 1$, *we have*

$$h_k(\langle u, x\rangle) = \left\langle \mathrm{H}_k(x), u^{\otimes k} \right\rangle.$$

**Lemma 5 (Generalized Stein's Lemma)**

$$\sqrt{k!} \cdot \mathbb{E}_{x\sim\gamma}\left[f(x)\mathrm{H}_k(x)\right] = \mathbb{E}_{x\sim\gamma}\left[\nabla_x^k f(x)\right]$$

- The stein's lemma implies that in Gaussian space, the expected $k$-order gradients of a function are equal to its corresponding $k$-th Hermite coefficients.

For a reference on Hermite Tensors, we refer to Note on N-dimensional hermite polynomials by Harold Grad (too old, we need a better reference.)

# Boolean Analysis [3]

- Let $\mathbb{F}_d = \{-1, 1\}^d$ be the Hamming cubes equiped with Hamming distance:

$$\Delta(x, y) = \#\{i : x_i \neq y_i, i \in [d]\}, \text{ for any } x, y \in \mathbb{F}_d.$$

[3] We refer to Analysis of Boolean Functions by O'Donnell for more details.

# Boolean Analysis [3]

- Let $\mathbb{F}_d = \{-1, 1\}^d$ be the Hamming cubes equiped with Hamming distance:

$$\Delta(x, y) = \#\{i : x_i \neq y_i, i \in [d]\}, \text{ for any } x, y \in \mathbb{F}_d.$$

- Let $\mathcal{F} = \{f : \mathbb{F}_d \mapsto \mathbb{R}\}$ be the set of real-valued functions over $\mathbb{F}_d$. Then, $\mathcal{F}$ is a $2^d$-dimensional space.

[3] We refer to Analysis of Boolean Functions by O'Donnell for more details.

# Boolean Analysis [3]

- Let $\mathbb{F}_d = \{-1, 1\}^d$ be the Hamming cubes equiped with Hamming distance:

$$\Delta(x, y) = \#\{i : x_i \neq y_i, i \in [d]\}, \text{ for any } x, y \in \mathbb{F}_d.$$

- Let $\mathcal{F} = \{f : \mathbb{F}_d \mapsto \mathbb{R}\}$ be the set of real-valued functions over $\mathbb{F}_d$. Then, $\mathcal{F}$ is a $2^d$-dimensional space.
- Typical boolean functions include

[3] We refer to Analysis of Boolean Functions by O'Donnell for more details.

# Boolean Analysis [3]

- Let $\mathbb{F}_d = \{-1, 1\}^d$ be the Hamming cubes equiped with Hamming distance:

$$\Delta(x, y) = \#\{i : x_i \neq y_i, i \in [d]\}, \text{ for any } x, y \in \mathbb{F}_d.$$

- Let $\mathcal{F} = \{f : \mathbb{F}_d \mapsto \mathbb{R}\}$ be the set of real-valued functions over $\mathbb{F}_d$. Then, $\mathcal{F}$ is a $2^d$-dimensional space.
- Typical boolean functions include
  - **Parity/Monomials**: $f(x) = \chi_S(x) = \prod_{i \in S} x_i$ for $S \subset [d]$. When $S = \emptyset$, define $\chi_S(x) = 1$. Parity functions can be understood as indicator functions.

[3] We refer to Analysis of Boolean Functions by O'Donnell for more details.

# Boolean Analysis [3]

- Let $\mathbb{F}_d = \{-1, 1\}^d$ be the Hamming cubes equiped with Hamming distance:

$$\Delta(x, y) = \#\{i : x_i \neq y_i, i \in [d]\}, \text{ for any } x, y \in \mathbb{F}_d.$$

- Let $\mathcal{F} = \{f : \mathbb{F}_d \mapsto \mathbb{R}\}$ be the set of real-valued functions over $\mathbb{F}_d$. Then, $\mathcal{F}$ is a $2^d$-dimensional space.
- Typical boolean functions include
  - **Parity/Monomials**: $f(x) = \chi_S(x) = \prod_{i \in S} x_i$ for $S \subset [d]$. When $S = \emptyset$, define $\chi_S(x) = 1$. Parity functions can be understood as indicator functions.
  - **Majority:**

$$\mathrm{Maj}_d(x) = \begin{cases} 1 & \text{if } \#\{i : x_i = 1\} \geq \#\{i : x_i = -1\} \\ -1 & \text{otherwise} \end{cases}.$$

[3] We refer to Analysis of Boolean Functions by O'Donnell for more details.

# Fourier Analysis over Hamming Cubes

- Recall $\pi_d = \mathrm{Unif}(\mathbb{F}_d)$. Define the $L^2(\pi_d)$ by

$$\langle f, g \rangle = \mathbb{E}_{x \sim \pi_d}[f(x)g(x)] = \frac{1}{2^d} \sum_{x \in \mathbb{F}_d} f(x)g(x).$$

**Proposition 6**

*The $2^d$ parity functions $\{\chi_S\}_{S \subset [d]}$ forms a complete orthonormal basis of $L^2(\pi_d)$.*

# Fourier Analysis over Hamming Cubes

- Recall $\pi_d = \mathrm{Unif}(\mathbb{F}_d)$. Define the $L^2(\pi_d)$ by

$$\langle f, g \rangle = \mathbb{E}_{x \sim \pi_d}[f(x)g(x)] = \frac{1}{2^d} \sum_{x \in \mathbb{F}_d} f(x)g(x).$$

**Proposition 6**

*The $2^d$ parity functions $\{\chi_S\}_{S \subset [d]}$ forms a complete orthonormal basis of $L^2(\pi_d)$.*

**Proof:** We only need to verify the orthonormal property: For any $S, T \subset [d]$

$$\chi_S(x)\chi_T(x) = \prod_{i \in S} x_i \prod_{j \in T} x_j = \prod_{i \in S \Delta T} x_i \prod_{j \in S \cap T} x_j^2 = \prod_{i \in S \Delta T} x_i = \chi_{S \Delta T}(x),$$

where $S \Delta T$ denote the symmetric difference. Taking expectation completes the proof.

# Fourier Analysis over Hamming Cubes (Cont'd)

**Definition 7**

For any $f \in \mathcal{F}$, its Fourier-Walsh expansion is given by

$$f(x) = \sum_{S \in [d]} \widehat{f}_S \chi_S(x),$$

where $\widehat{f}_S = \mathbb{E}[f(x)\chi_S(x)]$.

# Fourier Analysis over Hamming Cubes (Cont'd)

**Definition 7**

For any $f \in \mathcal{F}$, its Fourier-Walsh expansion is given by

$$f(x) = \sum_{S \in [d]} \widehat{f}_S \chi_S(x),$$

where $\widehat{f}_S = \mathbb{E}[f(x)\chi_S(x)]$.

**Example:**

$$\mathrm{Maj}_3(x_1, x_2, x_3) = \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_1x_2x_3. \tag{2}$$

# Fourier Analysis over Hamming Cubes (Cont'd)

**Definition 7**

For any $f \in \mathcal{F}$, its Fourier-Walsh expansion is given by

$$f(x) = \sum_{S \in [d]} \widehat{f}_S \chi_S(x),$$

where $\widehat{f}_S = \mathbb{E}[f(x)\chi_S(x)]$.

**Example:**

$$\mathrm{Maj}_3(x_1, x_2, x_3) = \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_1x_2x_3. \tag{2}$$

**Some important properties:**

- $\mathbb{E}[f^2] = \sum_{S \subset [d]} \widehat{f}_S^2$ and $\mathbb{E}[fg] = \sum_S \widehat{f}_S \widehat{g}_S$.
- $\mathbb{E}[f] = \widehat{f}_\emptyset$.
- $\mathrm{Var}[f] = \sum_{S \neq \emptyset} \widehat{f}_S^2$.

# Single/Multi-Index Models

- Consider a target function that takes the form:

$$f(x) = g(v \cdot x) \text{ with } g : \mathbb{R} \mapsto \mathbb{R} \text{ and } v \in \mathbb{R}^d.$$

  This type of models are called **single-index models** and $g$ is referred to as the **link function**.
  - When $g$ is known, it is tyically referred to as generalized linear model (GLM), e.g., logistic regression.
  - When $g$ is unknown but assumed to be monotonic, this is called "isotonic regression".
- The mutil-index model is a natural extesion:

$$f(x) = g(Ux) = g(u_1^\top x, u_2^\top x, \ldots, u_r^\top x),$$

  where $g : \mathbb{R}^r \mapsto \mathbb{R}$ and $U \in \mathbb{R}^{r \times d}$.

# Learning Multi-index Models

- Learning a multi-index model needs to capture both
  - the underlying "**feature**" $U \in \mathbb{R}^{r \times d}$;
  - the low-dimensional **link function** $g : \mathbb{R}^r \mapsto \mathbb{R}$.
- In Lecture 2, we already showed that two-layer networks can approximate and estimate multi-index models efficiently.
- The remaining question is about the optimization:
  - Whether does there exist an algorithm that can learn multi-index models efficiently?
  - Whether does standard algorithm such as SGD can learn multi-index models efficiently?

# The Setup of Learning Multi-index Models

- Suppose that inputs are drawn from $\gamma_d = \mathcal{N}(0, I_d)$ and $f^*(x) = g(Ux)$ with $UU^\top = I_k$.

# The Setup of Learning Multi-index Models

- Suppose that inputs are drawn from $\gamma_d = \mathcal{N}(0, I_d)$ and $f^*(x) = g(Ux)$ with $UU^\top = I_k$.
- Consider a two-layer learner network

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

with a square loss $L(\theta) = \mathbb{E}\left[(f_\theta(x) - f^*(x))^2\right]$.

# The Setup of Learning Multi-index Models

- Suppose that inputs are drawn from $\gamma_d = \mathcal{N}(0, I_d)$ and $f^*(x) = g(Ux)$ with $UU^\top = I_k$.
- Consider a two-layer learner network

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

  with a square loss $L(\theta) = \mathbb{E}\left[(f_\theta(x) - f^*(x))^2\right]$.
- Initialization:

# The Setup of Learning Multi-index Models

- Suppose that inputs are drawn from $\gamma_d = \mathcal{N}(0, I_d)$ and $f^*(x) = g(Ux)$ with $UU^\top = I_k$.
- Consider a two-layer learner network

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

  with a square loss $L(\theta) = \mathbb{E}\left[(f_\theta(x) - f^*(x))^2\right]$.
- Initialization:
  - suppose $a_j \in \{+1, -1\}$ and $w_j \sim \mathcal{N}(0, \delta^2 I_d)$ and $b_j = 0$.

# The Setup of Learning Multi-index Models

- Suppose that inputs are drawn from $\gamma_d = \mathcal{N}(0, I_d)$ and $f^*(x) = g(Ux)$ with $UU^\top = I_k$.
- Consider a two-layer learner network

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

  with a square loss $L(\theta) = \mathbb{E}\left[(f_\theta(x) - f^*(x))^2\right]$.
- Initialization:
  - suppose $a_j \in \{+1, -1\}$ and $w_j \sim \mathcal{N}(0, \delta^2 I_d)$ and $b_j = 0$.
  - we shall use a **symmetric trick** to ensure at initialization

$$f_\theta(x) \equiv 0.$$

# The Role of Symmetric Initialization

Under the symmetric initialization, we have at initialization,

$$\begin{aligned}\nabla_{w_j} L(\theta) &= \mathbb{E}\left[2\left(f_\theta(x) - f^\star(x)\right)\nabla_{w_j} f_\theta(x)\right] \\ &= -2\mathbb{E}\left[f^\star(x)\nabla_{w_j} f_\theta(x)\right] \\ &= -2a_j\,\mathbb{E}[f^*(x)\sigma'(w_j^\top x)x]\end{aligned}$$

The key observation is that

**The "feature $U$" can be decodered using $\nabla_{w_j} L(\theta)$ via appropriately tuning the scale parameter $\delta$.**

We will discussed the following two cases separately:

- small scale: $\delta \ll 1$;
- vanilla scale: $\delta = 1/d$.

# Small-scale Initialization

When $\delta \ll 1$, we have

$$\begin{aligned}\nabla_{w_j} L(\theta) &= -2a_j \, \mathbb{E}[f^*(x)\sigma'(w_j \cdot x)x] \\ &\approx -2\sigma'(0)a_j \, \mathbb{E}[f^*(x)x] = -2\sigma'(0)a_j \, \mathbb{E}[\nabla f^*(x)]\end{aligned}$$

# Small-scale Initialization

When $\delta \ll 1$, we have

$$\begin{aligned}\nabla_{w_j} L(\theta) &= -2a_j \, \mathbb{E}[f^*(x)\sigma'(w_j \cdot x)x] \\ &\approx -2\sigma'(0)a_j \, \mathbb{E}[f^*(x)x] = -2\sigma'(0)a_j \, \mathbb{E}[\nabla f^*(x)]\end{aligned}$$

- Recalling $f^*(x) = g(Ux)$, we have

$$\begin{aligned}\mathbb{E}[\nabla f^*(x)] &= U^\top \mathbb{E}[\nabla g(Ux)] \\ &= U \, \mathbb{E}_{z\sim\mathcal{N}(0,I_k)}[\nabla g(z)] = U \, \mathbb{E}[g(z)z] = UC_1(g),\end{aligned}$$

  where $C_1(g) = \langle g, \mathrm{H}_1 \rangle$.
- When $r = 1$, as long as $C_1(g) \neq 0$, one-step GD can capture the "feature".
- When $r > 1$, the gradient can only recover a subspace of $U$, i.e. $UC_1(g)$.

# Vanilla-scale Initialization in High Dimension

- Analogously, we need to look at

$$\nabla_{w_j} L(\theta) \propto \mathbb{E}_x \left[ f^\star(x) \sigma'(w_j^\top x) x \right]$$

# Vanilla-scale Initialization in High Dimension

- Analogously, we need to look at

$$\nabla_{w_j} L(\theta) \propto \mathbb{E}_x \left[ f^\star(x) \sigma'(w_j^\top x) x \right]$$

- Let $\sigma' = \sum_{k=0}^{\infty} c_k h_k$ be the Hermite expansion of $\sigma'$. Then, we have

$$\mathbb{E}_x \left[ f^*(x) x \sigma'(\langle w_j, x \rangle) \right] = \sum_{k \geq 0} c_k \, \mathbb{E}_x \left[ f^*(x) x h_k(\langle w_j, x \rangle) \right].$$

# Vanilla-scale Initialization in High Dimension

- Analogously, we need to look at

$$\nabla_{w_j} L(\theta) \propto \mathbb{E}_x \left[ f^\star(x) \sigma'(w_j^\top x) x \right]$$

- Let $\sigma' = \sum_{k=0}^{\infty} c_k h_k$ be the Hermite expansion of $\sigma'$. Then, we have

$$\mathbb{E}_x \left[ f^*(x) x \sigma'(\langle w_j, x \rangle) \right] = \sum_{k \geq 0} c_k \, \mathbb{E}_x \left[ f^*(x) x h_k(\langle w_j, x \rangle) \right].$$

- We assume $\mathbb{E}_x \left[ f^*(x) \right] = \mathbb{E}_x \left[ f^*(x) x \right] = \mathbb{E}_x \left[ \nabla f^*(x) \right] = 0$ (this can be done by a preprocessing step on the data, i.e., removing the linear part).

# Vanilla-scale Initialization in High Dimension

- Analogously, we need to look at

$$\nabla_{w_j} L(\theta) \propto \mathbb{E}_x \left[ f^\star(x) \sigma'(w_j^\top x) x \right]$$

- Let $\sigma' = \sum_{k=0}^{\infty} c_k h_k$ be the Hermite expansion of $\sigma'$. Then, we have

$$\mathbb{E}_x \left[ f^*(x) x \sigma'(\langle w_j, x \rangle) \right] = \sum_{k \geq 0} c_k \, \mathbb{E}_x \left[ f^*(x) x h_k(\langle w_j, x \rangle) \right].$$

- We assume $\mathbb{E}_x \left[ f^*(x) \right] = \mathbb{E}_x \left[ f^*(x) x \right] = \mathbb{E}_x \left[ \nabla f^*(x) \right] = 0$ (this can be done by a preprocessing step on the data, i.e., removing the linear part).
- Then, we have

$$\nabla_{w_j} L(\theta) = -2 a_j \sum_{k \geq 1} c_k \, \mathbb{E}_x \left[ f^*(x) x h_k(\langle w_j, x \rangle) \right]$$

# Vanilla-scale Initialization in High Dimension

- Analogously, we need to look at

$$\nabla_{w_j} L(\theta) \propto \mathbb{E}_x \left[ f^\star(x) \sigma'(w_j^\top x) x \right]$$

- Let $\sigma' = \sum_{k=0}^{\infty} c_k h_k$ be the Hermite expansion of $\sigma'$. Then, we have

$$\mathbb{E}_x \left[ f^*(x) x \sigma'(\langle w_j, x \rangle) \right] = \sum_{k \geq 0} c_k \, \mathbb{E}_x \left[ f^*(x) x h_k(\langle w_j, x \rangle) \right].$$

- We assume $\mathbb{E}_x \left[ f^*(x) \right] = \mathbb{E}_x \left[ f^*(x) x \right] = \mathbb{E}_x \left[ \nabla f^*(x) \right] = 0$ (this can be done by a preprocessing step on the data, i.e., removing the linear part).
- Then, we have

$$\nabla_{w_j} L(\theta) = -2 a_j \sum_{k \geq 1} c_k \, \mathbb{E}_x \left[ f^*(x) x h_k(\langle w_j, x \rangle) \right]$$

- **Question:** Why do we want to remove the linear part?

# Vanilla Initialization in High Dimension (Cont'd)

- For the calculation in the multi-index case, we need to use Hermite Tensors.

# Vanilla Initialization in High Dimension (Cont'd)

- For the calculation in the multi-index case, we need to use Hermite Tensors.
- For any $k \in \mathbb{N}$, we have

$$\begin{aligned}\mathbb{E}_x\left[f^*(x)xh_k(\langle w_j, x\rangle)\right] &= \mathbb{E}_x\left[f^*(x)x\langle \mathrm{H}_k(x), w_j^{\otimes k}\rangle\right] \\ &= \langle w_j^{\otimes k}, \mathbb{E}_x\left[f^*(x)x\mathrm{H}_k(x)\right] \\ &= \frac{1}{\sqrt{k!}}\langle w_j^{\otimes k}, \mathbb{E}_x\left[\nabla_x^k\left(f^*(x)x\right)\right]\rangle\end{aligned}$$

# Vanilla Initialization in High Dimension (Cont'd)

- For the calculation in the multi-index case, we need to use Hermite Tensors.
- For any $k \in \mathbb{N}$, we have

$$\begin{aligned}\mathbb{E}_x\left[f^*(x)xh_k(\langle w_j, x\rangle)\right] &= \mathbb{E}_x\left[f^*(x)x\langle \mathrm{H}_k(x), w_j^{\otimes k}\rangle\right] \\ &= \langle w_j^{\otimes k}, \mathbb{E}_x\left[f^*(x)x\mathrm{H}_k(x)\right] \\ &= \frac{1}{\sqrt{k!}}\langle w_j^{\otimes k}, \mathbb{E}_x\left[\nabla_x^k\left(f^*(x)x\right)\right]\rangle\end{aligned}$$

- Noticing that for $w \sim \mathrm{Unif}(\mathbb{S}^{d-1})$ and fixed $A \in (\mathbb{R}^d)^{\otimes k}$, we have w.h.p. that

$$\langle w^{\otimes k}, A\rangle \sim d^{-k/2}.$$

# Vanilla Initialization in High Dimension (Cont'd)

- For the calculation in the multi-index case, we need to use Hermite Tensors.
- For any $k \in \mathbb{N}$, we have

$$\begin{aligned}\mathbb{E}_x\left[f^*(x)xh_k(\langle w_j, x\rangle)\right] &= \mathbb{E}_x\left[f^*(x)x\langle \mathrm{H}_k(x), w_j^{\otimes k}\rangle\right] \\ &= \langle w_j^{\otimes k}, \mathbb{E}_x\left[f^*(x)x\mathrm{H}_k(x)\right] \\ &= \frac{1}{\sqrt{k!}}\langle w_j^{\otimes k}, \mathbb{E}_x\left[\nabla_x^k\left(f^*(x)x\right)\right]\rangle\end{aligned}$$

- Noticing that for $w \sim \mathrm{Unif}(\mathbb{S}^{d-1})$ and fixed $A \in (\mathbb{R}^d)^{\otimes k}$, we have w.h.p. that

$$\langle w^{\otimes k}, A\rangle \sim d^{-k/2}.$$

- Hence,

$$\nabla_{w_j} L(\theta) \approx -2a_j c_1\, \mathbb{E}[f^*(x)xh_1(\langle w, x\rangle)] = -2a_j c_1\, \mathbb{E}[f^*(x)x\langle w, x\rangle].$$

# Vanilla Initialization in High Dimension (Cont'd)

- Upon previous analysis, we have

$$\begin{aligned}\nabla_{w_j} L(\theta) &\sim \mathbb{E}\left[f^*(x)xx^\top\right] w_j \\ &= 2\,\mathbb{E}\left[f^*(x)(\mathrm{H}_2(x) + I_d)\right] w_j = 2\,\mathbb{E}\left[f^*(x)\mathrm{H}_2(x)\right] w_j \\ &= \mathbb{E}_x\left[\nabla^2 f^*(x)\right] w_j\end{aligned}$$

# Vanilla Initialization in High Dimension (Cont'd)

- Upon previous analysis, we have

$$\begin{aligned}\nabla_{w_j} L(\theta) &\sim \mathbb{E}\left[f^*(x)xx^\top\right] w_j \\ &= 2\,\mathbb{E}\left[f^*(x)(\mathrm{H}_2(x) + I_d)\right] w_j = 2\,\mathbb{E}\left[f^*(x)\mathrm{H}_2(x)\right] w_j \\ &= \mathbb{E}_x\left[\nabla^2 f^*(x)\right] w_j\end{aligned}$$

- Recall that $f^*(x) = g(Ux)$. We have

$$\begin{aligned}\mathbb{E}[\nabla^2 f^*(x)] &= U^\top\,\mathbb{E}[\nabla^2 g(Ux)]U = U^\top\,\mathbb{E}_{z\sim\gamma_r}[\nabla^2 g(z)]U \\ &= \sqrt{2}U^\top\,\mathbb{E}[g(z)\mathrm{H}_2(z)]U\end{aligned}$$

# Vanilla Initialization in High Dimension (Cont'd)

- Upon previous analysis, we have

$$\begin{aligned}\nabla_{w_j} L(\theta) &\sim \mathbb{E}\left[f^*(x) x x^\top\right] w_j \\ &= 2\,\mathbb{E}\left[f^*(x)(\mathrm{H}_2(x) + I_d)\right] w_j = 2\,\mathbb{E}\left[f^*(x)\mathrm{H}_2(x)\right] w_j \\ &= \mathbb{E}_x\left[\nabla^2 f^*(x)\right] w_j\end{aligned}$$

- Recall that $f^*(x) = g(Ux)$. We have

$$\begin{aligned}\mathbb{E}[\nabla^2 f^*(x)] &= U^\top\,\mathbb{E}[\nabla^2 g(Ux)]U = U^\top\,\mathbb{E}_{z\sim\gamma_r}[\nabla^2 g(z)]U \\ &= \sqrt{2}U^\top\,\mathbb{E}[g(z)\mathrm{H}_2(z)]U\end{aligned}$$

- Assume $Q_g := \mathbb{E}[g(z)\mathrm{H}_2(z)]$ is full-rank. Then, due to $w_j \sim \mathcal{N}(0, I_d)$, we have

$$\nabla_{w_j} L(\theta) \sim \mathcal{N}(0, \sqrt{2}U^\top Q_g U),$$

i.e., $\nabla_{w_j} L(\theta)$ approximately obeys **a Gaussian distribution with full-rank covariance matrix in the subspace**. This is necessary for our kernel step arguments and the final learnability guarantee.

# Results

Consider the modified gradient-based algorithm:

- **Phase I (feature Learning)**: $W^{(1)} = W^{(0)} - \eta_1 \nabla_W L(\theta^{(0)})$.
- **Phase II (sub-space random feature)**: re-initialize $b_j \sim \mathcal{N}(0, 1)$. Then, for $t = 2, \ldots, T$

$$a^{(t)} = a^{(t-1)} - \eta_2 [\nabla_a L(a^{(t)}, W^{(1)}, b) + \lambda_t a^{(t-1)}].$$

**Theorem 8 (Informal, Alex et al., COLT 2022)**

*Under previously mentioned condition, the above modified GD can learn the multi-index targets using*

- $\mathcal{O}(d^2)$ *samples*
- *polynomial time.*

*On the contrary, kernel methods suffer from the CoD for a general link function* $g$.

# Remarks

- The previous learnability requires the Hermite coeffient matrix $\mathbb{E}[g(z)\mathrm{H}_2(z)]$ to be full-rank.

# Remarks

- The previous learnability requires the Hermite coeffient matrix $\mathbb{E}[g(z)\mathrm{H}_2(z)]$ to be full-rank.
- Gradient retrieves features by using only the second-order information.

# Remarks

- The previous learnability requires the Hermite coeffient matrix $\mathbb{E}[g(z)\mathrm{H}_2(z)]$ to be full-rank.
- Gradient retrieves features by using only the second-order information.
- Pure high-order targets like $f^*(x) = h_k(\langle u, x\rangle)$ with $k \geq 3$ and $\|u\| = 1$ do not satisfy the above condition.

Towards a time/sample complexity estimation
for feature learning with standard algorithms

# Problem Setup

- In the first part of this section, we will study the time/sample complexity of learning the single-index models

$$h^*(x) = \phi(\langle w^*, x \rangle)$$

via gradient descent, where $w^*, x \in \mathbb{R}^d$ and $\|w^*\| = 1$. We assume the input distribution is $N(0, I_d)$.

# Problem Setup

- In the first part of this section, we will study the time/sample complexity of learning the single-index models

$$h^*(x) = \phi(\langle w^*, x \rangle)$$

via gradient descent, where $w^*, x \in \mathbb{R}^d$ and $\|w^*\| = 1$. We assume the input distribution is $N(0, I_d)$.

- The complexity of extracting the latent feature $w^*$ completely depends on the structure of $\phi$.

# Single-index Models and Information Exponent

Consider single-index models

$$f(x) = \phi(\langle w, x \rangle).$$

What properties of $\phi$ and $w$ determine the learnability?

# Single-index Models and Information Exponent

Consider single-index models

$$f(x) = \phi(\langle w, x \rangle).$$

What properties of $\phi$ and $w$ determine the learnability?

**Definition 9**

For a function $\phi \in L^2(\gamma)$, let $\phi = \sum_{k=0}^{\infty} \hat{\phi}_k h_k$ be the Hermite expansion of $\phi$. The **information exponent** of $\phi$ is defined as

$$k^\star(\phi) = \min\{k : \hat{\phi}_k \neq 0\}.$$

# Learning Setup and Result

- Consider the model $h_\theta(x) = \phi(\langle w, x\rangle)$ to learn the target $h^*(x) = \phi(\langle w^*, x\rangle)$. Here, we assume the link function is known.
- **Online spherical SGD**: $w_{t+1} = \mathbb{P}_{\mathbb{S}^{d-1}}\left(w_t - \eta \nabla_{\mathbb{S}^{d-1}} \hat{L}(w_t)\right)$ with random initialization $w_0 \sim \mathrm{Unif}(\mathbb{S}^{d-1})$. Here $\hat{L}$ is the minibatch loss.

# Learning Setup and Result

- Consider the model $h_\theta(x) = \phi(\langle w, x\rangle)$ to learn the target $h^*(x) = \phi(\langle w^*, x\rangle)$. Here, we assume the link function is known.
- **Online spherical SGD**: $w_{t+1} = \mathbb{P}_{\mathbb{S}^{d-1}}\left(w_t - \eta \nabla_{\mathbb{S}^{d-1}} \hat{L}(w_t)\right)$ with random initialization $w_0 \sim \mathrm{Unif}(\mathbb{S}^{d-1})$. Here $\hat{L}$ is the minibatch loss.

**Theorem 10 (Informal, Arous et al., JMLR 2021)**

*To make the final test loss $o_d(1)$, $\widetilde{\mathcal{O}}(d^{\max(k^*-1,1)})$ training steps suffice. Therefore, the sample complexity is $\widetilde{\mathcal{O}}(d^{\max(k^*-1,1)})$ and the time complexity is $\widetilde{\mathcal{O}}(d^{\max(k^*-1,1)+1})$.*

# Proof Intuition

- Population loss

$$L(w) := \frac{1}{2}\,\mathbb{E}\left[\left(\phi(\langle w, x\rangle) - \phi(\langle w^*, x\rangle)\right)^2\right]$$

# Proof Intuition

- Population loss

$$L(w) := \frac{1}{2}\mathbb{E}\left[(\phi(\langle w, x\rangle) - \phi(\langle w^*, x\rangle))^2\right]$$

- For technical simplicity, we assume $\|w_t\| = 1$ throughout the training (This can be done by using spherical gradient descent). In that case, we have

$$L(w) = \text{constant} - \mathbb{E}\left[\phi(\langle w, x\rangle)\phi(\langle w^*, x\rangle)\right] = \text{constant} - \sum_{k\geq 0} \hat{\phi}_k^2 \langle w, w^*\rangle^k$$

# Proof Intuition

- Population loss

$$L(w) := \frac{1}{2}\mathbb{E}\left[(\phi(\langle w, x\rangle) - \phi(\langle w^*, x\rangle))^2\right]$$

- For technical simplicity, we assume $\|w_t\| = 1$ throughout the training (This can be done by using spherical gradient descent). In that case, we have

$$L(w) = \text{constant} - \mathbb{E}\left[\phi(\langle w, x\rangle)\phi(\langle w^*, x\rangle)\right] = \text{constant} - \sum_{k\geq 0}\hat{\phi}_k^2\langle w, w^*\rangle^k$$

- From the above argument, we have

Minimizing $L(w)$ is equivalent with maximizing $\sum_{k\geq 0}\hat{\phi}_k^2\langle w, w^*\rangle^k$.

# Proof Intuition

- Training objective $\sum_{k\geq 0} \hat{\phi}_k^2 \langle w, w^* \rangle^k$ and gradient $\sum_{k\geq 1} k\hat{\phi}_k^2 \langle w, w^* \rangle^{k-1} w^*$.

# Proof Intuition

- Training objective $\sum_{k\geq 0} \hat{\phi}_k^2 \langle w, w^* \rangle^k$ and gradient $\sum_{k\geq 1} k\hat{\phi}_k^2 \langle w, w^* \rangle^{k-1} w^*$.
- At the initialization, $\langle w, w^* \rangle = \Theta(d^{-1/2})$. When $\langle w, w^* \rangle = \Theta(1)$, the training will significantly speed up since the gradient becomes much larger. Therefore, in order to characterize the time/sample complexity, we only need to focus on the training process when $\langle w, w^* \rangle = o_d(1)$.

# Proof Intuition

- Training objective $\sum_{k\geq 0}\hat{\phi}_k^2\langle w, w^*\rangle^k$ and gradient $\sum_{k\geq 1}k\hat{\phi}_k^2\langle w, w^*\rangle^{k-1}w^*$.
- At the initialization, $\langle w, w^*\rangle = \Theta(d^{-1/2})$. When $\langle w, w^*\rangle = \Theta(1)$, the training will significantly speed up since the gradient becomes much larger. Therefore, in order to characterize the time/sample complexity, we only need to focus on the training process when $\langle w, w^*\rangle = o_d(1)$.
- When $\langle w, w^*\rangle = o_d(1)$, the training objective approximates $\hat{\phi}_{k^*}^2\langle w, w^*\rangle^{k^*}$. Therefore, WLOG we can just consider the case $\phi = h_{k^*}$ where the population loss becomes

$$L(w) = 1 - \langle w, w^*\rangle^{k^*}.$$

# Proof Intuition

- Training objective $\sum_{k\geq 0}\hat{\phi}_k^2\langle w, w^*\rangle^k$ and gradient $\sum_{k\geq 1}k\hat{\phi}_k^2\langle w, w^*\rangle^{k-1}w^*$.
- At the initialization, $\langle w, w^*\rangle = \Theta(d^{-1/2})$. When $\langle w, w^*\rangle = \Theta(1)$, the training will significantly speed up since the gradient becomes much larger. Therefore, in order to characterize the time/sample complexity, we only need to focus on the training process when $\langle w, w^*\rangle = o_d(1)$.
- When $\langle w, w^*\rangle = o_d(1)$, the training objective approximates $\hat{\phi}_{k^*}^2\langle w, w^*\rangle^{k^*}$. Therefore, WLOG we can just consider the case $\phi = h_{k^*}$ where the population loss becomes

$$L(w) = 1 - \langle w, w^*\rangle^{k^*}.$$

- The spherical GF is given by

$$\dot{w}_t = -(I - w_t w_t^\top)\nabla L(w_t).$$

# Proof Intuition

- The spherical GF is

$$\dot{w}_t = k^* \left( \langle w_t, w_* \rangle^{k^*-1} w^* - \langle w_t, w_* \rangle^{k^*} w_t \right)$$

Let $R_t = \langle w_t, w^* \rangle$. Then,

$$\dot{R}_t = k^* R_t^{k^*-1} (1 - R_t^2).$$

# Proof Intuition

- The spherical GF is

$$\dot{w}_t = k^* \left( \langle w_t, w_* \rangle^{k^*-1} w^* - \langle w_t, w_* \rangle^{k^*} w_t \right)$$

Let $R_t = \langle w_t, w^* \rangle$. Then,

$$\dot{R}_t = k^* R_t^{k^*-1} (1 - R_t^2).$$

- Denote $R_t = \langle w_t, w^* \rangle$, then we have

$$R_{t+1} - R_t = \eta k^* R_t^{k^*-1} (1 - R_t^2)$$

with $R_0 = \Theta(d^{-1/2})$.

# Result

Let $T_{0.5} = \inf\{t : R_t^2 \geq 0.5\}$. Then,

- When $k^* = 1$, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}(R_t^2) = R_t^2(1 - R_t^2).$$

  Therefore, $R_t^2 = R_0^2 e^t$ when $t \in T_{0.5}$. Therefore, for this case, we only need steps

$$O(\log d).$$

- When $k^* > 1$, if $R_0 < 0$, $\lim_{t\to\infty} R_t \neq 1$. This means the learning can succeed iff

$$R_0 = \langle w_0, w^* \rangle > 0.$$

  By assuming $R_0 = O(d^{-1/2}) > 0$, we need

$$\Theta(\eta^{-1} d^{\frac{k^*}{2}-1})$$

  steps.

# From GF to SGD

- For online SGD, we have

$$\begin{aligned} R_{t+1} - R_t &= -\eta \langle \mathbb{P}_{w_t} \nabla \hat{L}_t(w_t), w^* \rangle \\ &= \eta k^* R_t^{k^*-1} - \eta \langle \nabla \hat{L}_t(w_t) - \nabla L(w_t), w^* \rangle \end{aligned}$$

# From GF to SGD

- For online SGD, we have

$$\begin{aligned} R_{t+1} - R_t &= -\eta \langle \mathbb{P}_{w_t} \nabla \hat{L}_t(w_t), w^* \rangle \\ &= \eta k^* R_t^{k^*-1} - \eta \langle \nabla \hat{L}_t(w_t) - \nabla L(w_t), w^* \rangle \end{aligned}$$

- $\sum_{1 \le t \le T} \eta \langle \nabla \hat{L}(w_t) - \nabla L(w_t), w^* \rangle$ is the martingale part. Due to $\langle \nabla \hat{L}_t(w_t) - \nabla L(w_t), w^* \rangle = \Theta(1)$, the martingale part should scale as $\eta\sqrt{T}$ where $T$ is the number of total steps.

# From GF to SGD

- For online SGD, we have

$$\begin{aligned} R_{t+1} - R_t &= -\eta \langle \mathbb{P}_{w_t} \nabla \hat{L}_t(w_t), w^* \rangle \\ &= \eta k^* R_t^{k^*-1} - \eta \langle \nabla \hat{L}_t(w_t) - \nabla L(w_t), w^* \rangle \end{aligned}$$

- $\sum_{1 \le t \le T} \eta \langle \nabla \hat{L}(w_t) - \nabla L(w_t), w^* \rangle$ is the martingale part. Due to $\langle \nabla \hat{L}_t(w_t) - \nabla L(w_t), w^* \rangle = \Theta(1)$, the martingale part should scale as $\eta\sqrt{T}$ where $T$ is the number of total steps.
- To keep the martingale (noise) part controllable, we should have $\eta\sqrt{T} = \Theta\left(\frac{1}{\sqrt{d}}\right)$, thus leading to $\eta = \Theta(d^{-k^*/2})$ and $T = \Theta(\eta^{-1} d^{\frac{k^*}{2}-1}) = \Theta(d^{k^*-1})$.

# Beyond Single-Index Models: Leap Complexity

Recall the targets

$$f(x) = g(Ux) \text{ with } g : \mathbb{R}^r \mapsto \mathbb{R}.$$

- In single-index model, we have shown the information exponent of link function determines the time complexity.

# Beyond Single-Index Models: Leap Complexity

Recall the targets

$$f(x) = g(Ux) \text{ with } g : \mathbb{R}^r \mapsto \mathbb{R}.$$

- In single-index model, we have shown the information exponent of link function determines the time complexity.
- For multi-index model, we shall show that the leap complexity of link function determines the time/sample complexity.

# Leap Complexity

- Any function in $L^2\left(\mu^{\otimes r}\right)$ can be expressed in the orthogonal basis of $L^2\left(\mu^{\otimes r}\right)$, i.e., the Hermite or Fourier-Walsh basis for $\mu \sim \mathrm{N}(0,1)$ and $\mu \sim \mathrm{Unif}(\{+1,-1\})$ respectively,

$$h_*(\boldsymbol{z}) = \sum_{S \in \mathcal{Z}^r} \hat{h}_*(S) \chi_S(\boldsymbol{z}),$$

where $\mathcal{Z} = \{0,1\}$ for the Boolean case and $\mathcal{Z} = \mathbb{Z}_+$for the Gaussian case, $\chi_S(\boldsymbol{z}) = \prod_{i \in [P]} \chi_{S_i}\left(z_i\right)$,

$$\chi_{S_i}\left(z_i\right) = \begin{cases} z_i^{S_i} & \text{(Boolean case)} \\ h_{S_i}\left(z_i\right) & \text{(Gaussian case)} \end{cases}$$

where $h_k$ is the $k$-th Hermite polynomial, $k \in \mathbb{Z}_+$.

# Leap Complexity

**Definition 11 (Leap Complexity, (Abbe et al., COLT2022))**

For $h : \{-1,+1\}^r \mapsto \mathbb{R}$, let $\mathcal{S}(h) := \{S_1, \ldots, S_m\}, m \in \mathbb{Z}_+$. We define the leap complexity of $h_*$ as

$$\mathrm{Leap}(h) := \min_{\pi \in \Pi_m} \max_{i \in [m]} \left\| S_{\pi(i)} \setminus \cup_{j=0}^{i-1} S_{\pi(j)} \right\|_1,$$

where, for $S_j = (S_j(1), \ldots, S_j(P))$ in $\{0,1\}^r$ or $\mathbb{Z}_+^r$ for the Boolean or Gaussian case respectively, $\left\| S_{\pi(i)} \setminus \cup_{j=0}^{i-1} S_{\pi(j)} \right\|_1 := \sum_{k \in [P]} S_{\pi(i)}(k) 1_{\{S_{\pi(j)}(k)=0, \forall j \in [i-1]\}}$, with $S_{\pi(0)} = 0^P$.

- In words, a function $h$ is leap-$k$ if its non-zero monomials can be ordered in a sequence such that each time a monomial is added, the support of $h$ grows by at most $k$ new coordinates, where each new coordinate is counted with multiplicity in the Gaussian case.

# Leap Complexity

- Some examples in the Boolean case.

$$\text{Leap}\,(z_1 + z_1z_2 + z_1z_2z_3 + z_1z_2z_3z_4) = 1, \quad \text{Leap}\,(z_1 + z_2 + z_2z_3z_4) = 2,$$
$$\text{Leap}\,(z_1 + z_1z_2z_3 + z_2z_3z_4z_5z_6z_7) = 4, \quad \text{Leap}\,(z_1z_2z_3 + z_2z_3z_4) = 3$$

# Leap Complexity

- Some examples in the Boolean case.

$$\text{Leap}\left(z_1 + z_1z_2 + z_1z_2z_3 + z_1z_2z_3z_4\right) = 1, \quad \text{Leap}\left(z_1 + z_2 + z_2z_3z_4\right) = 2,$$
$$\text{Leap}\left(z_1 + z_1z_2z_3 + z_2z_3z_4z_5z_6z_7\right) = 4, \quad \text{Leap}\left(z_1z_2z_3 + z_2z_3z_4\right) = 3$$

- Some examples on isotropic Gaussian data.

$$\text{Leap}\left(h_k\left(z_1\right)\right) = \text{Leap}\left(h_1\left(z_1\right)h_1\left(z_2\right)\cdots h_1\left(z_k\right)\right) = k$$
$$\text{Leap}\left(h_{k_1}\left(z_1\right) + h_{k_1}\left(z_1\right)h_{k_2}\left(z_2\right) + h_{k_1}\left(z_1\right)h_{k_2}\left(z_2\right)h_{k_3}\left(z_3\right)\right) = \max\left(k_1, k_2, k_3\right),$$
$$\text{Leap}\left(h_2\left(z_1\right) + h_2\left(z_2\right) + h_2\left(z_3\right) + h_3\left(z_1\right)h_8\left(z_3\right)\right) = 2$$

# Experiments

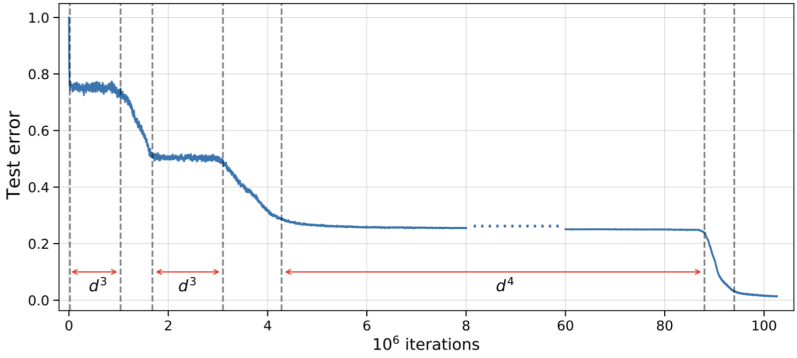



Figure 1: Test error versus the number of online-SGD steps to learn $h_*(\boldsymbol{z}) = z_1 + z_1z_2\cdots z_5 + z_1z_2\cdots z_9 + z_1z_2\cdots z_{14}$ in ambient dimension $d = 100$ on the hypercube. We take $M = 300$ neurons with shifted sigmoid activation and train both layers at once with constant step size $0.4/d$. The SGD dynamics follows a saddle-to-saddle dynamic and sequentially picks up the support and monomials $z_1$ in roughly $d$ steps, $z_1z_2\cdots z_5$ in $d^3$ steps (leap of size 4), $z_1z_2\cdots z_9$ in $d^3$ steps (leap of size 4) and $z_1z_2\cdots z_{14}$ in $d^4$ steps (leap of size 5).

# Conjectures

- Let $f_* : \mathbb{R}^d \to \mathbb{R}$ in $L_2\left(\mu^{\otimes d}\right)$ for $\mu$ either $\mathcal{N}(0,1)$ or $\mathrm{Unif}\{+1,-1\}$ satisfying $f_*(x) = g(Ux)$ where $U \in \mathbb{R}^{r\times d}$ and $r = O_d(1)$.

# Conjectures

- Let $f_* : \mathbb{R}^d \to \mathbb{R}$ in $L_2\left(\mu^{\otimes d}\right)$ for $\mu$ either $\mathcal{N}(0,1)$ or $\mathrm{Unif}\{+1,-1\}$ satisfying $f_*(x) = g(Ux)$ where $U \in \mathbb{R}^{r\times d}$ and $r = O_d(1)$.
- Let $\hat{f}^t_{\mathrm{NN}}$ be the output of training a FCN with $\mathrm{poly}(d)$ edges and rotationally-invariant weight initialization with $t$ steps of one-pass online-SGD on the square loss.

# Conjectures

- Let $f_* : \mathbb{R}^d \to \mathbb{R}$ in $L_2\left(\mu^{\otimes d}\right)$ for $\mu$ either $\mathcal{N}(0,1)$ or $\mathrm{Unif}\{+1,-1\}$ satisfying $f_*(x) = g(Ux)$ where $U \in \mathbb{R}^{r\times d}$ and $r = O_d(1)$.
- Let $\hat{f}^t_{\mathrm{NN}}$ be the output of training a FCN with $\mathrm{poly}(d)$ edges and rotationally-invariant weight initialization with $t$ steps of one-pass online-SGD on the square loss.
- **Claim**: For almost all targets, the risk is bounded by

$$\mathbb{E}_x\left[\left(\hat{f}^t_{\mathrm{NN}}(x) - f_*(x)\right)^2\right] \le \varepsilon \quad \text{if and only if} \quad t = \tilde{\Omega}_d\left(d^{(\mathrm{Leap}(h_*)-1)\vee 1}\right)\mathrm{poly}(1/\varepsilon).$$

# Conjectures

- Let $f_* : \mathbb{R}^d \to \mathbb{R}$ in $L_2\left(\mu^{\otimes d}\right)$ for $\mu$ either $\mathcal{N}(0,1)$ or $\mathrm{Unif}\{+1,-1\}$ satisfying $f_*(x) = g(Ux)$ where $U \in \mathbb{R}^{r\times d}$ and $r = O_d(1)$.
- Let $\hat{f}^t_{\mathrm{NN}}$ be the output of training a FCN with $\mathrm{poly}(d)$ edges and rotationally-invariant weight initialization with $t$ steps of one-pass online-SGD on the square loss.
- **Claim**: For almost all targets, the risk is bounded by

$$\mathbb{E}_x\left[\left(\hat{f}^t_{\mathrm{NN}}(x) - f_*(x)\right)^2\right] \le \varepsilon \quad \text{if and only if} \quad t = \tilde{\Omega}_d\left(d^{(\mathrm{Leap}(h_*)-1)\vee 1}\right)\mathrm{poly}(1/\varepsilon).$$

- This claim is proposed and partially proved in SGD learning on neural networks: leap complexity and saddle-to-saddle dynamics by Abbe, Boix-Adsera and Misiakiewicz.

# Learning a Single Monomial

- From here, with a little abuse of notation, we use $\mathrm{He}_k(z) := \sqrt{k!}h_k(z)$ to denote the Hermite polynomials in 1D without normalization. We first consider the case of learning a single monomial with Hermite exponents $k_1, \ldots, k_P$:

$$f_*(x) = h_*(x_{\leq r}) \text{ where } h_*(\boldsymbol{z}) = \mathrm{He}_{k_1}(z_1)\,\mathrm{He}_{k_2}(z_2)\cdots\mathrm{He}_{k_r}(z_r)\,.$$

  We assume $D = k_1 + \ldots + k_r \geq 2$ WLOG.

# Learning a Single Monomial

- From here, with a little abuse of notation, we use $\mathrm{He}_k(z) := \sqrt{k!}h_k(z)$ to denote the Hermite polynomials in 1D without normalization. We first consider the case of learning a single monomial with Hermite exponents $k_1, \ldots, k_P$:

$$f_*(x) = h_*(x_{\leq r}) \text{ where } h_*(\boldsymbol{z}) = \mathrm{He}_{k_1}(z_1)\,\mathrm{He}_{k_2}(z_2)\cdots\mathrm{He}_{k_r}(z_r)\,.$$

  We assume $D = k_1 + \ldots + k_r \geq 2$ WLOG.

- Network architecture

$$f_\theta(x) = \sum a_j \sigma(\langle w_j, x\rangle + b_j)$$

# Learning a Single Monomial

- Let us firstly train $w_i$ while setting $b_i$ zero and $a_i$ very small at initialization. In that case, we have

$$L(\theta) = \mathbb{E}\left[(f_\theta(x) - f_*(x))^2\right] \approx \text{ constant } - 2\,\mathbb{E}\left[f_\theta(x) f_*(x)\right]$$

# Learning a Single Monomial

- Let us firstly train $w_i$ while setting $b_i$ zero and $a_i$ very small at initialization. In that case, we have

$$L(\theta) = \mathbb{E}\left[(f_\theta(x) - f_*(x))^2\right] \approx \text{ constant } - 2\,\mathbb{E}\left[f_\theta(x) f_*(x)\right]$$

- $\mathbb{E}\left[f_\theta(x) f_*(x)\right]$ is the correlation loss. In this loss different neurons do not interact. Therefore, let us directly consider the correlation loss and track the dynamics of a unique neuron $(a, w)$.

# Learning a Single Monomial

- Let us firstly train $w_i$ while setting $b_i$ zero and $a_i$ very small at initialization. In that case, we have

$$L(\theta) = \mathbb{E}\left[(f_\theta(x) - f_*(x))^2\right] \approx \text{ constant } - 2\,\mathbb{E}\left[f_\theta(x) f_*(x)\right]$$

- $\mathbb{E}\left[f_\theta(x) f_*(x)\right]$ is the correlation loss. In this loss different neurons do not interact. Therefore, let us directly consider the correlation loss and track the dynamics of a unique neuron $(a, w)$.
- We assume that $w_1^0 = \ldots = w_d^0 = 1/\sqrt{d}$ and firstly consider gradient flow here for technical convenience. In GF, the dynamics is described by only two parameters

$$\alpha_1^t = w_1^t = \ldots = w_P^t, \quad \alpha_2^t = w_{P+1}^t = \ldots = w_d^t,$$

# Learning a Single Monomial

- We recall the following useful identities (where $G \sim \mathrm{N}(0,1)$ )

$$\mathbb{E}_G\left[\mathrm{He}_k(G) g(G)\right] = \mathbb{E}_G\left[g^{(k)}(G)\right], \quad x\mathrm{He}_k(x) = \mathrm{He}_{k+1}(x) + k\mathrm{He}_{k-1}(x).$$

# Learning a Single Monomial

- We recall the following useful identities (where $G \sim \mathrm{N}(0,1)$ )

$$\mathbb{E}_G\left[\mathrm{He}_k(G) g(G)\right] = \mathbb{E}_G\left[g^{(k)}(G)\right], \quad x\mathrm{He}_k(x) = \mathrm{He}_{k+1}(x) + k\mathrm{He}_{k-1}(x).$$

- In particular, by integration by parts, we have

$$\mathbb{E}\left[\prod_{j\in[P]} \mathrm{He}_{k_j}\left(x_j\right) \sigma'\left(\left\langle w^t, x\right\rangle\right)\right] = \left(\prod_{j\in[P]} \left(w_j^t\right)^{k_j}\right) \cdot \mathbb{E}_G\left[\sigma^{(1+k_1+\ldots+k_P)}\left(\left\|w^t\right\| G\right)\right].$$

# Learning a Single Monomial

- We recall the following useful identities (where $G \sim \mathrm{N}(0,1)$ )

$$\mathbb{E}_G\left[\mathrm{He}_k(G) g(G)\right] = \mathbb{E}_G\left[g^{(k)}(G)\right], \quad x\mathrm{He}_k(x) = \mathrm{He}_{k+1}(x) + k\mathrm{He}_{k-1}(x).$$

- In particular, by integration by parts, we have

$$\mathbb{E}\left[\prod_{j\in[P]} \mathrm{He}_{k_j}\left(x_j\right) \sigma'\left(\left\langle w^t, x\right\rangle\right)\right] = \left(\prod_{j\in[P]} \left(w_j^t\right)^{k_j}\right) \cdot \mathbb{E}_G\left[\sigma^{(1+k_1+\ldots+k_P)}\left(\left\|w^t\right\| G\right)\right].$$

- Furthermore, if $i \in [P]$

$$x_i f_*(x) = \left(k_i \,\mathrm{He}_{k_i-1}\left(x_i\right) + \mathrm{He}_{k_i+1}\left(x_i\right)\right)\left(\prod_{j\in[P], j\neq i} \mathrm{He}_{k_j}\left(x_j\right)\right)$$

# Learning a Single Monomial

- The gradient for $\alpha_i^t$, $i = 1, 2$:

$$\mathbb{E}\left[h_*(\boldsymbol{z})\sigma'\left(\langle w^t, x\rangle\right) x_1\right] \approx \left(\alpha_1^t\right)^{D-1} \mathbb{E}\left[\sigma^{(D)}\left(\|w^t\| G\right)\right] \approx \mu_D(\sigma)\left(\alpha_1^t\right)^{D-1}$$

$$\mathbb{E}\left[h_*(\boldsymbol{z})\sigma'\left(\langle w^t, x\rangle\right) x_{P+1}\right] \approx \left(\alpha_1^t\right)^{D} \alpha_2^t \mathbb{E}\left[\sigma^{(D+2)}\left(\|w^t\| G\right)\right] \approx \mu_{D+2}(\sigma)\left(\alpha_1^t\right)^{D} \alpha_2^t$$

where $\mu_k(\sigma)$ is the $k$-th Hermite coefficient of $\sigma$.

# Learning a Single Monomial

- The gradient for $\alpha_i^t$, $i = 1, 2$:

$$\mathbb{E}\left[h_*(\boldsymbol{z})\sigma'\left(\langle w^t, x\rangle\right) x_1\right] \approx \left(\alpha_1^t\right)^{D-1} \mathbb{E}\left[\sigma^{(D)}\left(\left\|w^t\right\| G\right)\right] \approx \mu_D(\sigma)\left(\alpha_1^t\right)^{D-1}$$

$$\mathbb{E}\left[h_*(\boldsymbol{z})\sigma'\left(\langle w^t, x\rangle\right) x_{P+1}\right] \approx \left(\alpha_1^t\right)^{D} \alpha_2^t \mathbb{E}\left[\sigma^{(D+2)}\left(\left\|w^t\right\| G\right)\right] \approx \mu_{D+2}(\sigma)\left(\alpha_1^t\right)^{D} \alpha_2^t$$

  where $\mu_k(\sigma)$ is the $k$-th Hermite coefficient of $\sigma$.

- We can approximate the above dynamics via the following ODEs

$$\frac{dR_t}{dt} = R_t^{D-1} \quad \frac{dS_t}{dt} = R_t^D S_t \quad R_0 = S_0 = \frac{1}{\sqrt{d}}$$

# Learning a Single Monomial

- The gradient for $\alpha_i^t$, $i = 1, 2$:

$$\mathbb{E}\left[h_*(\boldsymbol{z})\sigma'\left(\langle w^t, x\rangle\right) x_1\right] \approx \left(\alpha_1^t\right)^{D-1} \mathbb{E}\left[\sigma^{(D)}\left(\|w^t\| G\right)\right] \approx \mu_D(\sigma)\left(\alpha_1^t\right)^{D-1}$$

$$\mathbb{E}\left[h_*(\boldsymbol{z})\sigma'\left(\langle w^t, x\rangle\right) x_{P+1}\right] \approx \left(\alpha_1^t\right)^{D} \alpha_2^t \mathbb{E}\left[\sigma^{(D+2)}\left(\|w^t\| G\right)\right] \approx \mu_{D+2}(\sigma)\left(\alpha_1^t\right)^{D} \alpha_2^t$$

  where $\mu_k(\sigma)$ is the $k$-th Hermite coefficient of $\sigma$.

- We can approximate the above dynamics via the following ODEs

$$\frac{dR_t}{dt} = R_t^{D-1} \quad \frac{dS_t}{dt} = R_t^D S_t \quad R_0 = S_0 = \frac{1}{\sqrt{d}}$$

- Thus, in order to have $R_t = \Theta(1)$, we need total steps $T = \Theta(d^{D-1})$ using similar arguments in the previous subsection.

# Learning the Nested Monomials

- Second, consider the case of nested monomials

$$h_*(\boldsymbol{z}) = \sum_{l=1}^{L} \prod_{s\in[P_l]} \mathrm{He}_{k_s}(z_s)\,,$$

where $0 =: P_0 < P_1 < P_2 < \ldots < P_L =: P$ and $k_1, \ldots, k_P$ are positive integers. For $l \in [L]$, we denote $D_l = k_{P_{l-1}+1} + \ldots + k_{P_l}$, and $D = \max_{l\in[L]} D_l$ the size of the biggest leap.

# Learning the Nested Monomials

- Second, consider the case of nested monomials

$$h_*(\boldsymbol{z}) = \sum_{l=1}^{L} \prod_{s\in[P_l]} \mathrm{He}_{k_s}(z_s),$$

where $0 =: P_0 < P_1 < P_2 < \ldots < P_L =: P$ and $k_1, \ldots, k_P$ are positive integers. For $l \in [L]$, we denote $D_l = k_{P_{l-1}+1} + \ldots + k_{P_l}$, and $D = \max_{l\in[L]} D_l$ the size of the biggest leap.

- Example:

$$h_*(\boldsymbol{z}) = z_1 \cdots z_{P_1} + z_1 \cdots z_{P_2} + \ldots + z_1 \cdots z_{P_L}$$

# Learning the Nested Monomials

- For understanding simplicity, consider the above example for now. The intuition is basically the same with the single monomial cases by comparing the gradient magnitude of each coordinate. Firstly we learn the coordinates $1, \ldots, P_1$ within time $\Theta(d^{P_1-1})$.

# Learning the Nested Monomials

- For understanding simplicity, consider the above example for now. The intuition is basically the same with the single monomial cases by comparing the gradient magnitude of each coordinate. Firstly we learn the coordinates $1, \ldots, P_1$ within time $\Theta(d^{P_1-1})$.
- After we learnt the coordinates $1, \ldots, P_1$, the gradients of other coordinates become larger. Thus, we can learn the coordinates $P_1 + 1, \ldots, P_2$ within time $\Theta(d^{P_2-P_1-1})$. Repeating this process, the overall time complexity indeed should be $\Theta(d^{D-1})$ where $D$ is the largest leap.

# Open Questions

- Can neural networks efficiently learn non-linear features, i.e, targets like $g(p_1(x), \ldots, p_R(x))$ where $p_1, \ldots, p_R$ are nonlinear functions?
- Can those results be extended to other input distributions? For example, when the input distribution has some known or unknown low dimension structure, can gradient descent utilize this structure and do feature learning more efficiently?
- Feature learning in other architectures, like CNNs and transformers?
- ...

Thanks!