Token Selection in the Self-Attention Mechanism: Case Studies and Theoretical Understanding

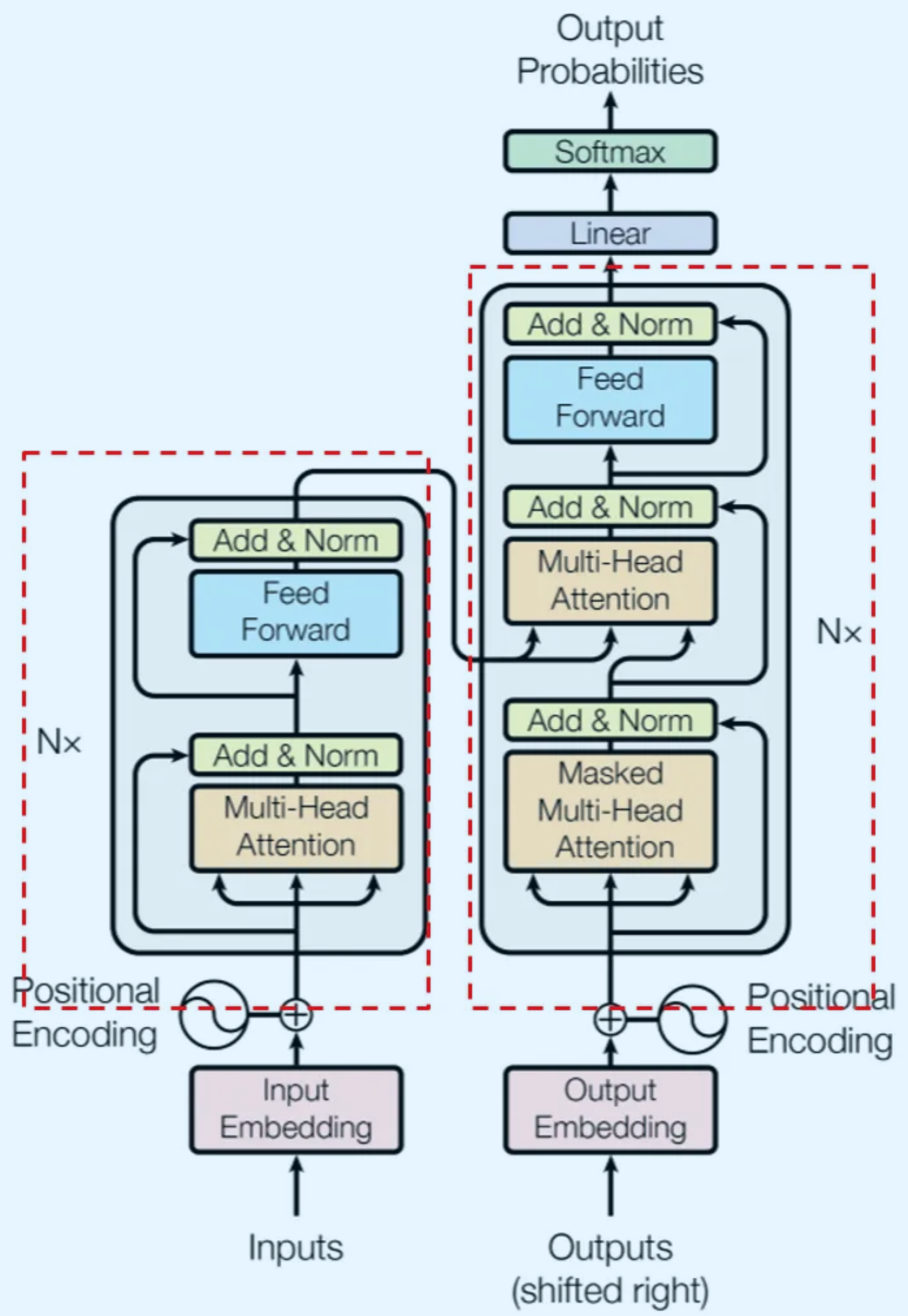
Yuan Cao

The University of Hong Kong

Joint work with: Zihao Li, Cheng Gao, Yihan He, Chenyang Zhang, Xuran Meng, Wei Shi, Han Liu, Jason Klusowski, Jianqing Fan, Mengdi Wang

Success of transformers

Theoretical understanding of transformers is limited



[Vaswani et al. 2017]

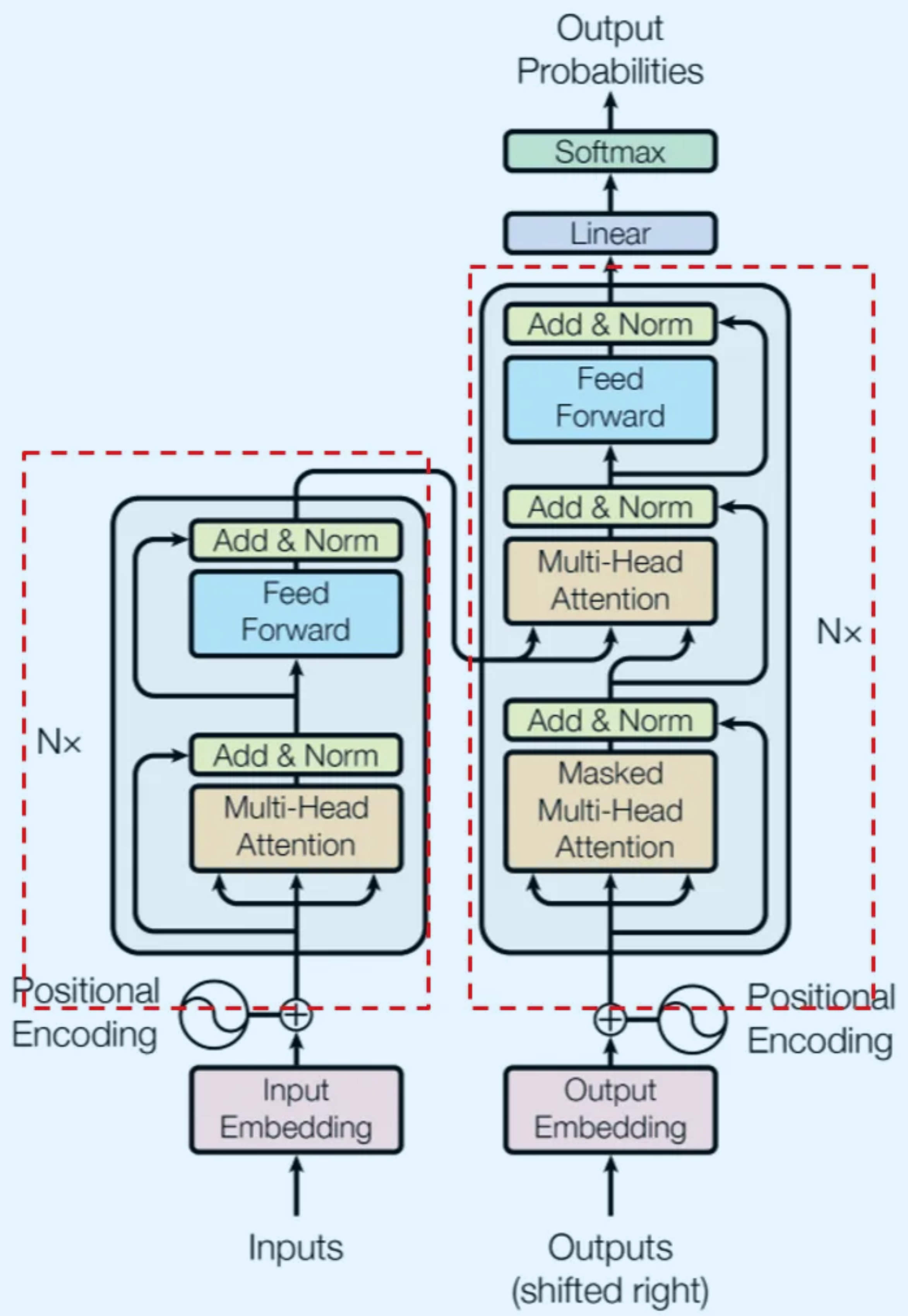
+

Large amount of natural language data

=

Powerful language model

Theoretical understanding of transformers is limited



[Vaswani et al. 2017]

+

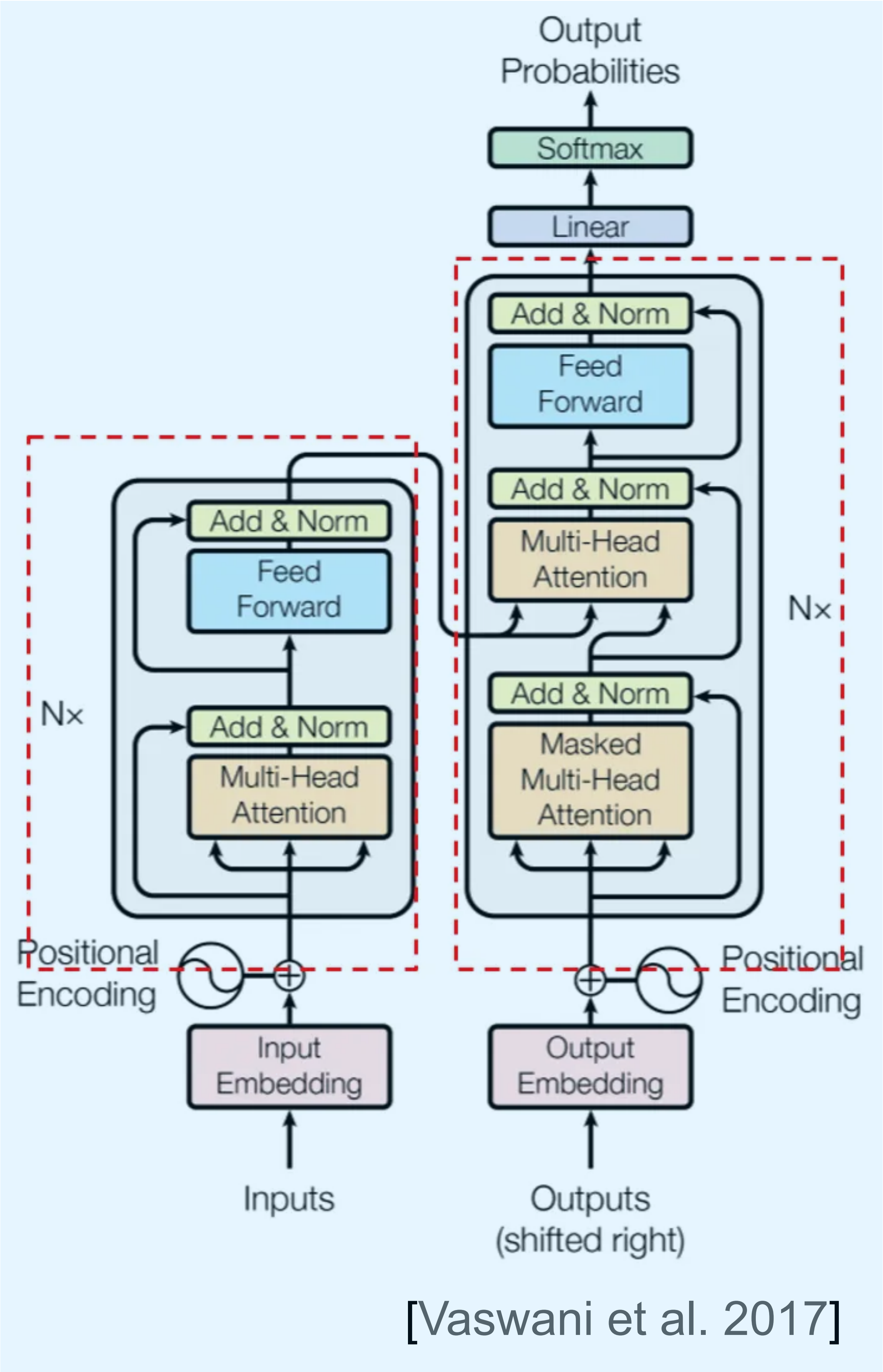
Large amount of natural language data

=

Powerful language model

Optimization/learning guarantees?

Theoretical understanding of transformers is limited



+

Large amount of natural language data

=

Powerful language model

Optimization/learning guarantees? **Interpretability?**

We consider...

Simple transformer $+$ **Data following classic statistical models** $= ?$

Nearest neighbor, group-sparse classification, random walk

We consider…

Simple transformer $+$ **Data following classic statistical models** $= ?$

Nearest neighbor, group-sparse classification, random walk

By considering such settings, we aim to understand transformers':

We consider…

Simple transformer $+$ **Data following classic statistical models** $= ?$

Nearest neighbor, group-sparse classification, random walk

By considering such settings, we aim to understand transformers':

Compatibility with classic models?

We consider…

Simple transformer $+$ **Data following classic statistical models** $= ?$

Nearest neighbor, group-sparse classification, random walk

By considering such settings, we aim to understand transformers':

Compatibility with classic models?

Adaptivity to a variety of classic tasks?

We consider…

Simple transformer $+$ **Data following classic statistical models** $= ?$

Nearest neighbor, group-sparse classification, random walk

By considering such settings, we aim to understand transformers':

Compatibility with classic models?

Adaptivity to a variety of classic tasks?

Capability to capture underlying statistical structures?

We consider…

Simple transformer $+$ **Data following classic statistical models** $= ?$

Nearest neighbor, group-sparse classification, random walk

By considering such settings, we aim to understand transformers':

Compatibility with classic models?

Adaptivity to a variety of classic tasks?

Capability to capture underlying statistical structures?

We will give **learning guarantees** & **interpretations of the trained model.**

Overview

Transformers as in-context one-nearest neighbor predictors

Zihao Li, Yuan Cao, Cheng Gao, Yihan He, Han Liu, Jason Klusowski, Jianqing Fan, and Mengdi Wang. "One-layer transformer provably learns one-nearest neighbor in context." NeurIPS 2024

Transformers as group-sparse linear predictors

Chenyang Zhang, Xuran Meng, and Yuan Cao. "Transformer learns optimal variable selection in group-sparse classification." ICLR 2025

Transformers as random walk predictors

Wei Shi and Yuan Cao. "Towards Understanding Transformers in Learning Random Walks." submitted.

Transformers Learn One-Nearest Neighbor In Context

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

A classic theoretical setup: in-context linear regression [Zhang et al., 2023, Bai et al. 2024, …]

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

A classic theoretical setup: in-context linear regression [Zhang et al., 2023, Bai et al. 2024, …]

Input matrix: $$\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \dots & y_N & 0 \\ \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_N & \mathbf{p}_{\text{query}} \end{bmatrix}$$

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

A classic theoretical setup: in-context linear regression [Zhang et al., 2023, Bai et al. 2024, …]

Input matrix: $$\mathbf{H} = \begin{bmatrix} \boxed{\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_N \quad \mathbf{x}_{\text{query}}} \\ y_1 \quad y_2 \quad \cdots \quad y_N \quad 0 \\ \mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_N \quad \mathbf{p}_{\text{query}} \end{bmatrix}$$ $\mathbf{x}_i, \mathbf{x}_{\text{query}} \in \mathbb{R}^d$

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

A classic theoretical setup: in-context linear regression [Zhang et al., 2023, Bai et al. 2024, …]

Input matrix: $\mathbf{H} = \begin{bmatrix} \boxed{\mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N & \mathbf{x}_{\text{query}}} \\ y_1 & y_2 & \cdots & y_N & 0 \\ \boxed{\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_N & \mathbf{p}_{\text{query}}} \end{bmatrix}$

$\mathbf{x}_i, \mathbf{x}_{\text{query}} \in \mathbb{R}^d$

"Positional encoding"

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

A classic theoretical setup: in-context linear regression [Zhang et al., 2023, Bai et al. 2024, …]

Input matrix: $$\mathbf{H} = \begin{bmatrix} \boxed{\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_N \quad \mathbf{x}_{\text{query}}} \\ y_1 \quad y_2 \quad \cdots \quad y_N \quad 0 \\ \boxed{\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_N \quad \mathbf{p}_{\text{query}}} \end{bmatrix}$$

$\mathbf{x}_i, \mathbf{x}_{\text{query}} \in \mathbb{R}^d$

"Positional encoding"

Output: $$[\text{self-attention}(\mathbf{H})]_{d+1,N+1}$$

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

A classic theoretical setup: in-context linear regression [Zhang et al., 2023, Bai et al. 2024, …]

Input matrix: $$\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \dots & y_N & \boxed{0} \\ \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_N & \mathbf{p}_{\text{query}} \end{bmatrix}$$

$\mathbf{x}_i, \mathbf{x}_{\text{query}} \in \mathbb{R}^d$

"Positional encoding"

Output: $$\boxed{[\text{self-attention}(\mathbf{H})]_{d+1,N+1}}$$

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

A classic theoretical setup: in-context linear regression [Zhang et al., 2023, Bai et al. 2024, …]

Input matrix: $\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \cdots & y_N & \boxed{0} \\ \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_N & \mathbf{p}_{\text{query}} \end{bmatrix}$

$\mathbf{x}_i, \mathbf{x}_{\text{query}} \in \mathbb{R}^d$

"Positional encoding"

Output: $\boxed{[\text{self-attention}(\mathbf{H})]_{d+1,N+1}}$

The desired output should give the result of:

(i) performing linear regression on $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and obtain linear model $\hat{\mathbf{w}}$;

(ii) calculating the predicted value $\langle \hat{\mathbf{w}}, \mathbf{x}_{\text{query}} \rangle$.

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

A classic theoretical setup: in-context linear regression [Zhang et al., 2023, Bai et al. 2024, …]

Input matrix: $\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \dots & y_N & \boxed{0} \\ \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_N & \mathbf{p}_{\text{query}} \end{bmatrix}$

$\mathbf{x}_i, \mathbf{x}_{\text{query}} \in \mathbb{R}^d$

"Positional encoding"

Output: $\boxed{[\text{self-attention}(\mathbf{H})]_{d+1,N+1}}$

The desired output should give the result of:

In-context linear regression

(i) performing linear regression on $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and obtain linear model $\hat{\mathbf{w}}$;

(ii) calculating the predicted value $\langle \hat{\mathbf{w}}, \mathbf{x}_{\text{query}} \rangle$.

In context learning

In Context Learning (ICL). Transformers can solve tasks solely relying on task-specific prompts, without the need for fine-tuning.

A classic theoretical setup: in-context linear regression [Zhang et al., 2023, Bai et al. 2024, …]

Input matrix: $\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \dots & y_N & \boxed{0} \\ \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_N & \mathbf{p}_{\text{query}} \end{bmatrix}$

$\mathbf{x}_i, \mathbf{x}_{\text{query}} \in \mathbb{R}^d$

"Positional encoding"

Output: $\boxed{[\text{self-attention}(\mathbf{H})]_{d+1,N+1}}$

The desired output should give the result of:

In-context linear regression

(i) performing linear regression on $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and obtain linear model $\hat{\mathbf{w}}$;

(ii) calculating the predicted value $\langle \hat{\mathbf{w}}, \mathbf{x}_{\text{query}} \rangle$.

Can transformers be trained to perform one-nearest neighbor prediction?

In-context one-nearest neighbor prediction

Input matrix: $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N, \mathbf{h}_{\text{query}}] = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \ldots & y_N & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix} \in \mathbb{R}^{(d+2)\times(N+1)}$,

In-context one-nearest neighbor prediction

Input matrix: $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N, \mathbf{h}_{\text{query}}] = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \ldots & y_N & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix} \in \mathbb{R}^{(d+2)\times(N+1)},$

Response: result of one nearest neighbor prediction

$$y_{i*}, \quad i* = \arg\min_{j\in[N]} \|\mathbf{x}_{\text{query}} - \mathbf{x}_j\|_2.$$

In-context one-nearest neighbor prediction

Input matrix: $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N, \mathbf{h}_{\mathrm{query}}] = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_N & \mathbf{x}_{\mathrm{query}} \\ y_1 & y_2 & \ldots & y_N & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix} \in \mathbb{R}^{(d+2)\times(N+1)}$,

Response: result of one nearest neighbor prediction

$$y_{i*}, \quad i* = \arg\min_{j\in[N]} \|\mathbf{x}_{\mathrm{query}} - \mathbf{x}_j\|_2.$$

We suppose that the data are drawn from a distribution satisfying:

- $\mathbf{x}_i \in \mathbb{R}^d$: i.i.d. sampled from $U(\mathbb{S}^{d-1})$
- $y_i \in \{\pm 1\}$: $\mathbb{E}[y_i y_j \,|\, \mathbf{x}_{1:N}] = 0, \quad \mathbb{E}[y_i^2 \,|\, \mathbf{x}_{1:N}] = 1,$
- $\mathbb{P}(\mathbf{y}_{1:N} \,|\, \mathbf{x}_{1:N}) = \mathbb{P}(\mathbf{y}_{1:N} \,|\, -\mathbf{x}_{1:N})$

One-layer transformer model

Self-attention layer with the value matrix fixed as identity:

$$\text{self-attention}(\mathbf{H}) = \mathbf{H} \cdot \text{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{H}),$$

$$f_{\mathbf{W}}(\mathbf{H}) = [\text{self-attention}(\mathbf{H})]_{d+1,N+1}$$

One-layer transformer model

Self-attention layer with the value matrix fixed as identity:

$$\text{self-attention}(\mathbf{H}) = \mathbf{H} \cdot \text{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{H}),$$
$$f_{\mathbf{W}}(\mathbf{H}) = [\text{self-attention}(\mathbf{H})]_{d+1,N+1}$$

We consider minimizing the population square loss with gradient descent:

Loss function: $$L(\mathbf{W}) = \frac{1}{2} \cdot \mathbb{E}_{\{\mathbf{x}_i,\mathbf{y}_i\}_{i\in[N]},\mathbf{x}_{\text{query}}}\left[\left(f_{\mathbf{W}}(\mathbf{H}) - \mathbf{y}_{i^*}\right)^2\right].$$

Gradient descent: $$\mathbf{W}^{(t+1)} - \mathbf{W}^{(t)} = \eta \cdot \nabla_{\mathbf{W}} L(\mathbf{W}^{(t)}), \quad \mathbf{W}^{(0)} = \begin{pmatrix} 0_{(d+1)\times(d+1)} & 0_{d+1} \\ 0_{d+1} & -\sigma \end{pmatrix}.$$

The constant $\sigma > 0$ serves as a mask to prevent the query from attending to itself.

One-layer transformer learns 1NN in context

Theorem. Suppose that

The mask σ in the initialization satisfies $\sigma = \Omega(\mathrm{poly}(d))$,

The length of context satisfies $N = \Omega\left(\sqrt{d}\log d\right)$,

Then $L(\mathbf{W}^{(t)})$ converges to zero.

One-layer transformer learns 1NN in context

Theorem. Suppose that

The mask σ in the initialization satisfies $\sigma = \Omega(\mathrm{poly}(d))$,

The length of context satisfies $N = \Omega\left(\sqrt{d}\log d\right)$,

Then $L(\mathbf{W}^{(t)})$ converges to zero.

One-layer transformers can be trained to perform in-context one-nearest neighbor prediction.

One-layer transformer learns 1NN in context

Theorem. Suppose that

> The mask σ in the initialization satisfies $\sigma = \Omega(\mathrm{poly}(d))$,
>
> The length of context satisfies $N = \Omega\big(\sqrt{d}\log d\big)$,

Then $L(\mathbf{W}^{(t)})$ converges to zero.

One-layer transformers can be trained to perform in-context one-nearest neighbor prediction.

Note that… Predictor: $f_{\mathbf{W}}(\mathbf{H}) = [y_1, \ldots, y_N, 0] \cdot \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_{\mathrm{query}})$.

Target: $y_{i*}, \quad i* = \arg\min_{j \in [N]} \|\mathbf{x}_{\mathrm{query}} - \mathbf{x}_j\|_2$.

$$\boxed{L(\mathbf{W}) = 0 \quad \text{if and only if} \quad \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_{\mathrm{query}})) \equiv \mathbf{e}_{i*}}$$

One-layer transformer learns 1NN in context

Theorem. Suppose that

The mask σ in the initialization satisfies $\sigma = \Omega(\mathrm{poly}(d))$,

The length of context satisfies $N = \Omega\left(\sqrt{d}\log d\right)$,

Then $L(\mathbf{W}^{(t)})$ converges to zero.

One-layer transformers can be trained to perform in-context one-nearest neighbor prediction.

Note that… Predictor: $f_{\mathbf{W}}(\mathbf{H}) = [y_1, \ldots, y_N, 0] \cdot \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_{\mathrm{query}})$.

Target: $y_{i*}, \quad i* = \arg\min_{j \in [N]} \|\mathbf{x}_{\mathrm{query}} - \mathbf{x}_j\|_2.$

$$\boxed{L(\mathbf{W}) = 0 \quad \text{if and only if} \quad \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_{\mathrm{query}})) \equiv \mathbf{e}_{i*}}$$

The transformer can be trained to perform appropriate token selection!

One-layer transformer learns 1NN in context

Theorem. Suppose that

The mask σ in the initialization satisfies $\sigma = \Omega(\mathrm{poly}(d))$,

The length of context satisfies $N = \Omega\left(\sqrt{d} \log d\right)$,

Then $L(\mathbf{W}^{(t)})$ converges to zero.

One-layer transformers can be trained to perform in-context one-nearest neighbor prediction.

Note that… Predictor: $f_{\mathbf{W}}(\mathbf{H}) = [y_1, \ldots, y_N, 0] \cdot \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_{\mathrm{query}})$.

Target: $y_{i*}, \quad i* = \arg\min_{j \in [N]} \|\mathbf{x}_{\mathrm{query}} - \mathbf{x}_j\|_2$.

"Nearest neighbor selector"

$$\boxed{L(\mathbf{W}) = 0 \quad \text{if and only if} \quad \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_{\mathrm{query}})) \equiv \mathbf{e}_{i*}}$$

The transformer can be trained to perform appropriate token selection!

Prediction performance under distribution shift

We also study the performance of the transformer trained by T gradient descent iterations on new test data with distribution shift.

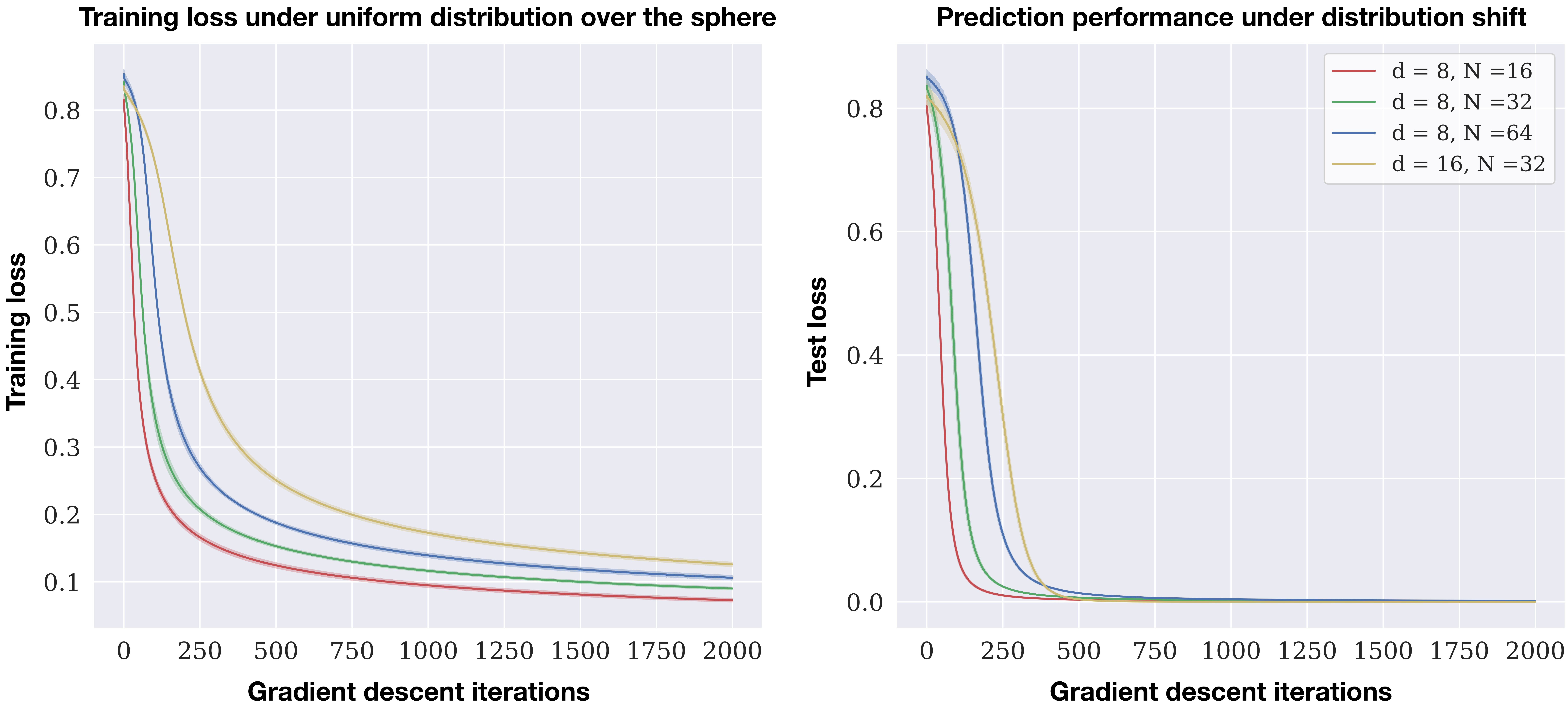
Theorem. For any data satisfying $|y_i| \leq R$, $\mathbf{x}_i \in \mathbb{S}^{d-1}$, and

$$\|\mathbf{x}_j - \mathbf{x}_{\text{query}}\|_2^2 \geq \|\mathbf{x}_{i*} - \mathbf{x}_{\text{query}}\|_2^2 + \delta \quad \text{for all } j \text{ with } y_j \neq y_{i*},$$

it holds that

$$\text{Test loss} \leq O\left(R^2 N^2 T^{-\text{poly}(N,d)\delta}\right).$$

Experiments



Training (left) and test loss (right) curves under gradient descent, with different dimensions and context sizes. The test contexts are generated with a boundary separation of $\delta = 0.1$.

Transformers Learn Optimal Variable Selection in Group-Sparse Classification

Group-sparse linear classification

Consider a classification task: $\overline{\mathbf{x}} \sim N(\mathbf{0}, \sigma_x^2 \cdot \mathbf{I}_p)$, $y = \operatorname{sign}(\langle \overline{\mathbf{x}}, \boldsymbol{\beta}^* \rangle)$.

Group-sparse linear classification

Consider a classification task: $\bar{\mathbf{x}} \sim N(\mathbf{0}, \sigma_x^2 \cdot \mathbf{I}_p)$, $y = \mathrm{sign}(\langle \bar{\mathbf{x}}, \boldsymbol{\beta}^* \rangle)$.

Suppose that index sets $G_1, \ldots, G_D$ give a predefined **partition** of $\{1, \ldots, p\}$.

Group-sparse linear classification

Consider a classification task: $\bar{\mathbf{x}} \sim N(\mathbf{0}, \sigma_x^2 \cdot \mathbf{I}_p)$, $y = \mathrm{sign}(\langle \bar{\mathbf{x}}, \boldsymbol{\beta}^* \rangle)$.

Suppose that index sets $G_1, \ldots, G_D$ give a predefined **partition** of $\{1, \ldots, p\}$.

The learning problem is "group sparse" if $\boldsymbol{\beta}^*$ satisfies that

$$\mathrm{supp}(\boldsymbol{\beta}^*) := \{k \in [p] : [\boldsymbol{\beta}^*]_k \neq 0\} \subset G_{j^*},$$

where $j^* \in [D]$ is the index of label-relevant group.

Solving group-sparse classification with transformers

Let $p = dD$ with d denoting the dimension of each group. We can then reshape the feature vector $\bar{\mathbf{x}}$ into

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D],$$

where each column $\mathbf{x}_j = [\bar{\mathbf{x}}]_{G_j} \sim \mathcal{N}(\mathbf{0}, \sigma_x^2 \mathbf{I}_d)$.

Solving group-sparse classification with transformers

Let $p = dD$ with d denoting the dimension of each group. We can then reshape the feature vector $\overline{\mathbf{x}}$ into

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D],$$

where each column $\mathbf{x}_j = [\overline{\mathbf{x}}]_{G_j} \sim \mathcal{N}(\mathbf{0}, \sigma_x^2 \mathbf{I}_d)$.

The desired output is then

$$y = \mathrm{sign}(\langle \mathbf{x}_{j^*}, \mathbf{v}^* \rangle),$$

where $\mathbf{v}^* = [\boldsymbol{\beta}^*]_{G_{j^*}} \in \mathbb{R}^d$.

Solving group-sparse classification with transformers

Let $p = dD$ with d denoting the dimension of each group. We can then reshape the feature vector $\bar{\mathbf{x}}$ into

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D],$$

where each column $\mathbf{x}_j = [\bar{\mathbf{x}}]_{G_j} \sim \mathcal{N}(\mathbf{0}, \sigma_x^2 \mathbf{I}_d)$.

The desired output is then

$$\boxed{y = \operatorname{sign}(\langle \mathbf{x}_{j^*}, \mathbf{v}^* \rangle),} \quad \textbf{Label}$$

where $\mathbf{v}^* = [\boldsymbol{\beta}^*]_{G_{j^*}} \in \mathbb{R}^d$.

Solving group-sparse classification with transformers

Let $p = dD$ with d denoting the dimension of each group. We can then reshape the feature vector $\bar{\mathbf{x}}$ into

$$\boxed{\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D],}$$

+ positional encodings

$\mathbf{p_j} \in \mathbb{R}^D$

where each column $\mathbf{x}_j = [\bar{\mathbf{x}}]_{G_j} \sim \mathcal{N}(\mathbf{0}, \sigma_x^2 \mathbf{I}_d)$.

The desired output is then

$$\boxed{y = \mathrm{sign}(\langle \mathbf{x}_{j^*}, \mathbf{v}^* \rangle),}$$

Label

where $\mathbf{v}^* = [\boldsymbol{\beta}^*]_{G_{j^*}} \in \mathbb{R}^d$.

Input matrix: $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_D] = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_D \\ \mathbf{p}_1 & \mathbf{p}_2 & \ldots & \mathbf{p}_D \end{bmatrix} \in \mathbb{R}^{(d+D)\times D}$.

One-layer transformer

Consider a scalar-output one-layer transformer model:

$$f(\mathbf{H}, \mathbf{v}, \mathbf{W}) = \sum_{j=1}^{D} \mathbf{v}^\top \mathbf{H} \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_j)$$

One-layer transformer

Consider a scalar-output one-layer transformer model:

$$f(\mathbf{H}, \mathbf{v}, \mathbf{W}) = \sum_{j=1}^{D} \mathbf{v}^\top \mathbf{H} \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_j)$$

Population cross-entropy loss:

$$L(\mathbf{v}, \mathbf{W}) = \mathbb{E}_{(\mathbf{X}, y)} \left[\ell(y \cdot f(\mathbf{H}, \mathbf{v}, \mathbf{W})) \right],$$

where $\ell(a) = \log(1 + \exp(-a))$ is the cross-entropy loss.

One-layer transformer

Consider a scalar-output one-layer transformer model:

$$f(\mathbf{H}, \mathbf{v}, \mathbf{W}) = \sum_{j=1}^{D} \mathbf{v}^\top \mathbf{H} \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_j)$$

Population cross-entropy loss:

$$L(\mathbf{v}, \mathbf{W}) = \mathbb{E}_{(\mathbf{X}, y)} \left[\ell(y \cdot f(\mathbf{H}, \mathbf{v}, \mathbf{W})) \right],$$

where $\ell(a) = \log(1 + \exp(-a))$ is the cross-entropy loss.

Gradient descent:

$$\mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} - \eta \nabla_{\mathbf{v}} L(\mathbf{v}^{(t)}, \mathbf{W}^{(t)}); \;\; \mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla_{\mathbf{W}} L(\mathbf{v}^{(t)}, \mathbf{W}^{(t)}),$$

with zero initialization: $\mathbf{v}^{(0)} = \mathbf{0}_{d+D}$, $\mathbf{W}^{(0)} = \mathbf{0}_{(d+D)\times(d+D)}$.

Transformers can solve group-sparse linear classification

Theorem. For any $\epsilon > 0$, suppose that $D = \omega(\log^2(1/\epsilon))$, $d \leq O(\mathrm{poly}(D))$, $\sigma_x, \eta = \Theta(1)$. Then there exists

$$T^* = \Theta\Big(D^3 \vee \frac{1}{D^3\epsilon^3}\Big),$$

such that the following conclusions hold:

Transformers can solve group-sparse linear classification

Theorem. For any $\epsilon > 0$, suppose that $D = \omega(\log^2(1/\epsilon))$, $d \leq O(\mathrm{poly}(D))$, $\sigma_x, \eta = \Theta(1)$. Then there exists

$$T^* = \Theta\big(D^3 \vee \frac{1}{D^3\epsilon^3}\big),$$

such that the following conclusions hold:

- Self-attention extracts the variables from the label-relevant group: w.h.p.,

$$\mathbf{S}_{j^*,j}^{(T^*)} \geq 1 - \exp(-\Theta(D)), \ \forall j \in [D]\,.$$

Transformers can solve group-sparse linear classification

Theorem. For any $\epsilon > 0$, suppose that $D = \omega(\log^2(1/\epsilon))$, $d \leq O(\text{poly}(D))$, $\sigma_x, \eta = \Theta(1)$. Then there exists

$$T^* = \Theta\big(D^3 \vee \frac{1}{D^3\epsilon^3}\big),$$

such that the following conclusions hold:

- Self-attention extracts the variables from the label-relevant group: w.h.p.,

$$\boxed{\mathbf{S}_{j^*,j}^{(T^*)} \geq 1 - \exp(-\Theta(D)),\ \forall j \in [D]\ .}$$ Variable selection

Transformers can solve group-sparse linear classification

Theorem. For any $\epsilon > 0$, suppose that $D = \omega(\log^2(1/\epsilon))$, $d \leq O(\mathrm{poly}(D))$, $\sigma_x, \eta = \Theta(1)$. Then there exists

$$T^* = \Theta\Big(D^3 \vee \frac{1}{D^3\epsilon^3}\Big),$$

such that the following conclusions hold:

- Self-attention extracts the variables from the label-relevant group: w.h.p.,

$$\boxed{\mathbf{S}^{(T^*)}_{j^*,j} \geq 1 - \exp(-\Theta(D)),\ \forall j \in [D]\,.}$$ Variable selection

- The value vector $\mathbf{v}$ successfully learns the ground truth classifier:

$$\mathbf{v}^{(T^*)} = [\mathbf{v}_1^{(T^*)\top}, \mathbf{0}_D^\top]^\top, \text{ and } \Big\|\mathrm{normalized}(\mathbf{v}_1^{(T^*)}) - \mathbf{v}^*\Big\|_2 \leq \epsilon D \exp(-\Theta(\sqrt{D}))\,.$$

Transformers can solve group-sparse linear classification

Theorem. For any $\epsilon > 0$, suppose that $D = \omega(\log^2(1/\epsilon))$, $d \leq O(\text{poly}(D))$, $\sigma_x, \eta = \Theta(1)$. Then there exists

$$T^* = \Theta\big(D^3 \vee \frac{1}{D^3 \epsilon^3}\big),$$

such that the following conclusions hold:

- Self-attention extracts the variables from the label-relevant group: w.h.p.,

$$\boxed{\mathbf{S}^{(T^*)}_{j^*,j} \geq 1 - \exp(-\Theta(D)), \ \forall j \in [D] \,.}$$

Variable selection

- The value vector $\mathbf{v}$ successfully learns the ground truth classifier:

$$\boxed{\mathbf{v}^{(T^*)} = [\mathbf{v}_1^{(T^*)\top}, \mathbf{0}_D^\top]^\top, \text{ and } \left\| \text{normalized}(\mathbf{v}_1^{(T^*)}) - \mathbf{v}^* \right\|_2 \leq \epsilon D \exp(-\Theta(\sqrt{D})) \,.}$$

Optimal linear classification on selected variables

Transformers can solve group-sparse linear classification

Theorem. For any $\epsilon > 0$, suppose that $D = \omega(\log^2(1/\epsilon))$, $d \le O(\text{poly}(D))$, $\sigma_x, \eta = \Theta(1)$. Then there exists

$$T^* = \Theta\Big(D^3 \vee \frac{1}{D^3\epsilon^3}\Big),$$

such that the following conclusions hold:

- Self-attention extracts the variables from the label-relevant group: w.h.p.,

$$\boxed{\mathbf{S}_{j^*,j}^{(T^*)} \ge 1 - \exp(-\Theta(D)),\ \forall j \in [D]\,.}$$

Variable selection

- The value vector $\mathbf{v}$ successfully learns the ground truth classifier:

$$\boxed{\mathbf{v}^{(T^*)} = [\mathbf{v}_1^{(T^*)\top}, \mathbf{0}_D^\top]^\top, \text{ and } \Big\|\text{normalized}(\mathbf{v}_1^{(T^*)}) - \mathbf{v}^*\Big\|_2 \le \epsilon D \exp(-\Theta(\sqrt{D}))\,.}$$

- The loss is sufficiently minimized:

Optimal linear classification on selected variables

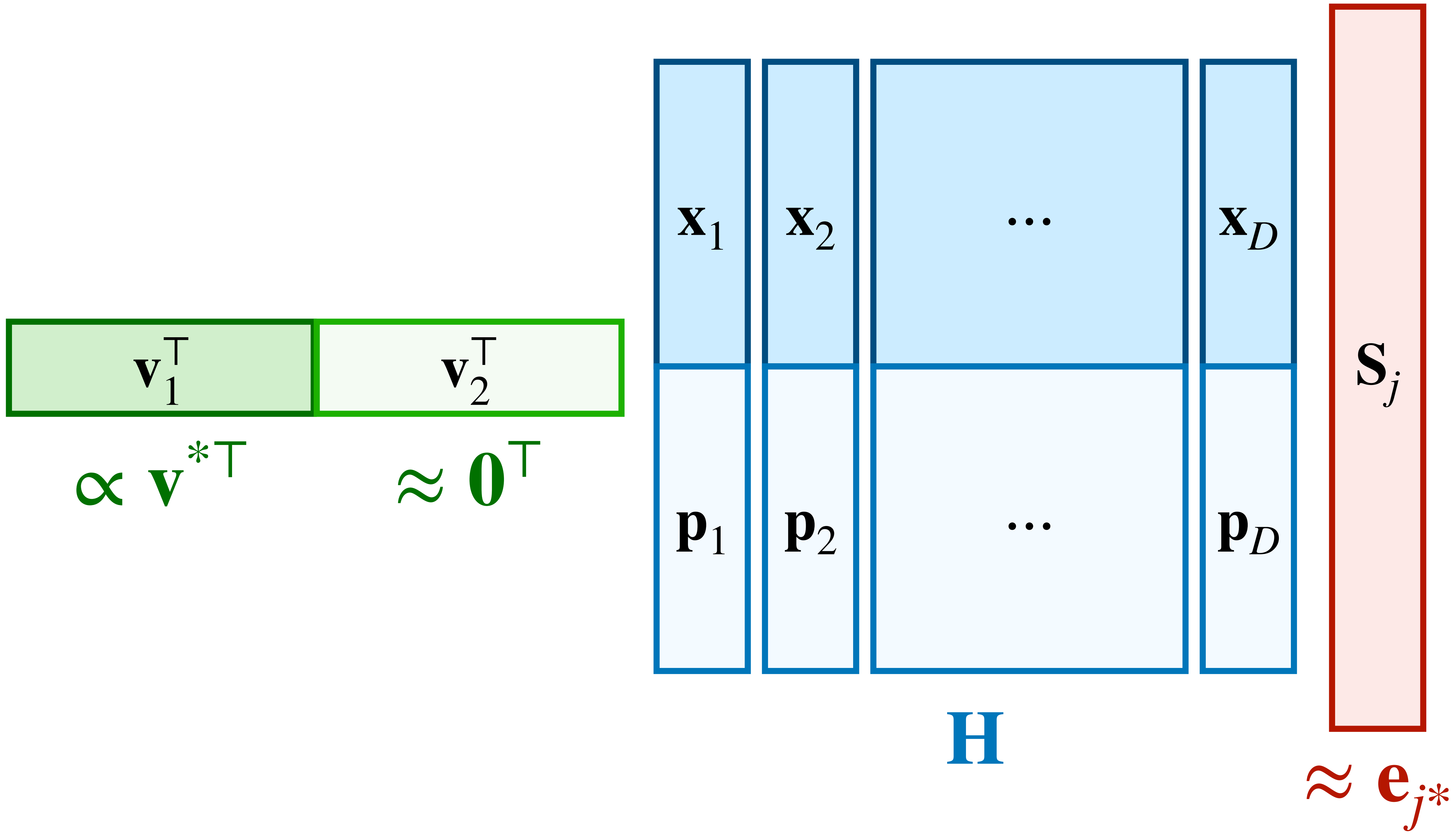
$$L(\mathbf{v}^{(T^*)}, \mathbf{W}^{(T^*)}) = \Theta(\epsilon \wedge D^{-2})\,.$$

Classification with variable selection

Recall the prediction model: $$f(\mathbf{H}, \mathbf{v}, \mathbf{W}) = \sum_{j=1}^{D} \mathbf{v}^\top \mathbf{H} \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_j)$$

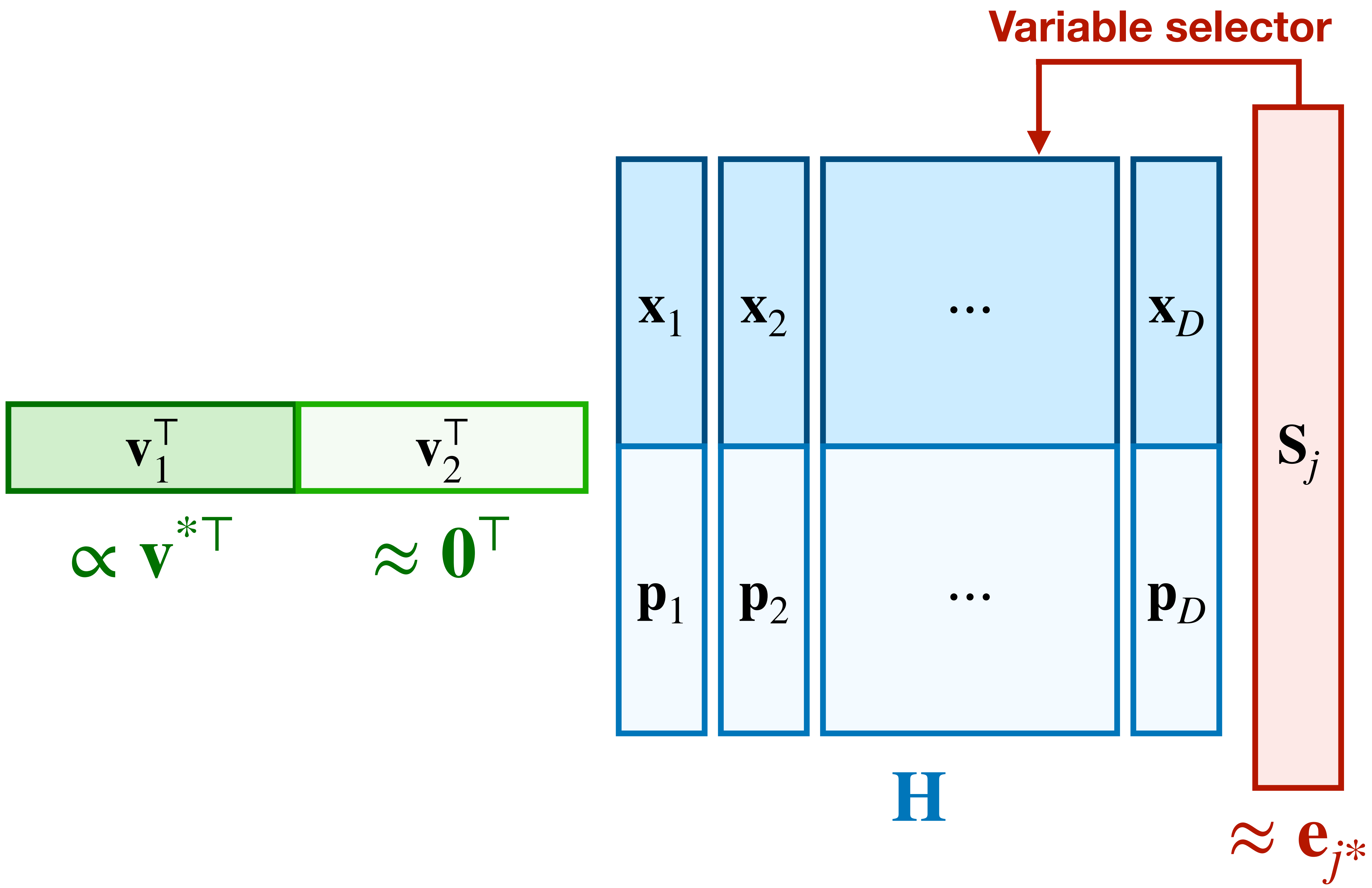
Classification with variable selection

Recall the prediction model: $$f(\mathbf{H}, \mathbf{v}, \mathbf{W}) = \sum_{j=1}^{D} \mathbf{v}^\top \mathbf{H} \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_j)$$



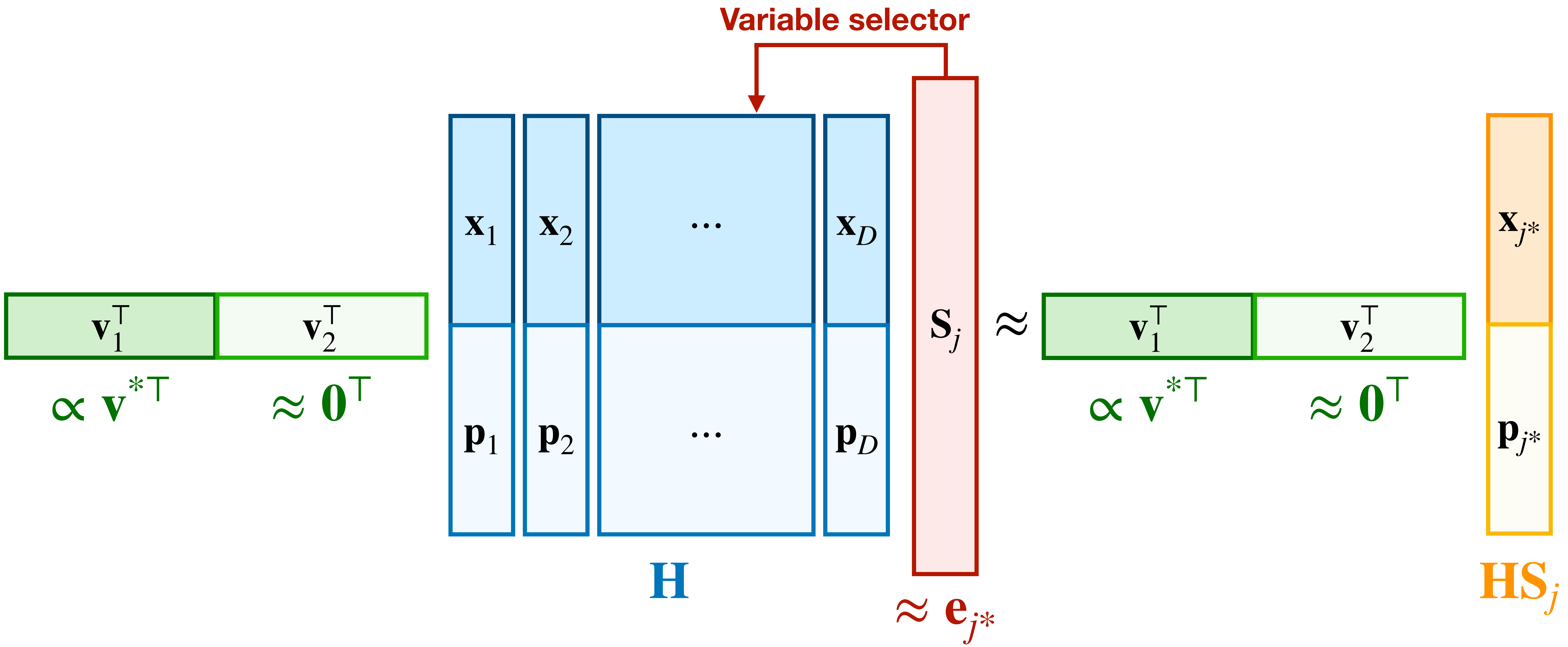
Classification with variable selection

Recall the prediction model: $$f(\mathbf{H}, \mathbf{v}, \mathbf{W}) = \sum_{j=1}^{D} \mathbf{v}^\top \mathbf{H}\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}\mathbf{h}_j)$$



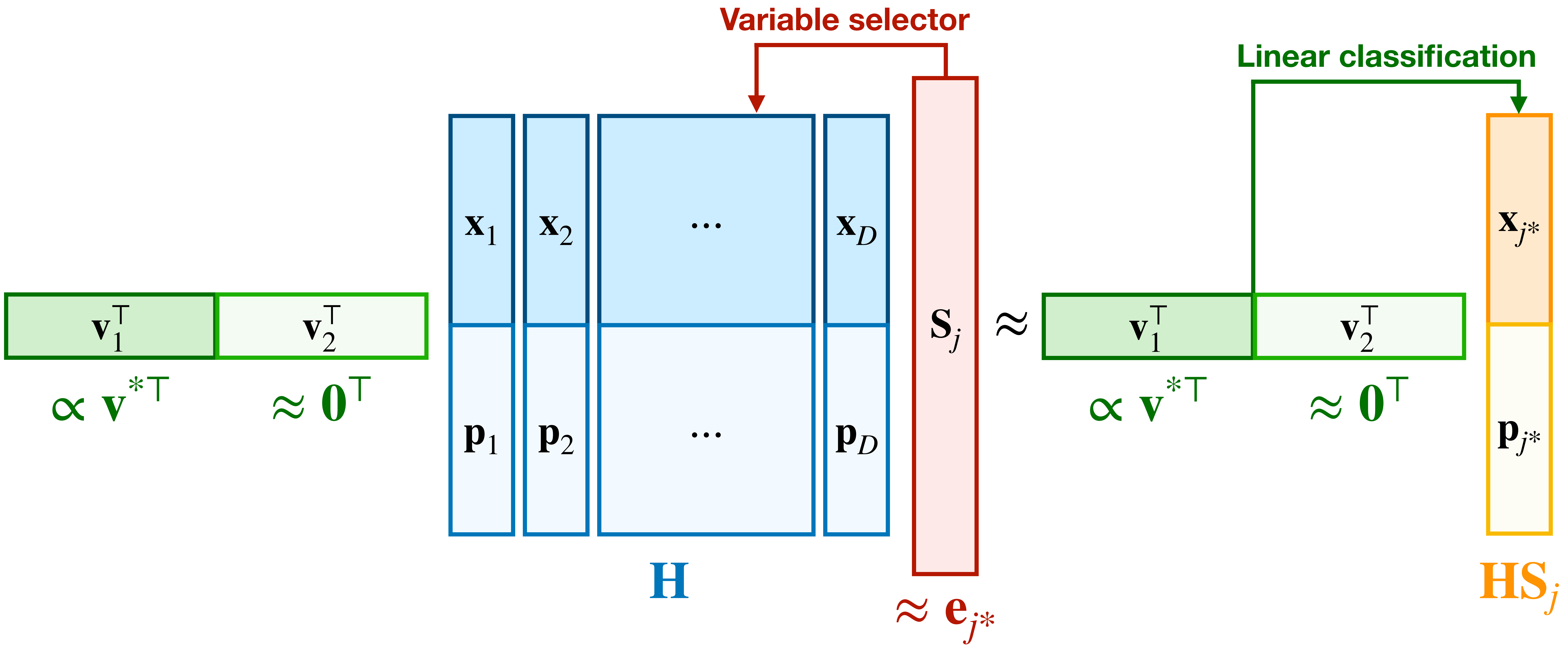
Classification with variable selection

Recall the prediction model: $f(\mathbf{H}, \mathbf{v}, \mathbf{W}) = \sum_{j=1}^{D} \mathbf{v}^\top \mathbf{H}\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}\mathbf{h}_j)$



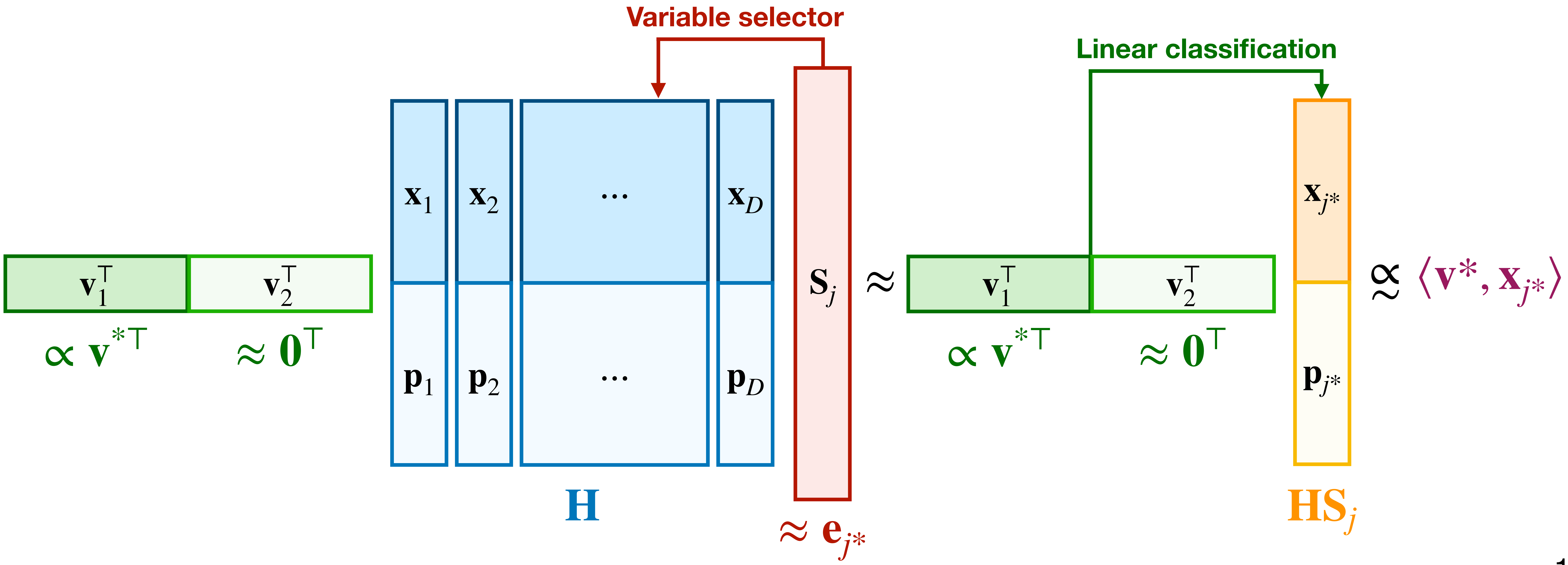
Classification with variable selection

Recall the prediction model: $f(\mathbf{H}, \mathbf{v}, \mathbf{W}) = \sum_{j=1}^{D} \mathbf{v}^\top \mathbf{H} \mathrm{softmax}(\mathbf{H}^\top \mathbf{W} \mathbf{h}_j)$

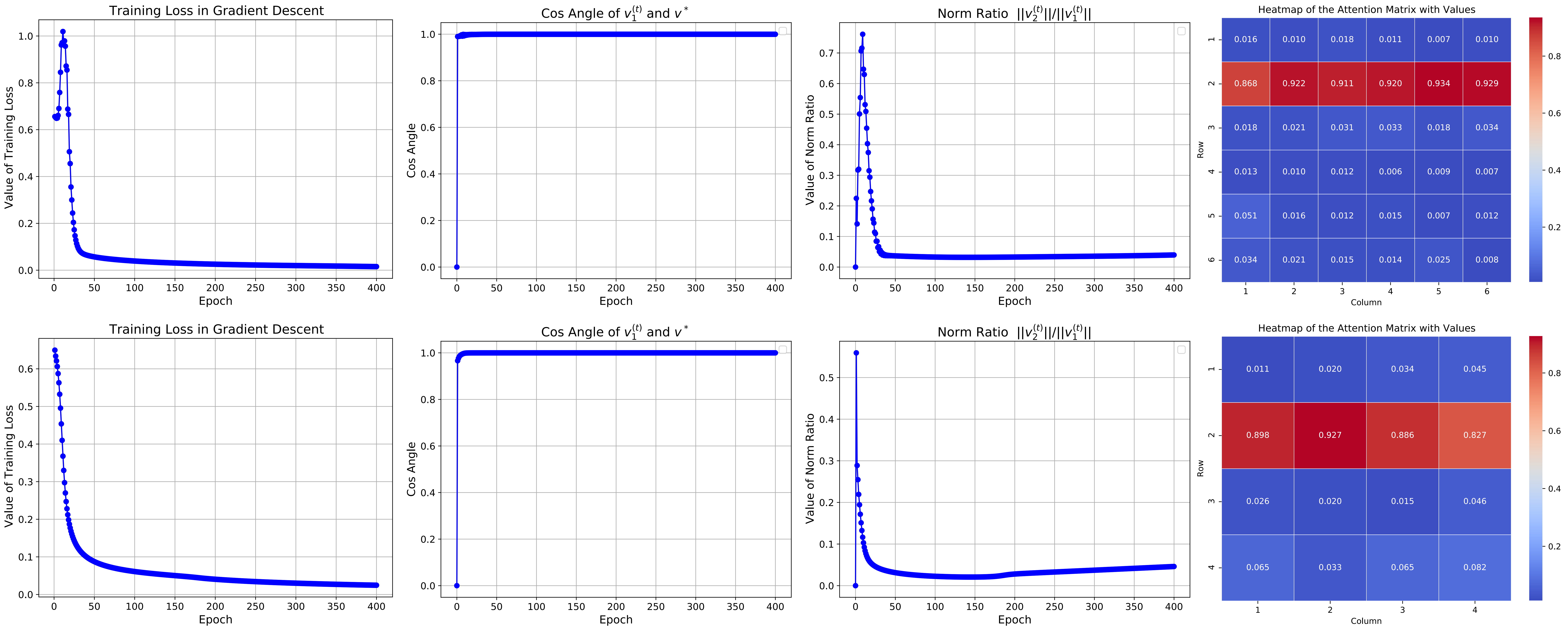


Classification with variable selection

Recall the prediction model: $$f(\mathbf{H}, \mathbf{v}, \mathbf{W}) = \sum_{j=1}^{D} \mathbf{v}^\top \mathbf{H}\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}\mathbf{h}_j)$$



Experiments - pretraining

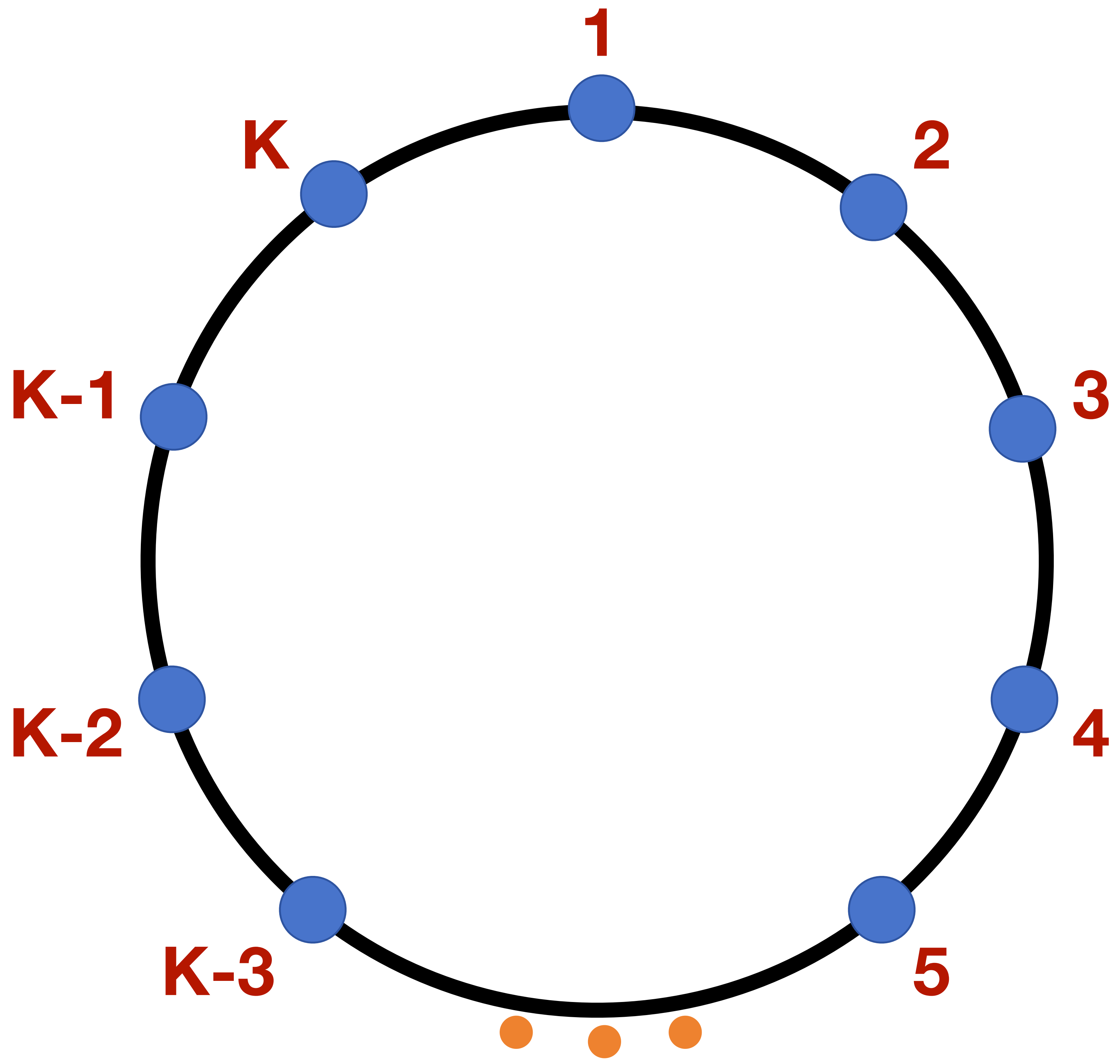


Training loss, cosine similarity, norm ratio, and attention score for $(n, d, D) = (500,4,6)$ and $(n, d, D) = (200,2,4)$ respectively when set $j^* = 2$.

Transformers Learn Random Walk Prediction by Attending to the Direct Parent State

A simple random walk prediction task

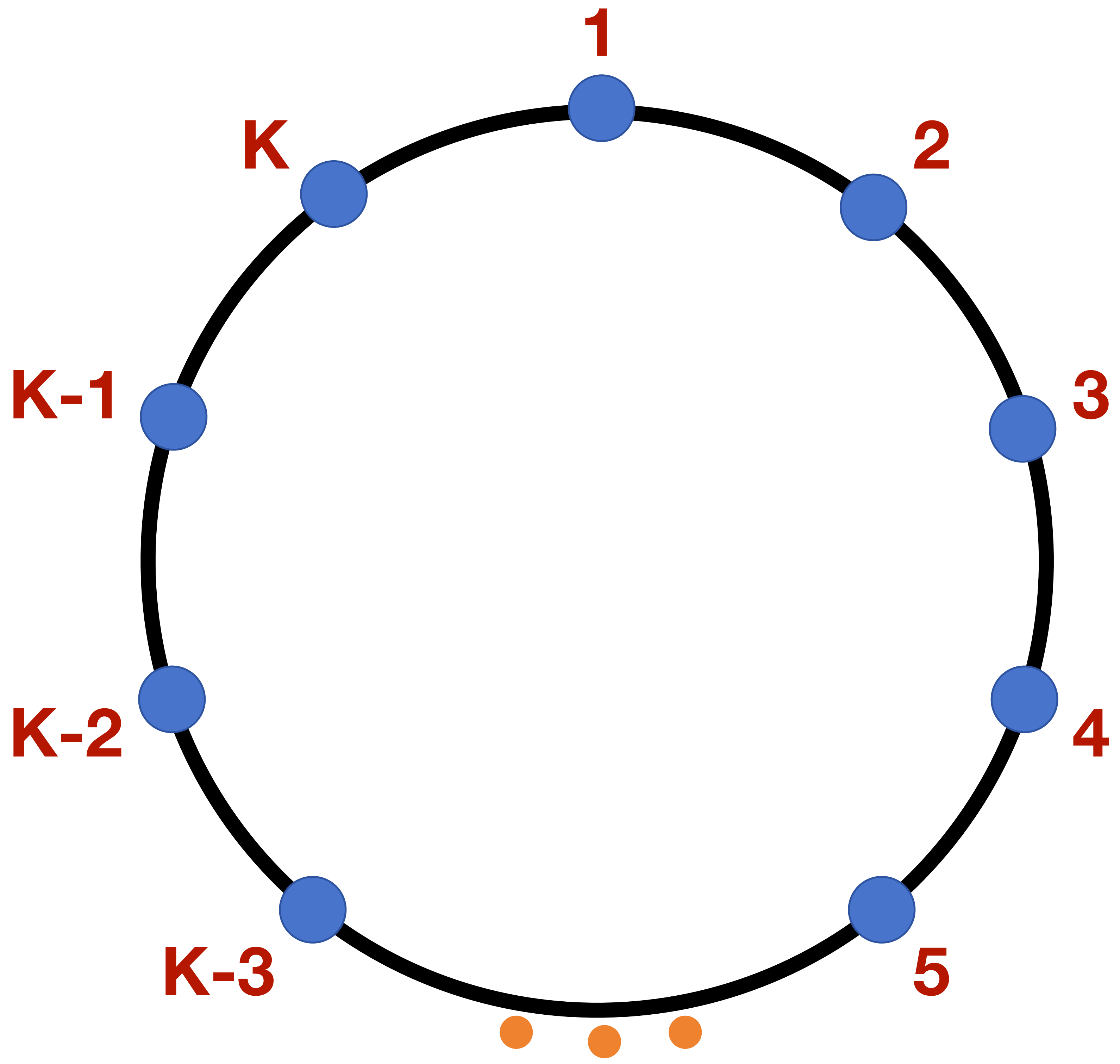
Consider a circle with K nodes.



A simple random walk prediction task

Consider a circle with K nodes.

A walk on the circle: the process where a ‘walker’ moves step-by-step among the nodes of the circle.

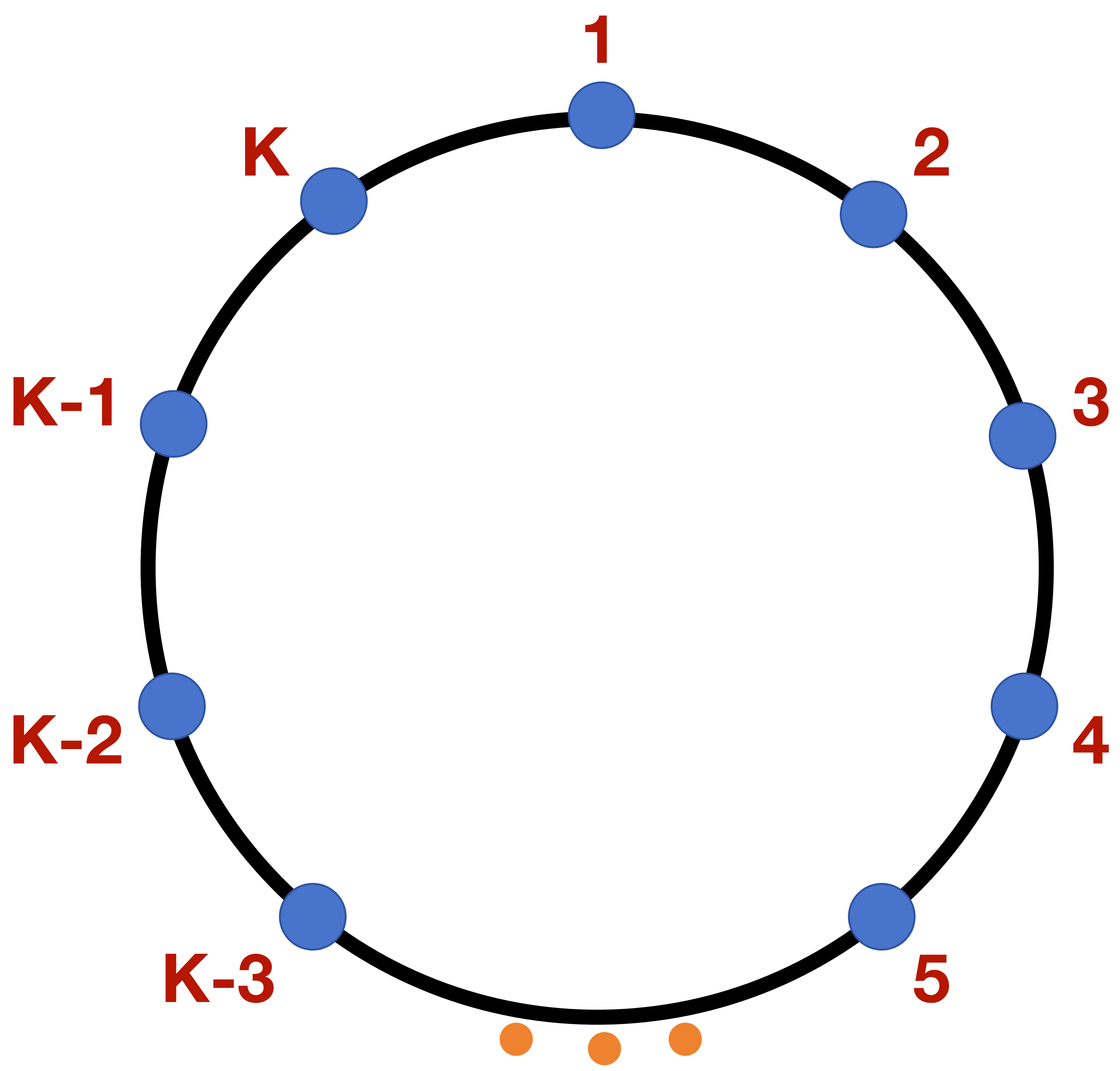


A simple random walk prediction task

Consider a circle with K nodes.

A walk on the circle: the process where a ‘walker’ moves step-by-step among the nodes of the circle.

State $s_i \in [K]$: the location of the walker at the i-th step ($i \in [N]$).



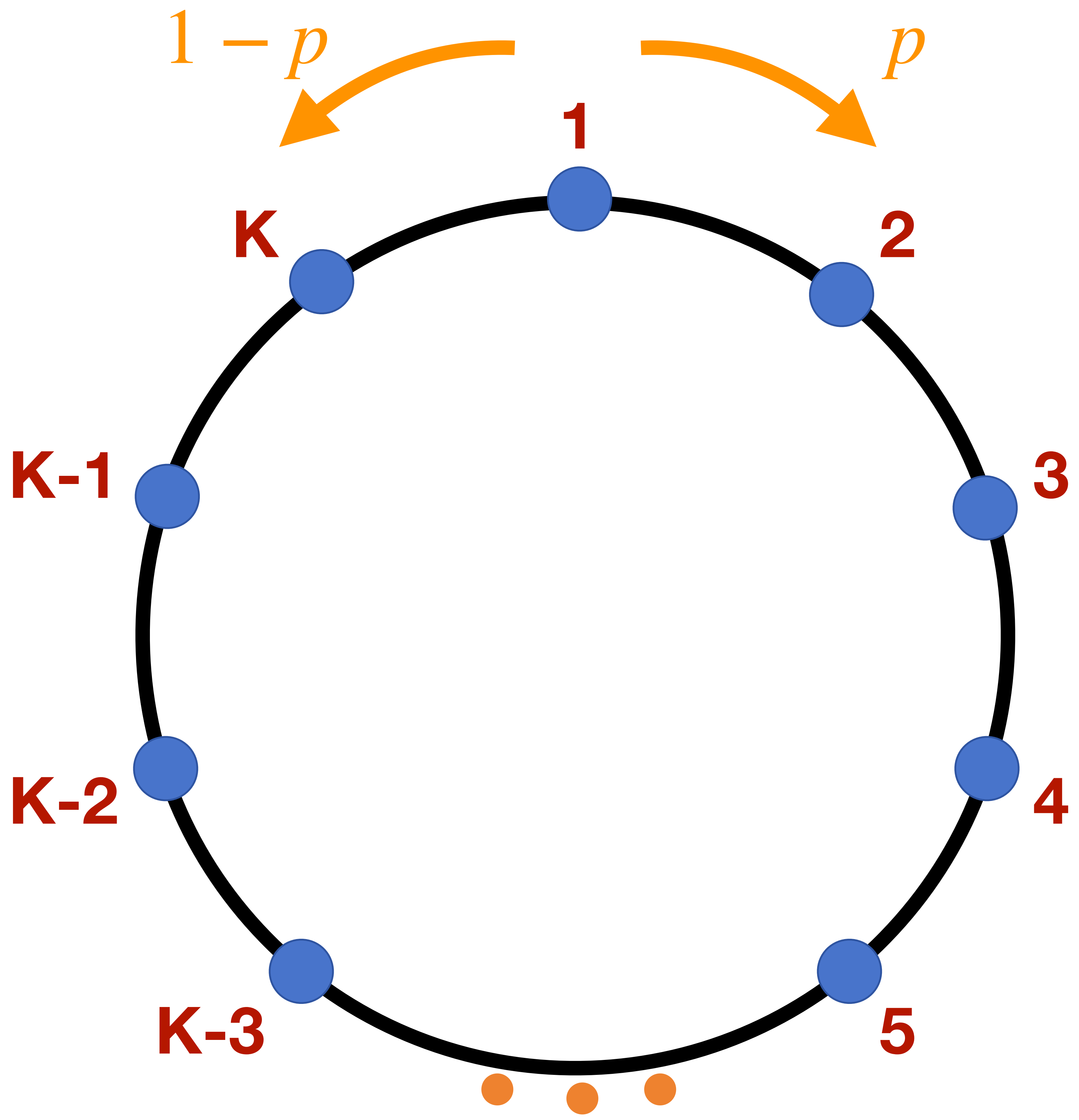
A simple random walk prediction task

Consider a circle with K nodes.

A walk on the circle: the process where a 'walker' moves step-by-step among the nodes of the circle.

State $s_i \in [K]$: the location of the walker at the i-th step ($i \in [N]$).

Suppose that $s_1 \in [K]$ is uniformly chosen. At each step, the walker goes **clockwise w.p. p** or **counter-clockwise w.p. $1 - p$**.



A simple random walk prediction task

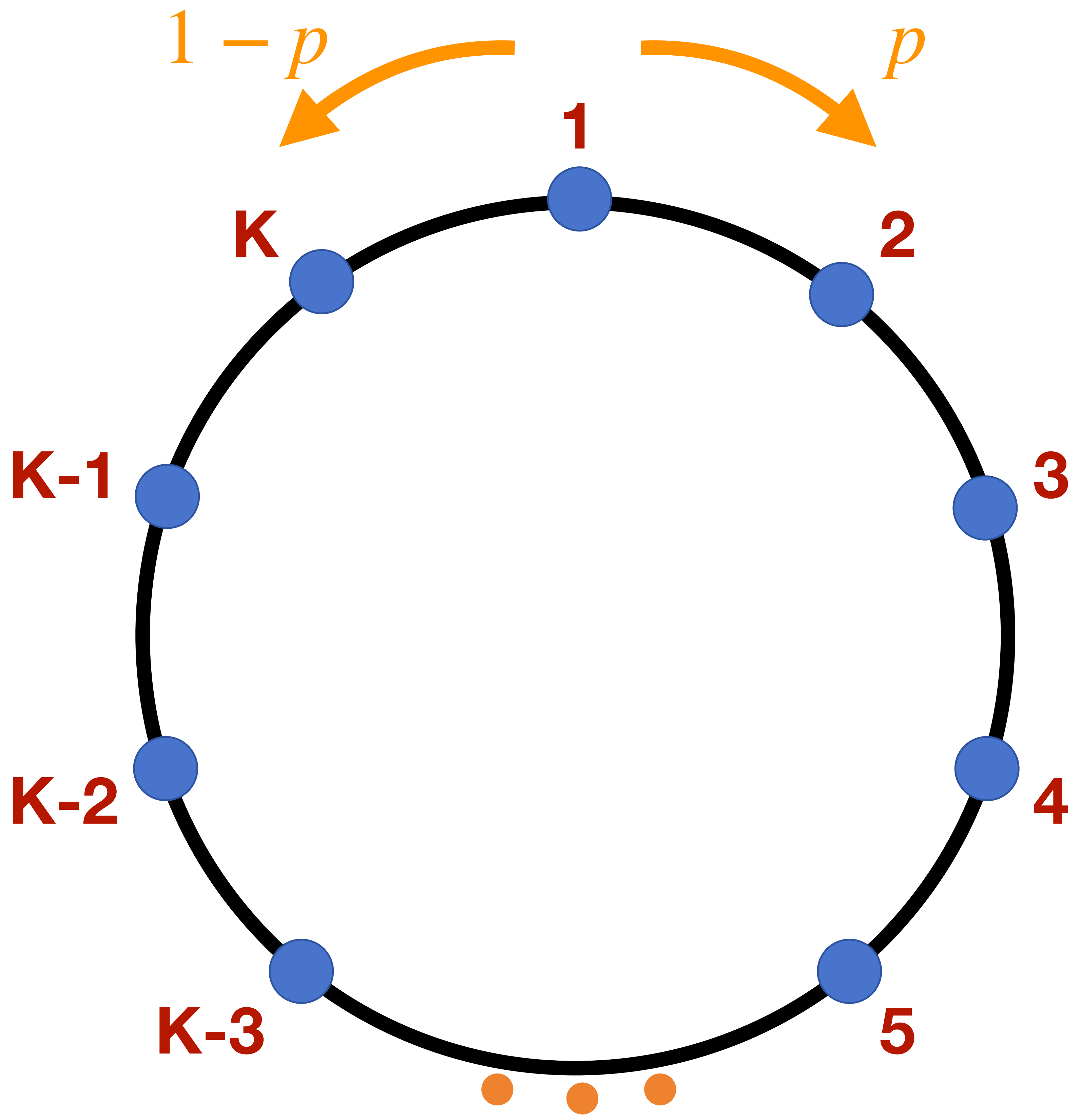
Consider a circle with K nodes.

A walk on the circle: the process where a ‘walker’ moves step-by-step among the nodes of the circle.

State $s_i \in [K]$: the location of the walker at the i-th step ($i \in [N]$).

Suppose that $s_1 \in [K]$ is uniformly chosen. At each step, the walker goes **clockwise w.p. p** or **counter-clockwise w.p. $1-p$**.

Goal: predict the location of the next step s_N based on the historical locations $s_1, \ldots, s_{N-1}$.

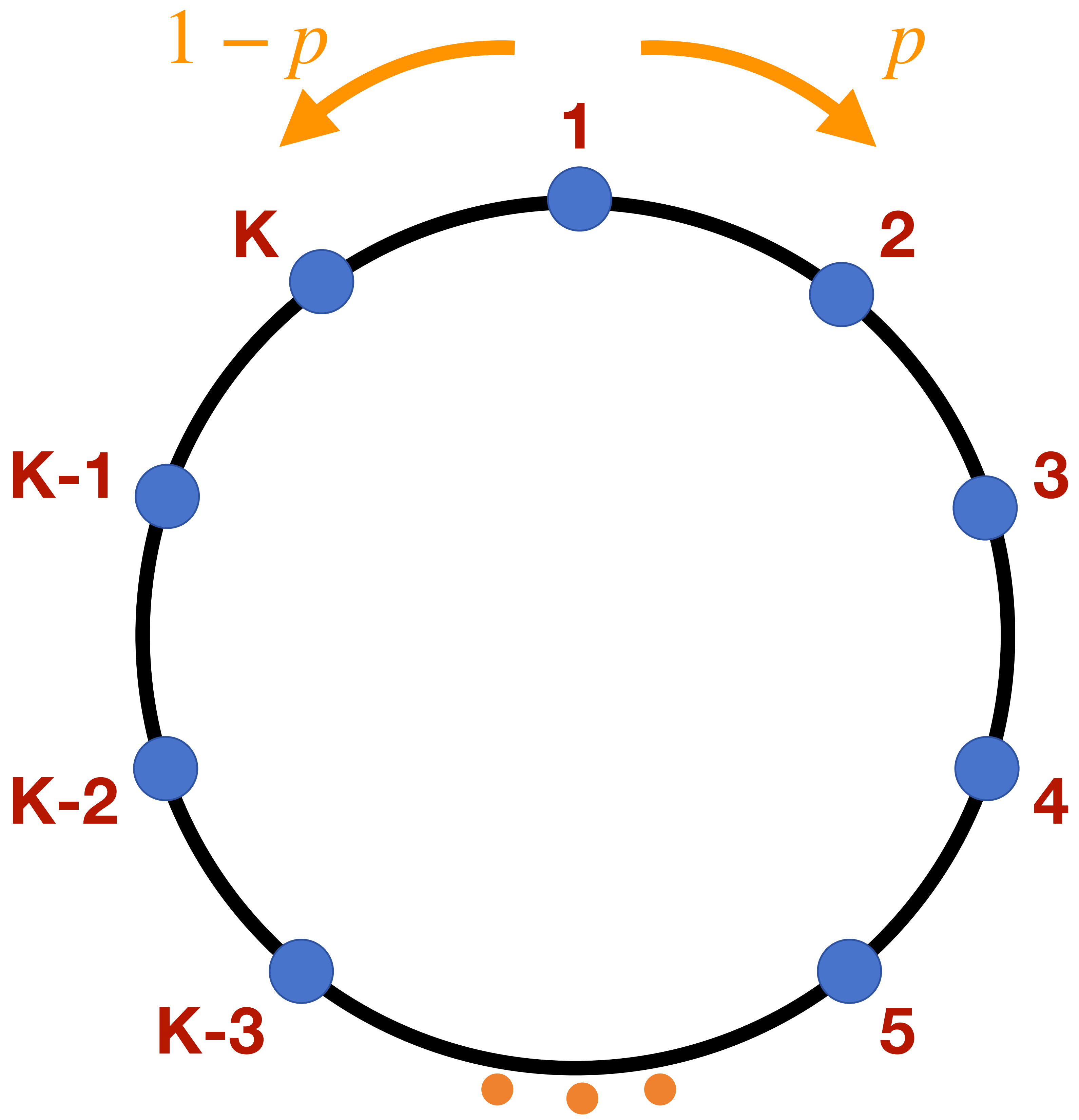


A simple random walk prediction task

Goal: predict the location of the next step s_N based on the historical locations $s_1, \ldots, s_{N-1}$.

For $i \in [N-1]$, denote by $\mathbf{x}_i \in \mathbb{R}^K$ the one-hot encoding of $s_i \in [K]$. Then

$$\mathbb{P}(\mathbf{x}_i \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}) = \mathbb{P}(\mathbf{x}_i \,|\, \mathbf{x}_{i-1}) = \mathbf{\Pi}^{*\top} \mathbf{x}_{i-1},$$

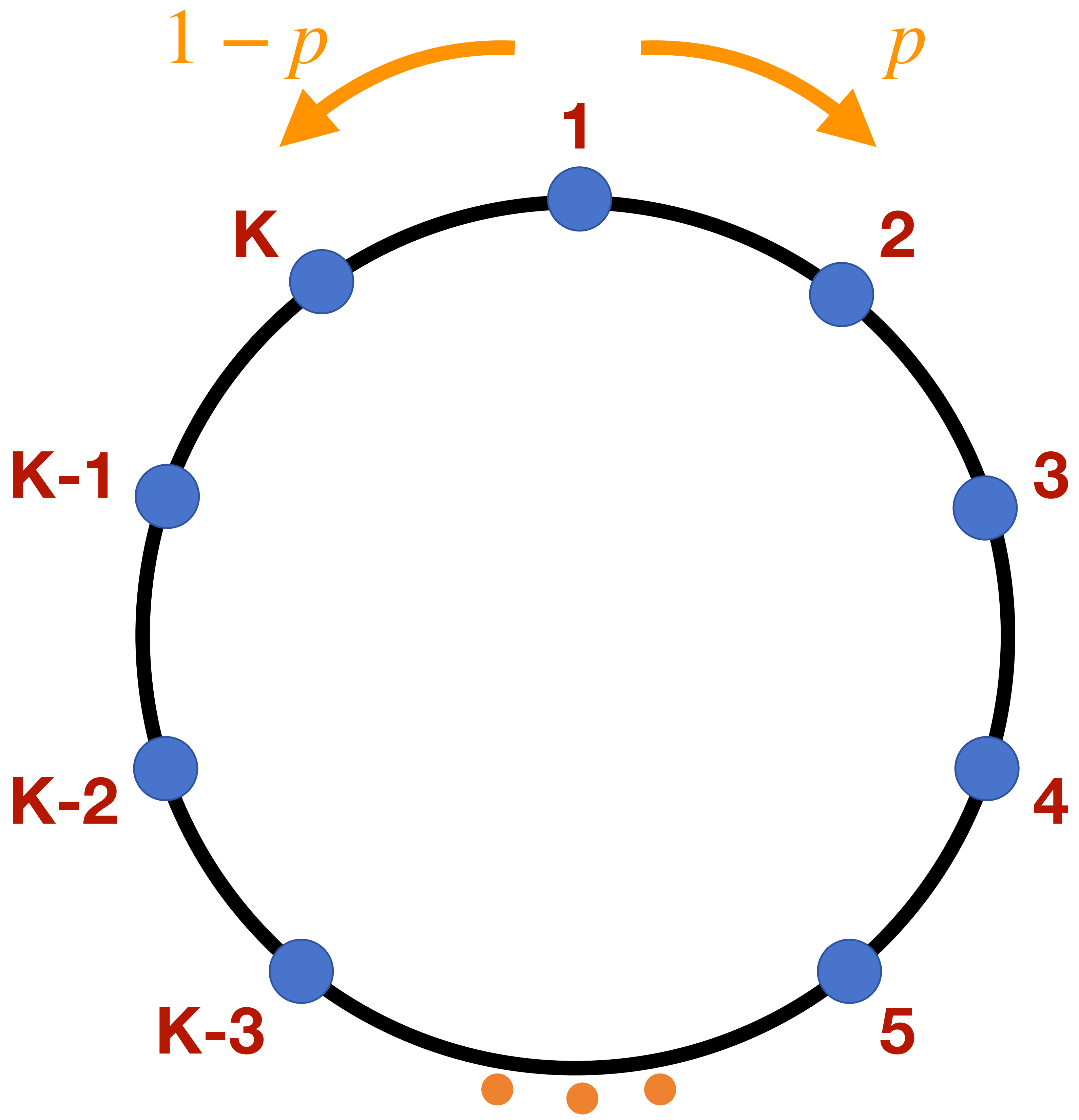


A simple random walk prediction task

Goal: predict the location of the next step s_N based on the historical locations $s_1, \ldots, s_{N-1}$.

For $i \in [N-1]$, denote by $\mathbf{x}_i \in \mathbb{R}^K$ the one-hot encoding of $s_i \in [K]$. Then

$$\underbrace{\boxed{\mathbb{P}(\mathbf{x}_i \mid \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}) = \mathbb{P}(\mathbf{x}_i \mid \mathbf{x}_{i-1})}}_{\text{Markov property}} = \mathbf{\Pi}^{*\top} \mathbf{x}_{i-1},$$



A simple random walk prediction task

Goal: predict the location of the next step s_N based on the historical locations $s_1, \ldots, s_{N-1}$.

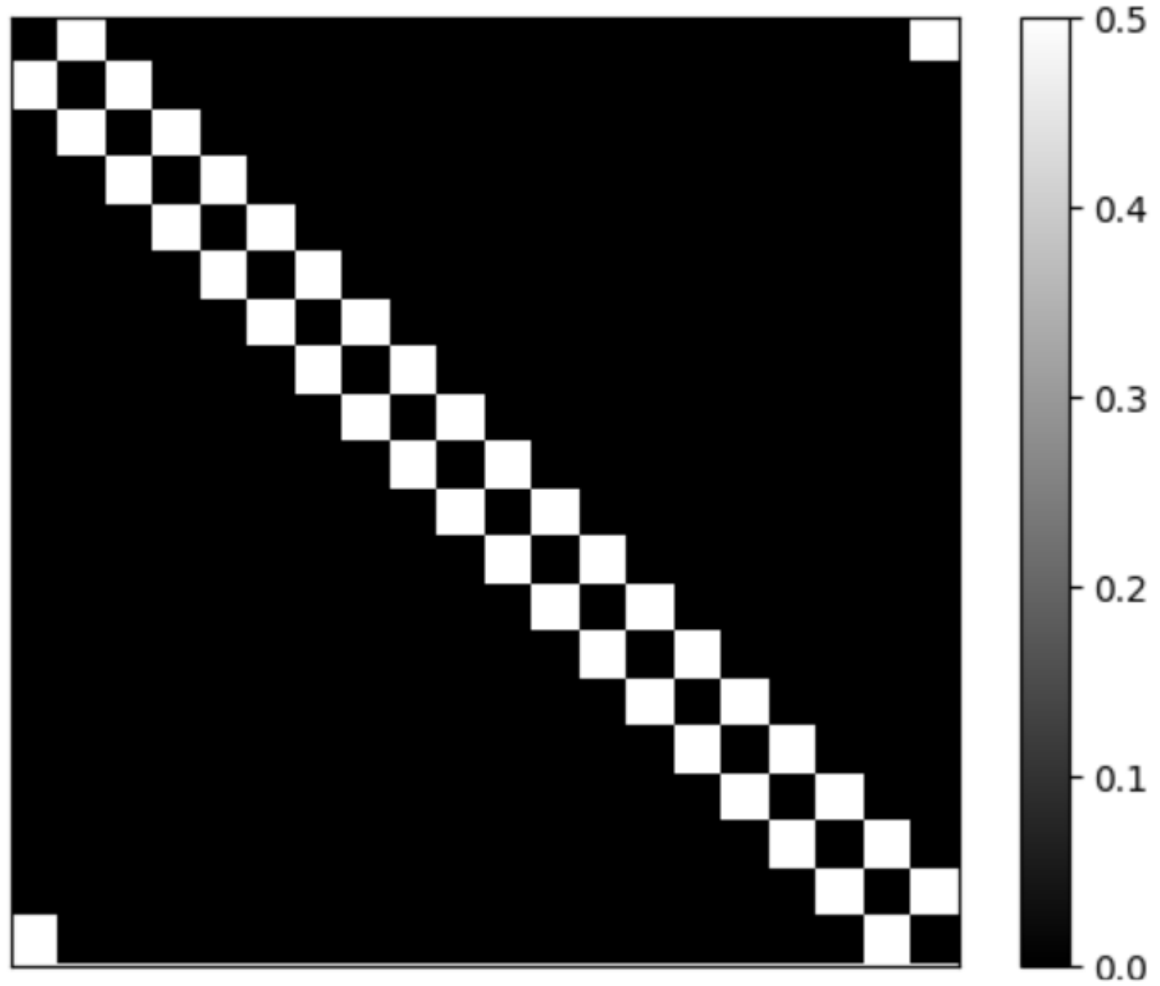
For $i \in [N-1]$, denote by $\mathbf{x}_i \in \mathbb{R}^K$ the one-hot encoding of $s_i \in [K]$. Then

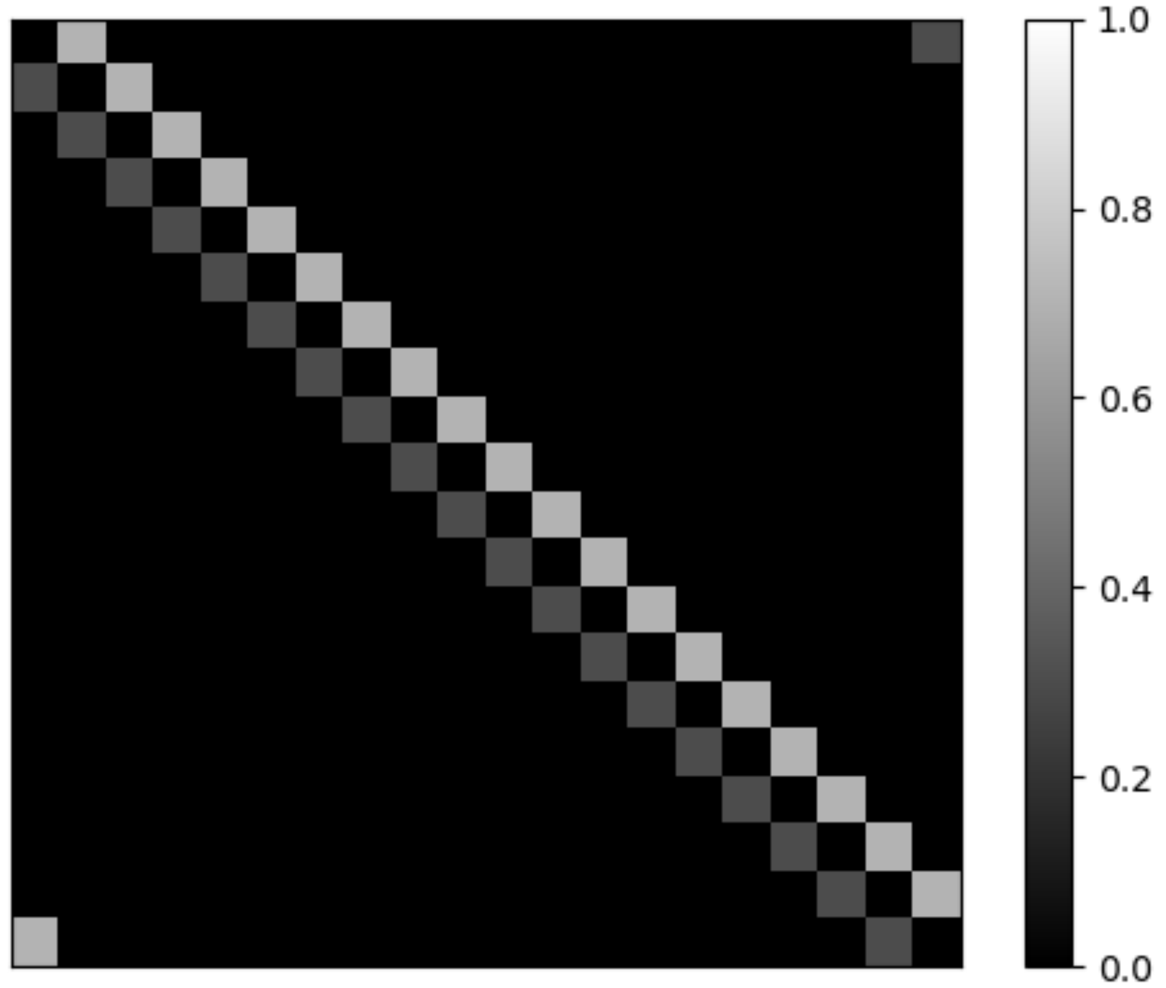
$$\boxed{\mathbb{P}(\mathbf{x}_i \mid \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}) = \mathbb{P}(\mathbf{x}_i \mid \mathbf{x}_{i-1})} = \mathbf{\Pi}^{*\top}\mathbf{x}_{i-1},$$

Markov property

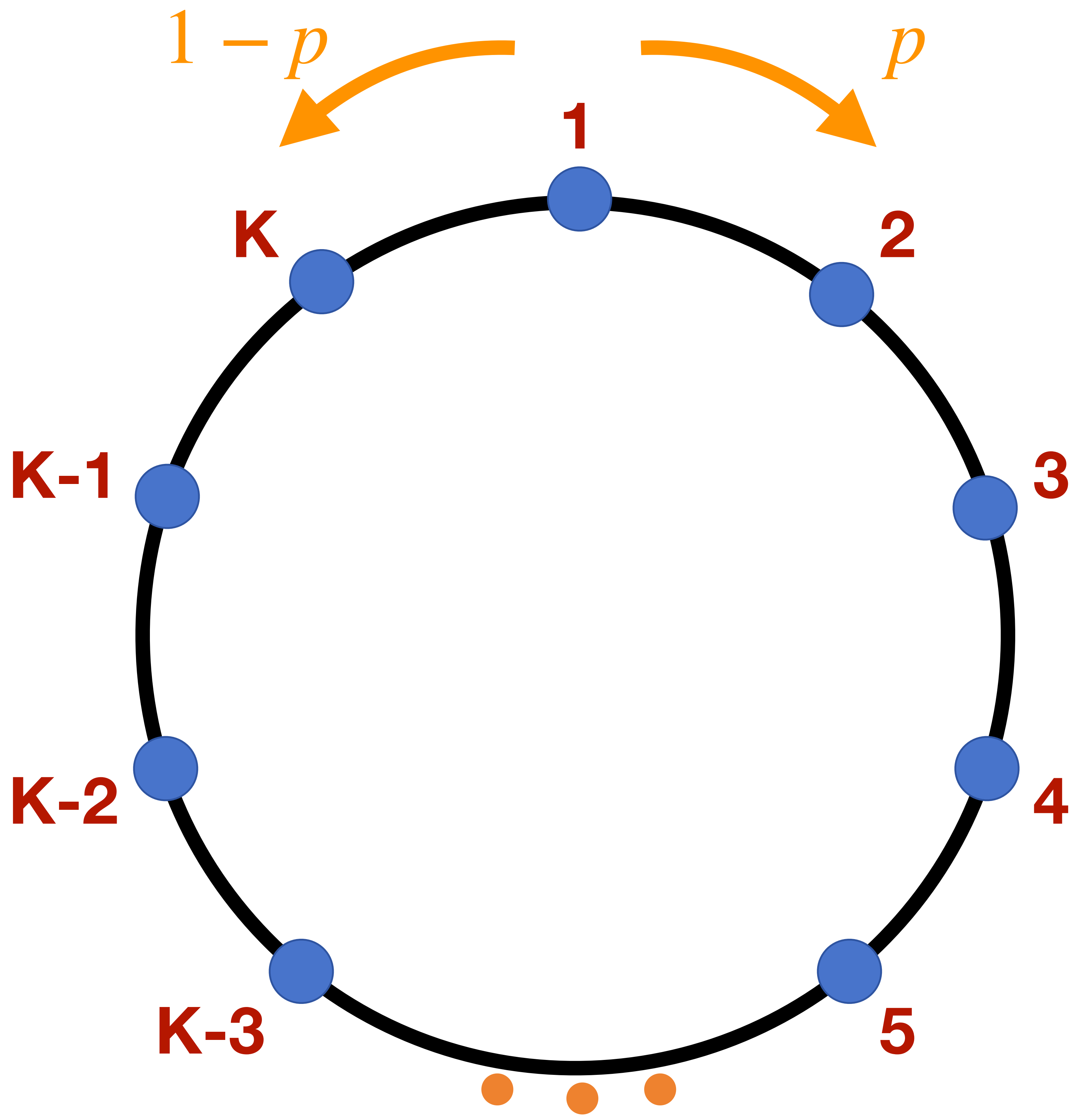
where $\mathbf{\Pi}^*$ is the ground-truth transition matrix:

$p = 0.5$





$p = 0.7$



Random walk prediction with transformers

Input matrix: $\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-1} & \mathbf{0} \\ \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_{N-1} & \mathbf{p}_N \end{bmatrix} \in \mathbb{R}^{(K+M)\times N}$

Random walk prediction with transformers

Input matrix: $\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-1} & \mathbf{0} \\ \boxed{\mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_{N-1} & \mathbf{p}_N} \end{bmatrix} \in \mathbb{R}^{(K+M)\times N}$

Mutually orthogonal positional encodings

Random walk prediction with transformers

Input matrix: $\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-1} & \mathbf{0} \\ \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_{N-1} & \mathbf{p}_N \end{bmatrix} \in \mathbb{R}^{(K+M)\times N}$

Mutually orthogonal positional encodings

Label: $y = s_N \sim \boldsymbol{\Pi}^{*\top}\mathbf{x}_{i-1}$

Random walk prediction with transformers

Input matrix: $\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-1} & \mathbf{0} \\ \boxed{\mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_{N-1} & \mathbf{p}_N} \end{bmatrix} \in \mathbb{R}^{(K+M)\times N}$

Mutually orthogonal positional encodings

Label: $y = s_N \sim \mathbf{\Pi}^{*\top}\mathbf{x}_{i-1}$

Transformer model: $\mathbf{f}(\mathbf{H}, \mathbf{V}, \mathbf{W}) = \mathbf{V}\mathbf{X}\text{softmax}(\mathbf{H}^\top \mathbf{W}\mathbf{h}_N)$

Random walk prediction with transformers

Input matrix: $\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-1} & \mathbf{0} \\ \boxed{\mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_{N-1} & \mathbf{p}_N} \end{bmatrix} \in \mathbb{R}^{(K+M)\times N}$

Mutually orthogonal positional encodings

Label: $y = s_N \sim \mathbf{\Pi}^{*\top}\mathbf{x}_{i-1}$

Transformer model: $\mathbf{f}(\mathbf{H}, \mathbf{V}, \mathbf{W}) = \mathbf{V}\boxed{\mathbf{X}}\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}\mathbf{h}_N)$

We do not include the positional encodings here.

Random walk prediction with transformers

Input matrix: $\mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-1} & \mathbf{0} \\ \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_{N-1} & \mathbf{p}_N \end{bmatrix} \in \mathbb{R}^{(K+M)\times N}$

Mutually orthogonal positional encodings

Label: $y = s_N \sim \mathbf{\Pi}^{*\top}\mathbf{x}_{i-1}$

Transformer model: $\mathbf{f}(\mathbf{H}, \mathbf{V}, \mathbf{W}) = \mathbf{V}\boxed{\mathbf{X}}\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}\mathbf{h}_N)$

We do not include the positional encodings here.

Population log loss: $L(\mathbf{V}, \mathbf{W}) = \mathbb{E}_{(\mathbf{X},y)} \log[\mathbf{e}_y^\top \mathbf{f}(\mathbf{H}, \mathbf{V}, \mathbf{W}) + \epsilon]$

Gradient descent:

$$\mathbf{V}^{(t+1)} = \mathbf{V}^{(t)} - \eta \nabla_{\mathbf{V}} L(\mathbf{V}^{(t)}, \mathbf{W}^{(t)});\ \ \mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla_{\mathbf{W}} L(\mathbf{V}^{(t)}, \mathbf{W}^{(t)}),$$

with zero initialization: $\mathbf{v}^{(0)} = \mathbf{0}_{K\times K}$, $\mathbf{W}^{(0)} = \mathbf{0}_{(K+M)\times(K+M)}$.

Transformers learn random walk prediction

Theorem. Suppose that $0 < p < 1$, and $\eta, \epsilon = \Theta(1)$. Under certain conditions, there exists $T_0 = \Theta(1)$, such that for any polynomial iteration number $T \geq T_0$, the following results hold:

Transformers learn random walk prediction

Theorem. Suppose that $0 < p < 1$, and $\eta, \epsilon = \Theta(1)$. Under certain conditions, there exists $T_0 = \Theta(1)$, such that for any polynomial iteration number $T \geq T_0$, the following results hold:

- Softmax attention selects the "direct parent" token:

$$\left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_{N-1} \geq 1 - \exp(-\Omega(N)), \quad \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_j \leq \exp(-\Omega(N)) \,.$$

Transformers learn random walk prediction

Theorem. Suppose that $0 < p < 1$, and $\eta, \epsilon = \Theta(1)$. Under certain conditions, there exists $T_0 = \Theta(1)$, such that for any polynomial iteration number $T \geq T_0$, the following results hold:

- Softmax attention selects the "direct parent" token: **Selector of the parent token**

$$\boxed{\left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_{N-1} \geq 1 - \exp(-\Omega(N)), \quad \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_{j} \leq \exp(-\Omega(N))}.$$

Transformers learn random walk prediction

Theorem. Suppose that $0 < p < 1$, and $\eta, \epsilon = \Theta(1)$. Under certain conditions, there exists $T_0 = \Theta(1)$, such that for any polynomial iteration number $T \geq T_0$, the following results hold:

- Softmax attention selects the "direct parent" token: **Selector of the parent token**

$$\boxed{\left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_{N-1} \geq 1 - \exp(-\Omega(N)), \quad \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_j \leq \exp(-\Omega(N))}.$$

- The value matrix converges to the true transition matrix in direction:

$$\left\| \frac{\mathbf{V}^{(T)}}{\|\mathbf{V}^{(T)}\|_F} - \frac{\mathbf{\Pi}^{*\top}}{\|\mathbf{\Pi}^{*\top}\|_F} \right\|_F = O\left(\frac{1}{\sqrt{T}}\right).$$

Transformers learn random walk prediction

Theorem. Suppose that $0 < p < 1$, and $\eta, \epsilon = \Theta(1)$. Under certain conditions, there exists $T_0 = \Theta(1)$, such that for any polynomial iteration number $T \geq T_0$, the following results hold:

- Softmax attention selects the "direct parent" token: **Selector of the parent token**

$$\boxed{\left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_{N-1} \geq 1 - \exp(-\Omega(N)), \quad \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_j \leq \exp(-\Omega(N))}.$$

- The value matrix converges to the true transition matrix in direction:

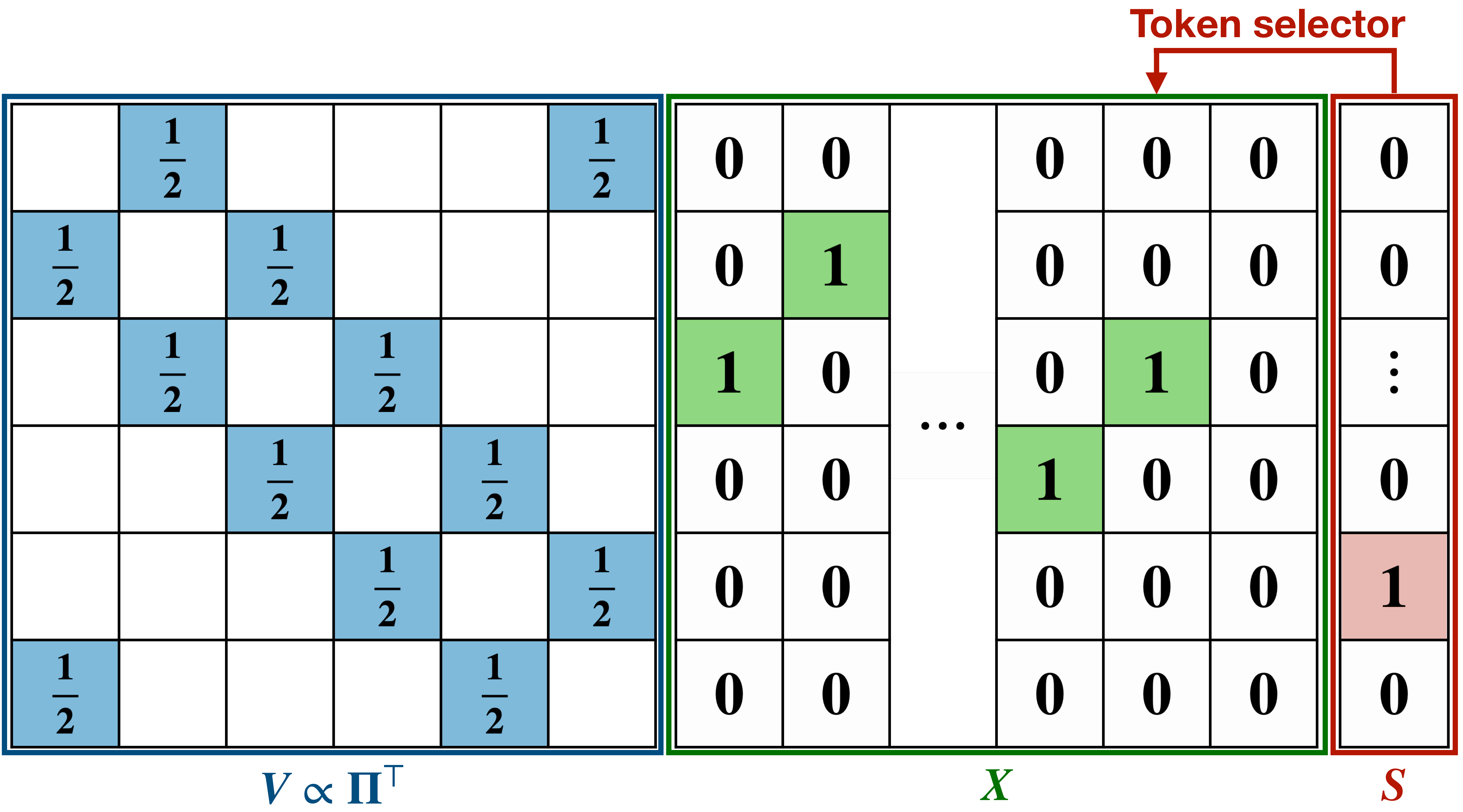
$$\boxed{\left\| \frac{\mathbf{V}^{(T)}}{\|\mathbf{V}^{(T)}\|_F} - \frac{\mathbf{\Pi}^{*\top}}{\|\mathbf{\Pi}^{*\top}\|_F} \right\|_F = O\left(\frac{1}{\sqrt{T}}\right)}.$$

Optimal probability transition on the parent token

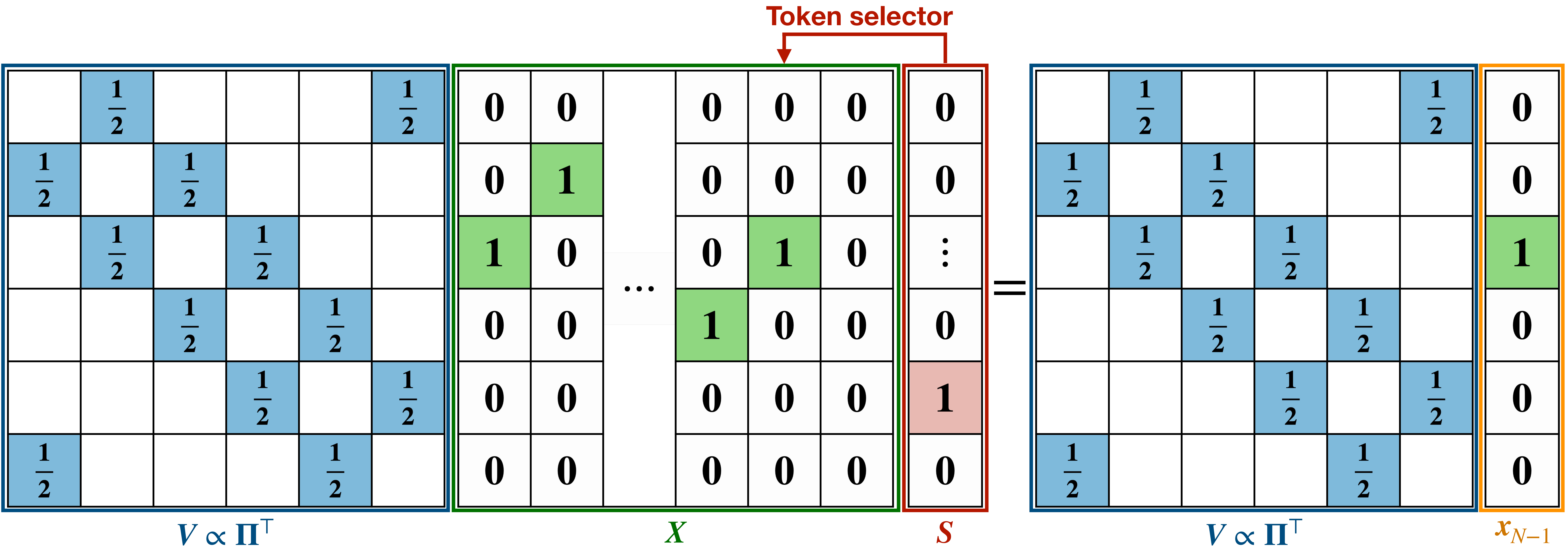
Transformers learn random walk prediction

$$
\underbrace{\begin{bmatrix} & \frac{1}{2} & & & & \frac{1}{2} \\ \frac{1}{2} & & \frac{1}{2} & & & \\ & \frac{1}{2} & & \frac{1}{2} & & \\ & & \frac{1}{2} & & \frac{1}{2} & \\ & & & \frac{1}{2} & & \frac{1}{2} \\ \frac{1}{2} & & & & \frac{1}{2} & \end{bmatrix}}_{\boldsymbol{V} \propto \boldsymbol{\Pi}^\top}
\underbrace{\begin{bmatrix} 0 & 0 & & 0 & 0 & 0 \\ 0 & 1 & & 0 & 0 & 0 \\ 1 & 0 & & 0 & 1 & 0 \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & & 0 & 0 & 0 \\ 0 & 0 & & 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{X}}
\underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \end{bmatrix}}_{\boldsymbol{S}}
$$

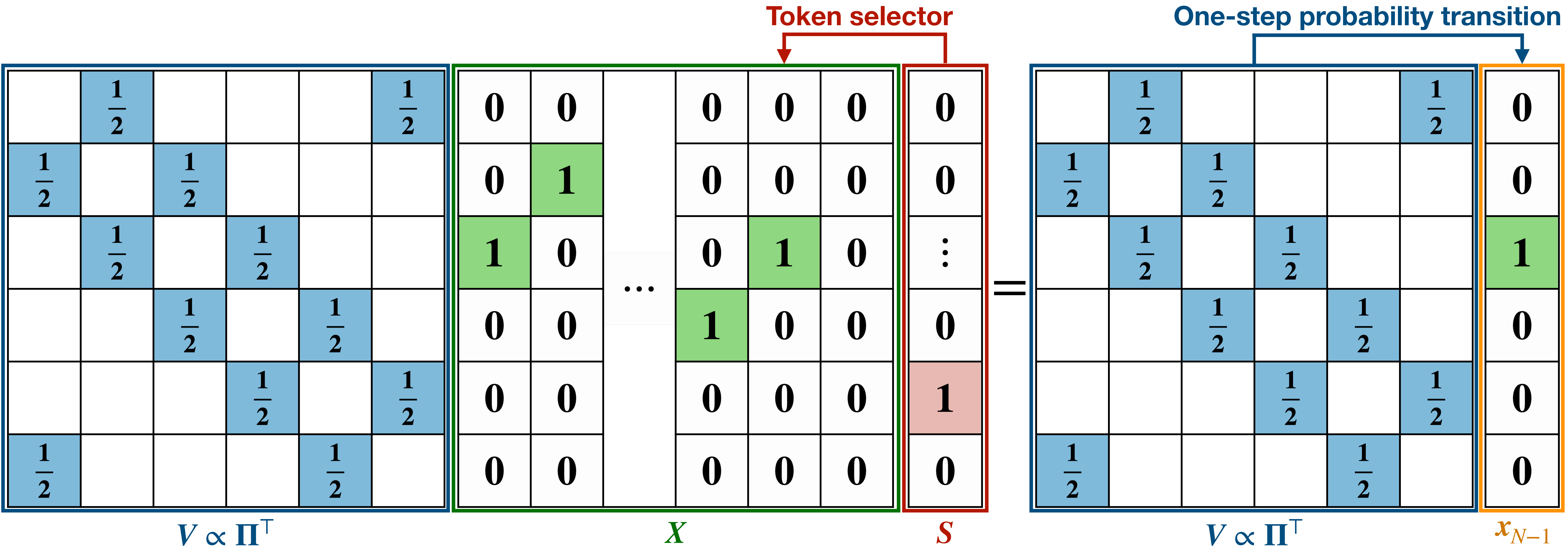
Transformers learn random walk prediction



Transformers learn random walk prediction



Transformers learn random walk prediction



Transformers learn random walk prediction

Transformers learn random walk prediction

Corollary. Suppose that $0 < p < 1$, and $\eta, \epsilon = \Theta(1)$. Under certain conditions, there exists $T_0 = \Theta(1)$, such that for any polynomial iteration number $T \geq T_0$, the following results hold:

Transformers learn random walk prediction

Corollary. Suppose that $0 < p < 1$, and $\eta, \epsilon = \Theta(1)$. Under certain conditions, there exists $T_0 = \Theta(1)$, such that for any polynomial iteration number $T \geq T_0$, the following results hold:

- The transformer converges to the optimal predictor:

$$\left\| \frac{\mathbf{f}(\mathbf{H}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})}{\|\mathbf{f}(\mathbf{H}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})\|_2} - \mathbf{\Pi}^{*\top}\mathbf{x}_{N-1} \right\|_2 = O\left(\frac{1}{\sqrt{T}}\right).$$

Transformers learn random walk prediction

Corollary. Suppose that $0 < p < 1$, and $\eta, \epsilon = \Theta(1)$. Under certain conditions, there exists $T_0 = \Theta(1)$, such that for any polynomial iteration number $T \geq T_0$, the following results hold:

- The transformer converges to the optimal predictor:

$$\left\| \frac{\mathbf{f}(\mathbf{H}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})}{\|\mathbf{f}(\mathbf{H}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})\|_2} - \boldsymbol{\Pi}^{*\top}\mathbf{x}_{N-1} \right\|_2 = O\left(\frac{1}{\sqrt{T}}\right).$$

- The trained transformer achieves optimal prediction accuracy:

$$\mathbb{P}_{(\mathbf{X},y)}\left[\mathrm{Pred}[\mathbf{f}(\mathbf{X}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})] = y\right] = \max\{p, 1-p\}.$$

Here we define: $\mathrm{Pred}(\mathbf{f}) = \min\left\{ j \in [K] : [\mathbf{f}]_j = \max_{i \in [K]}\{[\mathbf{f}]_i\} \right\}.$

Transformers learn random walk prediction

Corollary. Suppose that $0 < p < 1$, and $\eta, \epsilon = \Theta(1)$. Under certain conditions, there exists $T_0 = \Theta(1)$, such that for any polynomial iteration number $T \geq T_0$, the following results hold:

- The transformer converges to the optimal predictor:

$$\left\| \frac{\mathbf{f}(\mathbf{H}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})}{\|\mathbf{f}(\mathbf{H}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})\|_2} - \mathbf{\Pi}^{*\top}\mathbf{x}_{N-1} \right\|_2 = O\left(\frac{1}{\sqrt{T}}\right).$$

- The trained transformer achieves optimal prediction accuracy:

Optimal accuracy

$$\mathbb{P}_{(\mathbf{X},y)}\left[\mathrm{Pred}[\mathbf{f}(\mathbf{X}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})] = y\right] = \boxed{\max\{p, 1-p\}}$$

Here we define: $\mathrm{Pred}(\mathbf{f}) = \min\left\{ j \in [K] : [\mathbf{f}]_j = \max_{i \in [K]}\{[\mathbf{f}]_i\} \right\}.$

Failure in learning “deterministic walks” with $p = 0 \text{ or } 1$

Theorem. Suppose that $p = 0 \text{ or } 1$, K is a constant integer, and $N = rK + 1$ with $r \geq 1$. Then for any loss function $\ell(\,\cdot\,)$, any learning rate $\eta > 0$, and any $T \geq 0$, it holds that

$$\mathbb{P}_{(\mathbf{X},y)}\big[\mathrm{Pred}[\mathbf{f}(\mathbf{X}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})] = y\big] = \frac{1}{K}.$$

Failure in learning "deterministic walks" with $p = 0 \text{ or } 1$

Theorem. Suppose that $p = 0 \text{ or } 1$, K is a constant integer, and $N = rK + 1$ with $r \geq 1$. Then for any loss function $\ell(\cdot)$, any learning rate $\eta > 0$, and any $T \geq 0$, it holds that

$$\mathbb{P}_{(\mathbf{X},y)}\left[\mathrm{Pred}[\mathbf{f}(\mathbf{X}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})] = y\right] = \boxed{\frac{1}{K}}.$$

Random guess

Failure in learning "deterministic walks" with $p = 0$ or 1

Theorem. Suppose that $p = 0$ or 1, K is a constant integer, and $N = rK + 1$ with $r \geq 1$. Then for any loss function $\ell(\cdot)$, any learning rate $\eta > 0$, and any $T \geq 0$, it holds that

$$\mathbb{P}_{(\mathbf{X},y)}\left[\mathrm{Pred}[\mathbf{f}(\mathbf{X}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})] = y\right] = \boxed{\frac{1}{K}}.$$

Random guess

Moreover, with probability 1, for all $T \geq 0$, it holds that

$$\mathbf{V}^{(T)} \propto \mathbf{1}_{K \times K}, \quad \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_1 = \cdots = \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_{N-1}.$$

Failure in learning "deterministic walks" with $p = 0 \text{ or } 1$

Theorem. Suppose that $p = 0 \text{ or } 1$, K is a constant integer, and $N = rK + 1$ with $r \geq 1$. Then for any loss function $\ell(\cdot)$, any learning rate $\eta > 0$, and any $T \geq 0$, it holds that

$$\mathbb{P}_{(\mathbf{X},y)}\left[\mathrm{Pred}[\mathbf{f}(\mathbf{X}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})] = y\right] = \boxed{\frac{1}{K}}. \quad \textbf{Random guess}$$

Moreover, with probability 1, for all $T \geq 0$, it holds that

$$\mathbf{V}^{(T)} \propto \mathbf{1}_{K\times K}, \quad \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_1 = \cdots = \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_{N-1}.$$

At zero initialization, softmax attention serves as an average, and the average $\bar{\mathbf{x}} \propto \mathbf{1}$ is not informative at all!

Failure in learning "deterministic walks" with $p = 0 \text{ or } 1$

Theorem. Suppose that $p = 0 \text{ or } 1$, K is a constant integer, and $N = rK + 1$ with $r \geq 1$. Then for any loss function $\ell(\,\cdot\,)$, any learning rate $\eta > 0$, and any $T \geq 0$, it holds that

$$\mathbb{P}_{(\mathbf{X},y)}\big[\mathrm{Pred}[\mathbf{f}(\mathbf{X}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})] = y\big] = \boxed{\frac{1}{K}}. \quad \textbf{Random guess}$$

Moreover, with probability 1, for all $T \geq 0$, it holds that

$$\mathbf{V}^{(T)} \propto \mathbf{1}_{K\times K}, \quad \big[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\big]_1 = \cdots = \big[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\big]_{N-1}.$$

At zero initialization, softmax attention serves as an average, and the average $\bar{\mathbf{x}} \propto \mathbf{1}$ is not informative at all! $\implies$ Optimization is on a "**ridge**" of bad points.

Failure in learning "deterministic walks" with $p = 0 \text{ or } 1$

Theorem. Suppose that $p = 0 \text{ or } 1$, K is a constant integer, and $N = rK + 1$ with $r \geq 1$. Then for any loss function $\ell(\,\cdot\,)$, any learning rate $\eta > 0$, and any $T \geq 0$, it holds that

$$\mathbb{P}_{(\mathbf{X},y)}\left[\mathrm{Pred}[\mathbf{f}(\mathbf{X}, \mathbf{V}^{(T)}, \mathbf{W}^{(T)})] = y\right] = \boxed{\frac{1}{K}}.$$ **Random guess**

Moreover, with probability 1, for all $T \geq 0$, it holds that

$$\mathbf{V}^{(T)} \propto \mathbf{1}_{K\times K}, \quad \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_1 = \cdots = \left[\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(T)} \mathbf{h}_N)\right]_{N-1}.$$

At zero initialization, softmax attention serves as an average, and the average $\bar{\mathbf{x}} \propto \mathbf{1}$ is not informative at all! $\Longrightarrow$ Optimization is on a "**ridge**" of bad points.

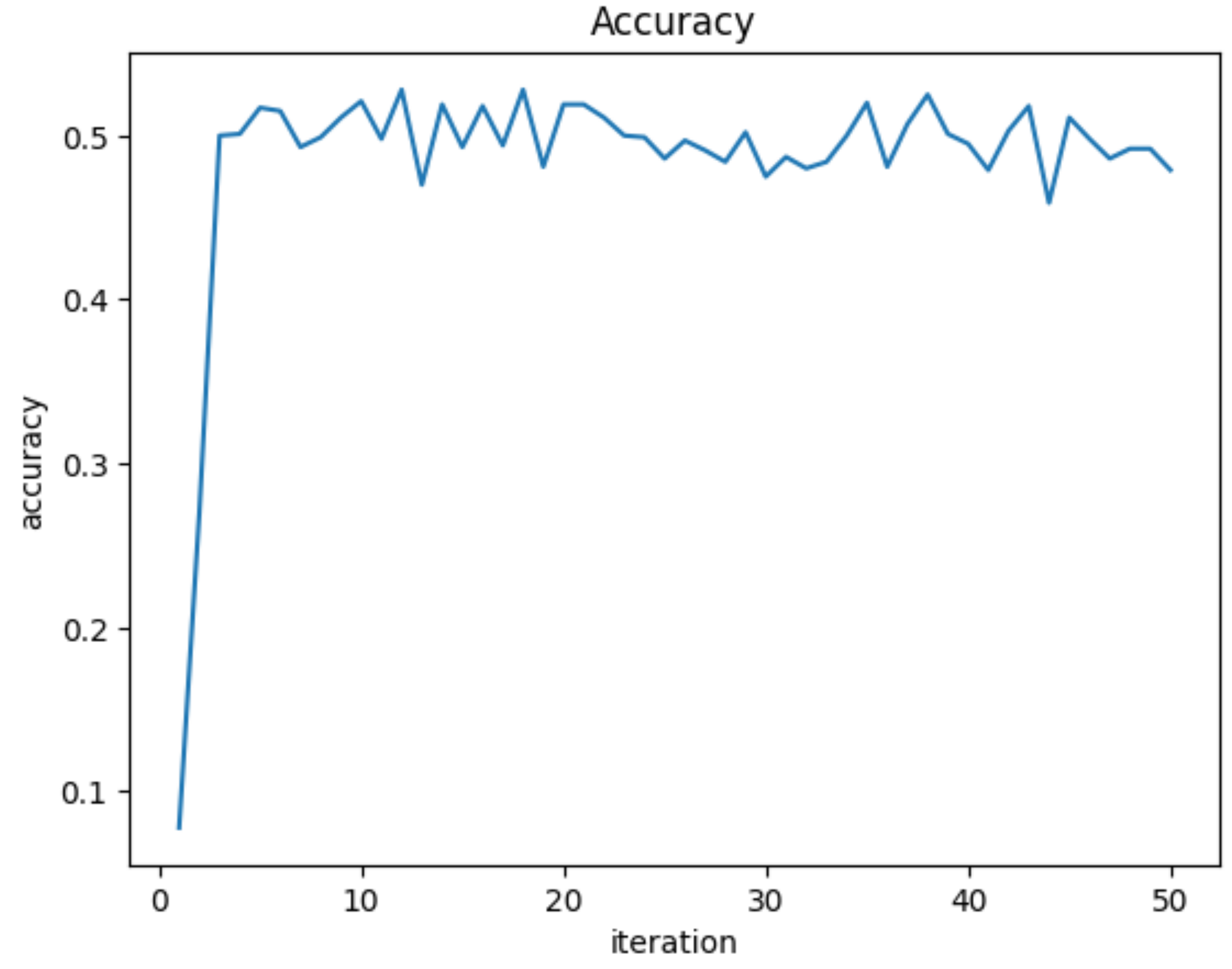
Random initialization overcomes the issue to a certain extent.

Experiments

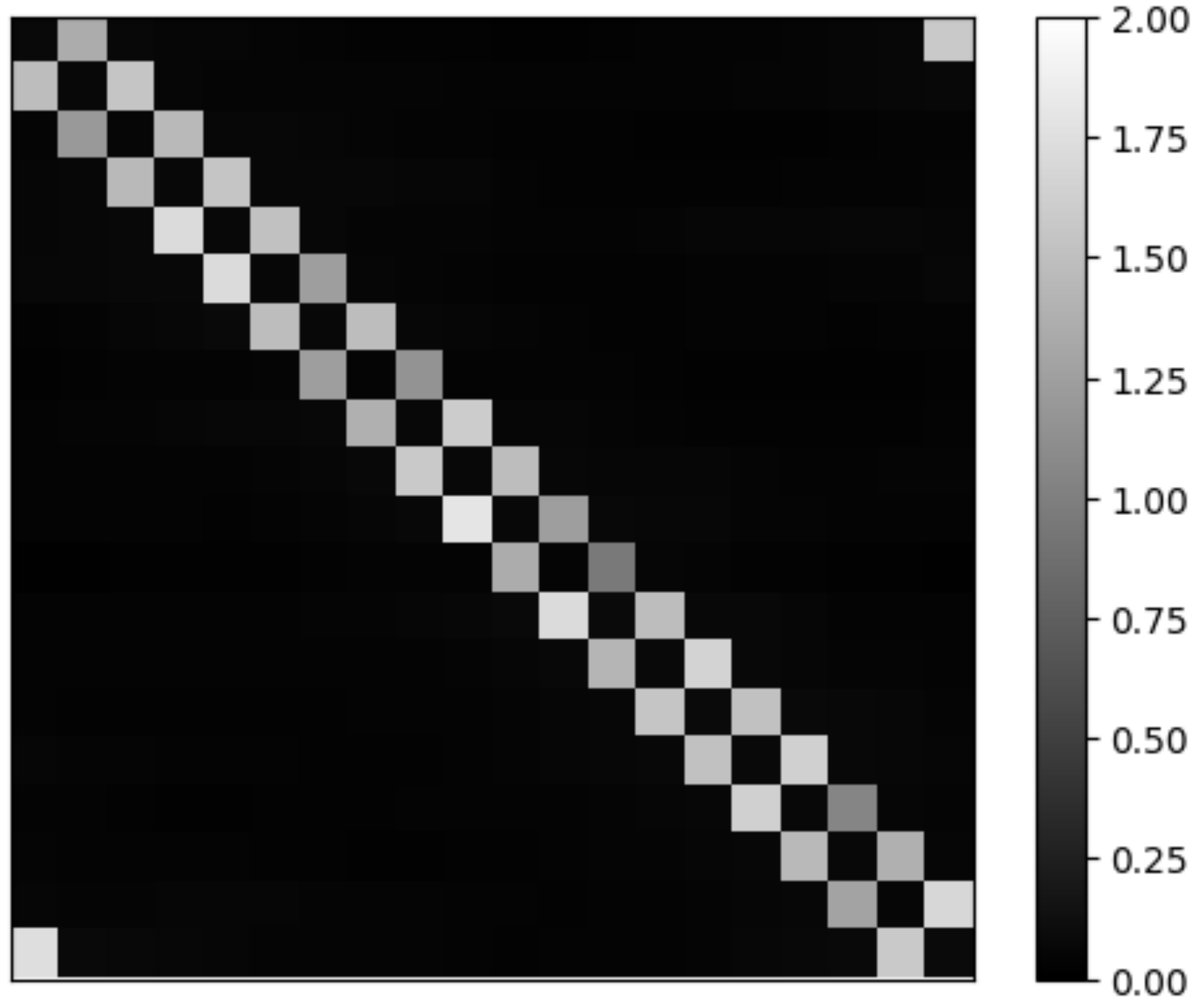
$p = 1/2$:

$K = 20,\ N = 101$

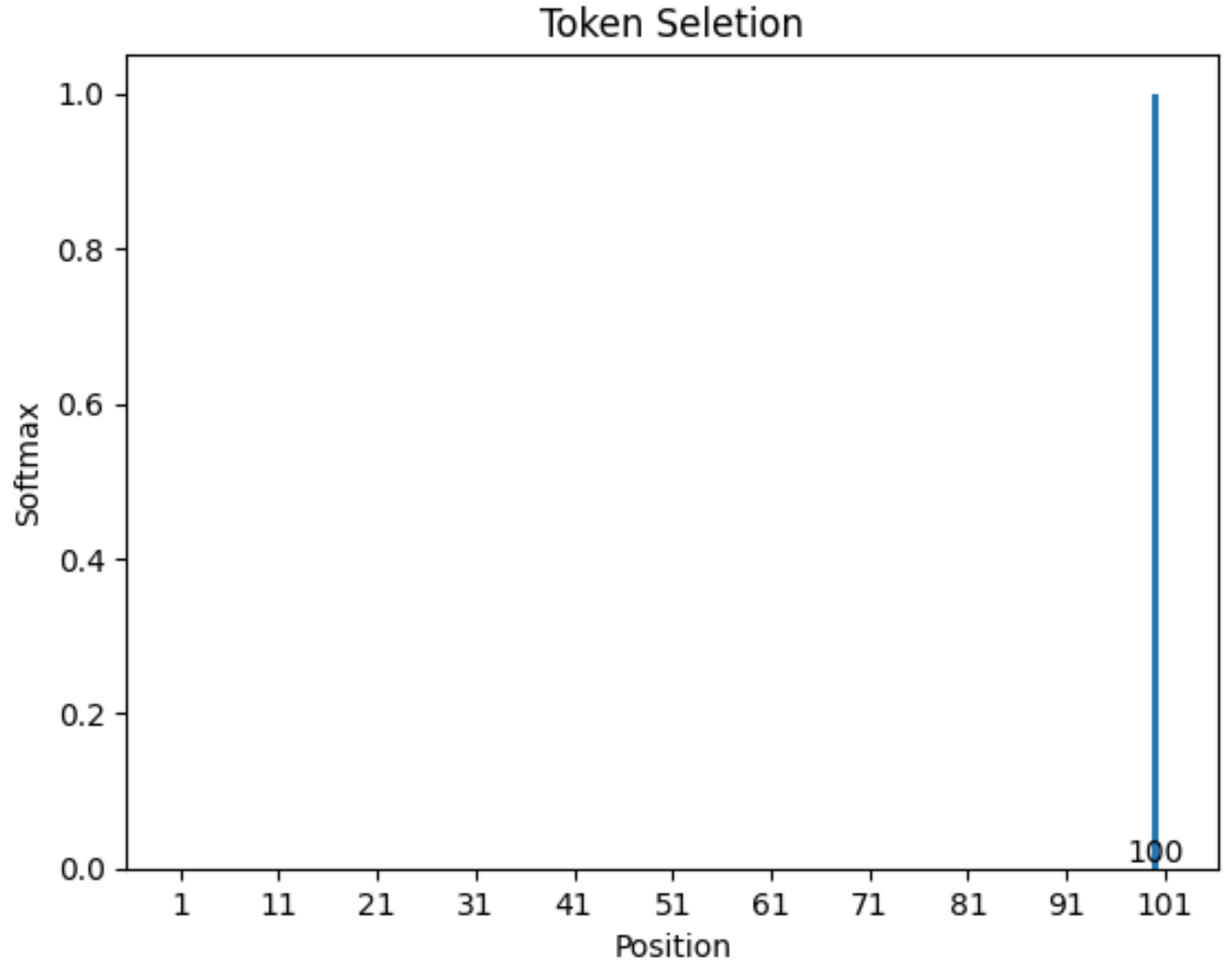
Accuracy



$\mathbf{V}^{(T)}$



Softmax attention

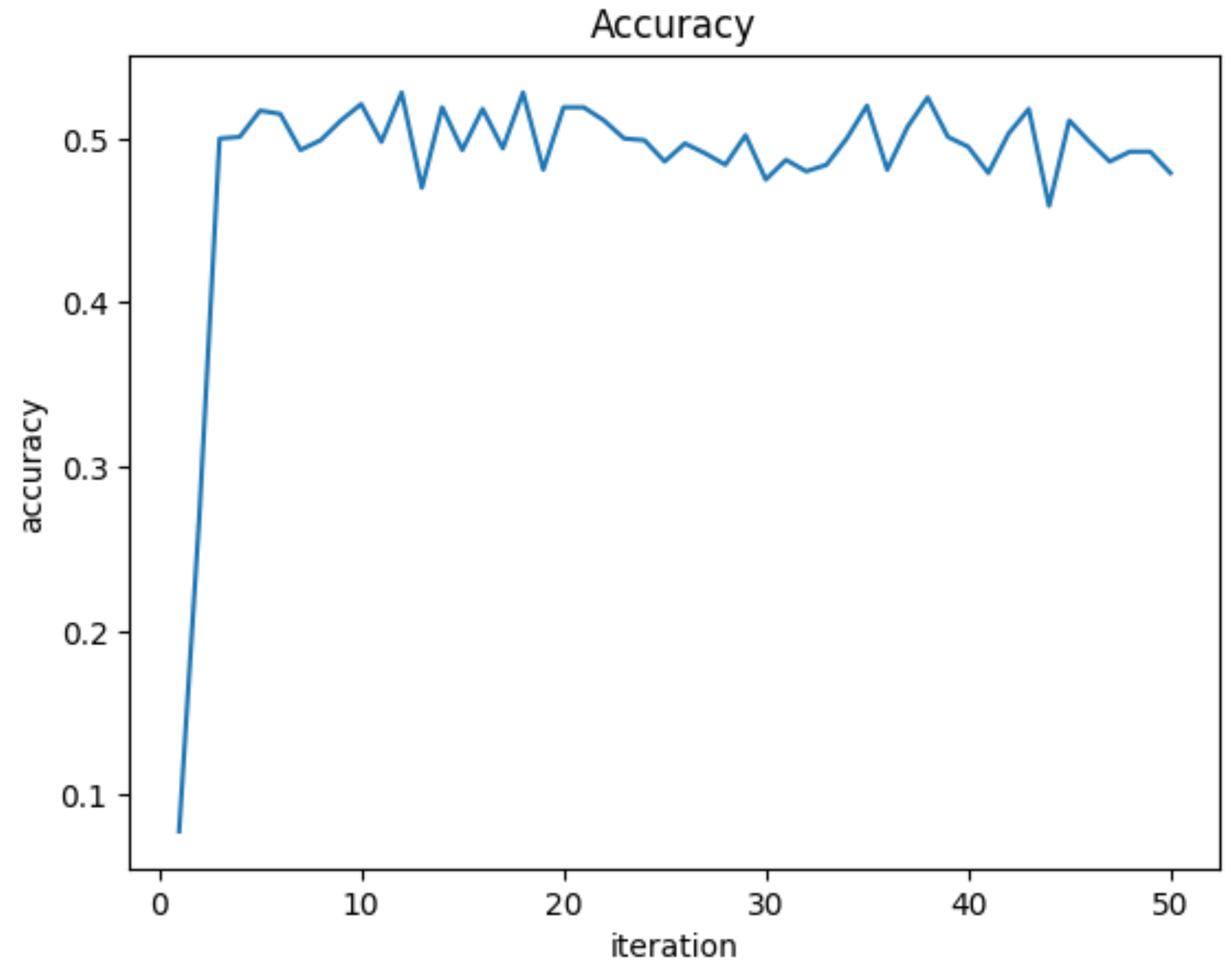


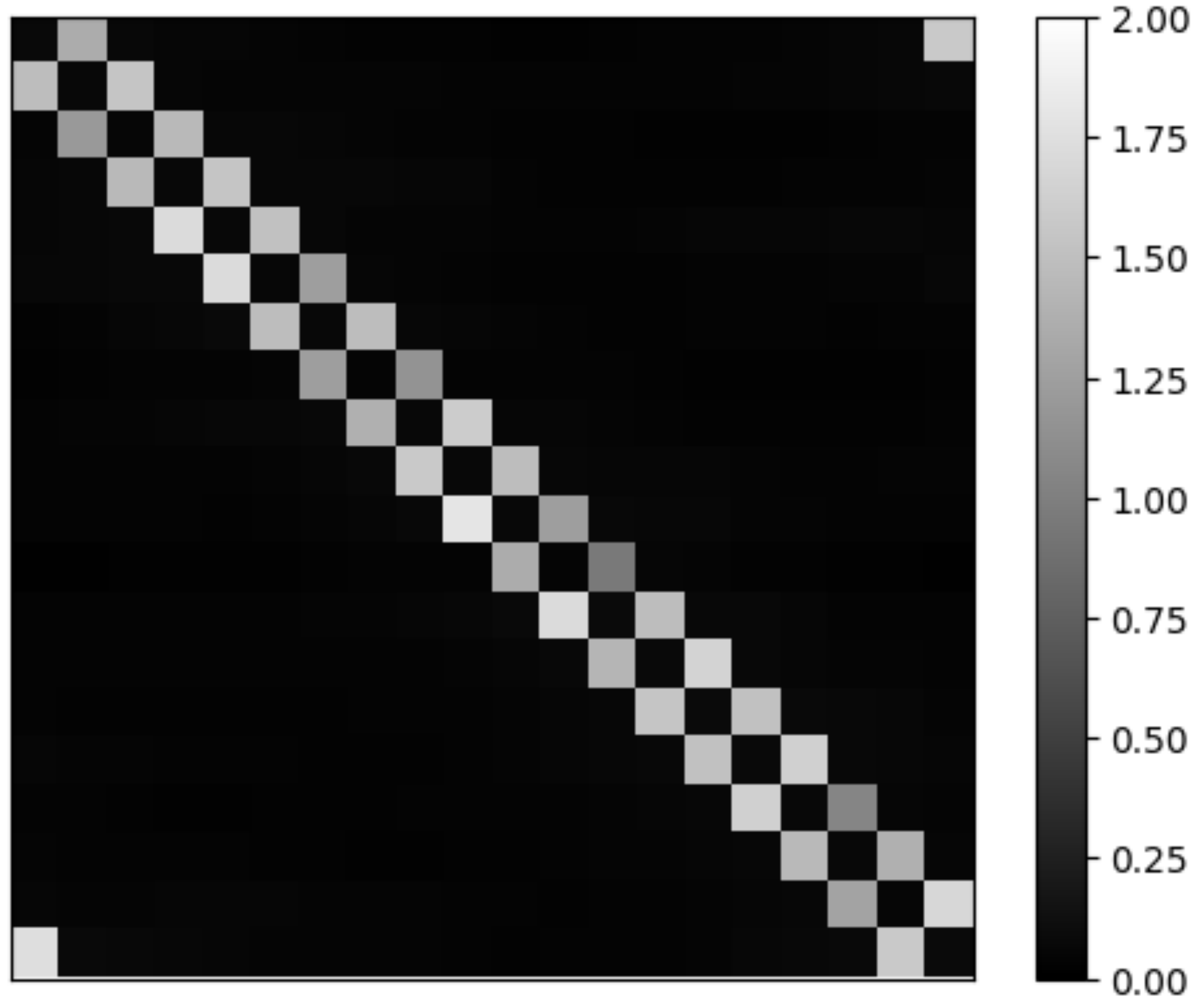
Experiments

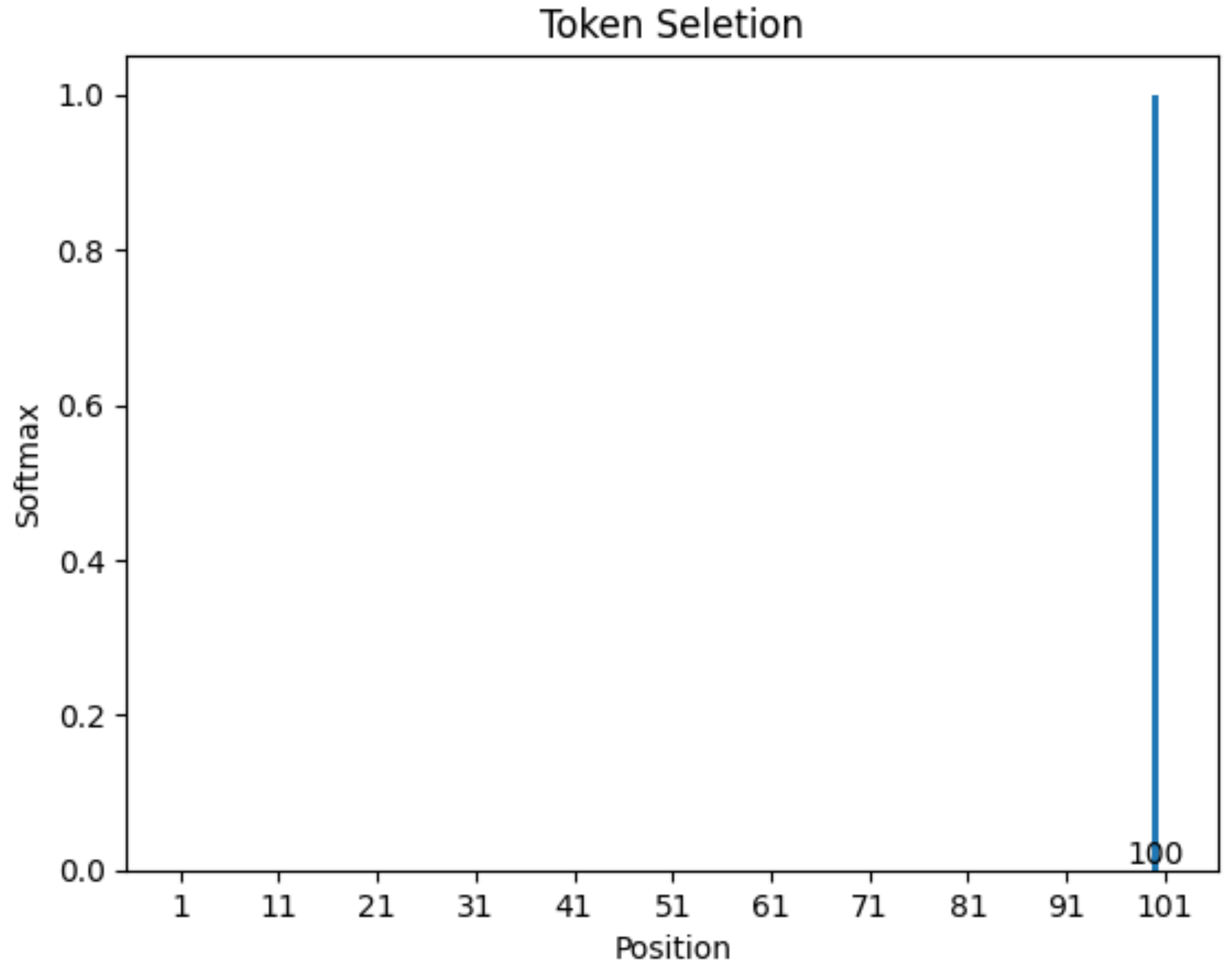
Accuracy | $\mathbf{V}^{(T)}$ | **Softmax attention**

$p = 1/2$:

$K = 20,\ N = 101$

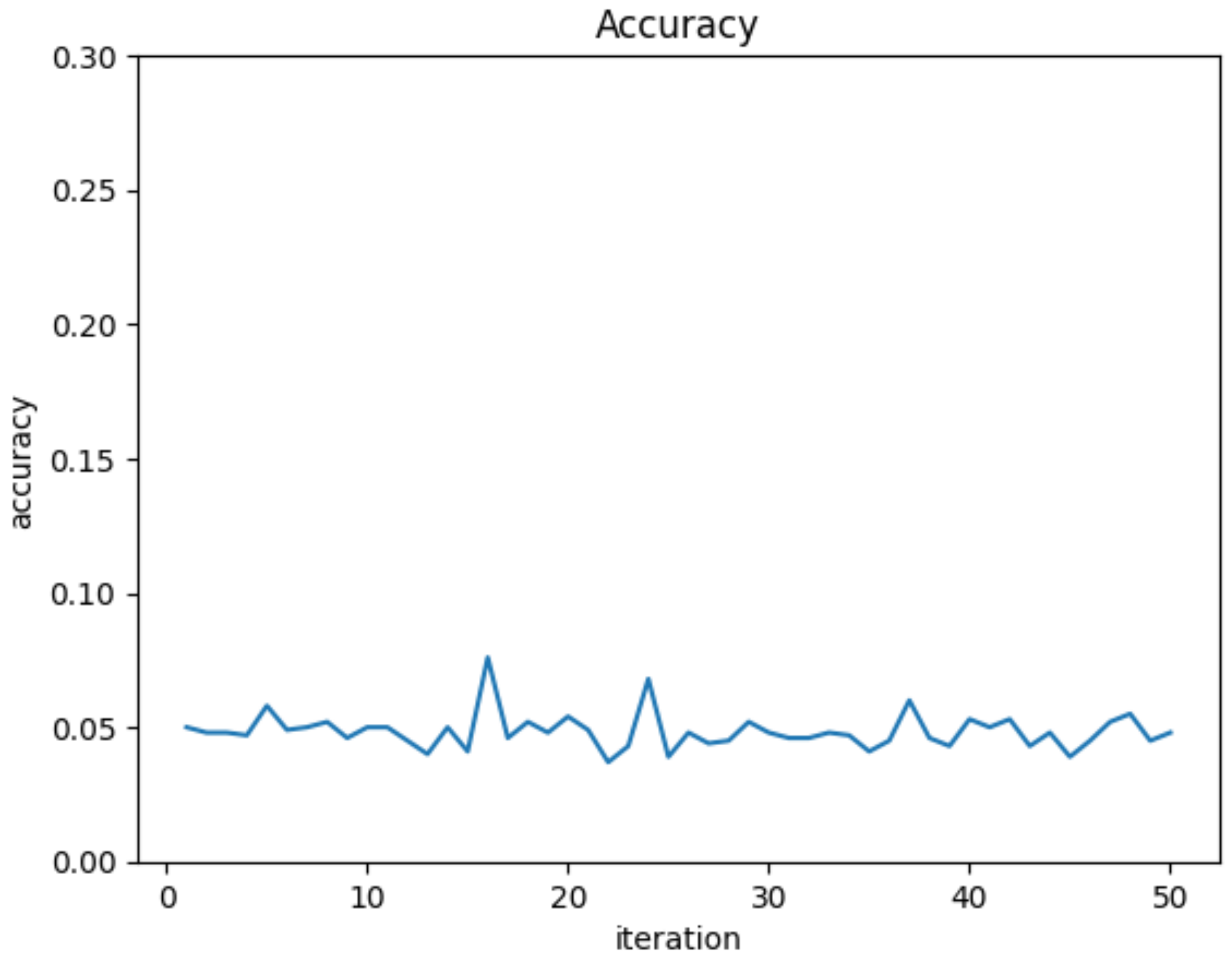


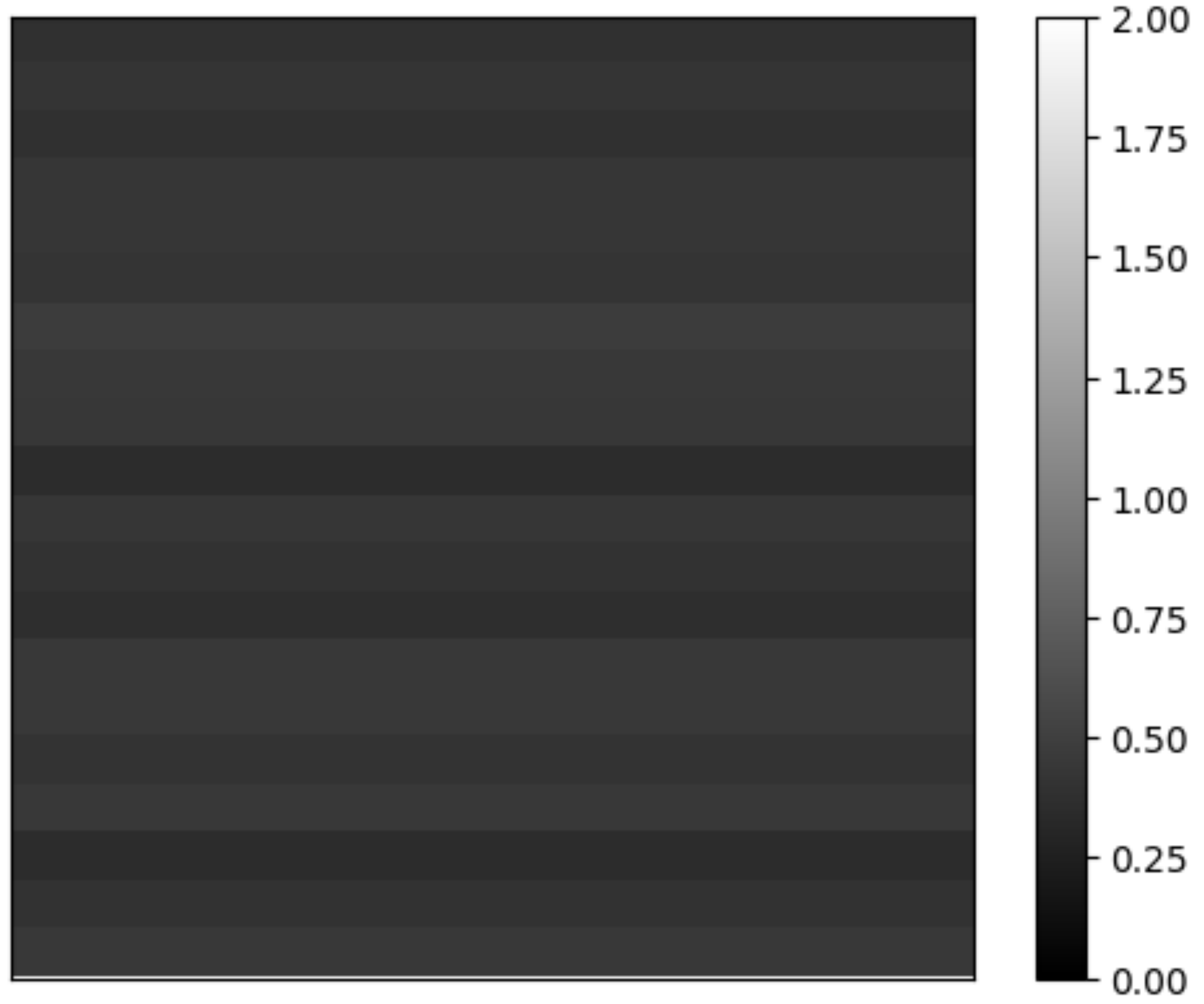


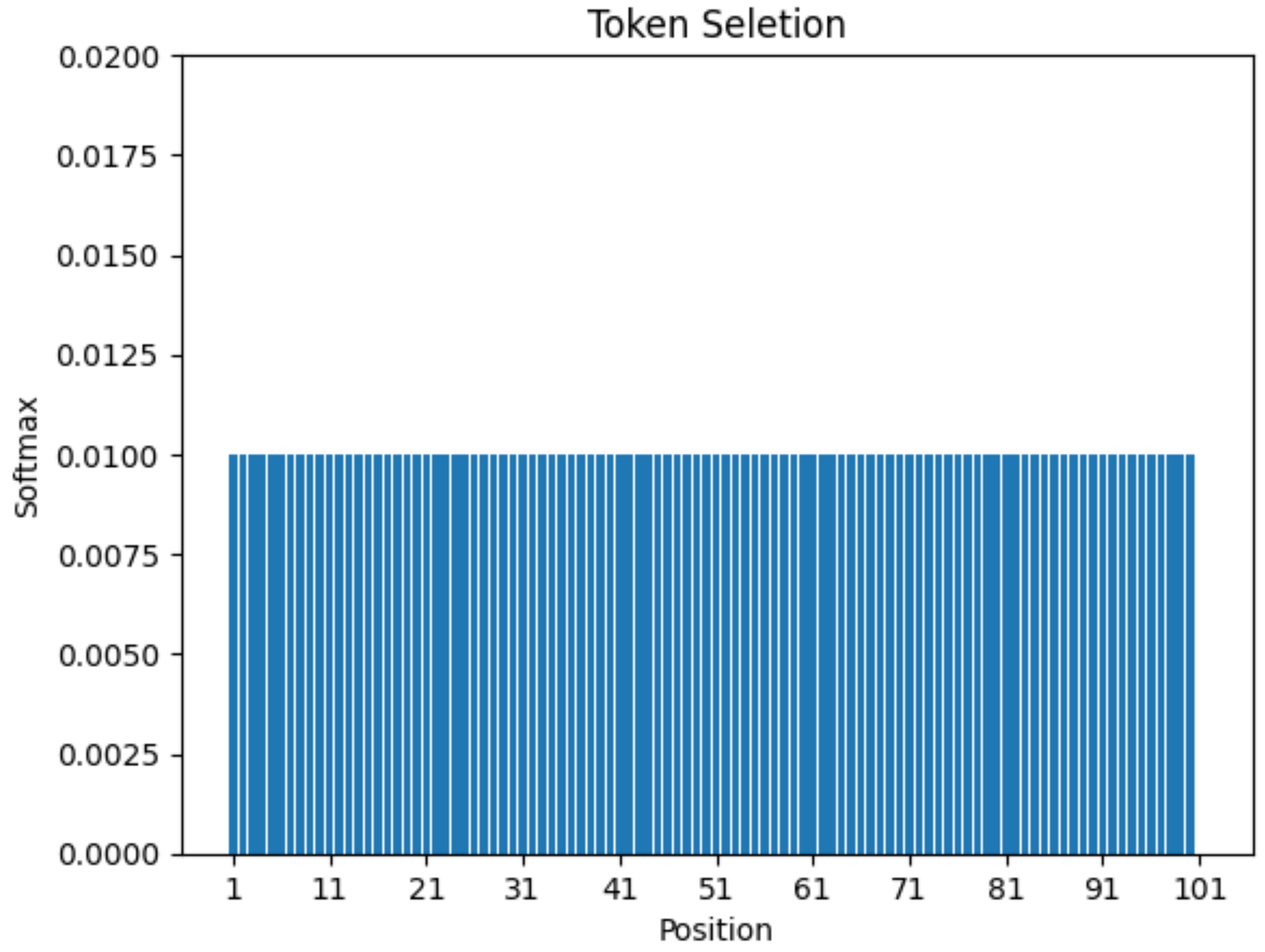


$p = 1$:

$K = 20,\ N = 101$







Summary

By gradient descent based training, a one-layer transformer can handle different classic statistical learning tasks:

Summary

By gradient descent based training, a one-layer transformer can handle different classic statistical learning tasks:

When the data follows a 1-NN model, the trained transformer can **learn the 1-NN prediction rule**, with softmax attention serves as a **nearest neighbor selector**;

Summary

By gradient descent based training, a one-layer transformer can handle different classic statistical learning tasks:

When the data follows a 1-NN model, the trained transformer can **learn the 1-NN prediction rule**, with softmax attention serves as a **nearest neighbor selector**;

When the data follows a group-sparse model, the trained transformer can **capture the sparsity pattern**, with softmax attention serves as a **variable selector**;

Summary

By gradient descent based training, a one-layer transformer can handle different classic statistical learning tasks:

When the data follows a 1-NN model, the trained transformer can **learn the 1-NN prediction rule**, with softmax attention serves as a **nearest neighbor selector**;

When the data follows a group-sparse model, the trained transformer can **capture the sparsity pattern**, with softmax attention serves as a **variable selector**;

When the data follows a random walk, the trained transformer can **capture the Markov property**, with softmax attention serves as a **parent token selector**.

Summary

By gradient descent based training, a one-layer transformer can handle different classic statistical learning tasks:

When the data follows a 1-NN model, the trained transformer can **learn the 1-NN prediction rule**, with softmax attention serves as a **nearest neighbor selector**;

When the data follows a group-sparse model, the trained transformer can **capture the sparsity pattern**, with softmax attention serves as a **variable selector**;

When the data follows a random walk, the trained transformer can **capture the Markov property**, with softmax attention serves as a **parent token selector**.

Thank you!

Self-attention in transformers

$$\text{self-attention}(\mathbf{X}) = \mathbf{W}_V\mathbf{X}\text{softmax}(\mathbf{X}^\top\mathbf{W}_K^\top\mathbf{W}_Q\mathbf{X}),$$

Self-attention in transformers

$$\text{self-attention}(\mathbf{X}) = \mathbf{W}_V\mathbf{X}\text{softmax}(\mathbf{X}^\top\mathbf{W}_K^\top\mathbf{W}_Q\mathbf{X}),$$

$$[\text{self-attention}(\mathbf{X})]_{\cdot i} = \mathbf{W}_V\mathbf{X}\text{softmax}(\mathbf{X}^\top\mathbf{W}_K^\top\mathbf{W}_Q\mathbf{x}_i)$$

Self-attention in transformers

$$\text{self-attention}(\mathbf{X}) = \mathbf{W}_V \mathbf{X} \text{softmax}(\mathbf{X}^\top \mathbf{W}_K^\top \mathbf{W}_Q \mathbf{X}),$$

$$[\text{self-attention}(\mathbf{X})]_{\cdot i} = \mathbf{W}_V \mathbf{X} \text{softmax}(\mathbf{X}^\top \mathbf{W}_K^\top \mathbf{W}_Q \mathbf{x}_i) = \sum_i \alpha_{ji} \cdot \mathbf{W}_V \mathbf{x}_j$$

Self-attention in transformers

$$\text{self-attention}(\mathbf{X}) = \mathbf{W}_V\mathbf{X}\text{softmax}(\mathbf{X}^\top\mathbf{W}_K^\top\mathbf{W}_Q\mathbf{X}),$$

Scores

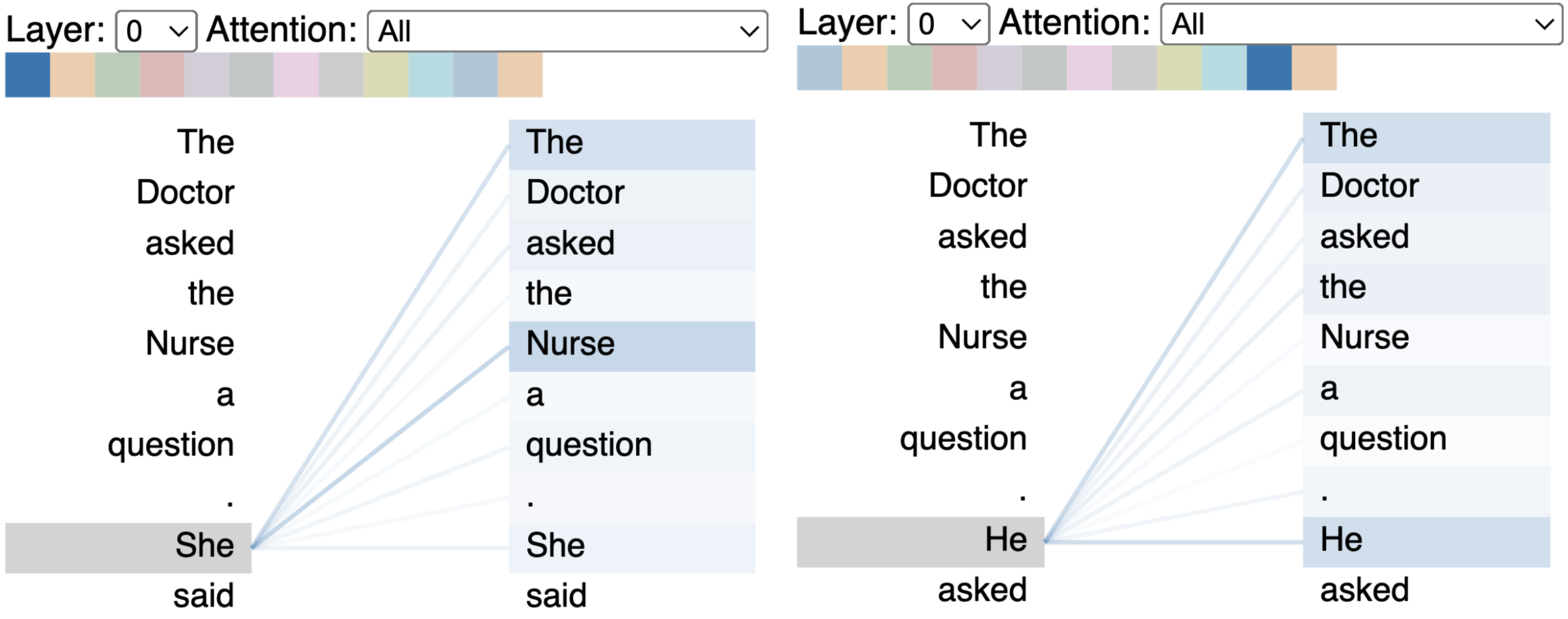
$$[\text{self-attention}(\mathbf{X})]_{\cdot i} = \mathbf{W}_V\mathbf{X}\boxed{\text{softmax}(\mathbf{X}^\top\mathbf{W}_K^\top\mathbf{W}_Q\mathbf{x}_i)} = \sum_i \boxed{\alpha_{ji}} \cdot \mathbf{W}_V\mathbf{x}_j$$

Self-attention in transformers

$$\text{self-attention}(\mathbf{X}) = \mathbf{W}_V\mathbf{X}\text{softmax}(\mathbf{X}^\top\mathbf{W}_K^\top\mathbf{W}_Q\mathbf{X}),$$

Scores

$$[\text{self-attention}(\mathbf{X})]_{\cdot i} = \mathbf{W}_V\mathbf{X}\boxed{\text{softmax}(\mathbf{X}^\top\mathbf{W}_K^\top\mathbf{W}_Q\mathbf{x}_i)} = \sum_i \boxed{\alpha_{ji}} \cdot \mathbf{W}_V\mathbf{x}_j$$



Diverse roles of self-attention

Scores

$$[\text{self-attention}(\mathbf{X})]_{.i} = \mathbf{W}_V \mathbf{X} \boxed{\text{softmax}(\mathbf{X}^\top \mathbf{W}_K^\top \mathbf{W}_Q \mathbf{x}_i)} = \sum_j \boxed{\alpha_{ji}} \cdot \mathbf{W}_V \mathbf{x}_j$$

$$\text{softmax}(\mathbf{z}) \approx \begin{cases} \text{average}, & \text{if } \mathbf{z} \approx \mathbf{0}, \\ \text{weighted average}, & \text{if } \|\mathbf{z}\|_2 \text{ is neither too large nor too small}, \\ \text{'hard max'}, & \text{if } \|\mathbf{z}\|_2 \to \infty, \end{cases}$$

Diverse roles of self-attention

Scores

$$[\text{self-attention}(\mathbf{X})]_{\cdot i} = \mathbf{W}_V\mathbf{X}\boxed{\text{softmax}(\mathbf{X}^\top\mathbf{W}_K^\top\mathbf{W}_Q\mathbf{x}_i)} = \sum_j \boxed{\alpha_{ji}} \cdot \mathbf{W}_V\mathbf{x}_j$$

$$\text{softmax}(\mathbf{z}) \approx \begin{cases} \text{average}, & \text{if } \mathbf{z} \approx \mathbf{0}, \\ \text{weighted average}, & \text{if } \|\mathbf{z}\|_2 \text{ is neither too large nor too small}, \\ \boxed{\text{'hard max'},} & \text{if } \|\mathbf{z}\|_2 \to \infty, \end{cases}$$

The largest entry in $\mathbf{z}$ $\longrightarrow$

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Diverse roles of self-attention

Scores

$$[\text{self-attention}(\mathbf{X})]_{\cdot i} = \mathbf{W}_V \mathbf{X} \boxed{\text{softmax}(\mathbf{X}^\top \mathbf{W}_K^\top \mathbf{W}_Q \mathbf{x}_i)} = \sum_j \boxed{\alpha_{ji}} \cdot \mathbf{W}_V \mathbf{x}_j$$

$$\text{softmax}(\mathbf{z}) \approx \begin{cases} \text{average}, & \text{if } \mathbf{z} \approx \mathbf{0}, \\ \text{weighted average}, & \text{if } \|\mathbf{z}\|_2 \text{ is neither too large nor too small}, \\ \boxed{\text{'hard max'},} & \text{if } \|\mathbf{z}\|_2 \to \infty, \end{cases}$$

The largest entry in $\mathbf{z}$ $\longrightarrow$

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Longrightarrow \text{"Token/variable selection"}$$

Proof sketch of in-context nearest neighbor predictor

Consider the parameter matrix $\mathbf{W}$ as a block matrix:

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} \\ \mathbf{W}_{31} & \mathbf{W}_{32} & \mathbf{W}_{33} \end{bmatrix}.$$

Proof sketch of in-context nearest neighbor predictor

Consider the parameter matrix $\mathbf{W}$ as a block matrix:

$$\mathbf{W} = \begin{array}{cc} \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} \\ \mathbf{W}_{31} & \mathbf{W}_{32} & \mathbf{W}_{33} \end{bmatrix} & \begin{array}{l} \}\, d \\ \}\, 1 \\ \}\, 1 \end{array} \\ \begin{array}{ccc} \underbrace{\hphantom{\mathbf{W}_{11}}}_{d} & \underbrace{\hphantom{\mathbf{W}_{12}}}_{1} & \underbrace{\hphantom{\mathbf{W}_{13}}}_{1} \end{array} & \end{array}$$

Proof sketch of in-context nearest neighbor predictor

Consider the parameter matrix $\mathbf{W}$ as a block matrix:

$$\mathbf{W} = \begin{array}{cc} \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} \\ \mathbf{W}_{31} & \mathbf{W}_{32} & \mathbf{W}_{33} \end{bmatrix} & \begin{array}{l} \} \, d \\ \} \, 1 \\ \} \, 1 \end{array} \\ \begin{array}{ccc} \quad \underbrace{\qquad}_{d} & \underbrace{\qquad}_{1} & \underbrace{\qquad}_{1} \end{array} & \end{array} \qquad \mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \dots & y_N & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}.$$

Proof sketch of in-context nearest neighbor predictor

Consider the parameter matrix $\mathbf{W}$ as a block matrix:

$$\mathbf{W} = \begin{array}{cc} \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} \\ \mathbf{W}_{31} & \mathbf{W}_{32} & \mathbf{W}_{33} \end{bmatrix} & \begin{array}{l} \}\, d \\ \}\, 1 \\ \}\, 1 \end{array} \\ \begin{array}{ccc} \underbrace{\quad}_{d} & \underbrace{\quad}_{1} & \underbrace{\quad}_{1} \end{array} & \end{array} \qquad \mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \dots & y_N & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}.$$

A key observation is that

$$\nabla_{\mathbf{W}_{11}} L(\mathbf{W}^{(t)}) = \mathbb{E}\left[\sum_{i=1}^{N} g_i^{(t)}(\mathbf{x}_i^\top \mathbf{x}_{N+1}) \cdot \mathbf{x}_i \mathbf{x}_{N+1}^\top + g_*^{(t)}(\mathbf{x}_{i*}^\top \mathbf{x}_{N+1}) \cdot \mathbf{x}_{i*} \mathbf{x}_{N+1}^\top\right],$$

where $g_i^k(x) : \mathbb{R} \to \mathbb{R}, i \in [N]$ are functions that map scalars to scalars.

Proof sketch of in-context nearest neighbor predictor

Consider the parameter matrix $\mathbf{W}$ as a block matrix:

$$\mathbf{W} = \begin{array}{cc} \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} \\ \mathbf{W}_{31} & \mathbf{W}_{32} & \mathbf{W}_{33} \end{bmatrix} & \begin{array}{l} \} d \\ \} 1 \\ \} 1 \end{array} \\ \begin{array}{ccc} \quad d \quad & \quad 1 \quad & \quad 1 \quad \end{array} & \end{array} \qquad \mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \dots & y_N & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}.$$

A key observation is that

$$\boxed{\nabla_{\mathbf{W}_{11}} L(\mathbf{W}^{(t)}) = \mathbb{E}\left[\sum_{i=1}^{N} g_i^{(t)}(\mathbf{x}_i^\top \mathbf{x}_{N+1}) \cdot \mathbf{x}_i \mathbf{x}_{N+1}^\top + g_*^{(t)}(\mathbf{x}_{i*}^\top \mathbf{x}_{N+1}) \cdot \mathbf{x}_{i*} \mathbf{x}_{N+1}^\top \right]},$$

where $g_i^k(x) : \mathbb{R} \to \mathbb{R}$, $i \in [N]$ are functions that map scalars to scalars.

$$\boxed{\mathbf{U} \nabla_{\mathbf{W}_{11}} L(\mathbf{W}^{(t)}) \mathbf{U}^\top = \nabla_{\mathbf{W}_{11}} L(\mathbf{W}^{(t)}) \text{ for all orthogonal matrix } \mathbf{U}! \implies \nabla_{\mathbf{W}_{11}} L(\mathbf{W}^{(t)}) \propto \mathbf{I}}$$

Proof sketch of in-context nearest neighbor predictor

Consider the parameter matrix $\mathbf{W}$ as a block matrix:

$$\mathbf{W} = \begin{array}{cl} \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} \\ \mathbf{W}_{31} & \mathbf{W}_{32} & \mathbf{W}_{33} \end{bmatrix} & \begin{array}{l} \}\, d \\ \}\, 1 \\ \}\, 1 \end{array} \\ \begin{array}{ccc} \underbrace{\qquad}_{d} & \underbrace{\qquad}_{1} & \underbrace{\qquad}_{1} \end{array} & \end{array} \qquad \mathbf{H} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N & \mathbf{x}_{\text{query}} \\ y_1 & y_2 & \dots & y_N & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}.$$

A key observation is that

$$\boxed{\nabla_{\mathbf{W}_{11}} L(\mathbf{W}^{(t)}) = \mathbb{E}\left[\sum_{i=1}^{N} g_i^{(t)}(\mathbf{x}_i^\top \mathbf{x}_{N+1}) \cdot \mathbf{x}_i \mathbf{x}_{N+1}^\top + g_*^{(t)}(\mathbf{x}_{i*}^\top \mathbf{x}_{N+1}) \cdot \mathbf{x}_{i*} \mathbf{x}_{N+1}^\top\right],}$$

where $g_i^k(x) : \mathbb{R} \to \mathbb{R}$, $i \in [N]$ are functions that map scalars to scalars.

$$\boxed{\mathbf{U}\, \nabla_{\mathbf{W}_{11}} L(\mathbf{W}^{(t)}) \mathbf{U}^\top = \nabla_{\mathbf{W}_{11}} L(\mathbf{W}^{(t)}) \text{ for all orthogonal matrix } \mathbf{U}! \implies \nabla_{\mathbf{W}_{11}} L(\mathbf{W}^{(t)}) \propto \mathbf{I}}$$

Moreover, $\nabla_{\mathbf{W}_{ij}} L(\mathbf{W}^k) = 0$ for all i, j except for $\mathbf{W}_{11}$ and $\mathbf{W}_{33}$!

Proof sketch of in-context nearest neighbor predictor

Proposition.

Along the optimization path of gradient descent, the weights has the form

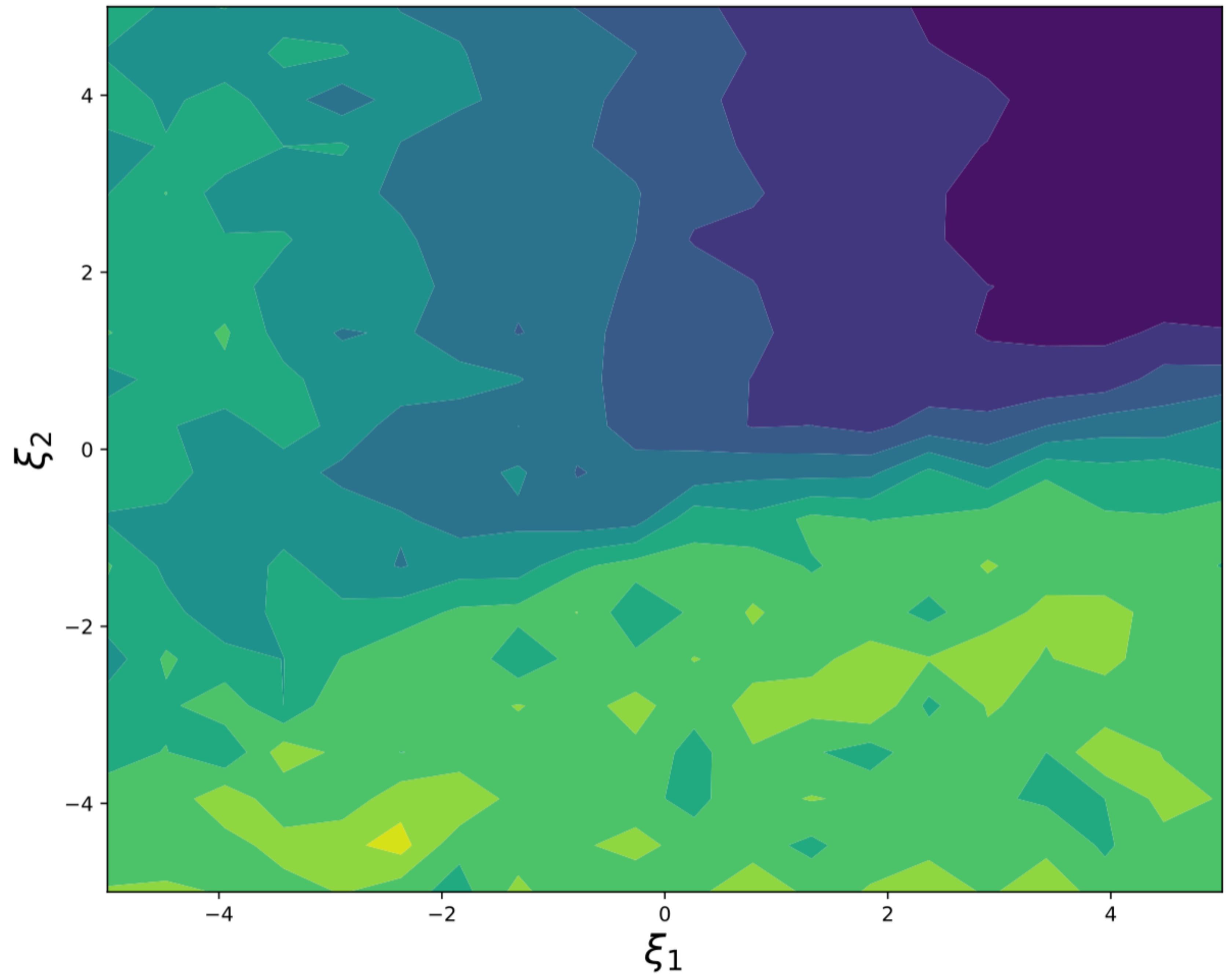
$$\mathbf{W}^{(t)} = \mathrm{diag}\{\underbrace{\xi_1^{(t)}, \ldots, \xi_1^{(t)}}_{d \text{ times}}, 0, -\xi_2^{(t)}\}, \text{ with } \xi_1^k, \xi_2^k > 0.$$

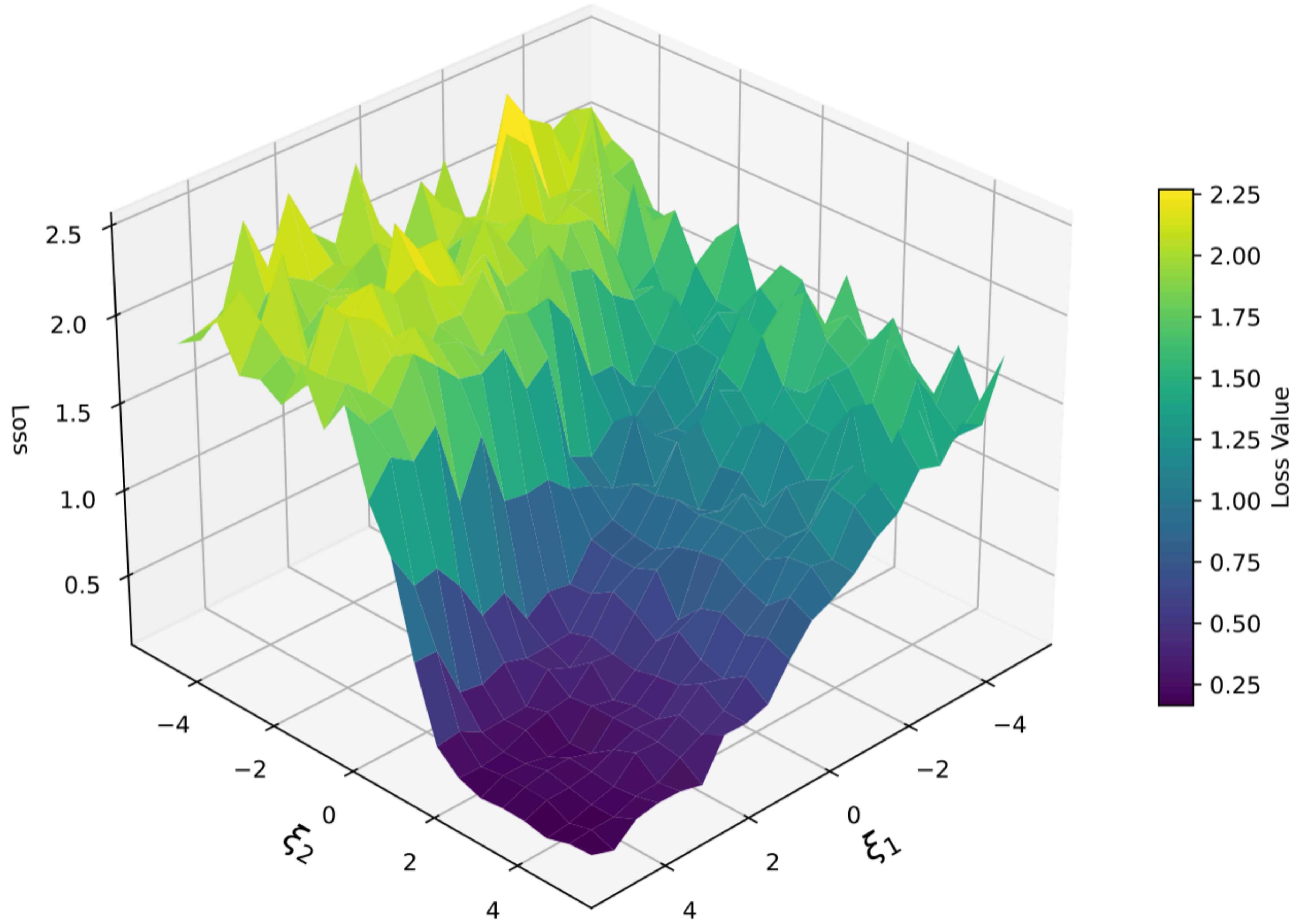
Proof sketch of in-context nearest neighbor predictor

Proposition.

Along the optimization path of gradient descent, the weights has the form

$$\mathbf{W}^{(t)} = \mathrm{diag}\{\underbrace{\xi_1^{(t)}, \ldots, \xi_1^{(t)}}_{d \text{ times}}, 0, -\xi_2^{(t)}\}, \text{ with } \xi_1^k, \xi_2^k > 0.$$





Proof sketch of in-context nearest neighbor predictor

$$\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(t)} \mathbf{H}) = \mathrm{softmax}\left(\begin{bmatrix} \xi_1^{(t)} \cdot \langle \mathbf{x}_1, \mathbf{x}_{\text{query}} \rangle \\ \xi_1^{(t)} \cdot \langle \mathbf{x}_2, \mathbf{x}_{\text{query}} \rangle \\ \vdots \\ \xi_1^{(t)} \cdot \langle \mathbf{x}_N, \mathbf{x}_{\text{query}} \rangle \\ -\xi_2^{(t)} \end{bmatrix} \right)$$

Proof sketch of in-context nearest neighbor predictor

$$\mathrm{softmax}(\mathbf{H}^\top \mathbf{W}^{(t)} \mathbf{H}) = \mathrm{softmax}\left(\begin{bmatrix} \xi_1^{(t)} \cdot \langle \mathbf{x}_1, \mathbf{x}_{\text{query}} \rangle \\ \xi_1^{(t)} \cdot \langle \mathbf{x}_2, \mathbf{x}_{\text{query}} \rangle \\ \vdots \\ \xi_1^{(t)} \cdot \langle \mathbf{x}_N, \mathbf{x}_{\text{query}} \rangle \\ -\xi_2^{(t)} \end{bmatrix}\right) \approx \mathrm{hardmax}\left(\begin{bmatrix} -\|\mathbf{x}_1 - \mathbf{x}_{\text{query}}\|_2^2 \\ -\|\mathbf{x}_2 - \mathbf{x}_{\text{query}}\|_2^2 \\ \vdots \\ -\|\mathbf{x}_N - \mathbf{x}_{\text{query}}\|_2^2 \\ -\infty \end{bmatrix}\right)$$

As $\xi_1^{(t)} \gg \xi_2^{(t)} \to +\infty$.

Downstream task for group-sparse classification

Consider a downstream task, where the data $\{(\tilde{\mathbf{X}}^{(i)}, \tilde{y}^{(i)})\}_{i=1}^{n}$ follow an arbitrary distribution satisfying (i) $\tilde{\mathbf{X}}$ is sub-Gaussian, and (ii) $\tilde{y} \cdot \langle \tilde{\mathbf{v}}^*, \tilde{\mathbf{x}}_{j^*} \rangle \geq \gamma$ almost surely.

Downstream task for group-sparse classification

Consider a downstream task, where the data $\{(\tilde{\mathbf{X}}^{(i)}, \tilde{y}^{(i)})\}_{i=1}^{n}$ follow an arbitrary distribution satisfying (i) $\tilde{\mathbf{X}}$ is sub-Gaussian, and (ii) $\tilde{y} \cdot \langle \tilde{\mathbf{v}}^*, \tilde{\mathbf{x}}_{j^*} \rangle \geq \gamma$ almost surely.

We only assume the label-relevant group index j^* to be the same as that in pre-training, while **the ground-truth linear vectors $\tilde{\mathbf{v}}^*$ and $\mathbf{v}^*$ can differ**.

Downstream task for group-sparse classification

Consider a downstream task, where the data $\{(\tilde{\mathbf{X}}^{(i)}, \tilde{y}^{(i)})\}_{i=1}^{n}$ follow an arbitrary distribution satisfying (i) $\tilde{\mathbf{X}}$ is sub-Gaussian, and (ii) $\tilde{y} \cdot \langle \tilde{\mathbf{v}}^*, \tilde{\mathbf{x}}_{j^*} \rangle \geq \gamma$ almost surely.

We only assume the label-relevant group index j^* to be the same as that in pre-training, while **the ground-truth linear vectors $\tilde{\mathbf{v}}^*$ and $\mathbf{v}^*$ can differ**.

Theorem. For any $\delta > 0$, under certain conditions, w.p. at least $1 - \delta$, the model fine-tuned with online SGD achieves:

$$\text{Test error} \leq O\left(\frac{(d + D)\log^2 n}{\gamma^2 n}\right) + \tilde{O}\left(\frac{\log(1/\delta)}{n}\right).$$

Downstream task for group-sparse classification

Consider a downstream task, where the data $\{(\tilde{\mathbf{X}}^{(i)}, \tilde{y}^{(i)})\}_{i=1}^{n}$ follow an arbitrary distribution satisfying (i) $\tilde{\mathbf{X}}$ is sub-Gaussian, and (ii) $\tilde{y} \cdot \langle \tilde{\mathbf{v}}^*, \tilde{\mathbf{x}}_{j^*} \rangle \geq \gamma$ almost surely.

We only assume the label-relevant group index j^* to be the same as that in pre-training, while **the ground-truth linear vectors $\tilde{\mathbf{v}}^*$ and $\mathbf{v}^*$ can differ**.

Theorem. For any $\delta > 0$, under certain conditions, w.p. at least $1 - \delta$, the model fine-tuned with online SGD achieves:

$$\text{Test error} \leq O\left(\frac{(d+D)\log^2 n}{\gamma^2 n}\right) + \tilde{O}\left(\frac{\log(1/\delta)}{n}\right).$$

Sample complexity: $\tilde{\Omega}((d+D)/\epsilon)$;

Downstream task for group-sparse classification

Consider a downstream task, where the data $\{(\tilde{\mathbf{X}}^{(i)}, \tilde{y}^{(i)})\}_{i=1}^{n}$ follow an arbitrary distribution satisfying (i) $\tilde{\mathbf{X}}$ is sub-Gaussian, and (ii) $\tilde{y} \cdot \langle \tilde{\mathbf{v}}^*, \tilde{\mathbf{x}}_{j^*} \rangle \geq \gamma$ almost surely.

We only assume the label-relevant group index j^* to be the same as that in pre-training, while **the ground-truth linear vectors $\tilde{\mathbf{v}}^*$ and $\mathbf{v}^*$ can differ**.

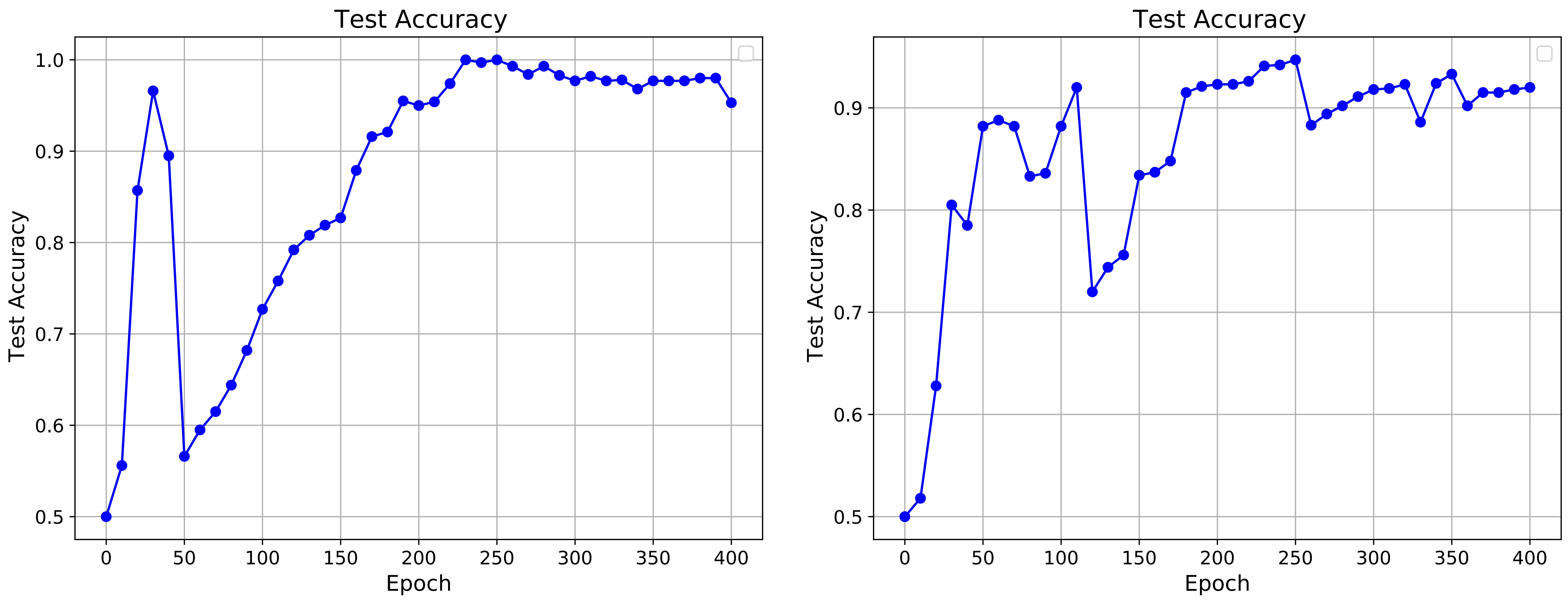
Theorem. For any $\delta > 0$, under certain conditions, w.p. at least $1 - \delta$, the model fine-tuned with online SGD achieves:

$$\text{Test error} \leq O\left(\frac{(d+D)\log^2 n}{\gamma^2 n}\right) + \tilde{O}\left(\frac{\log(1/\delta)}{n}\right).$$

Sample complexity: $\tilde{\Omega}((d+D)/\epsilon)$;

Sample complexity lower bound of linear logistic regression on $\mathrm{vec}(\tilde{\mathbf{X}})$ is $\tilde{\Omega}(dD/\epsilon)$.

Experiments - downstream task



Test accuracy in the downstream task when utilizing the pre-trained $\mathbf{W}^{(T^*)}$.