Transformers and Associative Memories

Alberto Bietti

Flatiron Institute, Simons Foundation

Computational Harmonic Analysis in Data Science and Machine Learning.
CMO-BIRS, Oaxaca, September 2024

Success of deep learning

$$f(x) = W_L \sigma(W_{L-1} \cdots \sigma(W_1 x) \cdots)$$

huge models $+$ **lots of data** $+$ **compute** $+$ **simple algorithms**

Success of deep learning

$$f(x) = W_L \sigma(W_{L-1} \cdots \sigma(W_1 x) \cdots)$$

huge models $+$ **lots of data** $+$ **compute** $+$ **simple algorithms**

Q: Why does it work?

Breaking the curse of dimensionality I: feature learning

Curse of dimensionality:

- Image/text/genomics/etc. data are **high-dimensional**: $x \in \mathbb{R}^d$, d large
- Curse of dimensionality $\implies$ need additional **structure** for learning

Breaking the curse of dimensionality I: feature learning

Curse of dimensionality:

- Image/text/genomics/etc. data are **high-dimensional**: $x \in \mathbb{R}^d$, d large
- Curse of dimensionality $\implies$ need additional **structure** for learning

Feature learning:

- **Single-index/multi-index** models:

$$\mathbb{E}[y|x] = f^*(w_1^\top x, \ldots, w_r^\top x), \qquad r \ll d$$

- Example: first layer of CNNs learns Gabor-like filters/features

Breaking the curse of dimensionality I: feature learning

Curse of dimensionality:

- Image/text/genomics/etc. data are **high-dimensional**: $x \in \mathbb{R}^d$, d large
- Curse of dimensionality $\implies$ need additional **structure** for learning

Feature learning:

- **Single-index/multi-index** models:

$$\mathbb{E}[y|x] = f^*(w_1^\top x, \dots, w_r^\top x), \qquad r \ll d$$

- Example: first layer of CNNs learns Gabor-like filters/features
- **Goal**: $O(n^{-1/r})$ instead of $O(n^{-1/d})$ rates (Bach, 2017)

Breaking the curse of dimensionality I: feature learning

Curse of dimensionality:

- Image/text/genomics/etc. data are **high-dimensional**: $x \in \mathbb{R}^d$, d large
- Curse of dimensionality $\implies$ need additional **structure** for learning

Feature learning:

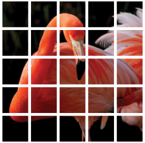
- **Single-index/multi-index** models:

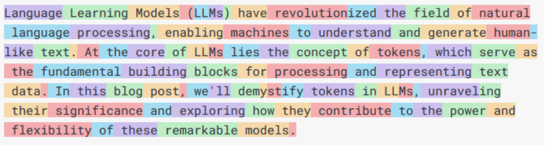
$$\mathbb{E}[y|x] = f^*(w_1^\top x, \dots, w_r^\top x), \qquad r \ll d$$

- Example: first layer of CNNs learns Gabor-like filters/features
- **Goal**: $O(n^{-1/r})$ instead of $O(n^{-1/d})$ rates (Bach, 2017)
- Gradient descent on **first layer** of shallow neural network can achieve this
  - Well-studied for Gaussian data, harmonic analysis over Hermite basis
  - (e.g., Ben Arous et al., 2021; Ba et al., 2022; B. et al., 2022; Damian et al., 2022; B. et al., 2023a)

Breaking the curse of dimensionality II: locality + architecture

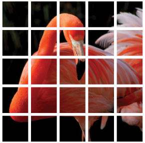
- **Local structure**: split input into small local patches / "tokens": $x = (x_1, \ldots, x_T)$

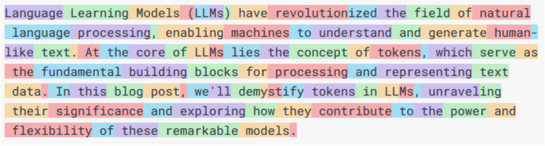




Breaking the curse of dimensionality II: locality + architecture

- **Local structure**: split input into small local patches / "tokens": $x = (x_1, \dots, x_T)$



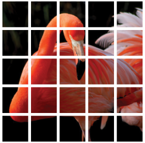


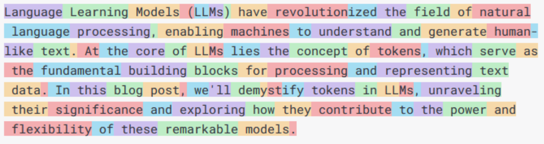
- Target may involve **interactions** between tokens, e.g. (Wahba, 1990)

$$\mathbb{E}[y|x] = \sum_i f_i^*(x_i) + \sum_{i,j} f_{ij}^*(x_i, x_j)$$

Breaking the curse of dimensionality II: locality + architecture

- **Local structure**: split input into small local patches / "tokens": $x = (x_1, \ldots, x_T)$





- Target may involve **interactions** between tokens, e.g. (Wahba, 1990)

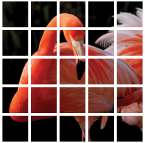
$$\mathbb{E}[y|x] = \sum_i f_i^*(x_i) + \sum_{i,j} f_{ij}^*(x_i, x_j)$$

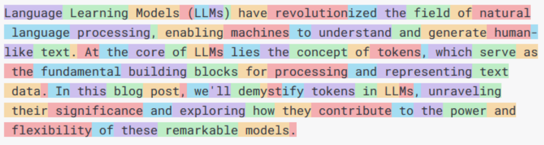
Role of architectures:

- "Feature learning" at intermediate layers $\leftrightarrow$ select relevant interactions (?)

Breaking the curse of dimensionality II: locality + architecture

- **Local structure**: split input into small local patches / "tokens": $x = (x_1, \ldots, x_T)$





- Target may involve **interactions** between tokens, e.g. (Wahba, 1990)

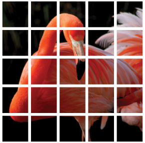
$$\mathbb{E}[y|x] = \sum_i f_i^*(x_i) + \sum_{i,j} f_{ij}^*(x_i, x_j)$$

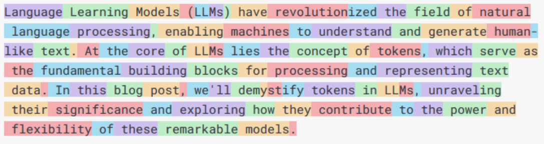
Role of architectures:

- "Feature learning" at intermediate layers $\leftrightarrow$ select relevant interactions (?)
- **Convolution**: local interactions at different scales

Breaking the curse of dimensionality II: locality + architecture

- **Local structure**: split input into small local patches / "tokens": $x = (x_1, \ldots, x_T)$



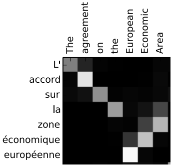


- Target may involve **interactions** between tokens, e.g. (Wahba, 1990)

$$\mathbb{E}[y|x] = \sum_i f_i^*(x_i) + \sum_{i,j} f_{ij}^*(x_i, x_j)$$

Role of architectures:

- "Feature learning" at intermediate layers $\leftrightarrow$ select relevant interactions (?)
- **Convolution**: local interactions at different scales
- **Attention**: non-local interactions



What are layers doing?

Embedding layer:

- map patches/tokens to embedding space
- capture **discrete/semantic features** encoded as different embedding directions (?)

What are layers doing?

Embedding layer:

- map patches/tokens to embedding space
- capture **discrete/semantic features** encoded as different embedding directions (?)

Unembedding (output) layer:

- map feature space back to tokens/labels

What are layers doing?

Embedding layer:

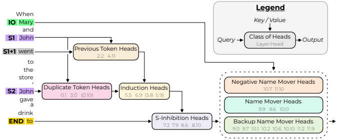
- map patches/tokens to embedding space
- capture **discrete/semantic features** encoded as different embedding directions (?)

Unembedding (output) layer:

- map feature space back to tokens/labels

What about intermediate layers?

- (discrete) communication/computation in feature space (??)
- $\Longrightarrow$ **associative** mappings (input-output, or across tokens)



(Wang et al., 2022)

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top$$

- We then have ${v_j}^\top W u_i \approx \alpha_{ij}$

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top$$

- We then have ${v_j}^\top W u_i \approx \alpha_{ij}$
  - e.g., classification, predict $\hat{f}(x) = \arg\max_j {v_j}^\top W u_x$

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top$$

- We then have ${v_j}^\top W u_i \approx \alpha_{ij}$
  - e.g., classification, predict $\hat{f}(x) = \arg\max_j {v_j}^\top W u_x$
- Related to Hopfield (1982); Kohonen (1972); Willshaw et al. (1969)

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top$$

- We then have ${v_j}^\top W u_i \approx \alpha_{ij}$
  - e.g., classification, predict $\hat{f}(x) = \arg\max_j {v_j}^\top W u_x$
- Related to Hopfield (1982); Kohonen (1972); Willshaw et al. (1969)

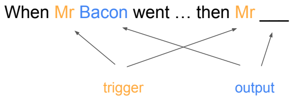
Q: how do these play a role in Transformers?

Outline

1. Application to Transformers (B., Cabannes, Bouchacourt, Jegou, and Bottou, 2023b)
2. Scaling laws for associative memories (Cabannes, Dohmatob, and B., 2024a)
3. Learning with gradient steps (B. et al., 2023b; Cabannes et al., 2024a,b)

The bigram data model for in-context reasoning

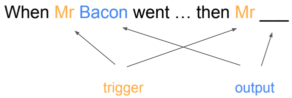
Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)



When Mr Bacon went to the mall, it started raining, then Mr Bacon decided to buy a raincoat and umbrella. He went to the store and bought a red raincoat and yellow polka dot umbrella.

The bigram data model for in-context reasoning

Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)

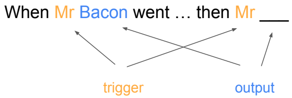


When Mr Bacon went to the mall, it started raining, then Mr Bacon decided to buy a raincoat and umbrella. He went to the store and bought a red raincoat and yellow polka dot umbrella.

Fix **trigger tokens**: $q_1, \ldots, q_K$

The bigram data model for in-context reasoning

Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)



When Mr Bacon went to the mall, it started raining, then Mr Bacon decided to buy a raincoat and umbrella. He went to the store and bought a red raincoat and yellow polka dot umbrella.

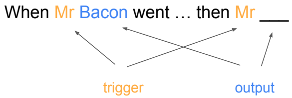
Fix **trigger tokens**: $q_1, \dots, q_K$

Sample each sequence $z_{1:T} \in [N]^T$ as follows

- **Output tokens**: $o_k \sim \pi_o(\cdot | q_k)$ (*random*)

The bigram data model for in-context reasoning

Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)



When Mr Bacon went to the mall, it started raining, then Mr Bacon decided to buy a raincoat and umbrella. He went to the store and bought a red raincoat and yellow polka dot umbrella.

Fix **trigger tokens**: $q_1, \ldots, q_K$

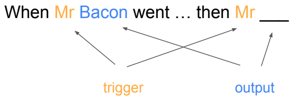
Sample each sequence $z_{1:T} \in [N]^T$ as follows

- **Output tokens**: $o_k \sim \pi_o(\cdot | q_k)$ (*random*)
- **Sequence-specific Markov model**: $z_1 \sim \pi_1$, $z_t | z_{t-1} \sim p(\cdot | z_{t-1})$ with

$$p(j|i) = \begin{cases} \mathbb{1}\{j = o_k\}, & \text{if } i = q_k, \quad k = 1, \ldots, K \\ \pi_b(j|i), & \text{o/w.} \end{cases}$$

The bigram data model for in-context reasoning

Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)



When Mr Bacon went to the mall, it started raining, then Mr Bacon decided to buy a raincoat and umbrella. He went to the store and bought a red raincoat and yellow polka dot umbrella.

Fix **trigger tokens**: $q_1, \dots, q_K$
Sample each sequence $z_{1:T} \in [N]^T$ as follows

- **Output tokens**: $o_k \sim \pi_o(\cdot|q_k)$ (*random*)
- **Sequence-specific Markov model**: $z_1 \sim \pi_1$, $z_t|z_{t-1} \sim p(\cdot|z_{t-1})$ with

$$p(j|i) = \begin{cases} \mathbb{1}\{j = o_k\}, & \text{if } i = q_k, \quad k = 1, \dots, K \\ \pi_b(j|i), & \text{o/w.} \end{cases}$$

π_b: **global bigrams** model (estimated from Karpathy's character-level Shakespeare)

Transformers I: embeddings and residual stream

- **Input sequence**: $[z_1, \dots, z_T] \in [N]^T$

Transformers I: embeddings and residual stream

- **Input sequence**: $[z_1, \ldots, z_T] \in [N]^T$
- **Embedding layer**:

$$x_t := w_E(z_t) + p_t \in \mathbb{R}^d$$

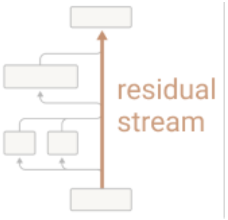
  - $w_E(z)$: token embedding of $z \in [N]$
  - p_t: positional embedding at position $t \in [T]$

Transformers I: embeddings and residual stream

- **Input sequence**: $[z_1, \ldots, z_T] \in [N]^T$
- **Embedding layer**:

$$x_t := w_E(z_t) + p_t \in \mathbb{R}^d$$

  - $w_E(z)$: token embedding of $z \in [N]$
  - p_t: positional embedding at position $t \in [T]$
- Intermediate layers: add new stuff to **residual stream** x_t
  - Repeat L times: **Attention** and **feed-forward** layers



Transformers I: embeddings and residual stream

- **Input sequence**: $[z_1, \dots, z_T] \in [N]^T$
- **Embedding layer**:

$$x_t := w_E(z_t) + p_t \in \mathbb{R}^d$$

  - $w_E(z)$: token embedding of $z \in [N]$
  - p_t: positional embedding at position $t \in [T]$
- Intermediate layers: add new stuff to **residual stream** x_t
  - Repeat L times: **Attention** and **feed-forward** layers
- **Unembedding layer**: logits for each $k \in [N]$,

$$(\xi_t)_k = w_U(k)^\top x_t$$

Transformers I: embeddings and residual stream

- **Input sequence**: $[z_1, \ldots, z_T] \in [N]^T$
- **Embedding layer**:

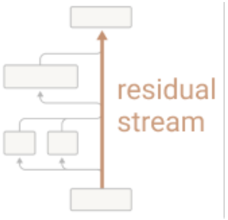
$$x_t := w_E(z_t) + p_t \in \mathbb{R}^d$$

  - $w_E(z)$: token embedding of $z \in [N]$
  - p_t: positional embedding at position $t \in [T]$
- Intermediate layers: add new stuff to **residual stream** x_t
  - Repeat L times: **Attention** and **feed-forward** layers
- **Unembedding layer**: logits for each $k \in [N]$,

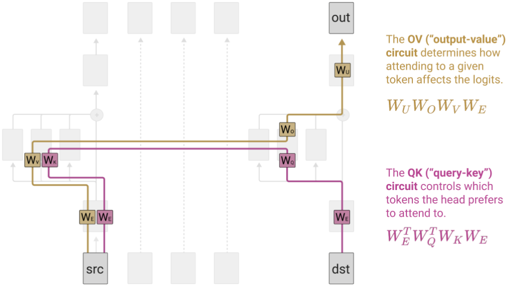
$$(\xi_t)_k = w_U(k)^\top x_t$$

- **Loss** for next-token prediction (ℓ: cross-entropy)

$$\sum_{t=1}^{T-1} \ell(z_{t+1}, \xi_t)$$



Transformers II: self-attention

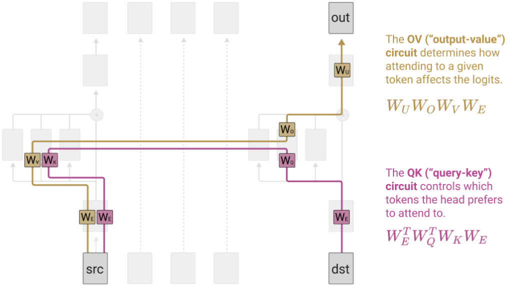


Causal self-attention layer (single head):

$$x'_t = \sum_{s=1}^{t} \beta_s W_O^T W_V x_s, \qquad \text{with } \beta_s = \frac{\exp(x_s{}^\top W_K^\top W_Q x_t)}{\sum_{s=1}^t \exp(x_s{}^\top W_K^\top W_Q x_t)}$$

- $W_K, W_Q \in \mathbb{R}^{d_h \times d}$: **key/query** matrices, $W_O, W_V \in \mathbb{R}^{d_h \times d}$: **output/value** matrices
- β_s: attention weights, $\sum_{s=1}^t \beta_s = 1$

Transformers II: self-attention

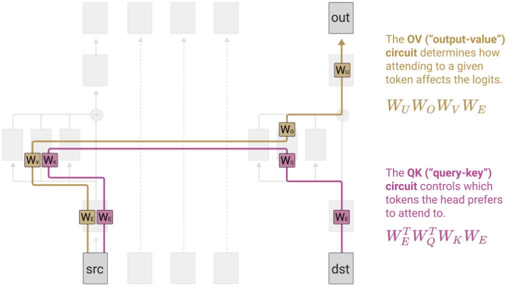


Causal self-attention layer (single head):

$$x'_t = \sum_{s=1}^{t} \beta_s W_{OV} x_s, \quad \text{with } \beta_s = \frac{\exp(x_s^\top W_{KQ} x_t)}{\sum_{s=1}^{t} \exp(x_s^\top W_{KQ} x_t)}$$

- $W_{KQ} \in \mathbb{R}^{d \times d}$: **key-query** matrix, $W_{OV} \in \mathbb{R}^{d \times d}$: **output-value** matrix
- β_s: attention weights, $\sum_{s=1}^{t} \beta_s = 1$

Transformers II: self-attention



Causal self-attention layer (single head):

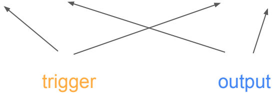
$$x'_t = \sum_{s=1}^{t} \beta_s W_{OV} x_s, \quad \text{with } \beta_s = \frac{\exp(x_s^\top W_{KQ} x_t)}{\sum_{s=1}^{t} \exp(x_s^\top W_{KQ} x_t)}$$

- $W_{KQ} \in \mathbb{R}^{d \times d}$: **key-query** matrix, $W_{OV} \in \mathbb{R}^{d \times d}$: **output-value** matrix
- β_s: attention weights, $\sum_{s=1}^{t} \beta_s = 1$
- Each x'_t is then added to the corresponding residual stream

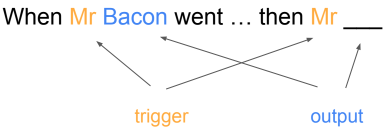
$$x_t := x_t + x'_t$$

Transformers on the bigram task

When Mr Bacon went … then Mr ___

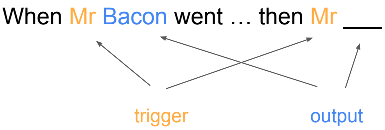


Transformers on the bigram task



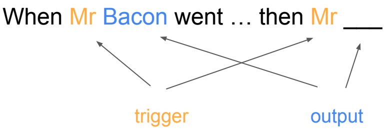
- **1-layer transformer fails**: $\sim 55\%$ accuracy on in-context output predictions

Transformers on the bigram task



- **1-layer transformer fails**: $\sim 55\%$ accuracy on in-context output predictions
- **2-layer transformer succeeds**: $\sim 99\%$ accuracy

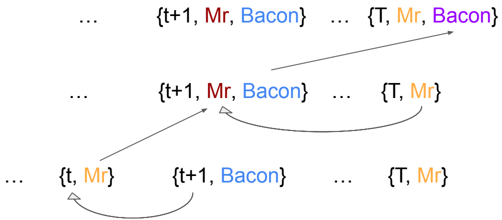
Transformers on the bigram task



- **1-layer transformer fails**: $\sim 55\%$ accuracy on in-context output predictions
- **2-layer transformer succeeds**: $\sim 99\%$ accuracy

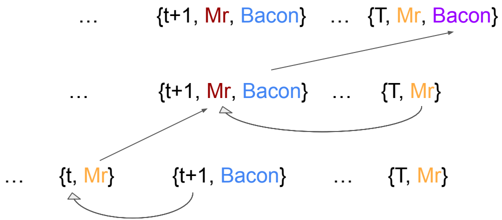
See also (Sanford, Hsu, and Telgarsky, 2023, 2024) for representational lower bounds

Induction head mechanism (Elhage et al., 2021; Olsson et al., 2022)



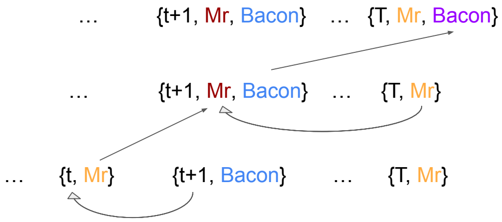
- 1st layer: **previous-token head**
  - attends to previous token and copies it to residual stream

Induction head mechanism (Elhage et al., 2021; Olsson et al., 2022)

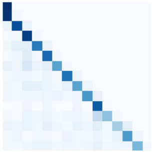


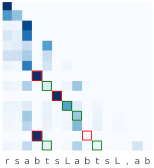
- 1st layer: **previous-token head**
  - attends to previous token and copies it to residual stream
- 2nd layer: **induction head**
  - attends to output of previous token head, copies attended token

Induction head mechanism (Elhage et al., 2021; Olsson et al., 2022)



- 1st layer: **previous-token head**
  - attends to previous token and copies it to residual stream
- 2nd layer: **induction head**
  - attends to output of previous token head, copies attended token
- Matches observed attention scores:





Random embeddings in high dimension

- We consider **random** embeddings u_i with i.i.d. $N(0, 1/d)$ entries and d large

$$\|u_i\| \approx 1 \quad \text{and} \quad u_i^\top u_j = O(1/\sqrt{d})$$

Random embeddings in high dimension

- We consider **random** embeddings u_i with i.i.d. $N(0, 1/d)$ entries and d large

$$\|u_i\| \approx 1 \quad \text{and} \quad u_i^\top u_j = O(1/\sqrt{d})$$

- **Remapping**: multiply by random matrix W with $\mathcal{N}(0, 1/d)$ entries:

$$\|Wu_i\| \approx 1 \quad \text{and} \quad u_i^\top W u_i = O(1/\sqrt{d})$$

Random embeddings in high dimension

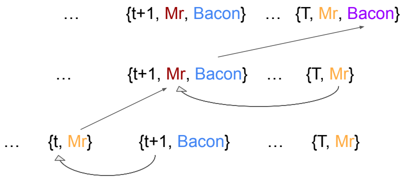
- We consider **random** embeddings u_i with i.i.d. $N(0, 1/d)$ entries and d large

$$\|u_i\| \approx 1 \quad \text{and} \quad u_i^\top u_j = O(1/\sqrt{d})$$

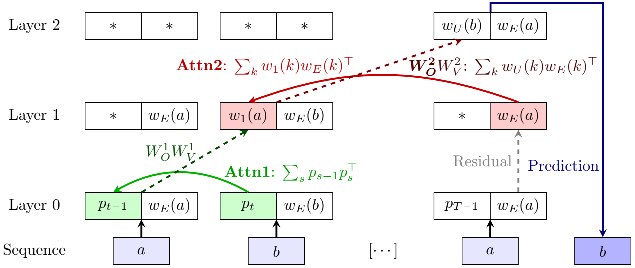
- **Remapping**: multiply by random matrix W with $\mathcal{N}(0, 1/d)$ entries:

$$\|Wu_i\| \approx 1 \quad \text{and} \quad u_i^\top W u_i = O(1/\sqrt{d})$$

- Value/Output matrices help with token remapping: Mr $\mapsto$ Mr, Bacon $\mapsto$ Bacon



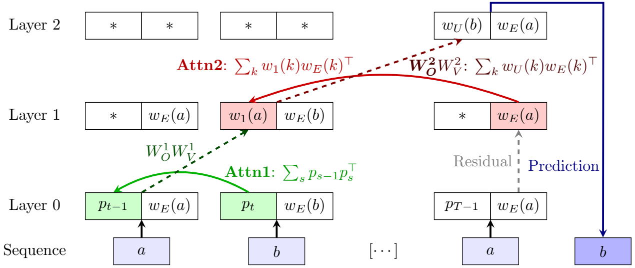
Induction head with associative memories



$$W_{KQ}^1 = \sum_{t=2}^{T} p_t p_{t-1}^\top, \quad W_{KQ}^2 = \sum_{k \in Q} w_E(k) w_1(k)^\top, \quad W_{OV}^2 = \sum_{k=1}^{N} w_U(k) w_E(k)^\top,$$

- Random embeddings $w_E(k)$, $w_U(k)$, random matrix W_{OV}^1 (frozen at init)
- **Remapped** previous tokens: $w_1(k) := W_{OV}^1 w_E(k)$

Induction head with associative memories



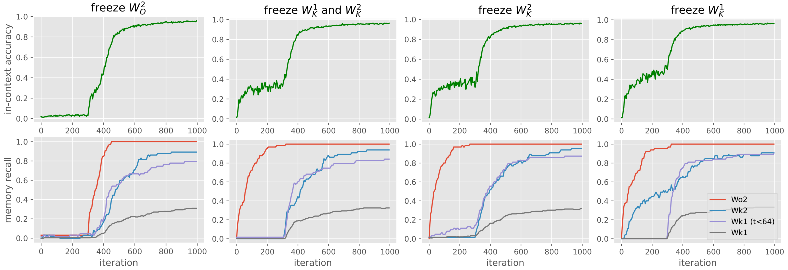
$$W_{KQ}^1 = \sum_{t=2}^{T} p_t p_{t-1}^\top, \quad W_{KQ}^2 = \sum_{k \in Q} w_E(k) w_1(k)^\top, \quad W_{OV}^2 = \sum_{k=1}^{N} w_U(k) w_E(k)^\top,$$

- Random embeddings $w_E(k)$, $w_U(k)$, random matrix W_{OV}^1 (frozen at init)
- **Remapped** previous tokens: $w_1(k) := W_{OV}^1 w_E(k)$

Q: Does this match practice?

Empirically probing the dynamics

Train only W_K^1, W_K^2, W_O^2, loss on deterministic output tokens only

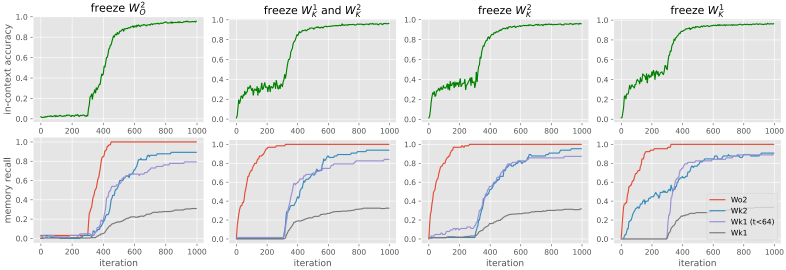


- "Memory recall **probes**": for target memory $W_* = \sum_{(i,j)\in\mathcal{M}} v_j u_i^\top$, compute

$$R(\hat{W}, W_*) = \frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \mathbb{1}\{j = \arg\max_{j'} v_{j'}^\top \hat{W} u_i\}$$

Empirically probing the dynamics

Train only W_K^1, W_K^2, W_O^2, loss on deterministic output tokens only



- "Memory recall **probes**": for target memory $W_* = \sum_{(i,j)\in\mathcal{M}} v_j u_i^\top$, compute

$$R(\hat{W}, W_*) = \frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \mathbb{1}\{j = \arg\max_{j'} v_{j'}^\top \hat{W} u_i\}$$

- Natural learning "**order**": W_{OV}^2 first, W_{KQ}^2 next, W_{KQ}^1 last
- Joint learning is faster (motivates our theory below)

Outline

Scaling laws (a.k.a. non-parametric rates)

Statistical setup

- Non-parametric setting: what if there are $N \to \infty$ elements?

Scaling laws (a.k.a. non-parametric rates)

Statistical setup

- Non-parametric setting: what if there are $N \to \infty$ elements?
- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \ldots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

Scaling laws (a.k.a. non-parametric rates)

Statistical setup

- Non-parametric setting: what if there are $N \to \infty$ elements?
- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \dots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

- Heavy-tailed token frequencies: **Zipf law** (typical for language where N is very large)

$$p(z) \propto z^{-\alpha}$$

Scaling laws (a.k.a. non-parametric rates)

Statistical setup

- Non-parametric setting: what if there are $N \to \infty$ elements?
- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \dots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

- Heavy-tailed token frequencies: **Zipf law** (typical for language where N is very large)

$$p(z) \propto z^{-\alpha}$$

- Hutter (2021): with infinite memory, we have

$$L(\hat{f}_n) \lesssim n^{-\frac{\alpha-1}{\alpha}}$$

Scaling laws (a.k.a. non-parametric rates)

Statistical setup

- Non-parametric setting: what if there are $N \to \infty$ elements?
- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \dots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

- Heavy-tailed token frequencies: **Zipf law** (typical for language where N is very large)

$$p(z) \propto z^{-\alpha}$$

- Hutter (2021): with infinite memory, we have

$$L(\hat{f}_n) \lesssim n^{-\frac{\alpha-1}{\alpha}}$$

Q: What about finite capacity?

How much can we store with finite d? Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$

How much can we store with finite d? Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- Consider

$$W = \sum_{z=1}^{N} v_{f^*(z)} u_z^\top \in \mathbb{R}^{d \times d}$$

How much can we store with finite d? Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- Consider

$$W = \sum_{z=1}^{N} v_{f^*(z)} u_z^\top \in \mathbb{R}^{d \times d}$$

- When can we recover $f^*(z)$ from z for all z?

$$\gamma_{z,y} := v_y^\top W u_z = \sum_{z'} v_y^\top v_{f^*(z')} u_z^\top u_{z'}$$

How much can we store with finite d? Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- Consider

$$W = \sum_{z=1}^{N} v_{f^*(z)} {u_z}^\top \in \mathbb{R}^{d \times d}$$

- When can we recover $f^*(z)$ from z for all z?

$$\gamma_{z,y} := {v_y}^\top W u_z = \sum_{z'} v_y^\top v_{f^*(z')} u_z^\top u_{z'}$$

We have

$$\mathbb{E}[\gamma_{z,y}] = \begin{cases} 1, & \text{if } y = f^*(z) \\ 0, & \text{otherwise.} \end{cases}$$

$$\mathrm{Var}[\gamma_{z,y}] \lesssim \frac{|\{z' : f^*(z') = y\}|}{d} + \frac{|\{z' : f^*(z') \neq y\}|}{d^2}$$

How much can we store with finite d? Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- Consider

$$W = \sum_{z=1}^{N} v_{f^*(z)} u_z^\top \in \mathbb{R}^{d \times d}$$

- When can we recover $f^*(z)$ from z for all z?

$$\gamma_{z,y} := v_y^\top W u_z = \sum_{z'} v_y^\top v_{f^*(z')} u_z^\top u_{z'}$$

We have

$$\mathbb{E}[\gamma_{z,y}] = \begin{cases} 1, & \text{if } y = f^*(z) \\ 0, & \text{otherwise.} \end{cases}$$

$$\mathrm{Var}[\gamma_{z,y}] \lesssim \frac{|\{z' : f^*(z') = y\}|}{d} + \frac{|\{z' : f^*(z') \neq y\}|}{d^2}$$

- Recover all associations when $\mathrm{Var}[\gamma_{z,y}] \lesssim 1$. Examples:
  - $f^*(z) = z$: can store up to $N \approx d^2$ associations
  - $f^*(z) = z \mod 2$: can store up to $N \approx d$ associations

Scaling laws with finite capacity

- **Random embeddings** $u_z, v_y \in \mathbb{R}^d$ with i.i.d. $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Scaling laws with finite capacity

- **Random embeddings** $u_z, v_y \in \mathbb{R}^d$ with i.i.d. $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Theorem (Cabannes, Dohmatob, B., 2024, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$

Scaling laws with finite capacity

- **Random embeddings** $u_z, v_y \in \mathbb{R}^d$ with i.i.d. $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Theorem (Cabannes, Dohmatob, B., 2024, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. *For* $q(z) = \mathbb{1}\{z \in S_n\}$, *and* $d \gg N$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ *for any* k

Scaling laws with finite capacity

- **Random embeddings** $u_z, v_y \in \mathbb{R}^d$ with i.i.d. $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Theorem (Cabannes, Dohmatob, B., 2024, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. *For* $q(z) = \mathbb{1}\{z \in S_n\}$, *and* $d \gg N$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ *for any* k
3. *For* $q(z) = \mathbb{1}\{z \text{ seen at least } s \text{ times in } S_n\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\alpha+1}$

Scaling laws with finite capacity

- **Random embeddings** $u_z, v_y \in \mathbb{R}^d$ with i.i.d. $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Theorem (Cabannes, Dohmatob, B., 2024, informal)

1. For $q(z) = \sum_i \mathbb{1}\{z = z_i\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. For $q(z) = \mathbb{1}\{z \in S_n\}$, and $d \gg N$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ for any k
3. For $q(z) = \mathbb{1}\{z \text{ seen at least } s \text{ times in } S_n\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\alpha+1}$

- $n^{-\frac{\alpha-1}{\alpha}}$ is the same as (Hutter, 2021)
- $q = 1$ is best if we have enough capacity
- Can store at most d memories (approximation error: $d^{-\alpha+1}$)

Scaling laws with finite capacity

- **Random embeddings** $u_z, v_y \in \mathbb{R}^d$ with i.i.d. $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Theorem (Cabannes, Dohmatob, B., 2024, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. *For* $q(z) = \mathbb{1}\{z \in S_n\}$, *and* $d \gg N$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ *for any* k
3. *For* $q(z) = \mathbb{1}\{z \text{ seen at least } s \text{ times in } S_n\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\alpha+1}$

- $n^{-\frac{\alpha-1}{\alpha}}$ is the same as (Hutter, 2021)
- $q = 1$ is best if we have enough capacity
- Can store at most d memories (approximation error: $d^{-\alpha+1}$)
- Extensions to md memories for MLPs with md parameters (w/ E. Nichani, J. Lee)

Outline

Gradients as associative memories

- **Simple model** to learn associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

Gradients as associative memories

- **Simple model** to learn associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthonormal input/output embeddings, assume fixed

Gradients as associative memories

- **Simple model** to learn associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthonormal input/output embeddings, assume fixed
- **Cross-entropy loss** for logits $\xi \in \mathbb{R}^M$: $\ell(y, \xi) = -\xi_y + \log(\sum_k \exp \xi_k)$

Gradients as associative memories

- **Simple model** to learn associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to (v_k{}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthonormal input/output embeddings, assume fixed
- **Cross-entropy loss** for logits $\xi \in \mathbb{R}^M$: $\ell(y, \xi) = -\xi_y + \log(\sum_k \exp \xi_k)$

Example: one gradient step

- **Data model**: $z \sim \mathrm{Unif}([N]), \quad y = f_*(z) \in [N]$
- After **one gradient step** from $W_0 = 0$, step-size η:

$$v_k{}^\top W_1 u_z \approx \frac{\eta}{N} \mathbb{1}\{f_*(z) = k\} + O\left(\frac{\eta}{N^2}\right)$$

Gradients as associative memories

- **Simple model** to learn associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthonormal input/output embeddings, assume fixed
- **Cross-entropy loss** for logits $\xi \in \mathbb{R}^M$: $\ell(y, \xi) = -\xi_y + \log(\sum_k \exp \xi_k)$

Example: one gradient step

- **Data model**: $z \sim \text{Unif}([N]), \quad y = f_*(z) \in [N]$
- After **one gradient step** from $W_0 = 0$, step-size η:

$${v_k}^\top W_1 u_z \approx \frac{\eta}{N} \mathbb{1}\{f_*(z) = k\} + O\left(\frac{\eta}{N^2}\right)$$

Note: related to (Ba et al., 2022; Damian et al., 2022; Yang and Hu, 2021)

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \dots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \ldots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections
- Some elements may be irrelevant for prediction

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \dots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections
- Some elements may be irrelevant for prediction

Example: filter out exogenous noise with one gradient step

- **Data model**: $y \sim \mathrm{Unif}([N]), \quad t \sim \mathrm{Unif}([T]), \quad x = u_y + n_t \in \mathbb{R}^d$
- After **one gradient step** from $W_0 = 0$

$$v_k{}^\top W_1 x \approx \frac{\eta}{N} \mathbb{1}\{k = y\} + O\left(\frac{\eta}{N^2}\right)$$

Gradient steps for the bigram task

Setting: transformer on the bigram task

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture
- Infinite width, infinite data, $N \gg T$

Gradient steps for the bigram task

Setting: transformer on the bigram task

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture
- Infinite width, infinite data, $N \gg T$

Theorem (B. et al., 2023b, informal)

In the setup above, we can recover the desired associative memories with ***3 gradient steps*** *on the population loss: first on* W_{OV}^2*, then* W_{KQ}^2*, then* W_{KQ}^1.

Gradient steps for the bigram task

Setting: transformer on the bigram task

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture
- Infinite width, infinite data, $N \gg T$

Theorem (B. et al., 2023b, informal)

*In the setup above, we can recover the desired associative memories with **3 gradient steps** on the population loss: first on W_{OV}^2, then W_{KQ}^2, then W_{KQ}^1.*

Key ideas

- Attention is uniform at initialization $\implies$ inputs are sums of embeddings
- W_O^2: correct output appears w.p. 1, while other tokens are noisy and cond. indep. of z_T
- $W_K^{1/2}$: correct associations lead to more focused attention

Imbalanced data, finite capacity

$$L(W) = \mathbb{E}_{z \sim p}[\ell(f^*(z), VWu_z)], \qquad \ell\text{: cross-entropy loss}$$

Imbalanced data, finite capacity

$$L(W) = \mathbb{E}_{z \sim p}[\ell(f^*(z), VWu_z)], \qquad \ell\text{: cross-entropy loss}$$

Optimization improves capacity with imbalance: (Cabannes, Dohmatob, and B., 2024a)

- One population gradient step is similar to $q(z) \approx p(z)$: **inefficient** for power law
- Many steps lead to $q(z) \approx 1$: **more efficient** (for $d \geq N$)

Imbalanced data, finite capacity

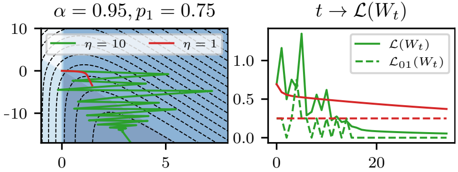
$$L(W) = \mathbb{E}_{z\sim p}[\ell(f^*(z), VWu_z)], \qquad \ell\text{: cross-entropy loss}$$

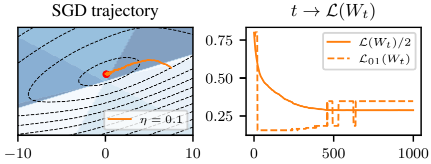
Optimization improves capacity with imbalance: (Cabannes, Dohmatob, and B., 2024a)

- One population gradient step is similar to $q(z) \approx p(z)$: **inefficient** for power law
- Many steps lead to $q(z) \approx 1$: **more efficient** (for $d \geq N$)

Benefits of large step-sizes + oscillations: (Cabannes, Simsek, and B., 2024b)

- Orthogonal embeddings $\implies$ logarithmic growth of margins for any step-size
- Correlated embeddings + imbalance $\implies$ oscillatory regimes
- Large step-sizes help reach perfect accuracy faster despite oscillations (empirically)
- Over-optimization can hurt in under-parameterized settings (empirically)





Discussion

Summary

- Weights of intermediate layers as associative memories
- Bigram model: simple model on discrete data for in-context reasoning
- Learning mechanisms via few gradient steps
- Statistical and optimization results

Discussion

Summary

- Weights of intermediate layers as associative memories
- Bigram model: simple model on discrete data for in-context reasoning
- Learning mechanisms via few gradient steps
- Statistical and optimization results

Future directions

- More complex “reasoning” tasks
- Learning embeddings
- Finer grained analysis of optimization dynamics and scaling laws
- Beyond text data: images and scientific data
- Applications: interpretability, model editing, factual recall, efficient fine-tuning

Discussion

Summary

- Weights of intermediate layers as associative memories
- Bigram model: simple model on discrete data for in-context reasoning
- Learning mechanisms via few gradient steps
- Statistical and optimization results

Future directions

- More complex "reasoning" tasks
- Learning embeddings
- Finer grained analysis of optimization dynamics and scaling laws
- Beyond text data: images and scientific data
- Applications: interpretability, model editing, factual recall, efficient fine-tuning

Thank you!

References I

A. B., J. Bruna, C. Sanford, and M. J. Song. Learning single-index models with shallow neural networks. *Advances in Neural Information Processing Systems*, 2022.

A. B., J. Bruna, and L. Pillaud-Vivien. On learning gaussian multi-index models with gradient flow. *arXiv preprint arXiv:2310.19793*, 2023a.

A. B., V. Cabannes, D. Bouchacourt, H. Jegou, and L. Bottou. Birth of a transformer: A memory viewpoint. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

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

F. Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research (JMLR)*, 18(19):1–53, 2017.

G. Ben Arous, R. Gheissari, and A. Jagannath. Online stochastic gradient descent on non-convex losses from high-dimensional inference. *Journal of Machine Learning Research (JMLR)*, 2021.

V. Cabannes, E. Dohmatob, and A. B. Scaling laws for associative memories. In *International Conference on Learning Representations (ICLR)*, 2024a.

V. Cabannes, B. Simsek, and A. B. Learning associative memories with gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024b.

References II

L. Chizat and F. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

A. Damian, J. Lee, and M. Soltanolkotabi. Neural networks can learn representations with gradient descent. In *Conference on Learning Theory (COLT)*, 2022.

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

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

M. Hutter. Learning curve theory. *arXiv preprint arXiv:2102.04074*, 2021.

T. Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, 1972.

S. Mei, T. Misiakiewicz, and A. Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory (COLT)*, 2019.

References III

E. Nichani, A. Damian, and J. D. Lee. Provable guarantees for nonlinear feature learning in three-layer neural networks. *arXiv preprint arXiv:2305.06986*, 2023.

C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, S. Johnston, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

C. Sanford, D. Hsu, and M. Telgarsky. Representational strengths and limitations of transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

C. Sanford, D. Hsu, and M. Telgarsky. One-layer transformers fail to solve the induction heads task. *arXiv preprint arXiv:2408.14332*, 2024.

G. Wahba. *Spline models for observational data*, volume 59. Siam, 1990.

K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222(5197):960–962, 1969.

G. Yang and E. J. Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials
- In practice, **discrete structure** and **memorization** are often crucial
  - language: words, syntactic rules, semantic concepts, facts
  - vision: “visual words”, features, objects

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials
- In practice, **discrete structure** and **memorization** are often crucial
  - language: words, syntactic rules, semantic concepts, facts
  - vision: "visual words", features, objects

Statistical learning setup:

- Data distribution $p(z, y)$ over pairs of **discrete tokens** $(z, y) \in [N] \times [M]$

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials
- In practice, **discrete structure** and **memorization** are often crucial
  - language: words, syntactic rules, semantic concepts, facts
  - vision: "visual words", features, objects

Statistical learning setup:

- Data distribution $p(z, y)$ over pairs of **discrete tokens** $(z, y) \in [N] \times [M]$
- We want a predictor $\hat{f} : [N] \to [M]$ with **small 0-1 loss**:

$$L_{01}(\hat{f}) = \mathbb{P}(y \neq \hat{f}(z))$$

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials
- In practice, **discrete structure** and **memorization** are often crucial
  - language: words, syntactic rules, semantic concepts, facts
  - vision: “visual words”, features, objects

Statistical learning setup:

- Data distribution $p(z, y)$ over pairs of **discrete tokens** $(z, y) \in [N] \times [M]$
- We want a predictor $\hat{f} : [N] \to [M]$ with **small 0-1 loss**:

$$L_{01}(\hat{f}) = \mathbb{P}(y \neq \hat{f}(z))$$

- Typically $\hat{f}(z) = \arg\max_y f_y(z)$ with $f_y : [N] \to \mathbb{R}$ for each $y \in [M]$

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i\in\mathcal{I}}$ and $\{v_j\}_{j\in\mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top$$

- We then have ${v_j}^\top W u_i \approx \alpha_{ij}$

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad u_i^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad v_i^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j u_i^\top$$

- We then have $v_j^\top W u_i \approx \alpha_{ij}$
- Computed in Transformers for logits in next-token prediction and self-attention

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top$$

- We then have ${v_j}^\top W u_i \approx \alpha_{ij}$
- Computed in Transformers for logits in next-token prediction and self-attention

note: closely related to Hopfield (1982); Kohonen (1972); Willshaw et al. (1969)

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthogonal input/output embeddings, assume fixed

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthogonal input/output embeddings, assume fixed
- **Cross-entropy loss** for logits $\xi \in \mathbb{R}^M$: $\ell(y, \xi) = -\xi_y + \log(\sum_k \exp \xi_k)$

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthogonal input/output embeddings, assume fixed
- **Cross-entropy loss** for logits $\xi \in \mathbb{R}^M$: $\ell(y, \xi) = -\xi_y + \log(\sum_k \exp \xi_k)$

Lemma (Gradients as memories)

Let p be a data distribution over $(z, y) \in [N] \times [M]$, and consider the loss

$$L(W) = \mathbb{E}_{(z,y)\sim p}[\ell(y, \xi_W(z))], \quad \xi_W(z)_k = {v_k}^\top W u_z,$$

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings.

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthogonal input/output embeddings, assume fixed
- **Cross-entropy loss** for logits $\xi \in \mathbb{R}^M$: $\ell(y, \xi) = -\xi_y + \log(\sum_k \exp \xi_k)$

Lemma (Gradients as memories)

Let p be a data distribution over $(z, y) \in [N] \times [M]$, and consider the loss

$$L(W) = \mathbb{E}_{(z,y)\sim p}[\ell(y, \xi_W(z))], \quad \xi_W(z)_k = {v_k}^\top W u_z,$$

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings. Then,

$$\nabla L(W) = \sum_{k=1}^{M} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k {u_z}^\top],$$

with $\hat{p}_W(y = k|z) = \exp(\xi_W(z)_k) / \sum_j \exp(\xi_W(z)_j)$.

Example: one gradient step

Data model: $z \sim \mathrm{Unif}([N]), \quad y = f_*(z) \in [N]$

Example: one gradient step

Data model: $z \sim \mathrm{Unif}([N]), \quad y = f_*(z) \in [N]$

- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^{N} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k u_z{}^\top]$$

$$= \eta \sum_{z,k} p(z)(p(y = k|z) - \hat{p}_W(y = k|z)) v_k u_z{}^\top$$

$$= \frac{\eta}{N} \sum_{z,k} (\mathbb{1}\{k = f^*(z)\} - \frac{1}{N}) v_k u_z{}^\top$$

Example: one gradient step

Data model: $z \sim \text{Unif}([N]), \quad y = f_*(z) \in [N]$

- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^{N} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k {u_z}^\top]$$
$$= \eta \sum_{z,k} p(z)(p(y = k|z) - \hat{p}_W(y = k|z)) v_k {u_z}^\top$$
$$= \frac{\eta}{N} \sum_{z,k} (\mathbb{1}\{k = f^*(z)\} - \frac{1}{N}) v_k {u_z}^\top$$

- Then, for any (z, k) we have

$$ {v_k}^\top W_1 u_z \approx \frac{\eta}{N} \mathbb{1}\{f_*(z) = k\} + O\left(\frac{\eta}{N^2}\right)$$

Example: one gradient step

Data model: $z \sim \mathrm{Unif}([N]), \quad y = f_*(z) \in [N]$

- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$
\begin{aligned}
W_1 &= W_0 - \eta \sum_{k=1}^{N} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k u_z{}^\top] \\
&= \eta \sum_{z,k} p(z)(p(y = k|z) - \hat{p}_W(y = k|z)) v_k u_z{}^\top \\
&= \frac{\eta}{N} \sum_{z,k} (\mathbb{1}\{k = f^*(z)\} - \frac{1}{N}) v_k u_z{}^\top
\end{aligned}
$$

- Then, for any (z, k) we have

$$
v_k{}^\top W_1 u_z \approx \frac{\eta}{N} \mathbb{1}\{f_*(z) = k\} + O\left(\frac{\eta}{N^2}\right)
$$

- **Corollary**: $\hat{f}(z) = \arg\max_k v_k{}^\top W_1 u_z$ has near-perfect accuracy

Example: one gradient step

Data model: $z \sim \text{Unif}([N]), \quad y = f_*(z) \in [N]$

- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^{N} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k u_z^\top]$$
$$= \eta \sum_{z,k} p(z)(p(y = k|z) - \hat{p}_W(y = k|z)) v_k u_z^\top$$
$$= \frac{\eta}{N} \sum_{z,k} (\mathbb{1}\{k = f^*(z)\} - \frac{1}{N}) v_k u_z^\top$$

- Then, for any (z, k) we have

$$v_k^\top W_1 u_z \approx \frac{\eta}{N} \mathbb{1}\{f_*(z) = k\} + O\left(\frac{\eta}{N^2}\right)$$

- **Corollary**: $\hat{f}(z) = \arg\max_k v_k^\top W_1 u_z$ has near-perfect accuracy

Note: related to (Ba et al., 2022; Damian et al., 2022; Yang and Hu, 2021)

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \ldots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \dots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections
- Some elements may be irrelevant for prediction

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \dots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections
- Some elements may be irrelevant for prediction

Lemma (Gradients with noisy inputs)

Let p be a data distribution over $(x, y) \in \mathbb{R}^d \times [N]$, and consider the loss

$$L(W) = \mathbb{E}_{(x,y)\sim p}[\ell(y, \xi_W(x))], \quad \xi_W(x)_k = {v_k}^\top W x.$$

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \dots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections
- Some elements may be irrelevant for prediction

Lemma (Gradients with noisy inputs)

Let p be a data distribution over $(x, y) \in \mathbb{R}^d \times [N]$, and consider the loss

$$L(W) = \mathbb{E}_{(x,y)\sim p}[\ell(y, \xi_W(x))], \quad \xi_W(x)_k = {v_k}^\top W x.$$

Denoting $\mu_k := \mathbb{E}[x|y = k]$ and $\hat{\mu}_k := \mathbb{E}_x[\frac{\hat{p}_W(k|x)}{p(y=k)} x]$, we have

$$\nabla_W L(W) = \sum_{k=1}^{N} p(y = k) v_k (\hat{\mu}_k - \mu_k)^\top.$$

Example: filter out exogenous noise

- **Data model**: $y \sim \mathrm{Unif}([N]), \quad t \sim \mathrm{Unif}([T]), \quad x = u_y + n_t \in \mathbb{R}^d$
  - where $\{n_t\}_{t=1}^T$ are another collection of embeddings, *e.g.*, positional embeddings

Example: filter out exogenous noise

- **Data model**: $y \sim \mathrm{Unif}([N]), \quad t \sim \mathrm{Unif}([T]), \quad x = u_y + n_t \in \mathbb{R}^d$
  - where $\{n_t\}_{t=1}^T$ are another collection of embeddings, *e.g.*, positional embeddings
- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$\begin{aligned} W_1 &= W_0 - \eta \sum_{k=1}^N p(y=k) v_k (\hat{\mu}_k - \mu_k)^\top \\ &= \frac{\eta}{N} \sum_{k=1}^N v_k (\mathbb{E}[u_y + n_t | y = k] - \mathbb{E}[u_y + n_t])^\top \\ &= \frac{\eta}{N} \sum_{k=1}^N v_k u_k^\top - \frac{\eta}{N^2} \sum_{k,j} v_k u_j^\top \end{aligned}$$

Example: filter out exogenous noise

- **Data model**: $y \sim \mathrm{Unif}([N])$, $t \sim \mathrm{Unif}([T])$, $x = u_y + n_t \in \mathbb{R}^d$
  - where $\{n_t\}_{t=1}^T$ are another collection of embeddings, *e.g.*, positional embeddings
- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^N p(y=k) v_k (\hat{\mu}_k - \mu_k)^\top$$
$$= \frac{\eta}{N} \sum_{k=1}^N v_k (\mathbb{E}[u_y + n_t | y = k] - \mathbb{E}[u_y + n_t])^\top$$
$$= \frac{\eta}{N} \sum_{k=1}^N v_k u_k^\top - \frac{\eta}{N^2} \sum_{k,j} v_k u_j^\top$$

- Then, for any $k, y, t, x = u_y + n_t$, we have

$$v_k^\top W_1 x \approx \frac{\eta}{N} \mathbb{1}\{k = y\} + O\left(\frac{\eta}{N^2}\right)$$

Example: filter out exogenous noise

- **Data model**: $y \sim \mathrm{Unif}([N])$, $t \sim \mathrm{Unif}([T])$, $x = u_y + n_t \in \mathbb{R}^d$
  - where $\{n_t\}_{t=1}^T$ are another collection of embeddings, *e.g.*, positional embeddings
- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^N p(y = k) v_k (\hat{\mu}_k - \mu_k)^\top$$

$$= \frac{\eta}{N} \sum_{k=1}^N v_k (\mathbb{E}[u_y + n_t | y = k] - \mathbb{E}[u_y + n_t])^\top$$

$$= \frac{\eta}{N} \sum_{k=1}^N v_k u_k^\top - \frac{\eta}{N^2} \sum_{k,j} v_k u_j^\top$$

- Then, for any $k, y, t, x = u_y + n_t$, we have

$$v_k^\top W_1 x \approx \frac{\eta}{N} \mathbb{1}\{k = y\} + O\left(\frac{\eta}{N^2}\right)$$

- **Corollary**: $\hat{f}(x) = \arg\max_k v_k^\top W_1 x$ has near-perfect accuracy

Link with feature learning

Maximal updates:

- First gradient update from standard initialization ($[W_0]_{ij} \sim \mathcal{N}(0, 1/d)$) take the form

$$W_1 = W_0 + \Delta W \in \mathbb{R}^{d \times d}, \quad \Delta W := \sum_j \alpha_j v_j u_j^\top, \quad \alpha_j = \Theta_d(1)$$

Link with feature learning

Maximal updates:

- First gradient update from standard initialization ($[W_0]_{ij} \sim \mathcal{N}(0, 1/d)$) take the form

$$W_1 = W_0 + \Delta W \in \mathbb{R}^{d \times d}, \quad \Delta W := \sum_j \alpha_j v_j u_j^\top, \quad \alpha_j = \Theta_d(1)$$

- For any input embedding u_j, we have, thanks to near-orthonormality

$$\|W_0 u_j\| = \Theta_d(1) \quad \text{and} \quad \|\Delta W u_j\| = \Theta_d(1)$$

Link with feature learning

Maximal updates:

- First gradient update from standard initialization ($[W_0]_{ij} \sim \mathcal{N}(0, 1/d)$) take the form

$$W_1 = W_0 + \Delta W \in \mathbb{R}^{d \times d}, \quad \Delta W := \sum_j \alpha_j v_j u_j^\top, \quad \alpha_j = \Theta_d(1)$$

- For any input embedding u_j, we have, thanks to near-orthonormality

$$\|W_0 u_j\| = \Theta_d(1) \quad \text{and} \quad \|\Delta W u_j\| = \Theta_d(1)$$

- Contribution of updates is of similar order to initialization (not true for NTK!)
- Related to μP/mean-field (Chizat and Bach, 2018; Mei et al., 2019; Yang and Hu, 2021)

Link with feature learning

Maximal updates:

- First gradient update from standard initialization ($[W_0]_{ij} \sim \mathcal{N}(0, 1/d)$) take the form

$$W_1 = W_0 + \Delta W \in \mathbb{R}^{d \times d}, \quad \Delta W := \sum_j \alpha_j v_j u_j^\top, \quad \alpha_j = \Theta_d(1)$$

- For any input embedding u_j, we have, thanks to near-orthonormality

$$\|W_0 u_j\| = \Theta_d(1) \quad \text{and} \quad \|\Delta W u_j\| = \Theta_d(1)$$

- Contribution of updates is of similar order to initialization (not true for NTK!)
- Related to μP/mean-field (Chizat and Bach, 2018; Mei et al., 2019; Yang and Hu, 2021)

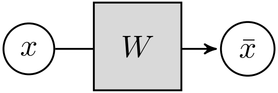
Large gradient steps on shallow networks:

- Useful for feature learning in **single-index** and **multi-index** models

$$y = f^*(x) + \text{noise}, \quad f^*(x) = g^*(Wx), \quad W \in \mathbb{R}^{r \times d}$$

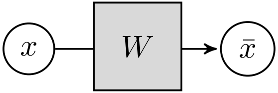
- Sufficient to break the curse of dimensionality when $r \ll d$
- (Ba et al., 2022; Damian et al., 2022; Dandi et al., 2023; Nichani et al., 2023)

Associative memories inside deep models



- Consider W that connects two nodes $x, \bar{x}$ in a feedforward computational graph

Associative memories inside deep models



- Consider W that connects two nodes $x, \bar{x}$ in a feedforward computational graph
- The loss gradient takes the form

$$\nabla_W L = \mathbb{E}[\nabla_{\bar{x}} \ell \cdot x^\top]$$

  where $\nabla_{\bar{x}} \ell$ is the **backward** vector (loss gradient w.r.t. $\bar{x}$)
- Often, this expectation may lead to associative memories as before
- A similar form can arise in attention matrices (see later!)

Questions

- **Finite capacity?** how much can we "store" with finite d?

Questions

- **Finite capacity?** how much can we "store" with finite d?
- **Finite samples?** how well can we learn with finite data?

Questions

- **Finite capacity?** how much can we "store" with finite d?
- **Finite samples?** how well can we learn with finite data?
- **Role of optimization algorithms?** multiple gradient steps? Adam?

Questions

- **Finite capacity?** how much can we "store" with finite d?
- **Finite samples?** how well can we learn with finite data?
- **Role of optimization algorithms?** multiple gradient steps? Adam?

$\Longrightarrow$ **study through scaling laws** (a.k.a. generalization bounds/statistical rates)

Setup with heavy-tailed data

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \dots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

Setup with heavy-tailed data

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \dots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

- Heavy-tailed token frequencies: Zipf law (typical for language where N is very large)

$$p(z) \propto z^{-\alpha}$$

Setup with heavy-tailed data

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \ldots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

- Heavy-tailed token frequencies: Zipf law (typical for language where N is very large)

$$p(z) \propto z^{-\alpha}$$

- Hutter (2021): with infinite memory, we have

$$L(\hat{f}_n) \lesssim n^{-\frac{\alpha-1}{\alpha}}$$

Setup with heavy-tailed data

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \dots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

- Heavy-tailed token frequencies: Zipf law (typical for language where N is very large)

$$p(z) \propto z^{-\alpha}$$

- Hutter (2021): with infinite memory, we have

$$L(\hat{f}_n) \lesssim n^{-\frac{\alpha-1}{\alpha}}$$

- **Q: What about finite capacity?**

Scaling laws with finite capacity

- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Scaling laws with finite capacity

- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

- Single population gradient step: $q(z) \approx p(z)$

Scaling laws with finite capacity

- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$

Scaling laws with finite capacity

- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. *For* $q(z) = \mathbb{1}\{z \in S_n\}$, *and* $d \gg N$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ *for any* k

Scaling laws with finite capacity

- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. *For* $q(z) = \mathbb{1}\{z \in S_n\}$*, and* $d \gg N$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ *for any* k
3. *For* $q(z) = \mathbb{1}\{z \text{ seen at least } s \text{ times in } S_n\}$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\alpha+1}$

Scaling laws with finite capacity

- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. *For* $q(z) = \mathbb{1}\{z \in S_n\}$, *and* $d \gg N$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ *for any* k
3. *For* $q(z) = \mathbb{1}\{z \text{ seen at least } s \text{ times in } S_n\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\alpha+1}$

- $n^{-\frac{\alpha-1}{\alpha}}$ is the same as (Hutter, 2021)
- $q = 1$ is best if we have enough capacity
- Can store at most d memories (approximation error: $d^{-\alpha+1}$)

Scaling laws with optimization algorithms

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

Scaling laws with optimization algorithms

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

- One step of SGD with large batch: $q(z) \approx p(z)$

Scaling laws with optimization algorithms

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$

Scaling laws with optimization algorithms

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$
- For $d \leq N$, smaller step-sizes can help later in training

Scaling laws with optimization algorithms

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

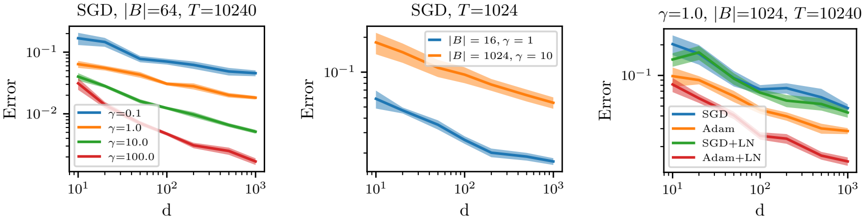
- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$
- For $d \leq N$, smaller step-sizes can help later in training
- Adam and layer-norm help with practical settings (large batch sizes + smaller step-size)

Scaling laws with optimization algorithms

$$W_{n,d} = \sum_{z=1}^{N} q(z) v_{f^*(z)} u_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$
- For $d \leq N$, smaller step-sizes can help later in training
- Adam and layer-norm help with practical settings (large batch sizes + smaller step-size)



Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

- **Nearest-neighbor** lookup: set $u_z = v_{f^*(z)}$ and take $\hat{f}(z) = \arg\max_y v_y^\top u_z$

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

- **Nearest-neighbor** lookup: set $u_z = v_{f^*(z)}$ and take $\hat{f}(z) = \arg\max_y v_y^\top u_z$
- **Attention**: soft-max instead of hard-max to retrieve from context

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

- **Nearest-neighbor** lookup: set $u_z = v_{f^*(z)}$ and take $\hat{f}(z) = \arg\max_y v_y^\top u_z$
- **Attention**: soft-max instead of hard-max to retrieve from context
- **MLP**: $\hat{f}(z) = \arg\max_y v_y^\top \sum_{z'=1}^N v_{f^*(z')} \sigma(u_{z'}^\top u_z - b)$

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

- **Nearest-neighbor** lookup: set $u_z = v_{f^*(z)}$ and take $\hat{f}(z) = \arg\max_y v_y^\top u_z$
- **Attention**: soft-max instead of hard-max to retrieve from context
- **MLP**: $\hat{f}(z) = \arg\max_y v_y^\top \sum_{z'=1}^N v_{f^*(z')} \sigma(u_{z'}^\top u_z - b)$

But: higher computational cost, more sensitive to noise, harder to learn