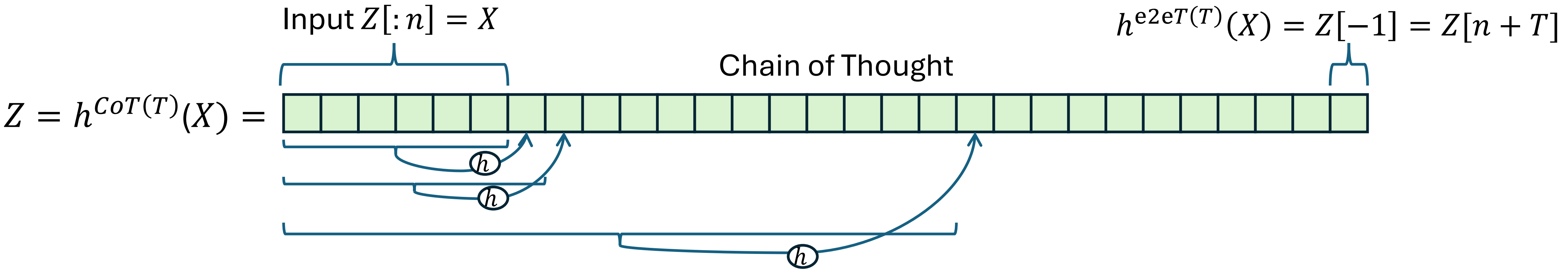
Learning Iterated Models

Nati Srebro (TTIC)

Based on ongoing joint work and discussions with

Nirmit Joshi (TTIC), **Gal Vardi** (TTIC→Weizmann),

Theo Misiakiewicz (TTIC→Yale), **Zhiyuan Li** (TTIC)



Iterating a function $h: \Sigma^* \to \Sigma$

Generally consider $h(X) = h(\,[0] \oplus X\,)$, and think of $h(X) = h([0,0,0,\dots,0] \oplus X)$ when defining $h(\cdot)$

For input of length $|X| = n$, start with $Z[0:n] = X$, and for $t \geq n$:

$$Z[t] = h(Z[:t])$$

$$h^{\mathrm{CoT}(0)}(X) = X$$

$$h^{\mathrm{CoT}(t+1)}(X) = h^{\mathrm{CoT}(t)}(X) \oplus \left[h\left(h^{\mathrm{CoT}(t)}(X)\right)\right]$$

i.e. $h^{\mathrm{CoT}(t)}(X) \in \Sigma^{|X|+t}$

$h^{\mathrm{e2e}(t)}(X) = h^{\mathrm{CoT}(t)}(X)[-1]$, i.e. the output of the last iteration

For a base class $\mathcal{H} = \{h: \Sigma^* \to \Sigma\} \subseteq \Sigma^{\Sigma^*}$ consider:

$$\mathcal{H}^{\mathrm{CoT}(T)} = \left\{\, h^{\mathrm{CoT}(T)}: \Sigma^* \to \Sigma^* \;\middle|\; h \in \mathcal{H} \,\right\} \qquad \mathcal{H}^{\mathrm{e2e}(T)} = \left\{\, h^{\mathrm{e2e}(T)}: \Sigma^* \to \Sigma \;\middle|\; h \in \mathcal{H} \,\right\} \subseteq \Sigma^{\Sigma^*}$$

(Realizable) Learning of an Iterated Class

End-to-end learning of $\mathcal{H}^{\mathbf{e2e}(T)}$: Given $\left(X_i, y_i = h^{\mathrm{e2e}(T)}(X_i)\right)$ for $X_1 \ldots X_m \sim iid\ \mathcal{D}$, learn $\hat{h}: \Sigma^* \to \Sigma$ with $\mathbb{E}_{X\sim\mathcal{D}}\left[err\left(\hat{h}(X), h^{\mathrm{e2e}(T)}(X)\right)\right] \le \epsilon$ $\boxed{err(\hat{y}, y) = 1\ iff\ \hat{y} \ne y}$

- Formally: Learning rule $A: (\Sigma^* \times \Sigma)^* \to \Sigma^{\Sigma^*}$ ϵ-e2e-learns $\mathcal{H}^{(T)}$ with sample complexity m on input distribution $\mathcal{D}$, if for any $h \in \mathcal{H}$, $\mathbb{E}_{S\sim\mathcal{D}^m_{h^{\mathrm{e2e}(T)}}} \mathbb{E}_{X\sim\mathcal{D}}\left[err\left(A(S)(X), h^{\mathrm{e2e}(T)}(X)\right)\right] \le \epsilon$
- Dist independent learning: Rule A ϵ-e2e-learns with sample complexity $m(\epsilon)$ for any distribution $\mathcal{D}$ over Σ^*
  Input length dependent: sample complexity $m(\epsilon, n)$ for any distribution $\mathcal{D}$ on Σ^n

$\equiv$ PAC-Learning $\mathcal{H}^{\mathrm{e2e}(T)}$

Chain-of-Thought learning of $\mathcal{H}^{\mathbf{e2e}(T)}$: Given $\left\{\mathrm{Z_i} = h^{\mathrm{CoT}(T)}(X_i)\right\}$ for $X_1 \ldots X_m \sim iid\ \mathcal{D}$, learn $\hat{h}: \Sigma^* \to \Sigma$ with $\mathbb{E}_{X\sim\mathcal{D}}\left[err\left(\hat{h}(X), h^{\mathrm{e2e}(T)}(X)\right)\right] \le \epsilon$

- Formally: Learning rule $A: (\Sigma^*)^* \to \Sigma^{\Sigma^*}$ ϵ-CoT-learns $\mathcal{H}^{(T)}$ with sample complexity m on input distribution $\mathcal{D}$, if for any $h \in \mathcal{H}$, $\mathbb{E}_{S\sim\mathcal{D}^m_{h^{\mathrm{CoT}(T)}}} \mathbb{E}_{X\sim\mathcal{D}}\left[err\left(A(S)(X), h^{\mathrm{e2e}(T)}(X)\right)\right] \le \epsilon$

$\approx$ AR-Learnable [Malach 24 "Auto-Regressive Next-Token Predictors are Universal Learners"]
(but we only care about final output)

Bounding the Sample Complexity in terms of $\mathcal{H}$

If $\mathcal{H}$ has bounded cardinality:

$$m_{\mathrm{CoT}(T)}(\mathcal{H}) \leq m_{\mathrm{e2e}(T)}\,(\mathcal{H}) \leq O\left(\log\left|\mathcal{H}^{\mathrm{e2e}(T)}\right| \cdot \frac{1}{\epsilon}\right) \leq O\left(\log|\mathcal{H}| \cdot \frac{1}{\epsilon}\right)$$

If $h \in \mathcal{H}$ has binary output, i.e. $\Sigma = \{0,1\}$, can we bound in terms of $VC(\mathcal{H})$?

$$m_{\mathrm{CoT}(T)}(\mathcal{H}) \leq m_{\mathrm{e2e}(T)}\,(\mathcal{H}) \leq \tilde{O}\left(VC\left(\mathcal{H}^{\mathrm{e2e}(T)}\right) \cdot \frac{1}{\epsilon}\right) \leq \tilde{O}\left(T \cdot VC(\mathcal{H}) \cdot \frac{1}{\epsilon}\right)$$

Tight: $\forall_{D,T}$ exists $\mathcal{H}$ with $VC(\mathcal{H}) = D$ but $m_{\mathrm{e2e}(T)}(\mathcal{H}) = \Omega\left(VC\left(\mathcal{H}^{\mathrm{e2e}(T)}\right)\right) = \Omega(TD)$,
even over $\{0,1\}^n, n = O(\log TD)$

Over $\Sigma = \mathbb{R}$ and relying on generalization of VC, even worse:

There exists $\mathcal{H}$ with subgraph dimension (aka Pollard pseudo-dimension) 1, but $\mathcal{H}^{\mathrm{e2e}(T)}$ has infinite subgraph dim, for any $T > 1$.

Iterated Linear Thresholds

$$\mathcal{L}_d = \{\ h_w : Z \mapsto \operatorname{sign}(\langle w, [1] \oplus Z[-d:]\rangle)\ \middle|\ w \in \mathbb{R}^{d+1}\ \}\ \text{ over } \Sigma = \{\pm 1\}$$

Theorem: $VC\left(\mathcal{L}_d^{\mathrm{e2e}(T)}\right) \le O(\, dT \wedge\ d^2\,)$

e2e Learning

Find w s.t. $y_i = h_w^{\mathrm{e2e}(T)}(X_i)$

For $i = 1..m$

$\boldsymbol{m = \tilde{O}\left(\left(dT \wedge d^2\right) \cdot \frac{1}{\epsilon}\right)}$

Open: is this tight?
Even $m = O(d)$ possible

Hard!

No $poly(n, d, T)$ time learning (even improper, if ∃crypto)

CoT Learning

Find w s.t. $Z_i[-t] = h_w(Z_i[:-t])$

For $i = 1..m,\ \ t = 1..T$

With m contexts, mT total examples for $\mathcal{H}$

➔ Suggests $m = \frac{d}{T} \cdot \frac{1}{\epsilon}$ might be enough

... but examples not iid (only d independent)

... AND need error at each step to be $\frac{\epsilon}{T}$

➔ **Actual sample complexity $\boldsymbol{\tilde{O}\left(\left(dT \wedge d^2\right) \cdot \frac{1}{\epsilon}\right)}$**

Easy!

LP-Satisfiability

Open: is this tight?
No stat advantage over e2e?
Advantage over separate w_i ?

Separate $\boldsymbol{w_i}$ (improper)

Find w_t s.t. $Z_i[-t] = h_{w_t}(Z_i[:-t])$

For $i = 1..m,\ \ t = 1..T$ [Malach 24]

m ind. Samples per w_i, but need err$\le \frac{\epsilon}{T}$

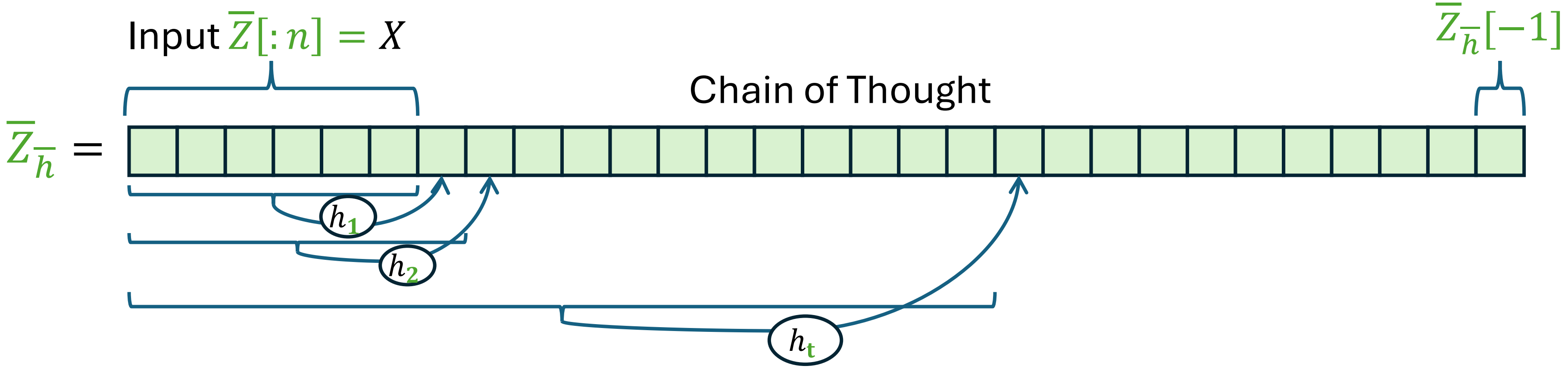
➔ **Sample complexity $\boldsymbol{\tilde{O}\left(dT \cdot \frac{1}{\epsilon}\right)}$**

Simulating Circuits with Iterated Linear Thresholds

Recall $\boldsymbol{Z} = \boldsymbol{h}^{\boldsymbol{CoT}(\boldsymbol{T})}(\boldsymbol{X})$ is defined as $\boldsymbol{Z}[\boldsymbol{t}] = \boldsymbol{h}(\boldsymbol{X}[:\boldsymbol{t}])$

Contrast with $\bar{Z}_{\bar{h}}$ specified by $\bar{h} = (h_1, \dots, h_T)$ defined as $\bar{Z}[t] = h_t(\bar{Z}[:t])$ initialized with $\bar{Z}[:|X|] = X$

$$\mathcal{H}^{\overline{\text{CoT}}(T)} = \left\{X \mapsto \bar{Z}_{\bar{h}} \middle| \bar{h} = (h_1, \dots, h_T) \in \mathcal{H}^T\right\}, \qquad \mathcal{H}^{\overline{\text{e2e}}(T)} = \left\{X \mapsto \bar{Z}_{\bar{h}}[-1] \middle| \bar{h} = (h_1, \dots, h_T) \in \mathcal{H}^T\right\}$$



Simulating Circuits with Iterated Linear Thresholds

Recall $\boldsymbol{Z} = \boldsymbol{h}^{\boldsymbol{CoT}(\boldsymbol{T})}(\boldsymbol{X})$ is defined as $\boldsymbol{Z}[\boldsymbol{t}] = \boldsymbol{h}(\boldsymbol{X}[:\boldsymbol{t}])$

Contrast with $\overline{Z}_{\overline{h}}$ specified by $\overline{h} = (h_1, \dots, h_T)$ defined as $\overline{Z}[t] = h_t(\overline{Z}[:t])$ initialized with $\overline{Z}[:|X|] = X$

$$\mathcal{H}^{\overline{\mathrm{CoT}}(T)} = \left\{X \mapsto \overline{Z}_{\overline{h}} \middle| \overline{h} = (h_1, \dots, h_T) \in \mathcal{H}^T\right\}, \qquad \mathcal{H}^{\overline{\mathrm{e2e}}(T)} = \left\{X \mapsto \overline{Z}_{\overline{h}}[-1] \middle| \overline{h} = (h_1, \dots, h_T) \in \mathcal{H}^T\right\}$$

Any (logical or linear threshold) circuit of size S over $\{\pm 1\}^n$ can be simulated by $\mathcal{L}_{n+T}^{\overline{\mathrm{CoT}}(T=S)}$,
i.e. for any circuit, there is a sequence $\overline{h}$ of lin thresholds s.t. $\overline{Z}_{\overline{h}}[i]$ is the output of unit i on input $\overline{Z}_{\overline{h}}[:n] = X$
[Malach 24]

<u>Theorem</u>: For any $w_1, \dots, w_T \in \mathbb{R}^d$, i.e. $\overline{h} \in \mathcal{L}_d^T$, there exists $\boldsymbol{w} \in \mathbb{R}^{\boldsymbol{O}((\boldsymbol{n}+\boldsymbol{T})^2)}$ s.t. for all $X \in \{\pm 1\}^n$,

$$\boldsymbol{h}_{\boldsymbol{w}}^{\boldsymbol{e2e}\left(\boldsymbol{O}((\boldsymbol{n}+\boldsymbol{T})^2)\right)}\left(\begin{bmatrix}-1 & 1^{\oplus (n+T)^2}\end{bmatrix} \oplus X\right) = \overline{Z}_{\overline{h}}(X)[-1]$$

<u>Conclusion</u>: Any circuit of size S can be simulated by $\boldsymbol{\mathcal{L}}_{\boldsymbol{d}}^{\mathbf{e2e}(\boldsymbol{T})}$ with $\boldsymbol{d}, \boldsymbol{T} = \boldsymbol{O}\left(\boldsymbol{S}^{\mathbf{2}}\right)$ and a fixed input expansion.

Learning poly-size circuits is hard ➔ learning $\mathcal{L}_d^{\mathrm{e2e}(T)}$ in time $poly(n, d, T)$ is hard

What I'd Really Like to Learn

- Ultimate: $DESC_S$ = { any function describable with S bits }
  ➔$\log|DESC_S| \leq S$, hence learnable with $O(S)$ samples

- More specifically: $PROG_S$ = { programs with of length S }
  ➔ $\log|PROG_S| = O(S)$, hence learnable with $O(S)$ samples
  But: Not tractable computationally + output not useful

- How about: $TIME_T$ = { programs with runtime T }
  Also learnable with $O(T)$ samples, learning $\in NP$
  $TIME_T \subseteq CIRCUT_{poly(T)} \subseteq NN_{poly(T)}$
  ➔We can learn $TIME_T$ (with poly-opt sample size) by learning a Feed-Forward Neural Nets

- But what about $PROG_{S,T}$ ={programs of length S and runtime T}
  - Learning with $O(S)$ samples is $\in NP$, as long as $T = poly(n)$
  - "Goal": learn with $poly(S \log T)$ samples and "training time" $poly(T)$
  - Can we construct a simple and "trainable" class $\mathcal{H} \supseteq PROG_{S,T}$ of complexity $poly(S \log T)$?

$$\mathcal{L}_d = \left\{ h_w : Z \mapsto \mathrm{sign}(\langle w, [1] \oplus Z[-d:] \rangle) \middle| w \in \mathbb{R}^{d+1} \right\}$$

$$Z = h_w^{CoT}(Z[:n]) \text{ with } Z[t] = h_w(Z[:t]) \text{ for } t \geq n$$

Problem with $\mathcal{L}_d^{\mathrm{CoT}(T)}$: context length is $O(d)$

➔ i.e. $Z[t]$ only depends on $Z[t-d:t] \in \{\pm 1\}^d$

➔ Computation has state space of size 2^d, i.e. need $d = \Omega(MEMORY)$

How can we get context length $\gg$ complexity?

Simplest attempt: sparse linear thresholds:

$$\mathcal{L}_{d,k} = \left\{ h_w : Z \mapsto \mathrm{sign}(\langle w, [1] \oplus Z[-d:] \rangle) \middle| w \in \mathbb{R}^{d+1}, \|w\|_0 \leq k \right\}$$

- Complexity $VC\left(\mathcal{L}_{d,k}^{\mathrm{e2e}(T)}\right) = O(k^2 \log d)$ with context length d
- Tractably CoT learnable for $k = O(1)$ (is this enough?) or with a sparse-learning oracle.
- What is the expressive power of $\mathcal{L}_{d,k}^{\mathrm{CoT}(T)}$??

- Can we describe a simple class $\mathcal{H}$ s.t.:
  - $PROG_{S,T} \subseteq \mathcal{H}^{\mathrm{e2e}(T)}$
  - $VC\left(\mathcal{H}^{\mathrm{e2e}(T)}\right) = poly(S \log T)$
  - CoT learnable with $poly(S \log T)$ and time $poly(T)$

The Turing Iterative Model

For an internal state space size S (and wlog tape alphabet $\{0,1\}$) consider a iterative model on alphabet

$$\Sigma = [S] \times \{0,1\} \times \{-1,0,1\}$$

parametrized by a Turing Machine transition function $A: (state, tape\ sym) \mapsto (new\ state, new\ sym, move)$:

$$\mathcal{H}_S = \left\{\ h_A: \Sigma^* \to \Sigma \ \middle|\ A \in \left([S] \times \{0,1\} \times \{-1,0,1\}\right)^{[S]\times\{0,1\}}\ \right\}$$

Where $h_A\big((s_0, r_0, m_0), \dots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$ computes the next symbol to be written, state and move for a TM specified by transition table A

$h_A\big(\ Z = (s_0, r_0, m_0), \dots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$:
For each $i = 0 \dots t$, calculate $pos[i] = \sum_{j<i} s_j$
Find $i^* = \max i\ \ s.t.\, pos[i] = pos[t]$
If exists, $r = r_{i^*}$, else $r = 0$
output $(s_t, r_t, m_t) = A(s_{t-1}, r)$

Claim: $PROG_{poly(S),poly(T)} = TM_{S,T} \subseteq \mathcal{H}_S^{\mathrm{e2e}(T)}$
With a fixed input embedding $Z[i] = (0, X[i], 1)$
And fixed output decoding (proj to 2nd component)

Claim: $\log|\mathcal{H}_S| \leq 2S \log 6S$
➔ e2e learnable with $m = O(S \log S)$

Claim: CoT learnable with $m = O(S \log S)$ in time $\tilde{O}(ST)$ by memorizing size-$O(S)$ table

The Turing Iterative Model

For an internal state space size S (and wlog tape alphabet $\{0,1\}$) consider an iterative model on alphabet

$$\Sigma = [S] \times \{0,1\} \times \{-1,0,1\}$$

parametrized by a Turing Machine transition function $A: (state, tape\ sym) \mapsto (new\ state, new\ sym, move)$:

$$\mathcal{H}_S = \left\{ h_A : \Sigma^* \to \Sigma \;\middle|\; A \in ([S] \times \{0,1\} \times \{-1,0,1\})^{[S]\times\{0,1\}} \right\}$$

Where $h_A\big((s_0, r_0, m_0), \dots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$ computes the next symbol to be written, state and move for a TM specified by transition table A

$h_A\big(Z = (s_0, r_0, m_0), \dots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$:

For each $i = 0 \dots t$, calculate $pos[i] = \sum_{j<i} s_j$

Find $i^* = \max i \;\; s.t. pos[i] = pos[t]$

If exists, $r = r_{i^*}$, else $r = 0$

output $(s_t, r_t, m_t) = A(s_{t-1}, r)$

$pos[i] = i \cdot Attn\big(k[j] = 0, q[i] = 0, v[j] = s_j \big)$

Keys: $k[i] = (i, 100T\ pos[i])$

Values: $v[i] = (pos[i], r_i)$

Query: $q[t] = (t, 100T\ pos[i])$

Obtain $(p, r) = Attn(k, q[t], v)$

If $p \neq pos[i]$, set $r \leftarrow 0$

$Attn(k, q[i], v)$:

$j^* = \arg\min_{j<i} \|k[j] - q[i]\|$

return $v[j^*]$

The Turing Iterative Model

For an internal state space size S (and wlog tape alphabet $\{0,1\}$) consider an iterative model on alphabet

$$\Sigma = [S] \times \{0,1\} \times \{-1,0,1\}$$

parametrized by a Turing Machine transition function $A: (state, tape\ sym) \mapsto (new\ state, new\ sym, move)$:

$$\mathcal{H}_S = \left\{\ h_A: \Sigma^* \to \Sigma \ \middle|\ A \in ([S] \times \{0,1\} \times \{-1,0,1\})^{[S]\times\{0,1\}}\ \right\}$$

Where $h_A\big((s_0, r_0, m_0), \dots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$ computes the next symbol to be written, state and move for a TM specified by transition table A

$h_A(\ Z = (s_0, r_0, m_0), \dots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$:

- For each $i = 0 \dots t$, calculate $pos[i] = \sum_{j<i} s_j$
- Find $i^* = \max i\ \ s.t. pos[i] = pos[t]$
  - If exists, $r = r_{i^*}$, else $r = 0$
- output $(s_t, r_t, m_t) = A(s_{t-1}, r)$

$pos[i] = i \cdot Attn(\ k[j] = 0, q[i] = 0, v[j] = s_j)$

Keys: $k[i] = (i, 100T\ pos[i])$
Values: $v[i] = (pos[i], r_i)$
Query: $q[t] = (t, 100T\ pos[i])$
Obtain $(p, r) = Attn(k, q[t], v)$
If $p \neq pos[i]$, set $r \leftarrow 0$

$Attn(k, q[i], v)$:

$w[j] = 1\ iff$

$j \in \arg\min_{j<i} \|k[j] - q[i]\|$

return $\frac{\sum_{j<i} w[j] v[j]}{\sum_{j<i} w[j]}$

The Turing Iterative Model

For an internal state space size S (and wlog tape alphabet $\{0,1\}$) consider an iterative model on alphabet

$$\Sigma = [S] \times \{0,1\} \times \{-1,0,1\}$$

parametrized by a Turing Machine transition function $A: (state, tape\ sym) \mapsto (new\ state, new\ sym, move)$:

$$\mathcal{H}_S = \left\{ \ h_A: \Sigma^* \to \Sigma \ \middle| \ A \in ([S] \times \{0,1\} \times \{-1,0,1\})^{[S]\times\{0,1\}} \ \right\}$$

Where $h_A\big((s_0, r_0, m_0), \dots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$ computes the next symbol to be written, state and move for a TM specified by transition table A

$h_A\big(Z = (s_0, r_0, m_0), \dots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$:

For each $i = 0 \dots t$, calculate $pos[i] = \sum_{j<i} s_j$

Find $i^* = \max i \ \ s.t. pos[i] = pos[t]$

If exists, $r = r_{i^*}$, else $r = 0$

output $(s_t, r_t, m_t) = A(s_{t-1}, r)$

$pos[i] = i \cdot Attn\big(k[j] = 0, q[i] = 0, v[j] = s_j \big)$

Keys: $k[i] = (i, 100T\ pos[i])$

Values: $v[i] = (pos[i], r_i)$

Query: $q[t] = (t, 100T\ pos[i])$

Obtain $(p, r) = Attn_{\beta\to\infty}(k, q[t], v)$

If $p \neq pos[i]$, set $r \leftarrow 0$

$Attn_\beta(k, q[i], v)$:

$w[j] = e^{\beta \|k[j] - q[i]\|}$

return $\dfrac{\sum_{j<i} w[j] v[j]}{\sum_{j<i} w[j]}$

The Turing Iterative Model

For an internal state space size S (and wlog tape alphabet $\{0,1\}$) consider an iterative model on alphabet

$$\Sigma = [S] \times \{0,1\} \times \{-1,0,1\}$$

parametrized by a Turing Machine transition function $A: (state, tape\ sym) \mapsto (new\ state, new\ sym, move)$:

$$\mathcal{H}_S = \left\{\ h_A: \Sigma^* \to \Sigma\ \middle|\ A \in ([S] \times \{0,1\} \times \{-1,0,1\})^{[S]\times\{0,1\}}\ \right\}$$

Where $h_A\big((s_0, r_0, m_0), \ldots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$ computes the next symbol to be written, state and move for a TM specified by transition table A

$h_A\big(\ Z = (s_0, r_0, m_0), \ldots, (s_{t-1}, r_{t-1}, m_{t-1})\big)$:

- For each $i = 0 \ldots t$, calculate $pos[i] = \sum_{j<i} s_j$
- Find $i^* = \max i\ \ s.t. pos[i] = pos[t]$
  - If exists, $r = r_{i^*}$, else $r = 0$
- output $(s_t, r_t, m_t) = A(s_{t-1}, r)$

$pos[i] = i \cdot Attn\big(\ k[j] = 0, q[i] = 0, v[j] = s_j\big)$

Keys: $k[i] = (i, pos[i]^2, T^2 - pos[i]^2)$
Values: $v[i] = (pos[i], r_i)$
Query: $q[t] = \left(\frac{1}{100T^4}, pos[t]^2, T^2 - pos[t]^2\right)$
Obtain $(p, r) = Attn_{\beta\to\infty}(k, q[t], v)$
If $p \neq pos[i]$, set $r \leftarrow 0$

$Attn(k, q[i], v)$:
$w[j] = e^{\beta\langle q[i], v\rangle}$
return $\frac{\sum_{j<i} w[j]v[j]}{\sum_{j<i} w[j]}$

$\mathcal{H}_S \subseteq$ { Transformer with poly(S log T)-sized MLP and O(log T) precision } (with simple fixed encoding)

[Jorge Pérez, Pablo Barceló, Javier Marinkovic '21, Attention is Turing-Complete]
[Colin Wei, Yining Chen, Tengyu Ma '22, Statistically Meaningful…Approximating TM with Transformers]
[William Merrill, Ashish Sabharwal '24, The Expressive Power of Transformers with Chain of Thought]

A (Minimal) Transformer?

$$f_w(X) = \begin{cases} sign(\langle w, Z[-d:]\rangle) & if\ Z[-5:] \neq [1\ 1\ 1\ 1\ 1] \\ \text{Attn}(Z) & if\ Z[-5:] = [1\ 1\ 1\ 1\ 1] \end{cases}$$

Attn(X):
$q = X[-r:0]$
$k[i] = X[-2ri: -2ri + r]$
$v[i] = X[-2ri + r]$
return $Attn(keys = k, query = q, vals = v)$

$$\mathcal{H}_{d,r} = \{ f_w \mid w \in \mathbb{R}^d \}$$

Claim: $\mathcal{H}_{d,r}^{(T)}$ CoT learnable with $m = O(d^2)$ samples in time $poly(d, T)$

What can it represent? $PROG_{S,T} \in \mathcal{H}_{d,r=poly(S \log T)}^{\mathrm{e2e}(poly(T))}$???

Summary

- Study of (Stationary) Iterative Models essential for understanding:
  - Autoregressive learning as it is actually done
  - Learning with large context length, and sample complexity independent of context (input + output) length
  - Length generalization
  - Learning with sample complexity scaling with program length, not runtime
- Open questions even on simple models
- What's the right view of non-realizable learning?
  - What is the goal/reward/error?
  - Discriminative/reward-based view of iterative models vs generative view