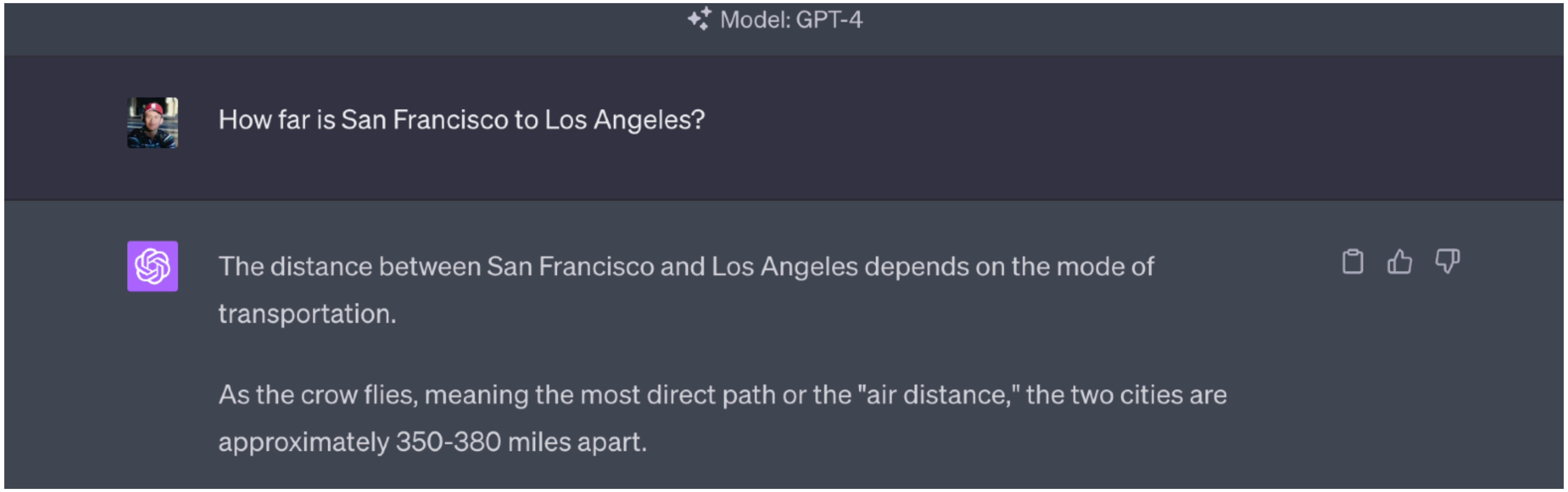
Transformers as Statisticians: Provable In-Context Learning with In-Context Algorithm Selection

Song Mei
UC Berkeley

Joint work with
Yu Bai (Salesforce Research), Fan Chen (Peking U -> MIT PhD), Huan Wang (Salesforce Research), Caiming Xiong (Salesforce Research)

Large Language model



- Task: next word prediction.
- Input: sequence of words. Output: the next word.
- Training data: texts from the whole internet, e.g., Wikipedia, Reddit, Shakespeare books, etc.
- Network architecture: transformers.

An emergent capability: In-context learning (ICL)

Apple -> Fruit. Sofa -> Furniture. Bird -> ?

Animal.

Circulation revenue has increased by 5% in Finland.
Positive.
Tom didn't disclose the purchase price.
Neutral.
Paying off the national debt will be extremely painful.

Negative.

Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

Supervised learning vs In-context learning

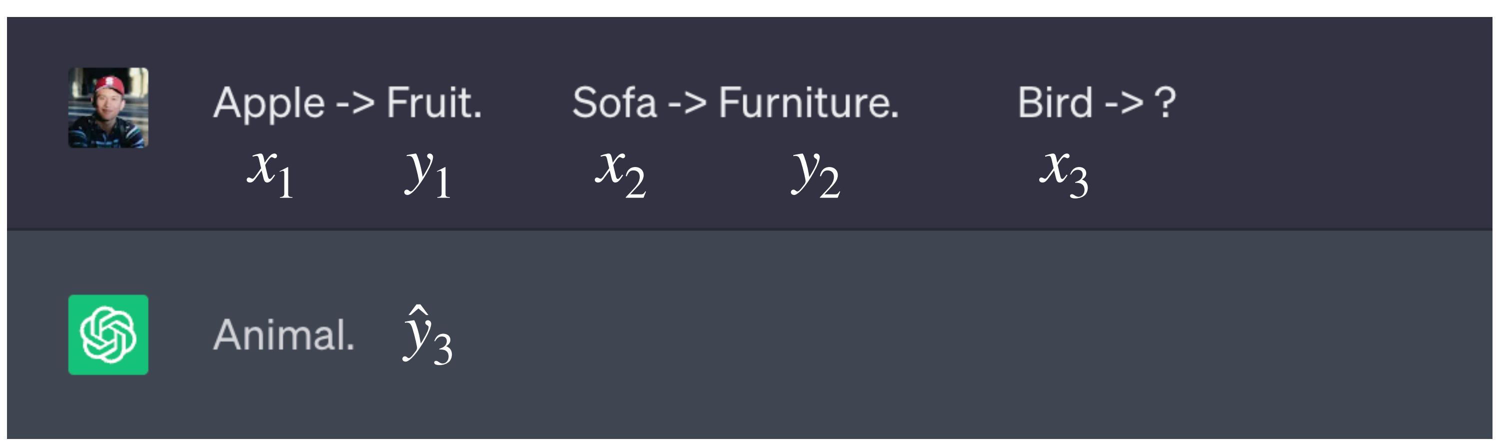
Dataset : $\{(x_i, y_i)\}_{i \in [N]}, (x_{N+1}, y_{N+1}) \sim \mathbb{P}$

- Supervised learning:

1. Train a model $y = f(x; \hat{w})$ using the training set $\{(x_i, y_i)\}_{1 \le i \le N}$.
2. Output prediction $\hat{y}_{N+1} = f(x_{N+1}; \hat{w}) \approx y_{N+1}$

- In-context learning:

1. Pre-train a model $h = f(H; \hat{\theta})$ using an enormous meta-dataset.
2. Take $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N, x_{N+1}]$ as the context input.
3. Output prediction $\hat{y}_{N+1} = f(H; \hat{\theta}) \approx y_{N+1}$.

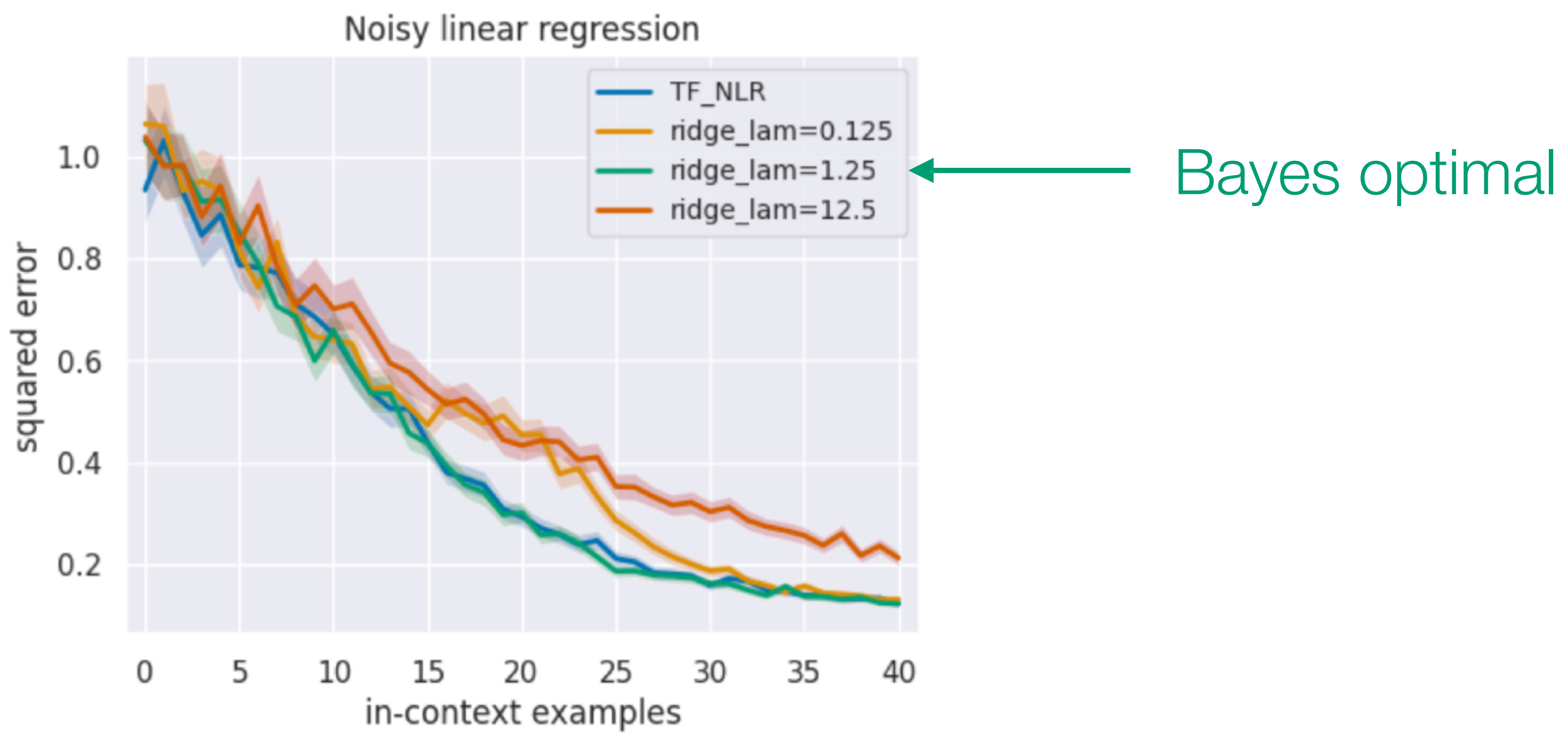


An ICL experiment on a synthetic dataset

A Task : $\{(x_i, y_i)\}_{i\in[N]}$, $\beta \sim \mathcal{N}(0, I_d/d)$,

$$x_i \sim \mathcal{N}(0, I_d), \quad y_i = x_i^\top \beta + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- A dataset of (size N) is a meta-datapoint: $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N]$.
- A meta-dataset (size n): $\{H^{(j)} = [x_1^{(j)}, y_1^{(j)}, x_2^{(j)}, y_2^{(j)}, \ldots, x_N^{(j)}, y_N^{(j)}]\}_{j\in[n]}$.
- Train the GPT2 model using $\{H^{(j)}\}_{j\in[n]}$ (a smaller version of ChatGPT).
- $d = 5, \quad N = 40, \quad n = 19{,}200{,}000$
- Evaluate the test performance of GPT2 on a new independent task.



- Trained GPT2 performs as good as Bayesian predictor!

Today

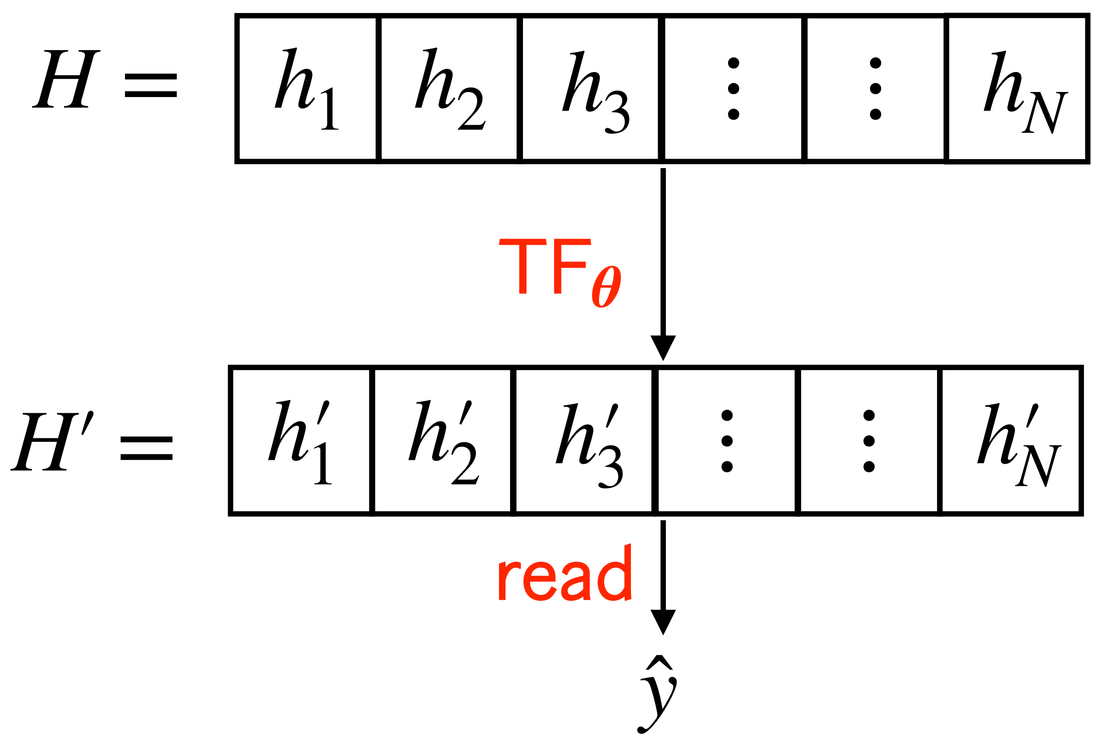
Why can GPT (transformers) perform in-context learning (ICL)?

[Bai, Chen, Wang, Xiong, Mei, 2023].

Related work: [Xie et al., 2021], [Garg et al. 2022], [Akyurek et al., 2022], [von Oswald et al., 2022], [Dai et al., 2022]

Transformers

- A transformer* is a sequence-to-sequence neural network $\mathsf{TF}_{\boldsymbol{\theta}} : \mathbb{R}^{D\times N} \to \mathbb{R}^{D\times N}$.
- Input sequence: $H = [h_1, h_2, \ldots, h_N] \in \mathbb{R}^{D\times N}$; each $h_i \in \mathbb{R}^D$ is called a token.



- Transformer output: $H' = \mathsf{TF}_{\boldsymbol{\theta}}(H) = [h'_1, \cdots, h'_N] \in \mathbb{R}^{d\times N}$.
- Final output: $\hat{y} = \mathsf{read}(\mathsf{TF}_{\boldsymbol{\theta}}(H))$.

** Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).*
We focus on the encoder architecture.

Transformers: Feedforward layer

- A transformer is an iterative composition of FF layers and Attention layers

  $\mathrm{TF}_{\boldsymbol{\theta}}(\cdot) = \mathrm{FF}_{W^{(L)}} \circ \mathrm{ATTN}_{A^{(L)}} \circ \cdots \circ \mathrm{FF}_{W^{(1)}} \circ \mathrm{ATTN}_{A^{(1)}}$

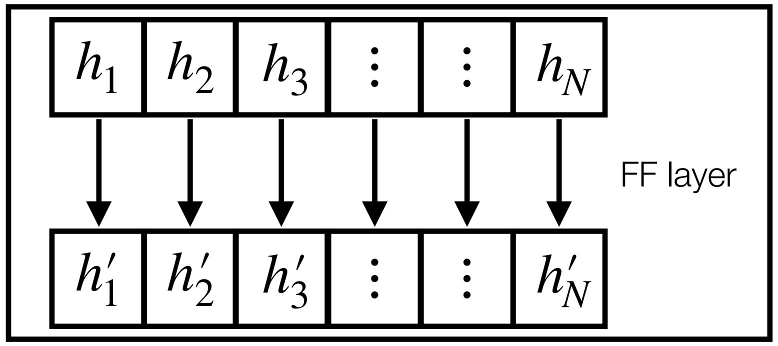
  $\boldsymbol{\theta} = (W^{(L)}, A^{(L)}, \ldots, W^{(1)}, A^{(1)})$

- Feedforward layer: $H' = \mathrm{FF}_W(H) : \mathbb{R}^{D\times N} \to \mathbb{R}^{D\times N}$,

$$W = (W_1, W_2), \qquad W_1, W_2^\top \in \mathbb{R}^{D'\times D}$$

- Token-wise function:

$$h_i' = h_i + W_2 \cdot \sigma(W_1 h_i)$$



* Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformers: Attention layer

- A transformer is an iterative composition of FF layers and Attention layers

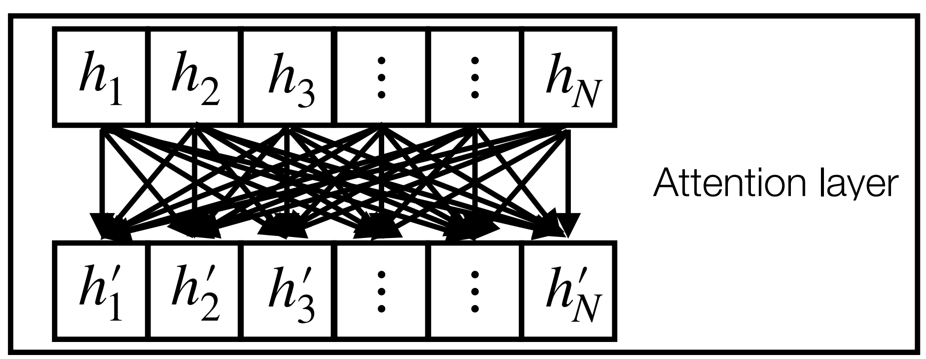
  $\mathsf{TF}_{\boldsymbol{\theta}}(\cdot) = \mathsf{FF}_{W^{(L)}} \circ \mathsf{ATTN}_{A^{(L)}} \circ \cdots \circ \mathsf{FF}_{W^{(1)}} \circ \mathsf{ATTN}_{A^{(1)}}$

  $\boldsymbol{\theta} = (W^{(L)}, A^{(L)}, \ldots, W^{(1)}, A^{(1)})$

- Attention layer: $H' = \mathsf{ATTN}_A(H) : \mathbb{R}^{D\times N} \to \mathbb{R}^{D\times N}$

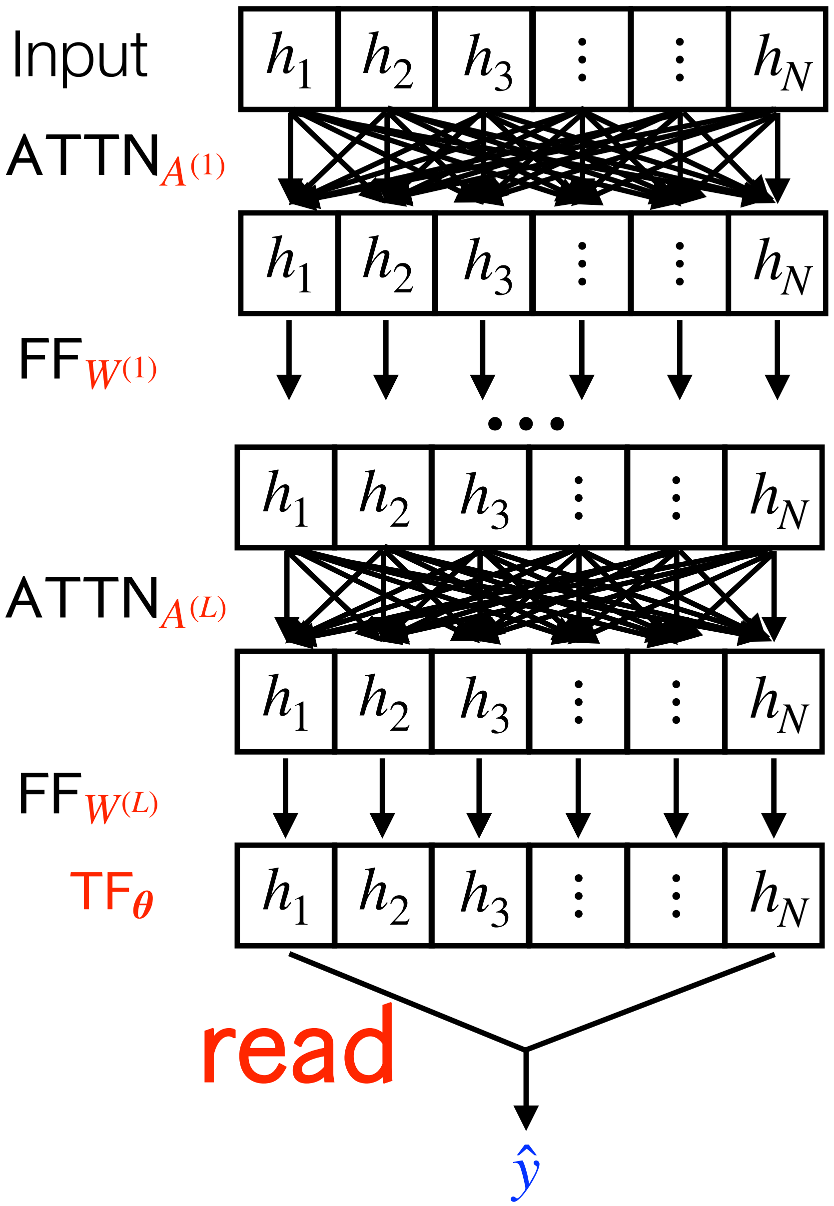
  $$A = (Q_m, K_m, V_m)_{m\in[M]}, \qquad Q_m, K_m, V_m \in \mathbb{R}^{D\times D}$$

- Single-head attention layer: $h_i' = h_i + \frac{1}{N}\sum_{j=1}^{N} \sigma(\langle Q h_i, K h_j\rangle) V h_j$
- Multi-head attention layer: $h_i' = h_i + \sum_{m=1}^{M} \frac{1}{N}\sum_{j=1}^{N} \sigma(\langle Q_m h_i, K_m h_j\rangle) V_m h_j$



* Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformers: the whole structure



- BERT (2018): $L = 24,\ M = 12,\ D = 1024,\ \#\text{para} = 340\text{M}$
- GPT2 (2019): $L = 48,\ M = 25,\ D = 1600,\ \#\text{para} = 1.5\text{B}$
- GPT3 (2020) : $L = 96,\ M = 96,\ D = 12288,\ \#\text{para} = 175\text{B}$
- PALM (2022) : $L = 118,\ M = 48,\ D = 11432,\ \#\text{param} = 540\text{B}$

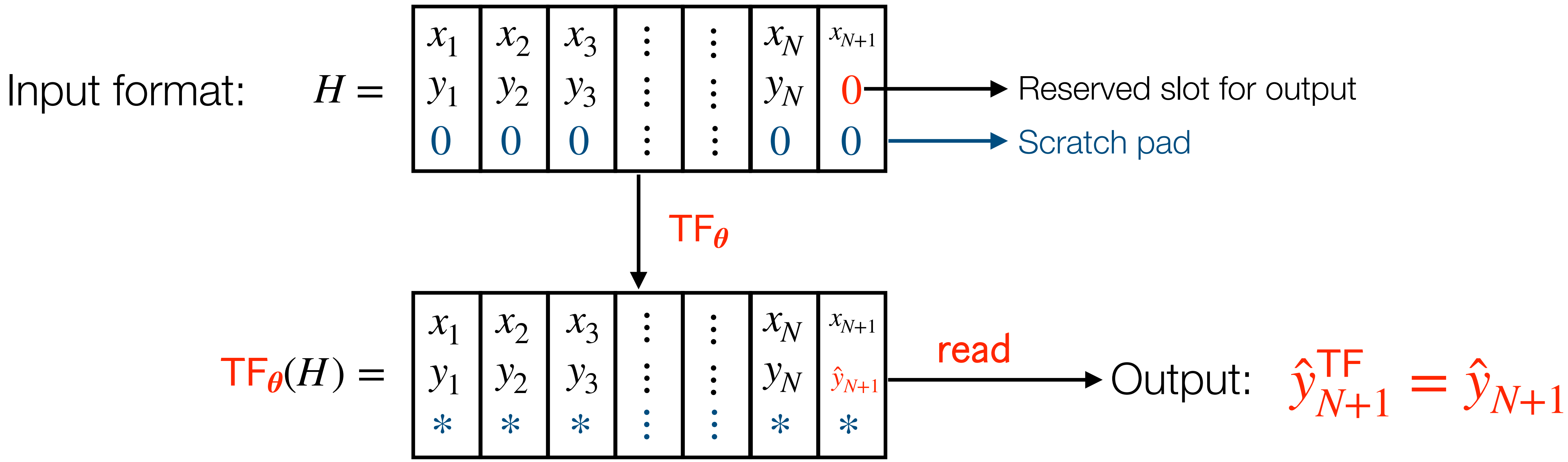
Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019): 9.
Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.
Chowdhery, Aakanksha, et al. "Palm: Scaling language modeling with pathways." arXiv preprint arXiv:2204.02311 (2022).

In-context learning (ICL) by Transformers

ICL by transformers

$$\{(x_i, y_i)\}_{i\in[N+1]} \sim_{iid} \mathsf{P}, \qquad y_i = f(\langle x_i, w_* \rangle) + \varepsilon_i$$

- ICL by transformers:

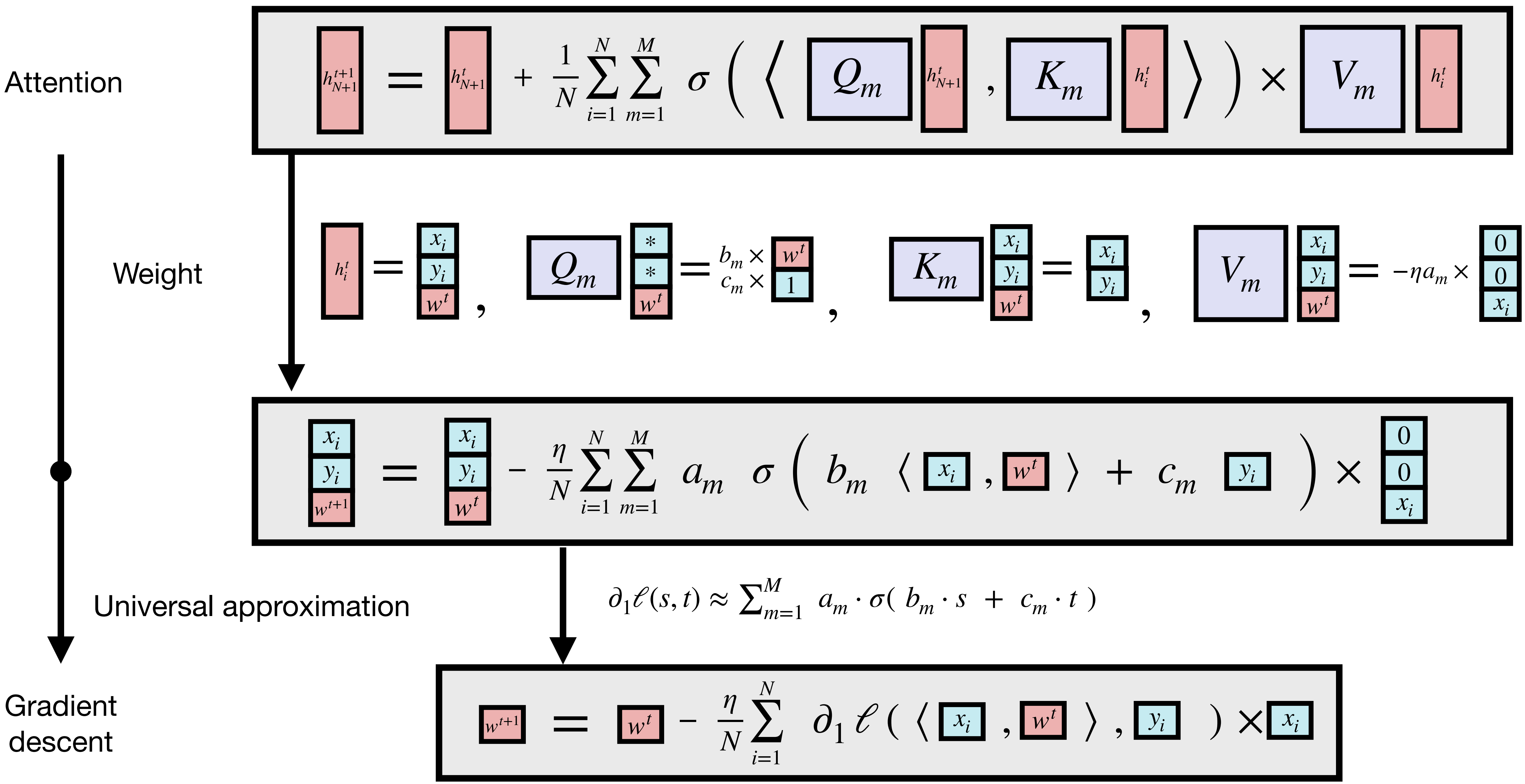


- Empirical observation: on pre-trained Transformers, $\mathbb{E}\ell(\hat{y}^{\mathsf{TF}}_{N+1}, y_{N+1})$ is small.
- Explanation: transformers can approximate GD on $\widehat{R}_N(w) = \frac{1}{N}\sum_{i=1}^N \ell(x_i^\top w, y_i)$

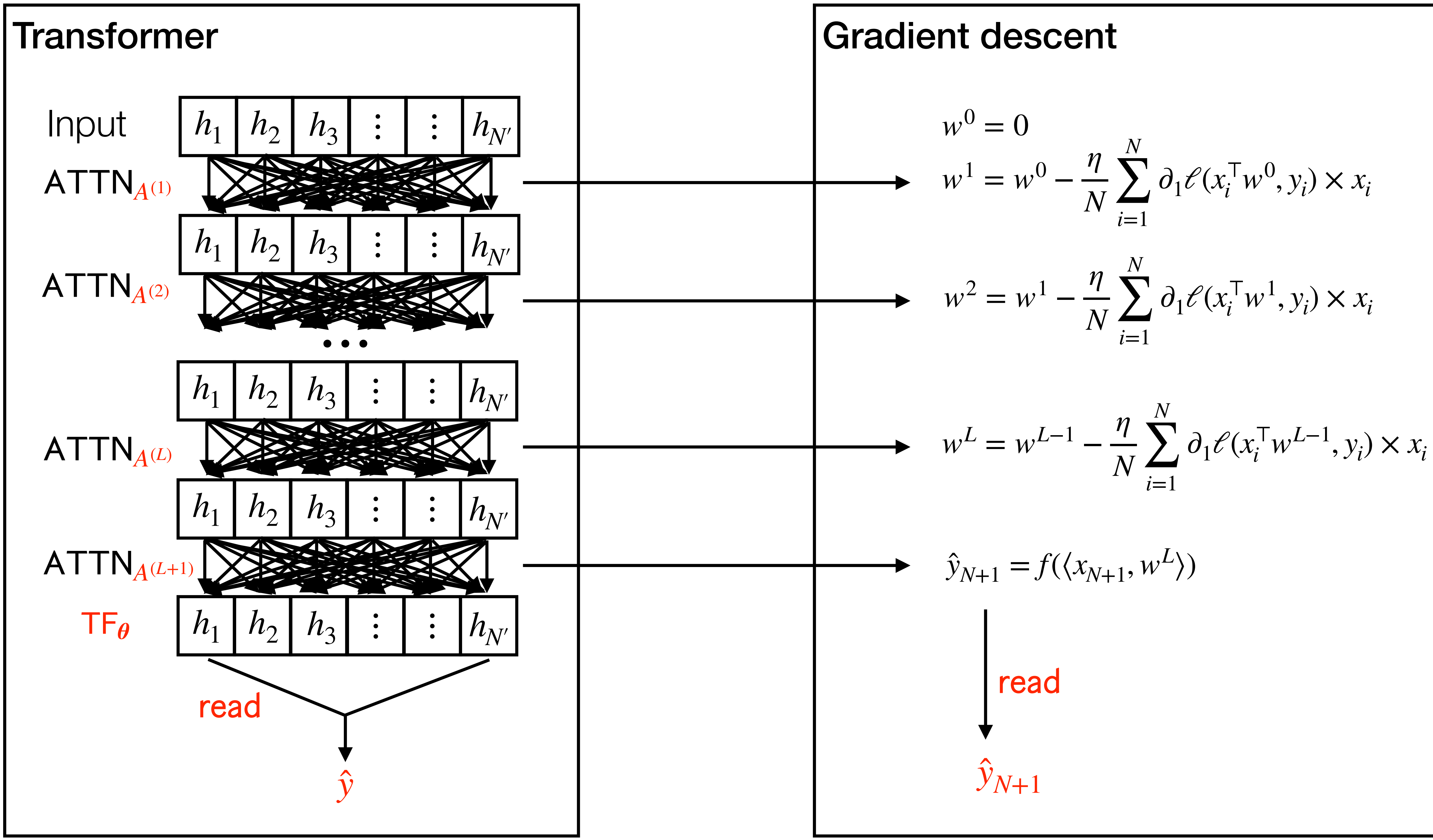
$$w^{t+1} = w^t - \frac{\eta}{N}\sum_{i=1}^N \partial_1 \ell(x_i^\top w^t, y_i) \times x_i$$

[Akyurek et al., 2022], [von Oswald et al., 2022], [Dai et al., 2022] have rough ideas of this kind.

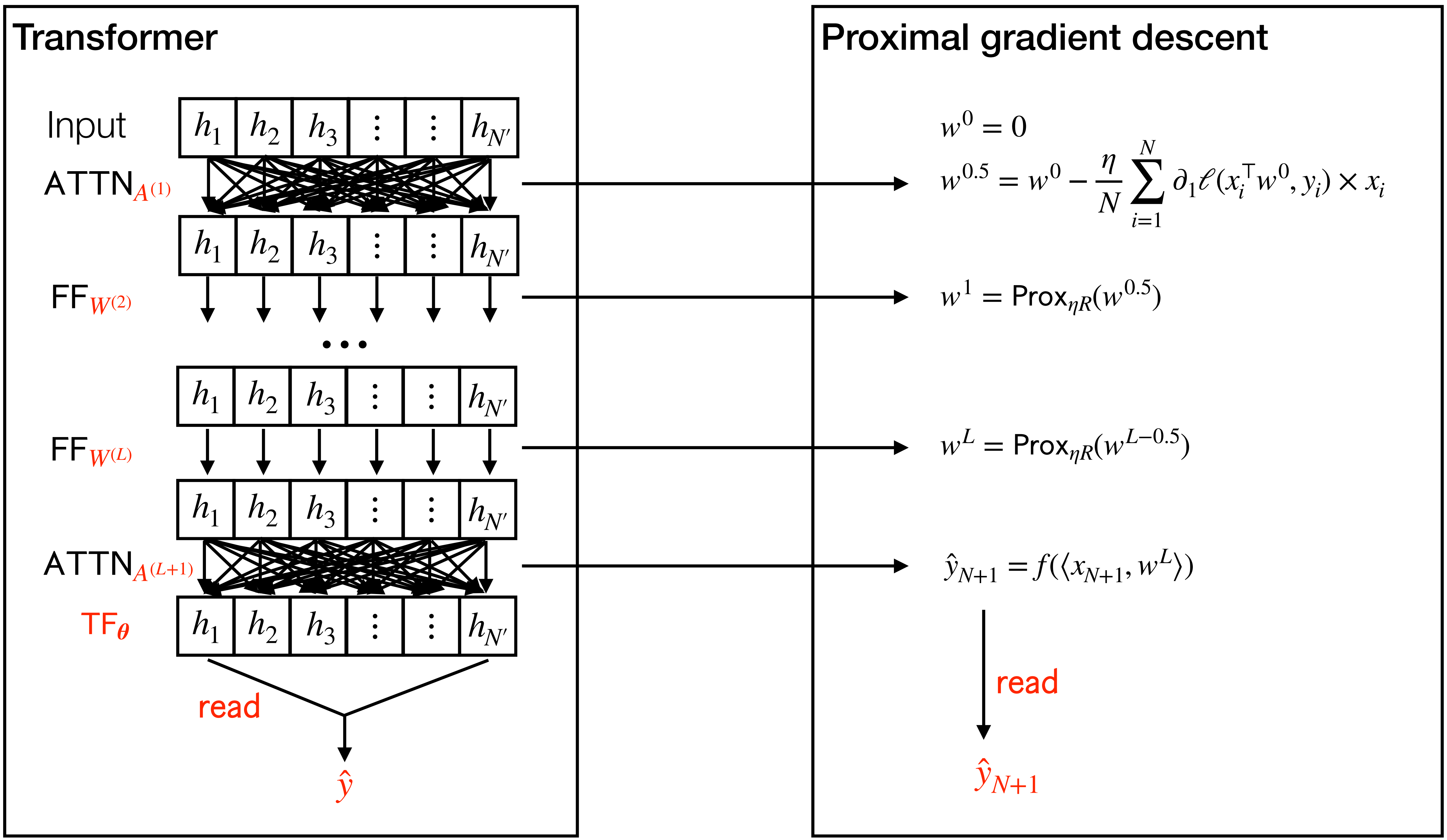
One-step gradient descent by an attention layer



Transformer versus multi-step GD



Transformer versus proximal gradient descent



In-context GD

Important parameters:

Embedding dimensions D, number of layers L, number of ATTN heads M, FF width D', and norm of parameters $|||\theta|||$

Thm [In-context gradient descent]: There exists a transformer with

$$D = \mathcal{O}(d), L = \mathcal{O}(1/\eta), M = M(\varepsilon), D' = \mathcal{O}(d), |||\theta||| = \mathcal{O}(\mathrm{Poly}(D, L, M, D', N)),$$

that output $\hat{w}^L$ that is close to the GD iterates w^L_{GD}

$$\|\hat{w}^L - w^L_{\mathrm{GD}}\|_2 \leq L\eta\varepsilon.$$

- TFs with competitive or better performance can be efficiently found by performing pre-training using particular loss functions (more details later).
- Some empirical evidence that pre-trained TF are indeed performing theoretically-constructed algorithms (more details later).

In-context ridge, logistic, LASSO

Thm: There exists three transformers with $D = D' = \mathcal{O}(d), \|\!| \theta \|\!| = \mathcal{O}(\mathrm{Poly}(\cdot))$

$$
\begin{aligned}
\text{Ridge}: \quad & L = \mathcal{O}(\log(N)), & M = 3, \\
\text{Logistic}: \quad & L = \mathcal{O}(\log(N/d)), & M = N/d, \\
\text{LASSO}: \quad & L = \mathcal{O}(1 + d/N), & M = 2,
\end{aligned}
$$

that output $\hat{y}_{N+1}$ implementing Ridge, Logistic, LASSO, with

$$
\begin{aligned}
\text{Ridge}: \quad & \mathbb{E}[(\hat{y}_{N+1} - y_{N+1})^2] \le \mathcal{O}(d/N), \\
\text{Logistic}: \quad & \mathbb{E}[(\hat{y}_{N+1} - \mathbb{E}[y_{N+1} \,|\, x_{N+1}])^2] \le \mathcal{O}(d/N), \\
\text{LASSO}: \quad & \mathbb{E}[(\hat{y}_{N+1} - y_{N+1})^2] \le \mathcal{O}(s \log d/N).
\end{aligned}
$$

Generalization bound for pre-training

Setting of pre-training:

Meta-dataset $\{Z^{(j)}\}_{j\in[n]} \sim_{iid} \pi$, with $Z^{(j)} = \{(x_i^{(j)}, y_i^{(j)})\}_{i\in[N]}$ iid.

Form empirical risk $\widehat{L}_{\mathrm{icl}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{j=1}^{n} \ell_{\mathrm{icl}}(Z^{(j)}, \boldsymbol{\theta})$. Consider TF class

$$\Theta_{L,M,D',B} = \left\{\boldsymbol{\theta} : L \text{ layers}, M \text{ heads}, D' \text{ width}, B \text{ norm}\right\}$$

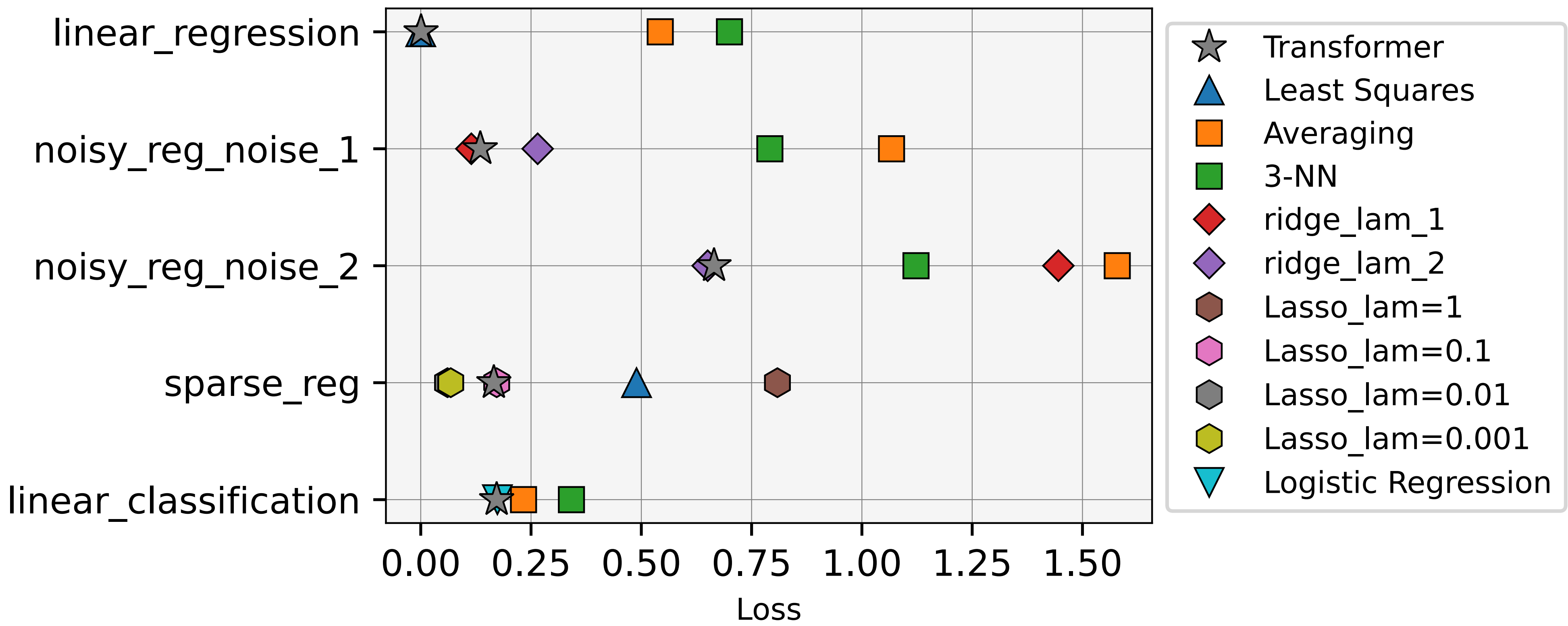
Thm [Generalization for pre-training]: The generalization bound gives

$$\sup_{\boldsymbol{\theta}\in\Theta_{L,M,D',B}} \left| \widehat{L}_{\mathrm{icl}}(\boldsymbol{\theta}) - \mathbb{E}[\widehat{L}_{\mathrm{icl}}(\boldsymbol{\theta})] \right| \leq \mathcal{O}\left(\sqrt{\frac{L^2(MD^2 + DD')\log B}{n}}\right).$$

For example, if π is sparse linear model (LASSO), the overall error with N in-context sample and n meta-training samples gives

$$L_{\mathrm{icl}}(\hat{\boldsymbol{\theta}}) - \sigma^2 = \tilde{\mathcal{O}}\left(\sqrt{\frac{d^4}{n}} + \frac{s\log d}{N}\right)$$

Simulations



Comparison of ICL error of pre-trained transformers. Parameters $D = D' = 64, L = 12, M = 8$, ReLU attention. Use Adam with learning rate $1e-4$, batch size 64 for $300k$ steps.

More surprising experiments

A surprising experiment

- Two meta-tasks:

1. Regression tasks $\hat{\pi}_{\text{reg}}$: $\beta \sim \mathcal{N}(0, I_d/d), \quad y_i = x_i^\top \beta.$
2. Classification tasks $\hat{\pi}_{\text{cls}}$: $\beta \sim \mathcal{N}(0, I_d/d), \quad y_i = \text{Logit}(x_i^\top \beta).$

- Three Transformers (same architecture but different pre-training):

1. Train TF_reg using $\{Z_{\text{reg}}^{(j)}\} \sim_{iid} \hat{\pi}_{\text{reg}}$.
2. Train TF_cls using $\{Z_{\text{cls}}^{(j)}\} \sim_{iid} \hat{\pi}_{\text{cls}}$.
3. Train a TF_unnamed using mixture $\{Z_{\text{reg}}^{(j)}\} \sim_{iid} \hat{\pi}_{\text{reg}}$ and $\{Z_{\text{cls}}^{(j)}\} \sim_{iid} \hat{\pi}_{\text{cls}}$.

- As demonstrated:

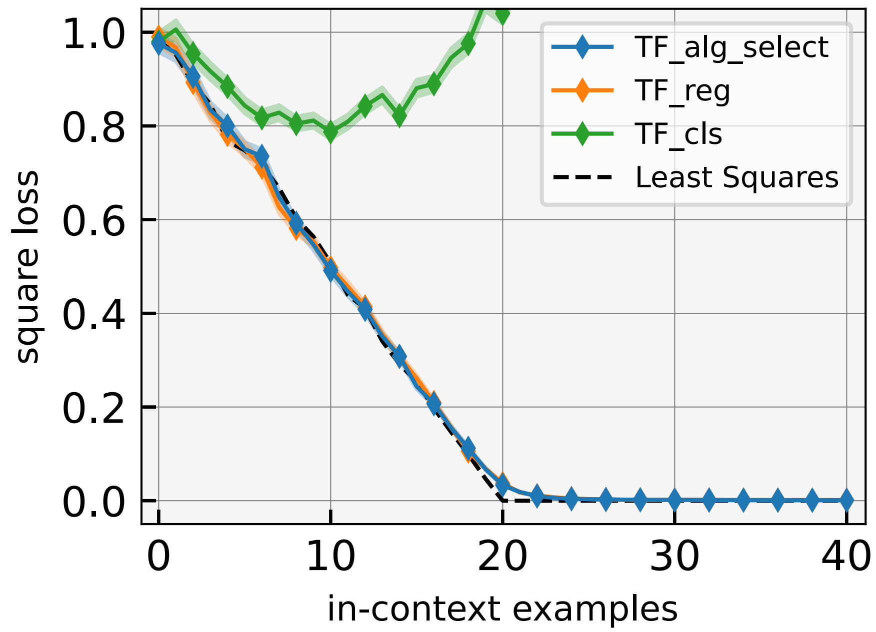
  TF_reg performs linear regression.

  TF_cls performs logistic regression.

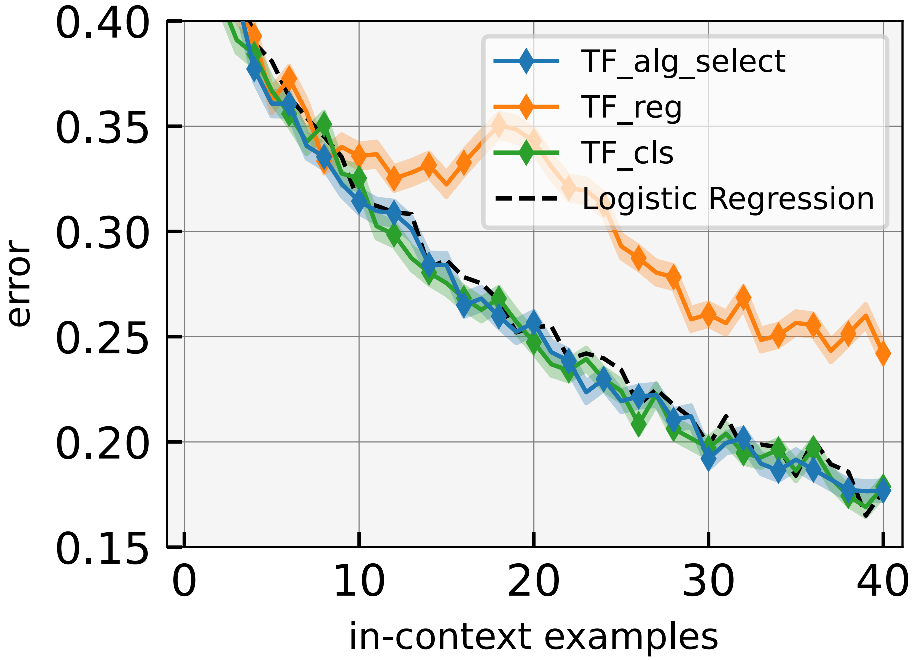
  What algorithm does TF_unnamed perform?

Algorithm selection capability of pre-trained TF

- We give TF_unnamed a name TF_alg_select.



Regression task

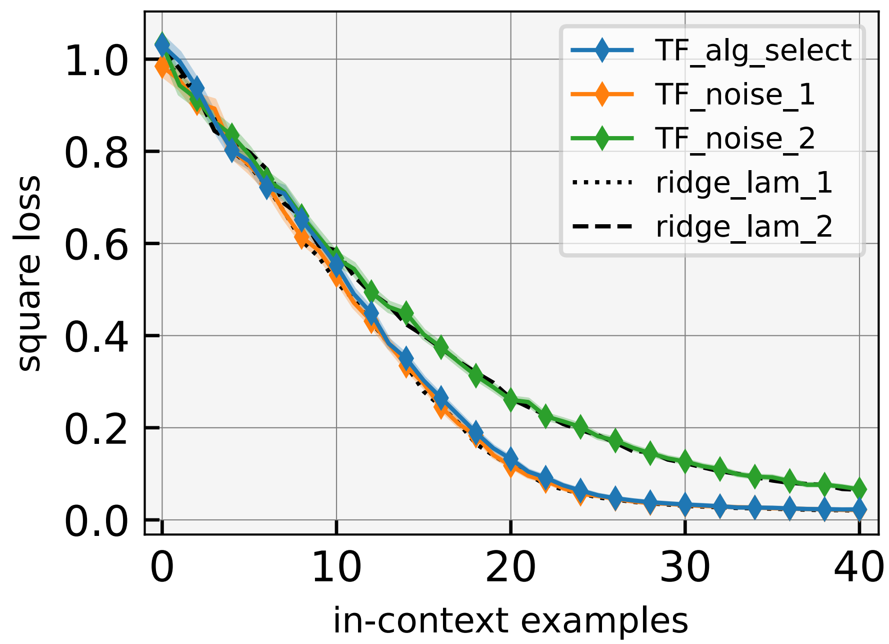


Classification task

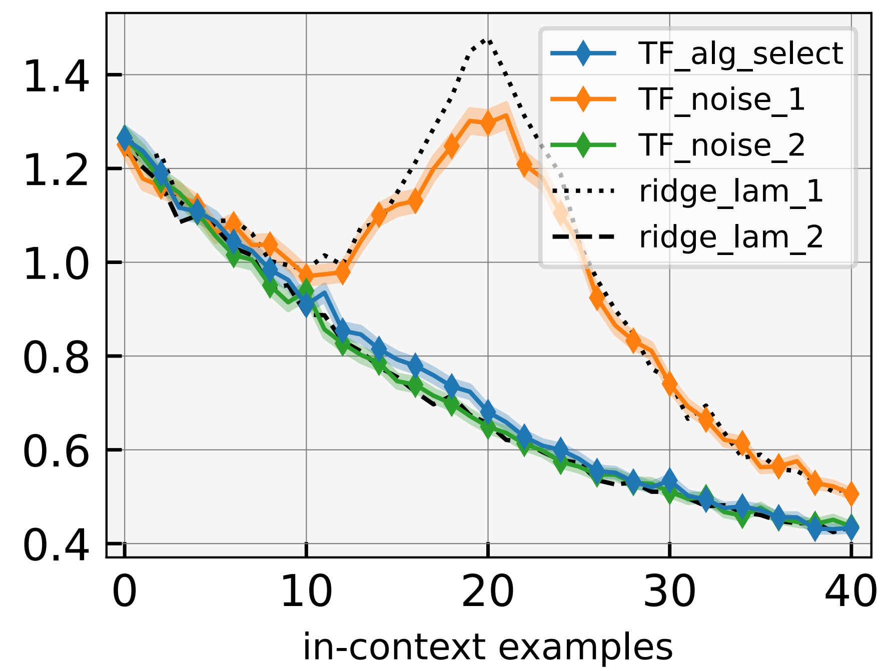
- TF_alg_select selects a proper algorithm on any task.

This is what statisticians do in data analysis

Another surprising experiment



Linear model with small noise



Linear model with large noise

- TF_alg_select selects ridge regression with optimal regularization.

  This is what statisticians do in data analysis

How statisticians perform algorithms selection?

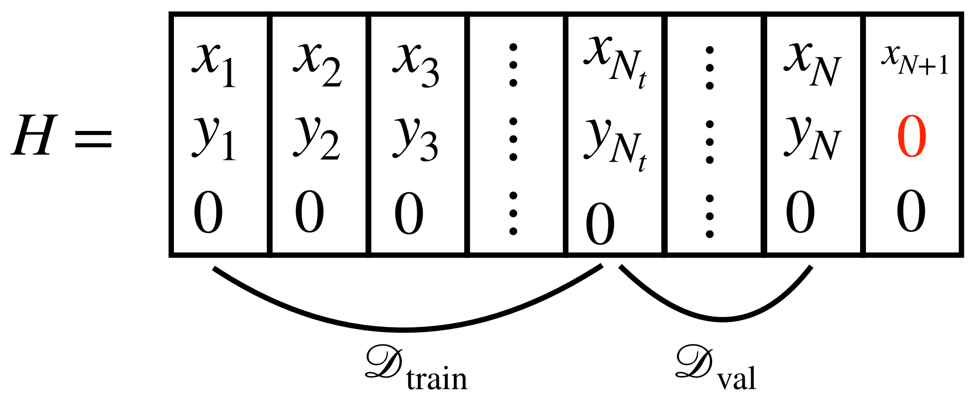
Two strategies:

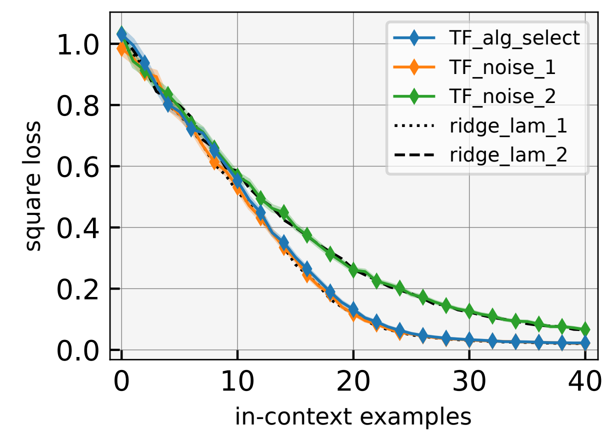
- Strategy 1: Run K algorithms in parallel. Select the algorithm with the smallest validation error.
- Strategy 2: Perform a hypothesis test to select the algorithm.

Transformers efficiently implement:

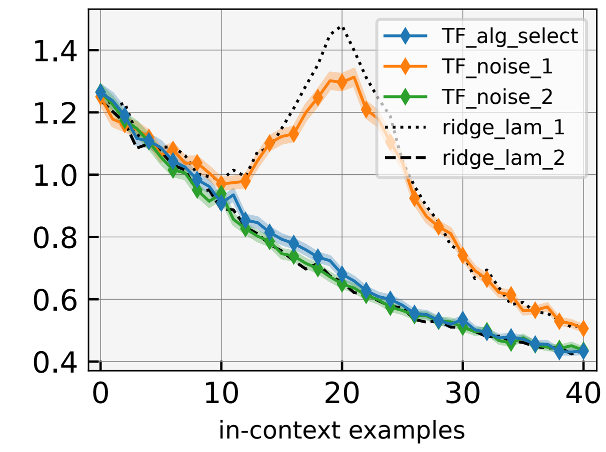
- Mechanism 1: Post-ICL validation.
- Mechanism 2: Pre-ICL testing.

Post-ICL validation mechanism





Linear model with small noise



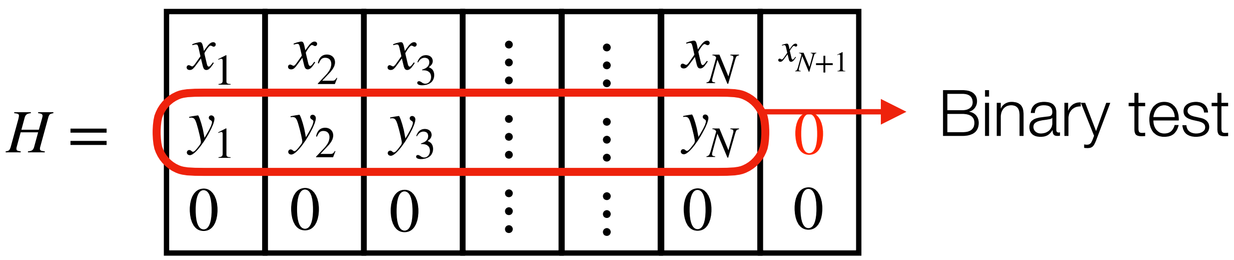
Linear model with large noise

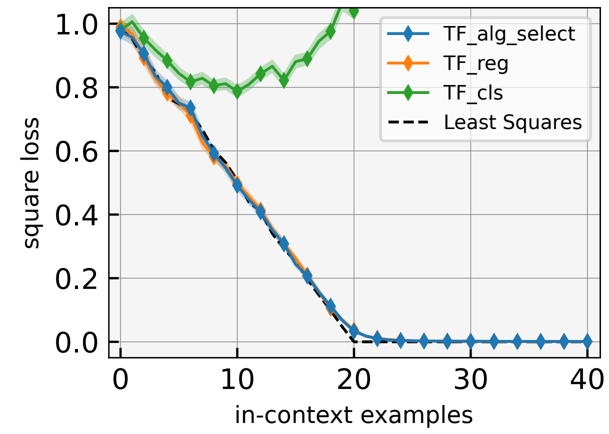
- There exists a transformer that performs the following:

1. Run K different algorithms on $\mathscr{D}_{\text{train}}$, using L steps GD with L layers TF.
2. Choose the prediction function that minimizes the loss on the validation set $\mathscr{D}_{\text{val}}$, and predict $\hat{y}_{N+1}$. This uses in total 3 layers.

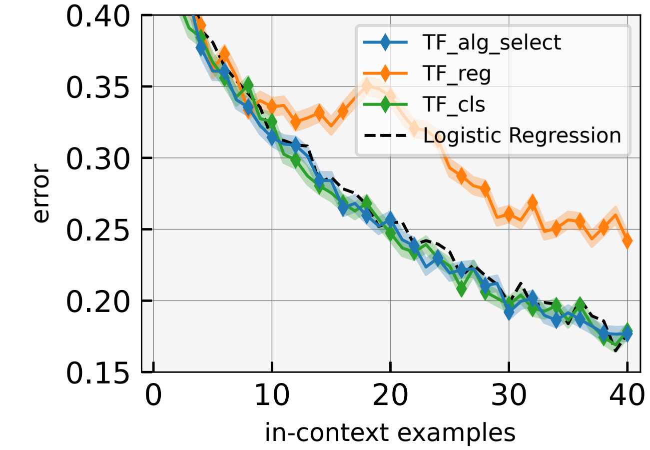
Thm [Select optimal ridge regularization; informal]: For Bayesian linear model with K different noise levels, there exists a transformer that uses the post-ICL validation mechanism to efficiently implement the ridge regression with optimal regularization parameter. Such a transformer can be pre-trained efficiently.

Pre-ICL testing mechanism





Regression task



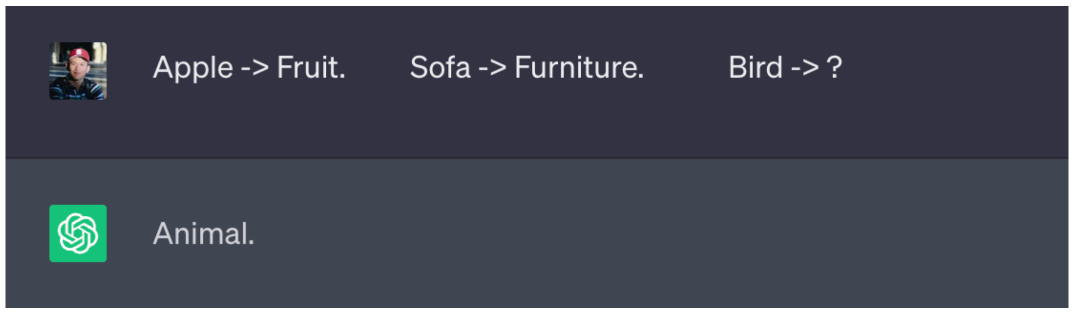
Classification task

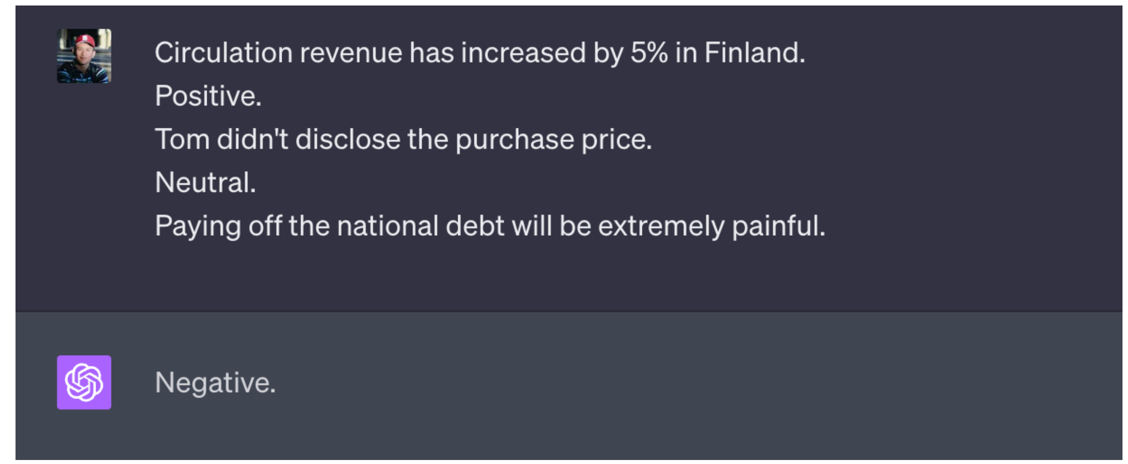
- There exists a transformer that performs the following:

1. Perform a simple hypothesis test on the dataset (binary test/correlation test, etc) and decide which algorithm to choose. This uses constant depth TF.
2. Run the selected algorithm on the whole dataset, and predict $\hat{y}_{N+1}$.

Thm [Select regression vs classification; informal]: There exists a TF that uses the pre-ICL testing mechanism to efficiently implement linear regression on regression tasks, and implement logistic regression on classification tasks. Such a transformer can be pre-trained efficiently.

Summary





- Transformers can efficiently implement basic ICL algorithms using the gradient descent mechanism.
- Transformers can efficiently implement algorithm selection (pre-ICL validation, post-ICL testing), similar to a statistician.
- Such transformers can be pre-trained statistically efficiently.
- Paper will come out this week.