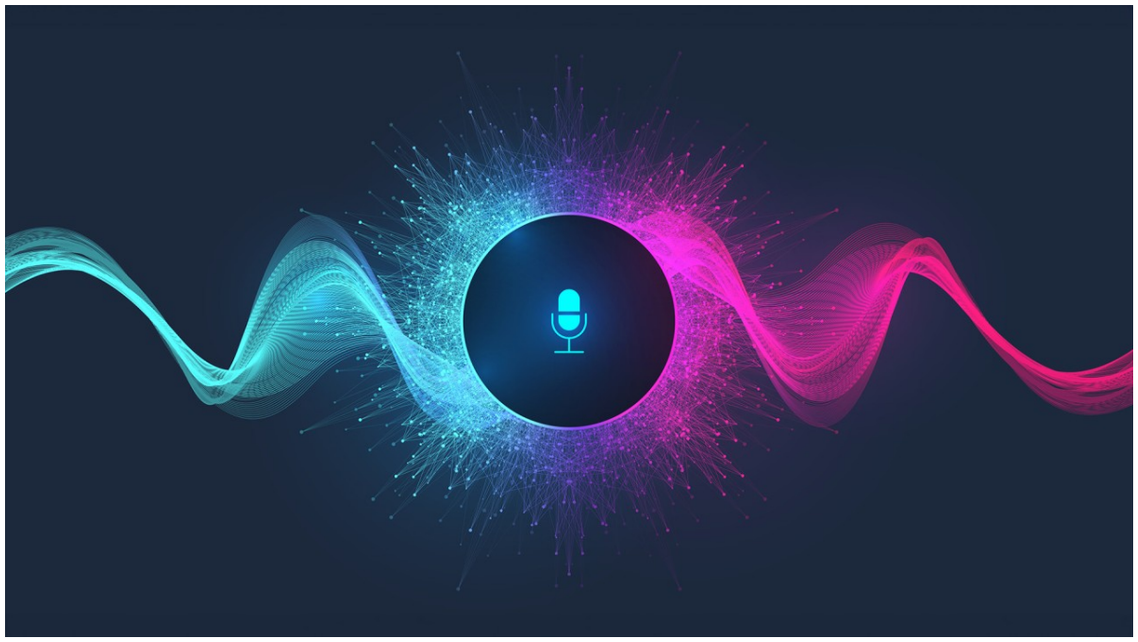
From Markov to Laplace:

A Markovian Tale of Large Language Models (LLMs)

Ashok Vardhan Makkuva

Joint work with Chanakya Ekbote, Marco Bondaschi, Nived Rajaraman, Adway Girish, Alliot Nagle, Martin Jaggi, Hyeji Kim, and Michael Gastpar

LLMs are part of daily lives



Not fully Trustworthy

Impressive language skills

Programming

Arithmetic reasoning

Hallucinations

Need of the hour

LLM Interpretability

What do they learn?

How do they learn?

Challenges

Challenges

- Complex and opaque
- How/why do they work?
- Expensive

Too many tuning knobs

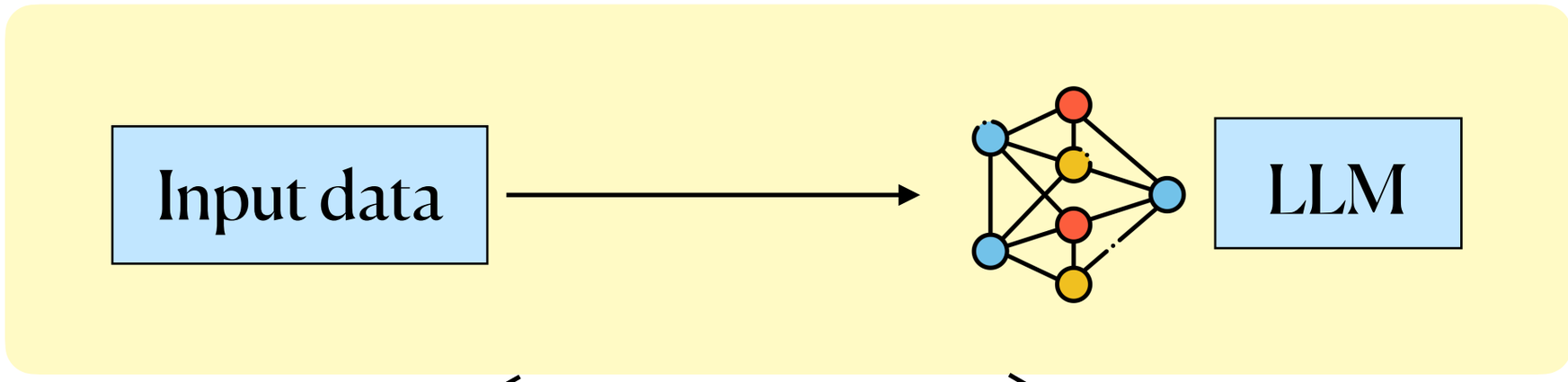
$63M

Need

- Complex and opaque —Principles→ Clarity & scientific understanding
- How/why do they work? —Transparency→ Core learning mechanisms
- Expensive —Efficiency→ Lighter & efficient models, algos.

LLM Interpretability

State of the art



What data can they represent?

How do they learn them?

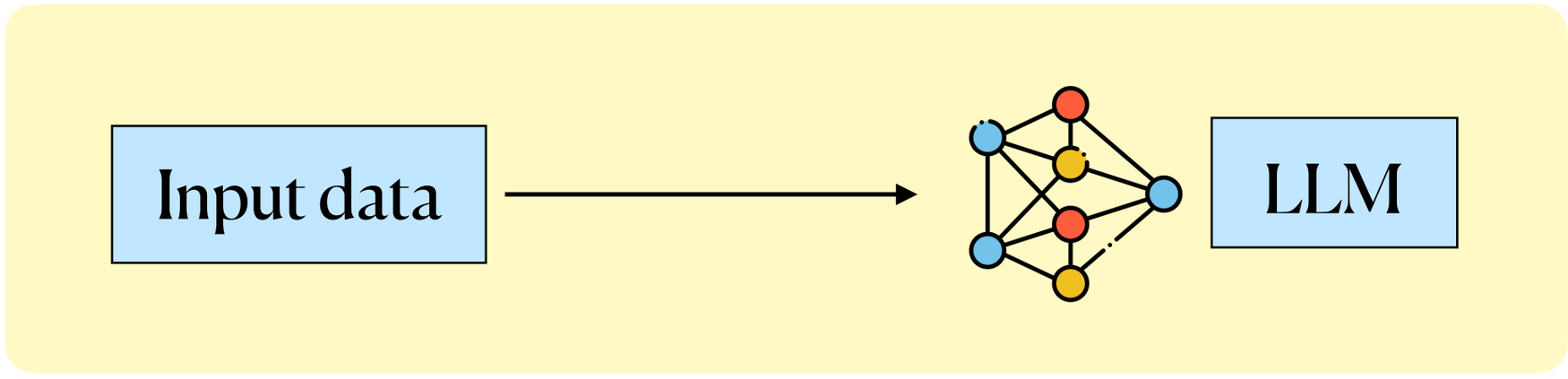
- Key limitation
  - **Lack generality:** specific data and models (e.g. transformers), restrictive assumptions

[Yun et al. 19, Perez et al. 20, Liu et al. 23, Weiss et al. 21, Merrill & Sabharwal 22-24, Li et al. 24, Wei et al. 24, Nichani et al. 24, Bietti et al. 23, Guo et al. 24, Edelman et al. 24, Li et al. 23, Wang et al. 25]

Our approach

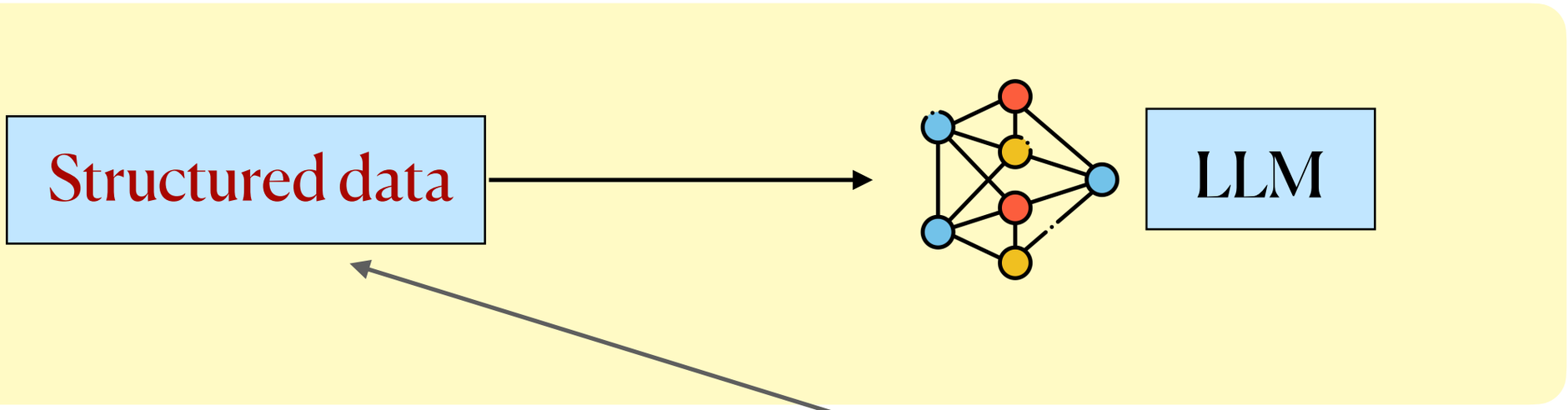
LLM Interpretability via Structured Data

Our approach



LLM Interpretability via Structured Data

Our approach

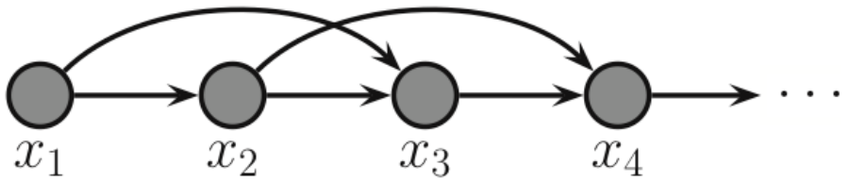


LLM Interpretability via Structured Data

Our approach

Structured data → LLM

Markov:



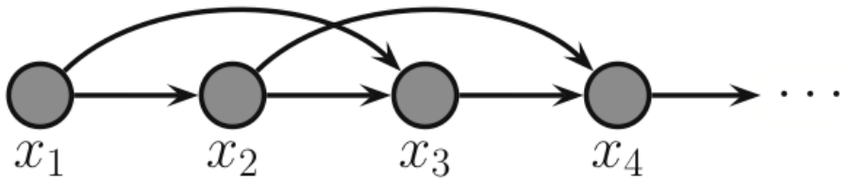
Mathematically controlled

Insightful → complex real-world data

Our approach

Structured data → LLM

Markov:



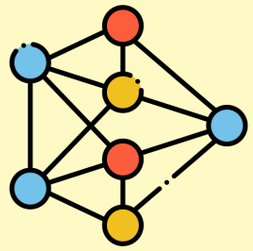
Statistics & Information theory

well studied

optimal estimators

Our approach

Structured data

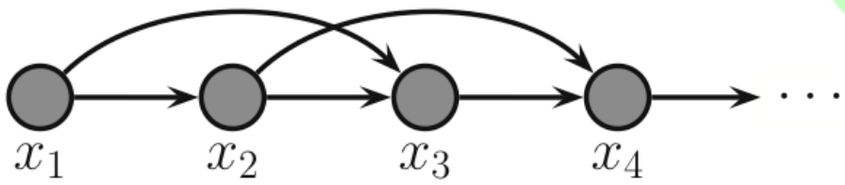


LLM

How does the model learn?

Minimal size needed?

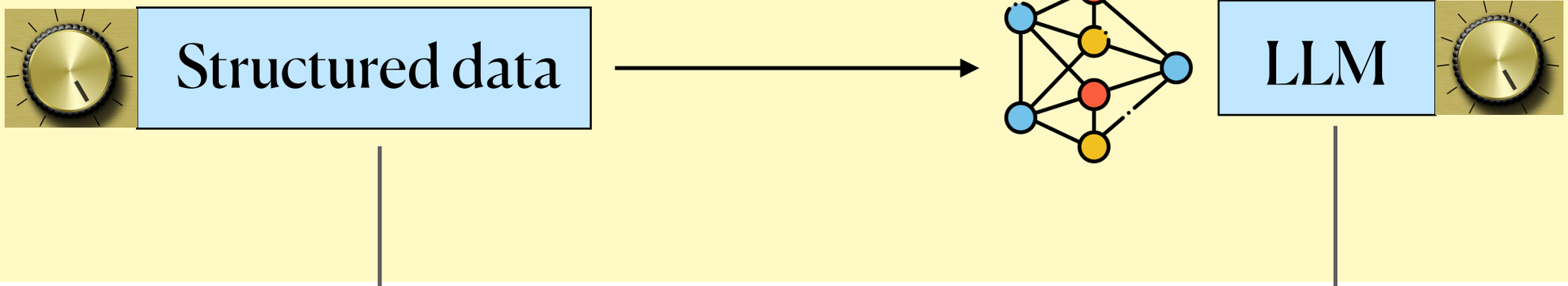
Markov:



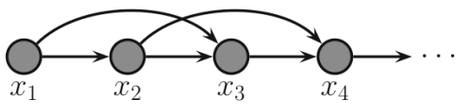
optimal estimators

Statistics & Information theory

This talk



Markov processes

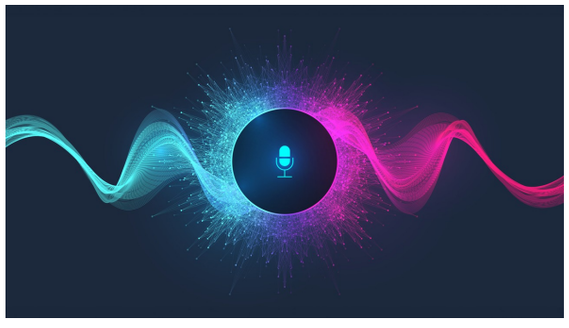


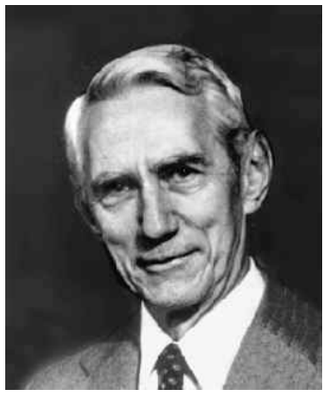
Transformers

State-Space Models

Why Markovian?

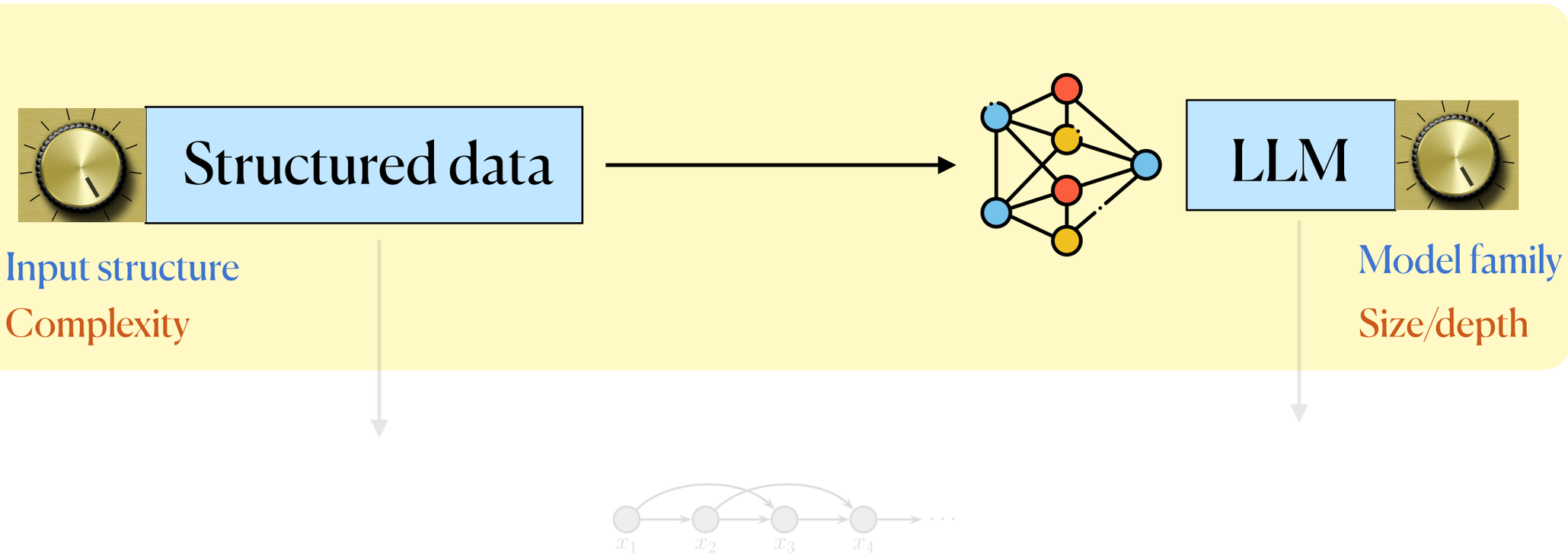
Markov processes





Shannon, 1948

Key features

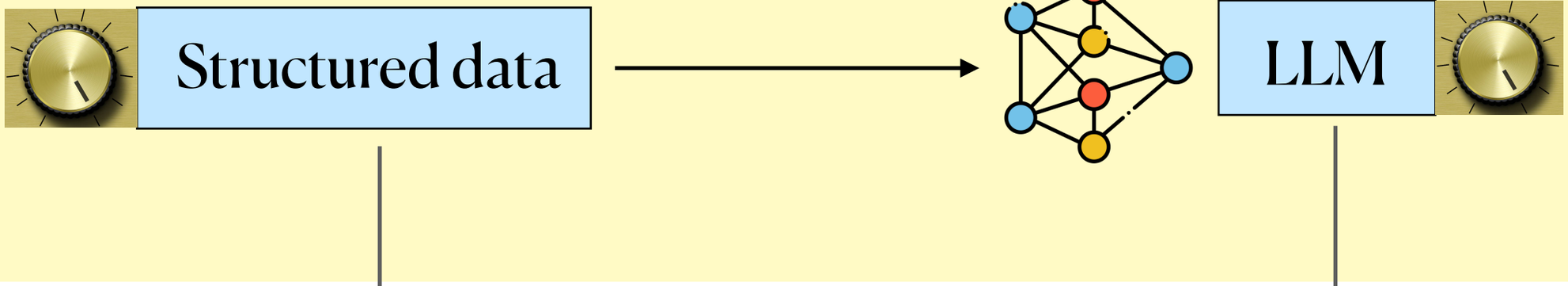


- Comprehensive: broad family of (data, models) ⟶ compare various architectures
- Fine-grained analysis

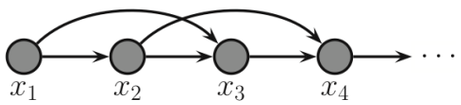
What do they learn?

How do they learn?

This talk



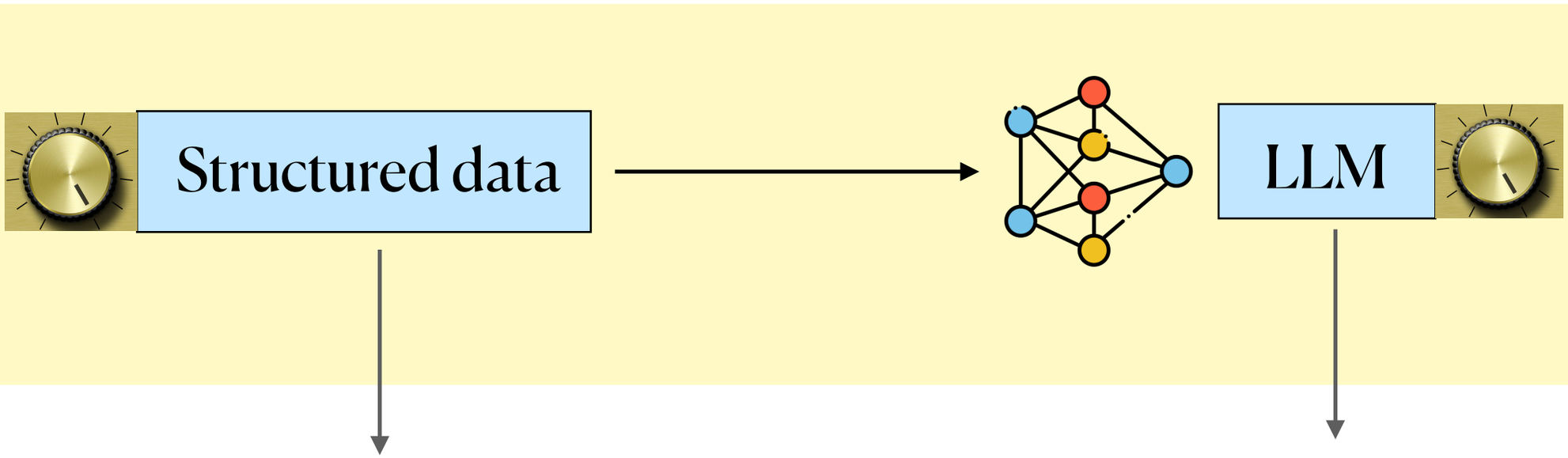
Markov processes



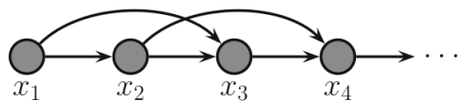
Transformers

State-Space Models

Part I

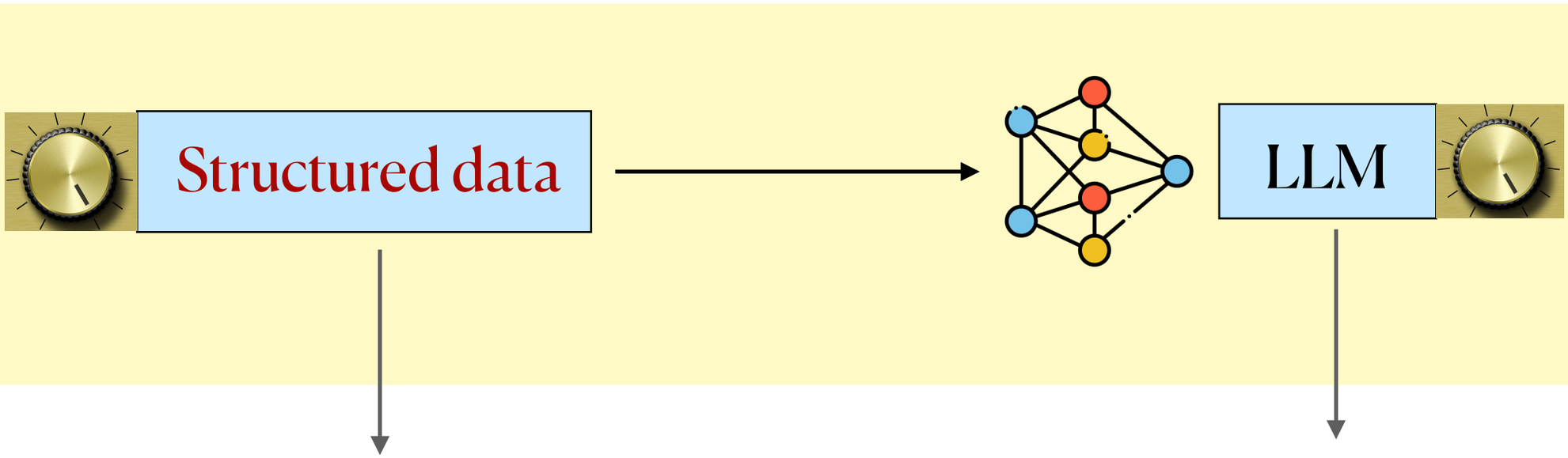


Markov processes

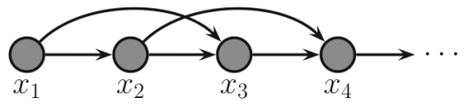


Transformers

Part II

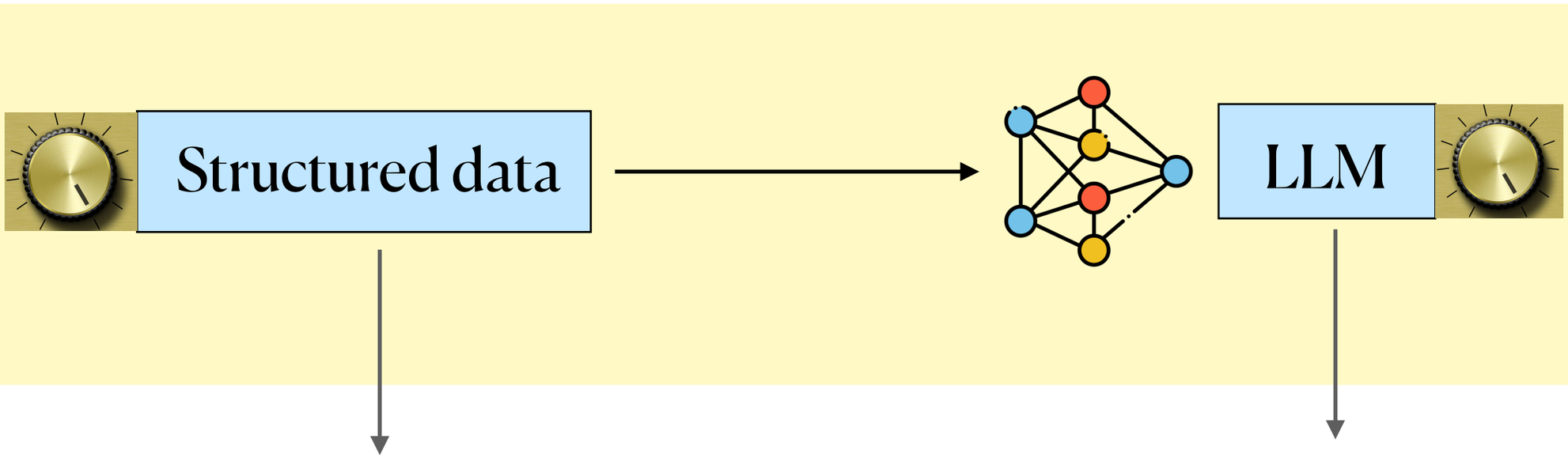


Markov processes

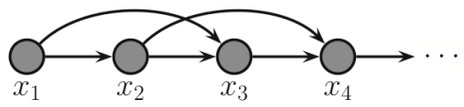


State-Space Models

Part I

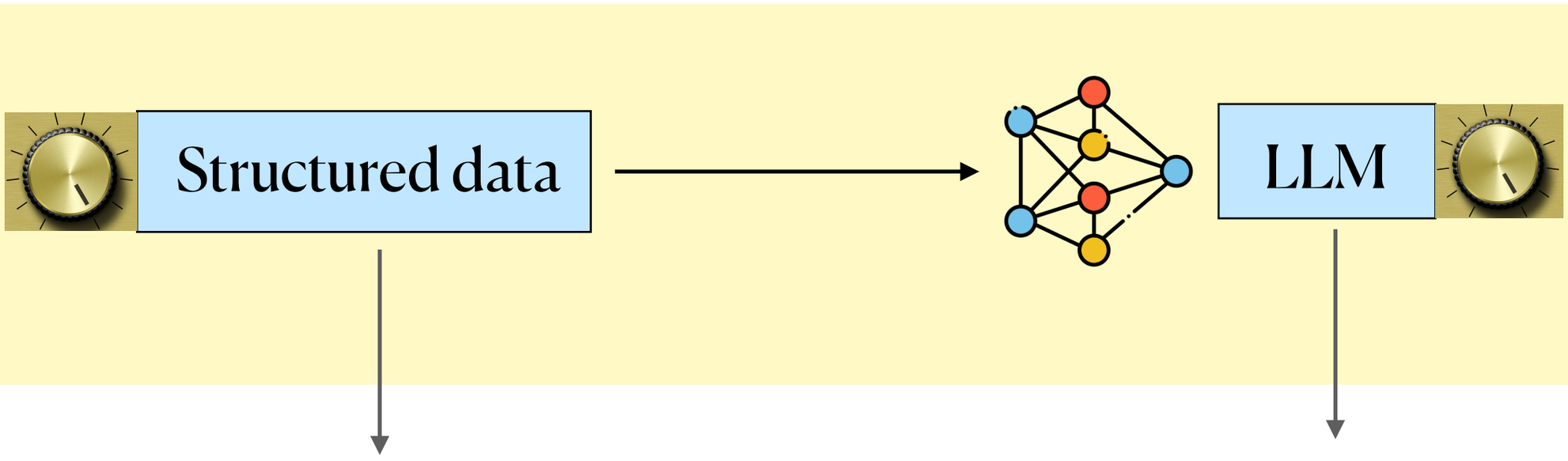


Markov processes

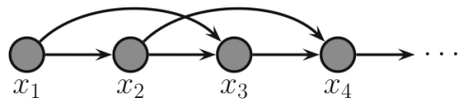


Transformers

Part I



Markov processes



Transformers

Attention with Markov

Attention with Markov

Markovian inputs

Memory

Transformers

Depth

What do they learn?

How do they learn?

Depth and non-linearity play a crucial role in transformer functionality

1-layer transformer sometimes fails to learn even first-order Markov chains! *

2-layer transformer can learn Markov chains of all orders!

What One Cannot Two Can: Two-Layer Transformers Provably Represent Induction Heads on Any-Order Markov Chains

NeurIPS 2025 Spotlight

Attention with Markov: A Curious Case of Single-layer Transformers

ICLR 2025 Spotlight

Local to Global: Learning Dynamics and Effect of Initialization for Transformers

NeurIPS 2024

Transformers on Markov Data: Constant Depth Suffices

NeurIPS 2024

Attention with Markov

Markovian inputs

Transformers

Memory

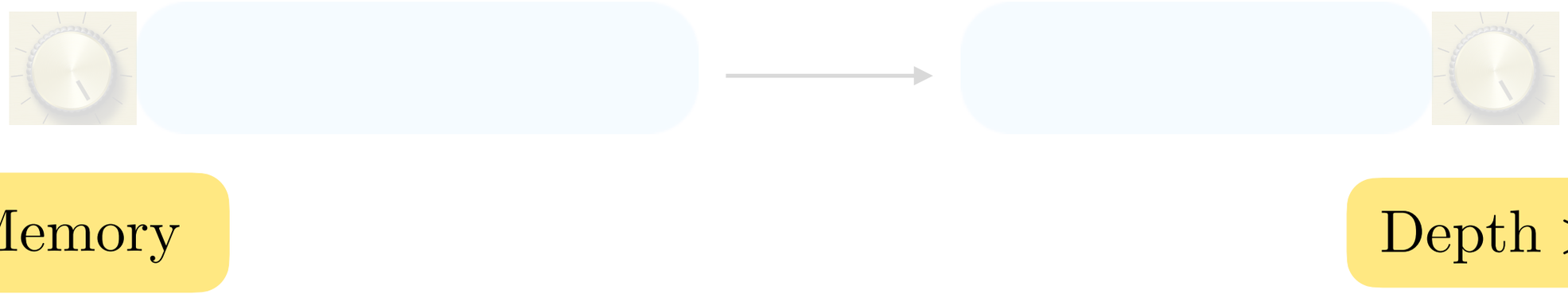
Depth

Part I.a

Memory = 1

Depth = 1

Part I.b



Part I.a

Memory = 1

Depth = 1

Part I.a

Markovian inputs

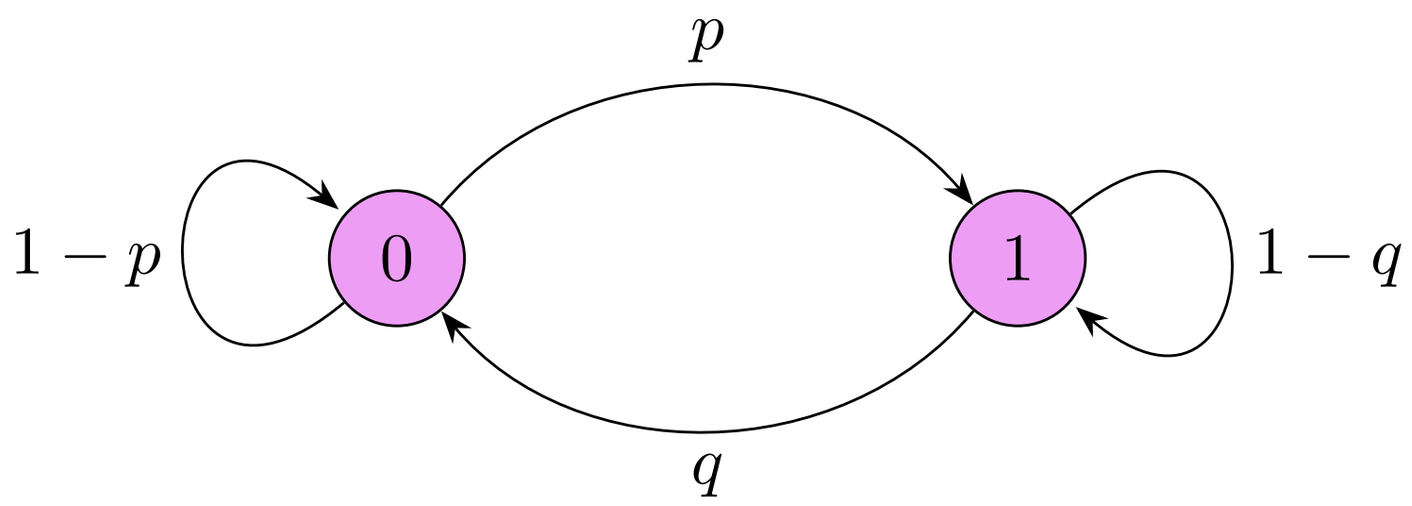
Memory = 1

Depth = 1

Input data: First-order Markov chain

Input data: First-order Markov chain

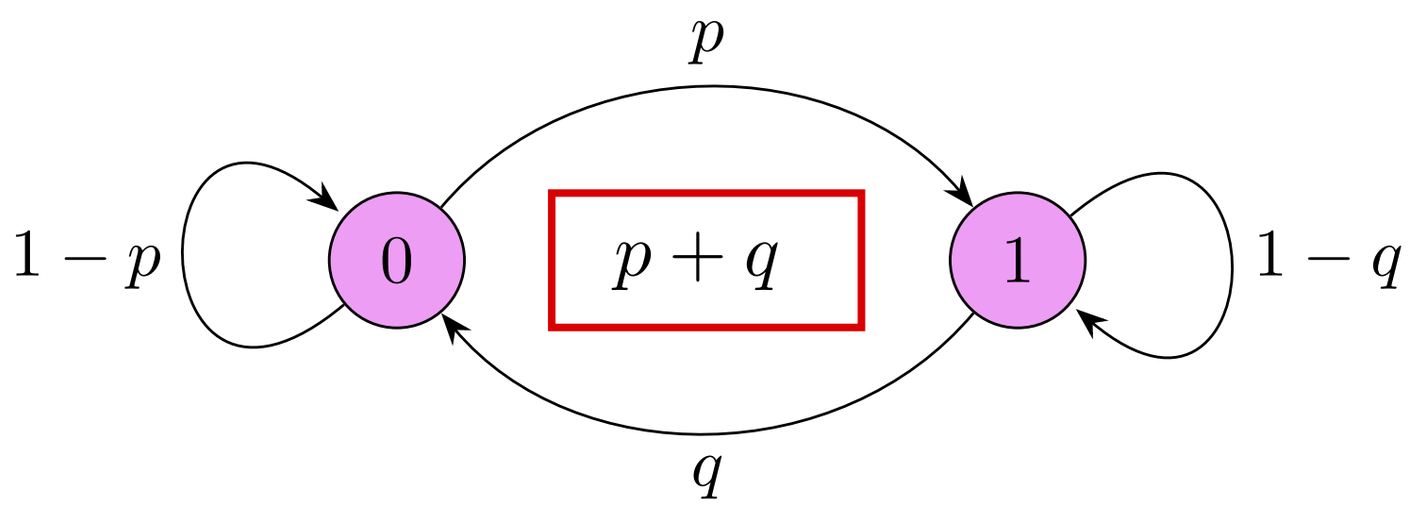
$$(x_n)_{n\geq 1} \sim (\boldsymbol{\pi}, \boldsymbol{P}) \quad \Leftrightarrow$$



$$\boldsymbol{\pi} = (\pi_0, \pi_1) = \left(\frac{q}{p+q}, \frac{p}{p+q} \right), \qquad \boldsymbol{P} = (\boldsymbol{P}_{ij}) = \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix}.$$

Input data: First-order Markov chain

$$(x_n)_{n\geq 1} \sim (\boldsymbol{\pi}, \boldsymbol{P}) \quad \Leftrightarrow$$



$$\boldsymbol{\pi} = (\pi_0, \pi_1) = \left(\frac{q}{p+q}, \frac{p}{p+q} \right), \quad \boldsymbol{P} = (\boldsymbol{P}_{ij}) = \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix}.$$

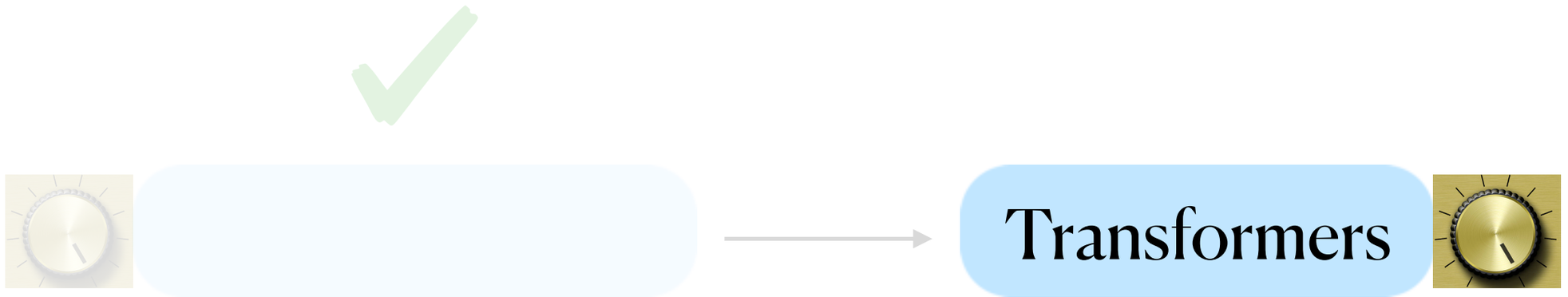
Part I.a

Markovian inputs

Memory = 1

Depth = 1

Part I.a



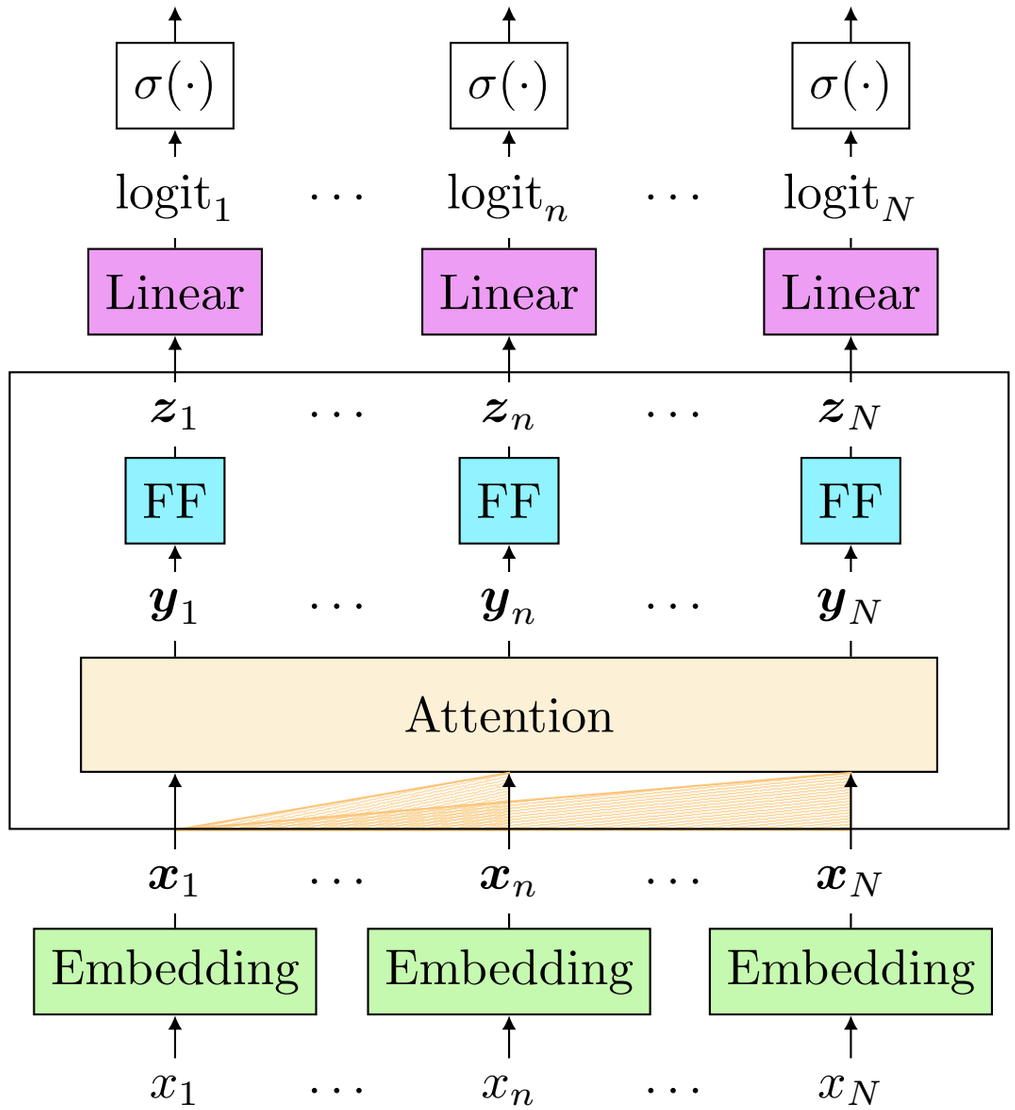
Memory = 1

Depth = 1

Transformers

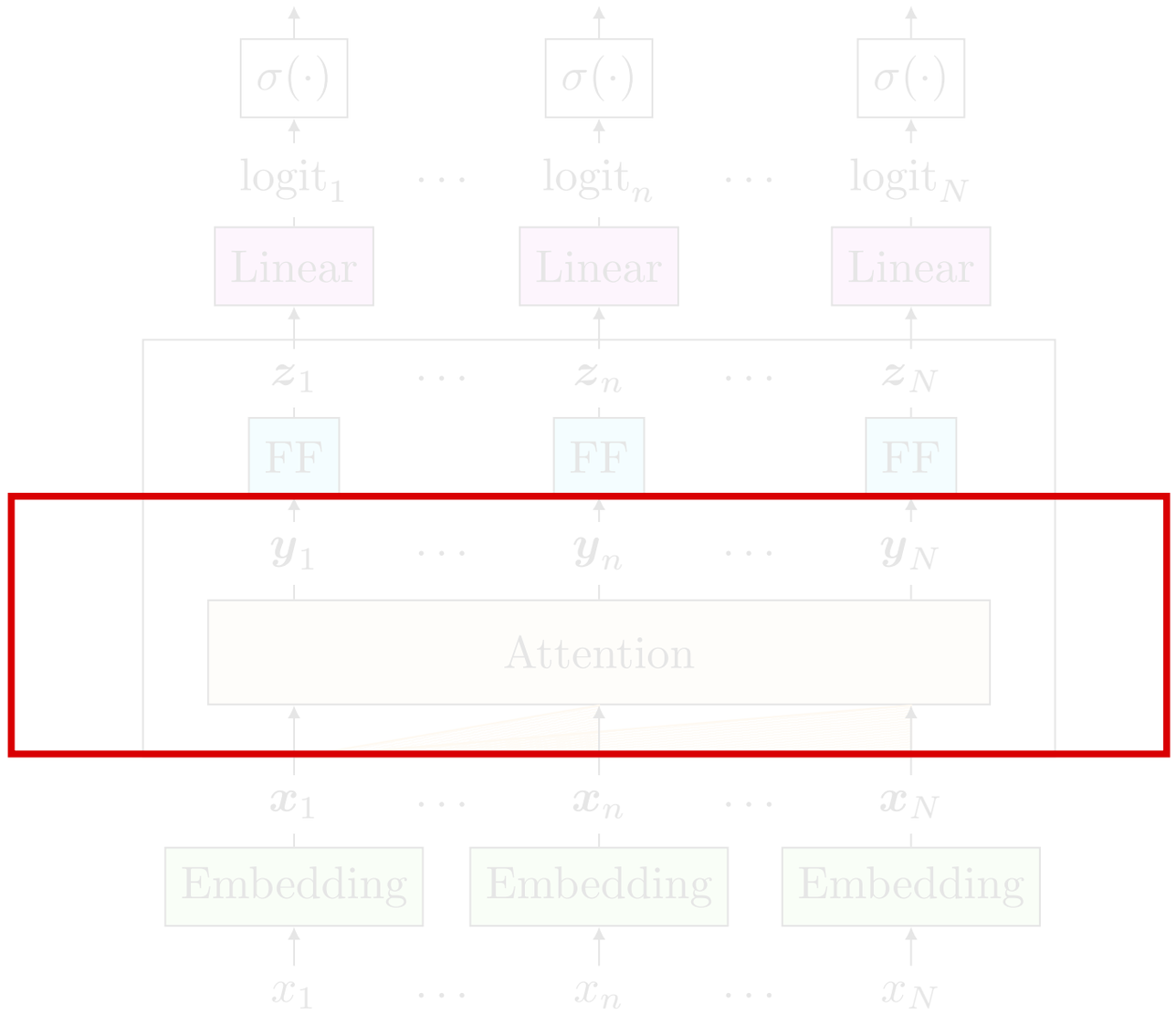
Transformers

Transformers

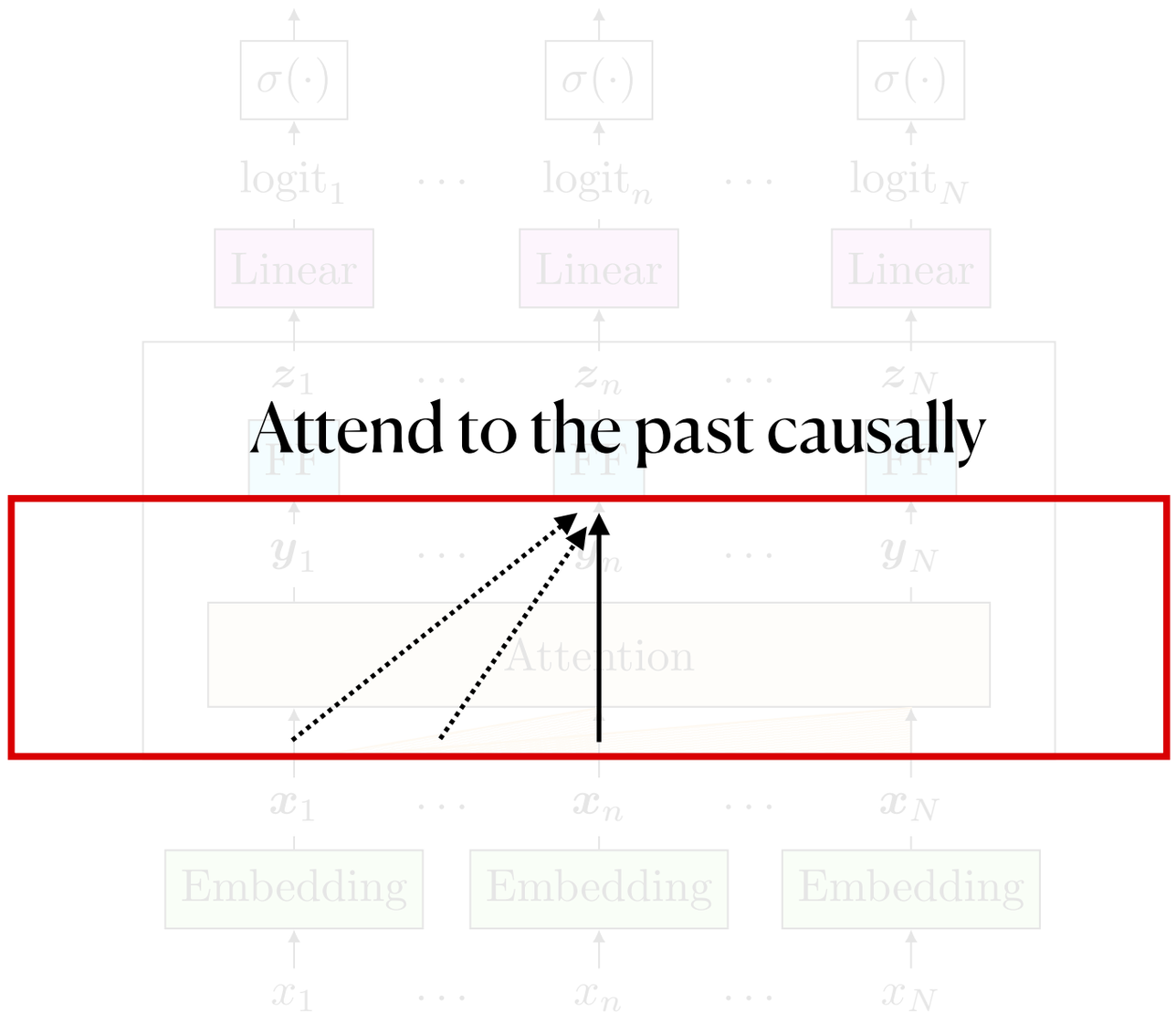


(Vaswani et. al., 2017)

Transformers



Transformers



Attend to the past causally

Single-layer transformer

$$f_{\boldsymbol{\theta}}(x_1^n) = \mathbb{P}_{\boldsymbol{\theta}}(x_{n+1} = 1 \mid x_1^n) = \sigma(\text{logit}_n)$$

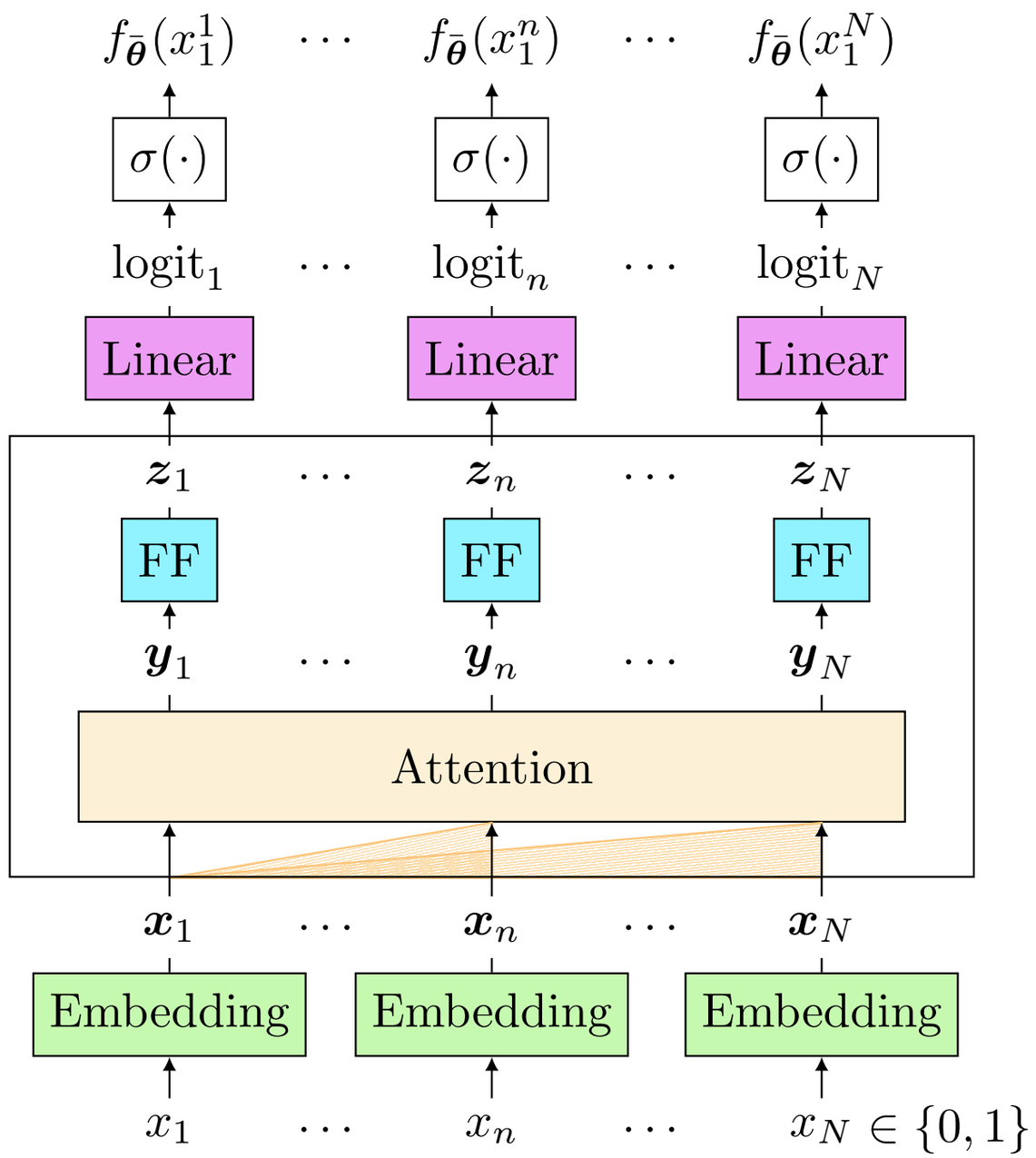
$$\text{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

$$\boldsymbol{z}_n = \boldsymbol{y}_n + \boldsymbol{W}_2 \,\text{ReLU}(\boldsymbol{W}_1 \boldsymbol{y}_n)$$

$$\boldsymbol{y}_n = \boldsymbol{x}_n + \boldsymbol{W}_O \sum_{i=1}^{n} \text{att}_{n,i} \cdot \boldsymbol{W}_V \boldsymbol{x}_i$$

$$\boldsymbol{x}_n = x_n \cdot \boldsymbol{e} + \boldsymbol{p}_n$$

$\boldsymbol{\theta}$



Single-layer transformer

$$f_{\boldsymbol{\theta}}(x_1^n) = \mathbb{P}_{\boldsymbol{\theta}}(x_{n+1} = 1 \mid x_1^n) = \sigma(\text{logit}_n)$$

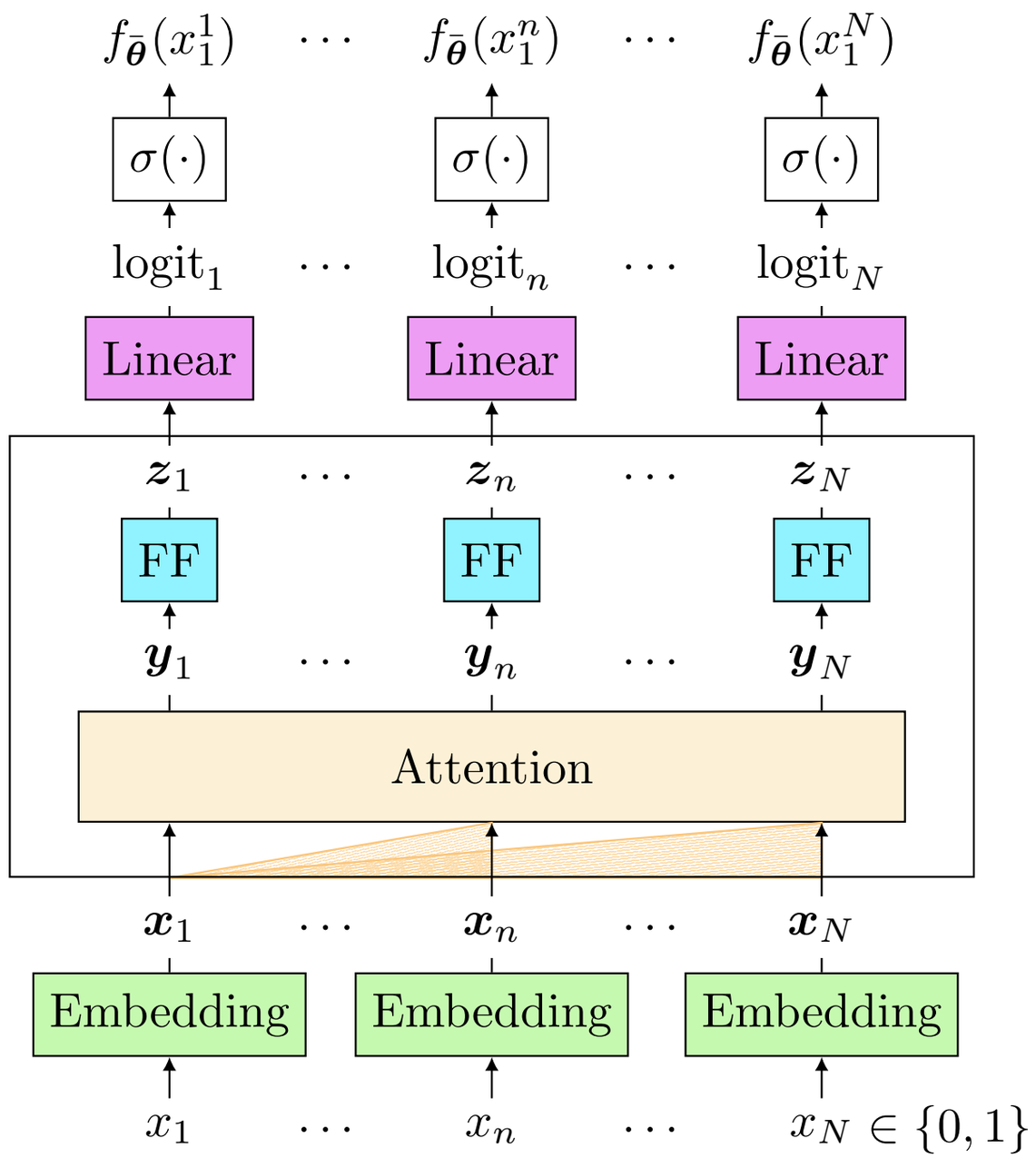
$$\text{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

$$\boldsymbol{z}_n = \boldsymbol{y}_n + \boldsymbol{W}_2 \,\text{ReLU}(\boldsymbol{W}_1 \boldsymbol{y}_n)$$

$$\boldsymbol{y}_n = \boldsymbol{x}_n + \boldsymbol{W}_O \sum_{i=1}^{n} \text{att}_{n,i} \cdot \boldsymbol{W}_V \boldsymbol{x}_i$$

$$\boldsymbol{x}_n = x_n \cdot \boldsymbol{e} + \boldsymbol{p}_n$$

$\boldsymbol{\theta}$



Single-layer transformer

$$f_{\boldsymbol{\theta}}(x_1^n) = \mathbb{P}_{\boldsymbol{\theta}}(x_{n+1} = 1 \mid x_1^n) = \sigma(\text{logit}_n)$$

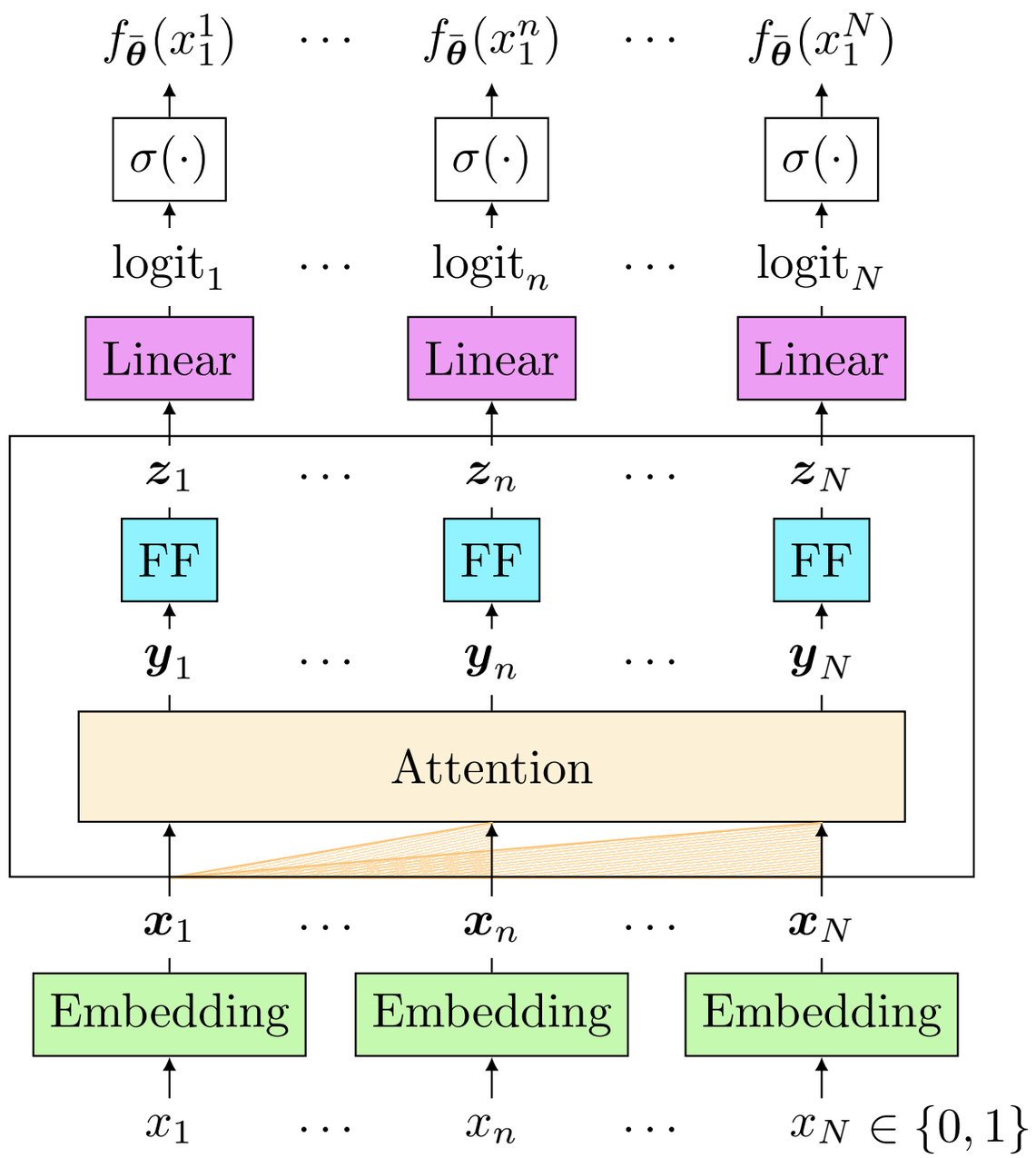
$$\text{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

$$\boldsymbol{z}_n = \boldsymbol{y}_n + \boldsymbol{W}_2 \,\text{ReLU}(\boldsymbol{W}_1 \boldsymbol{y}_n)$$

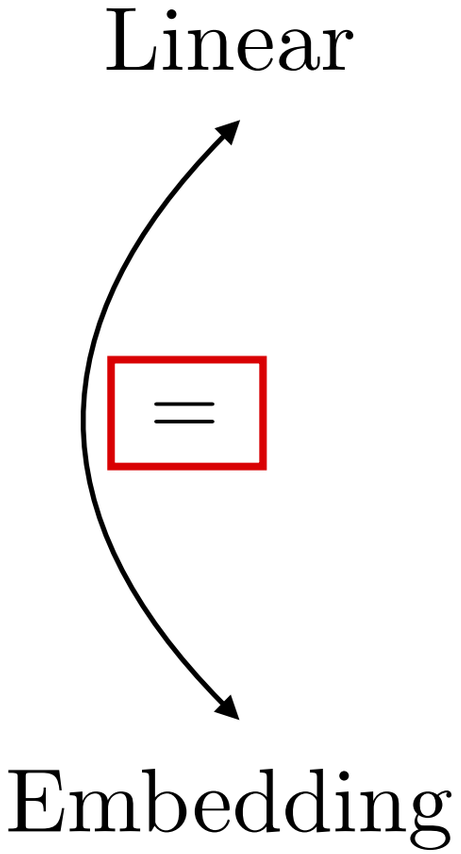
$$\boldsymbol{y}_n = \boldsymbol{x}_n + \boldsymbol{W}_O \sum_{i=1}^{n} \text{att}_{n,i} \cdot \boldsymbol{W}_V \boldsymbol{x}_i$$

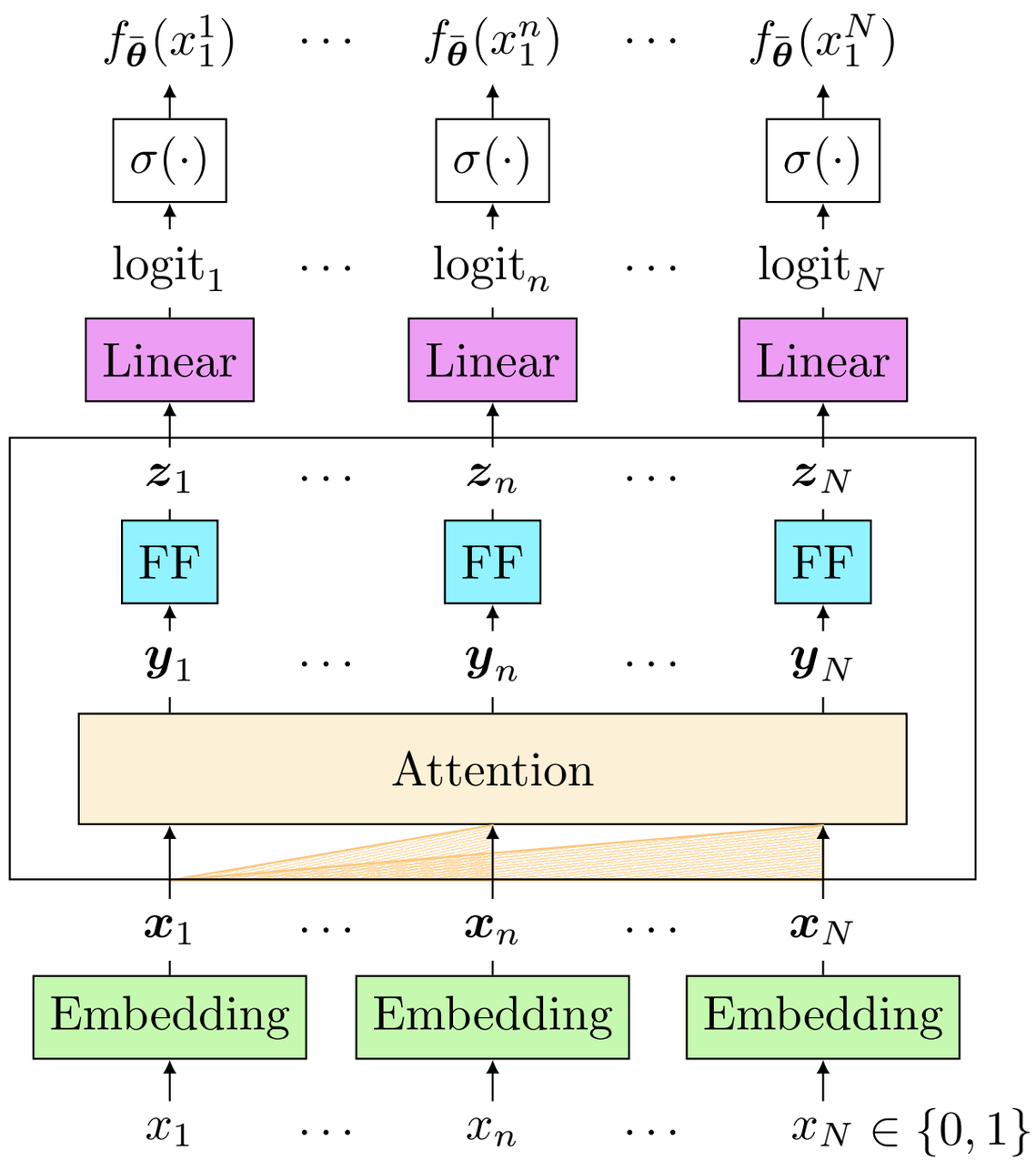
$$\boldsymbol{x}_n = x_n \cdot \boldsymbol{e} + \boldsymbol{p}_n$$

$\boldsymbol{\theta}$



Weight tying

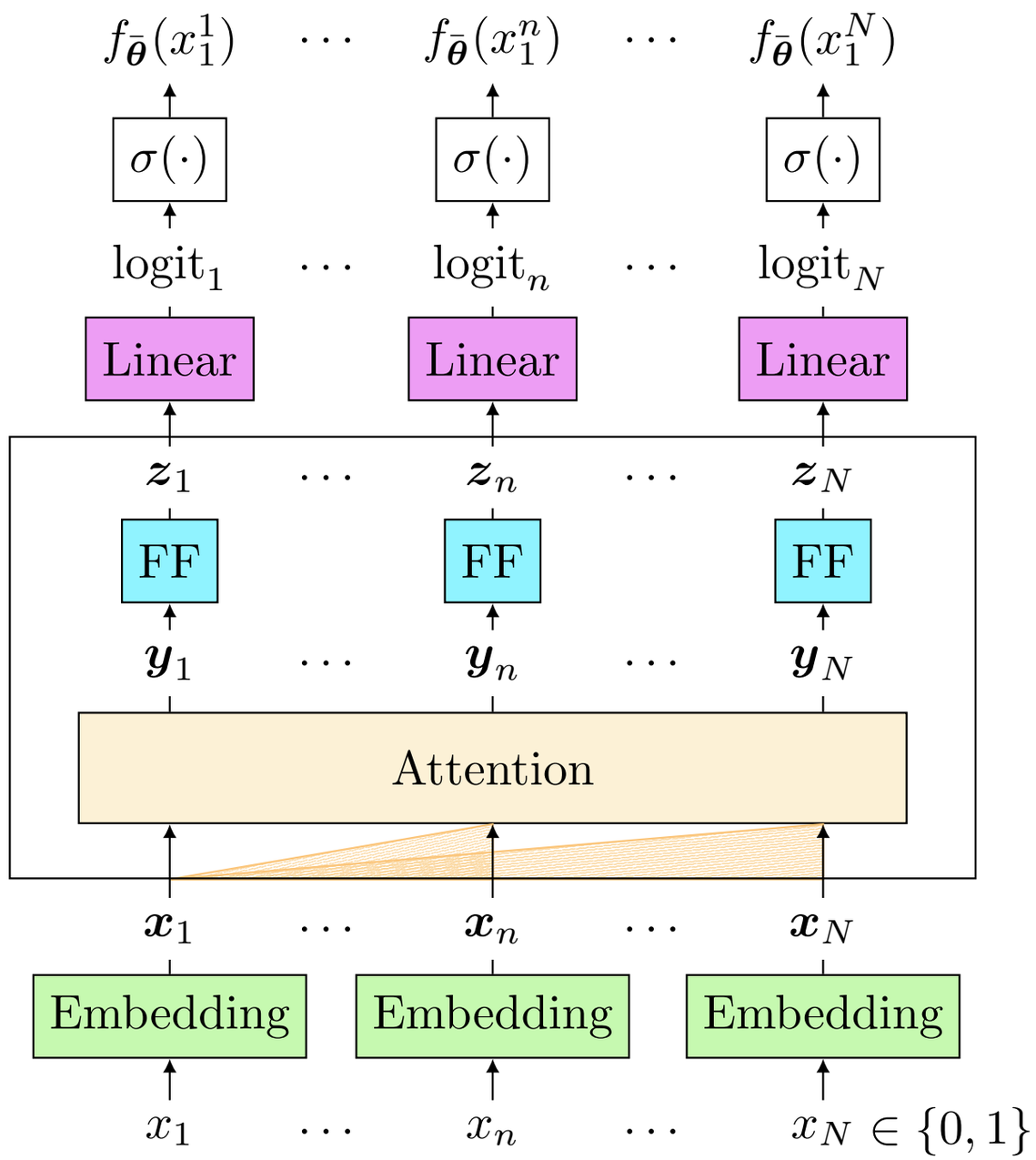




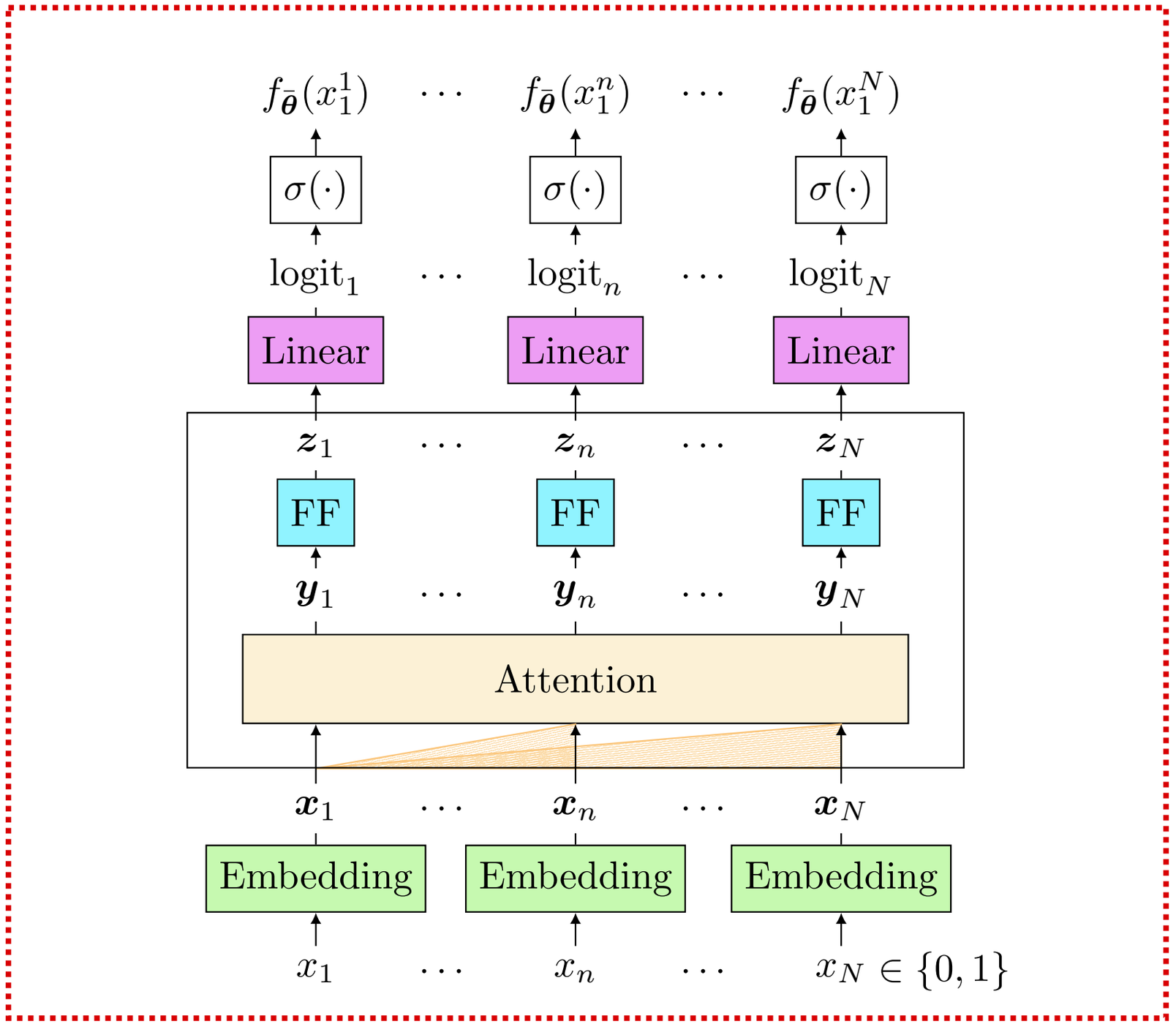
No weight tying

Linear

Embedding

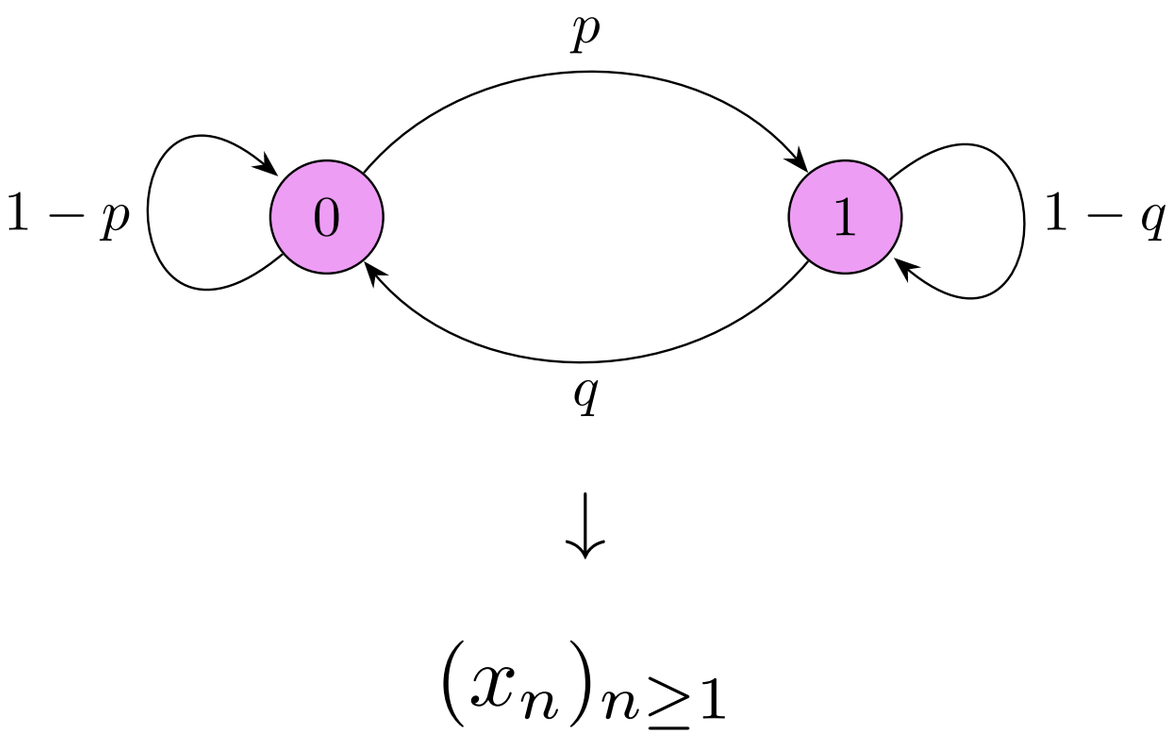


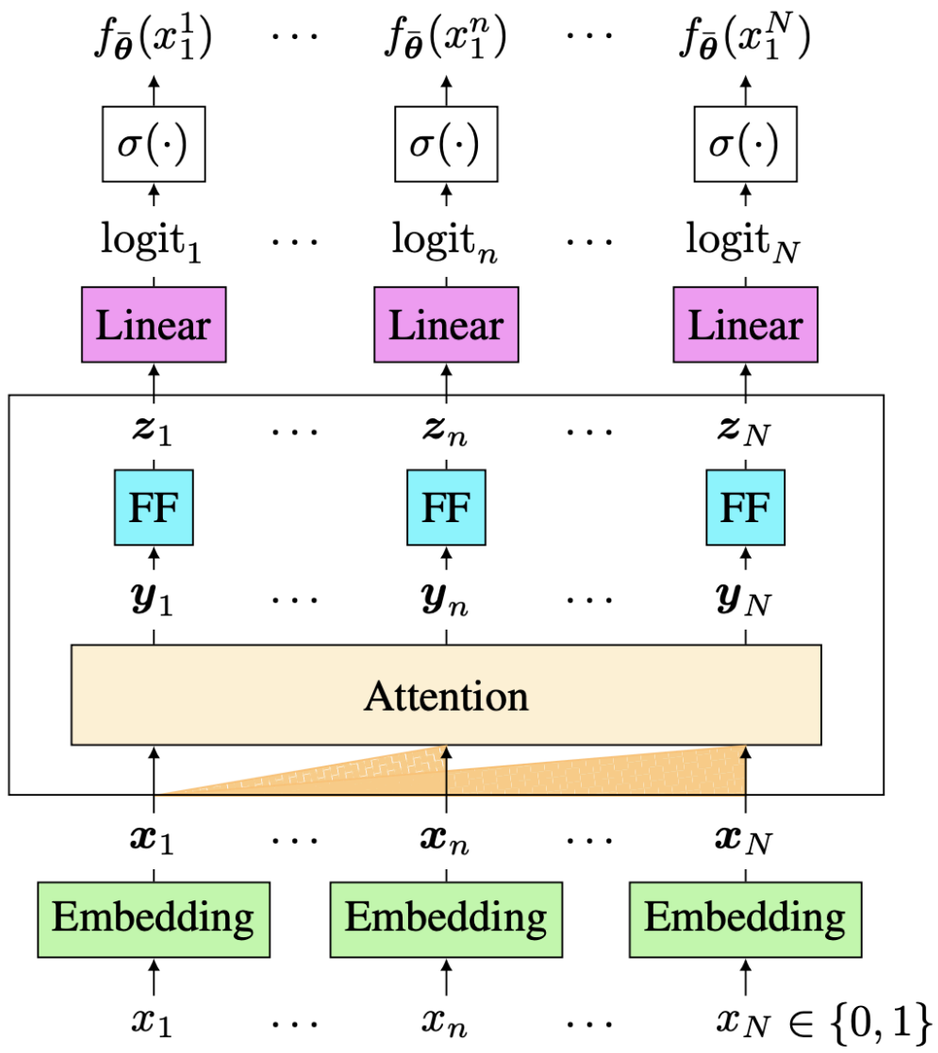
Single-layer transformer



$\boldsymbol{\theta}$

First-order Markov chain + Single-layer transformer





Input data: First-order Markov chain

Model: Depth = 1

Next-token prediction loss

Next-token prediction loss

$$L(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{x_1^{n+1}} \left[x_{n+1} \cdot \log f_{\boldsymbol{\theta}}(x_1^n) + (1 - x_{n+1}) \cdot \log(1 - f_{\boldsymbol{\theta}}(x_1^n)) \right]$$

Cross-entropy loss between x_{n+1} and prediction probability $f_{\boldsymbol{\theta}}(x_1^n)$

Ideally...

Prediction probability

Markov kernel

$$L(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{x_1^{n+1}}\left[x_{n+1} \cdot \log f_{\boldsymbol{\theta}}(x_1^n) + (1 - x_{n+1}) \cdot \log(1 - f_{\boldsymbol{\theta}}(x_1^n))\right]$$

$$f_{\boldsymbol{\theta}}(x_1^n) \approx \mathbb{P}(x_{n+1} = 1 \mid x_n)$$

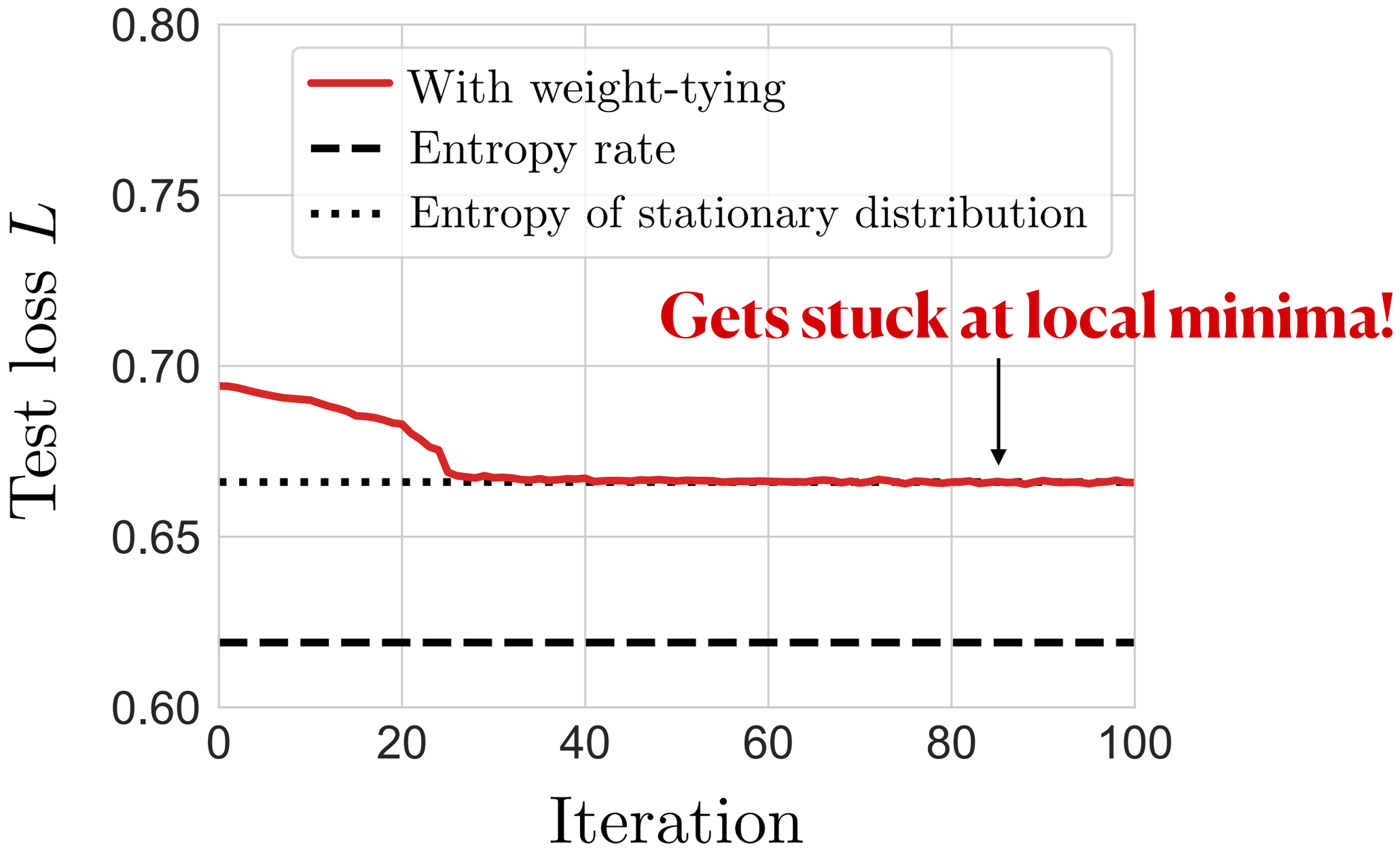
x_{n+1} $f_{\boldsymbol{\theta}}(x_1^n)$

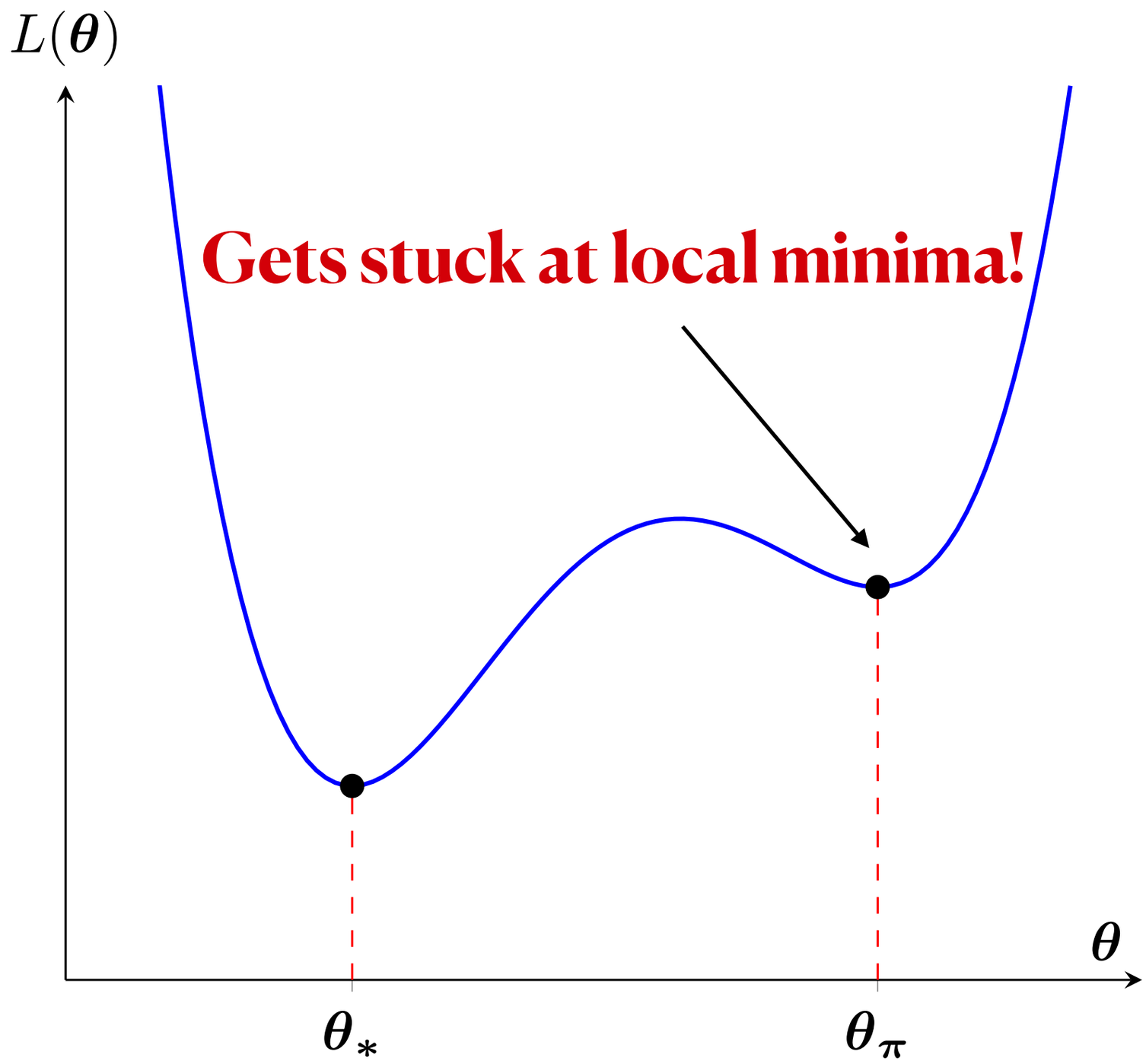
But...

But...

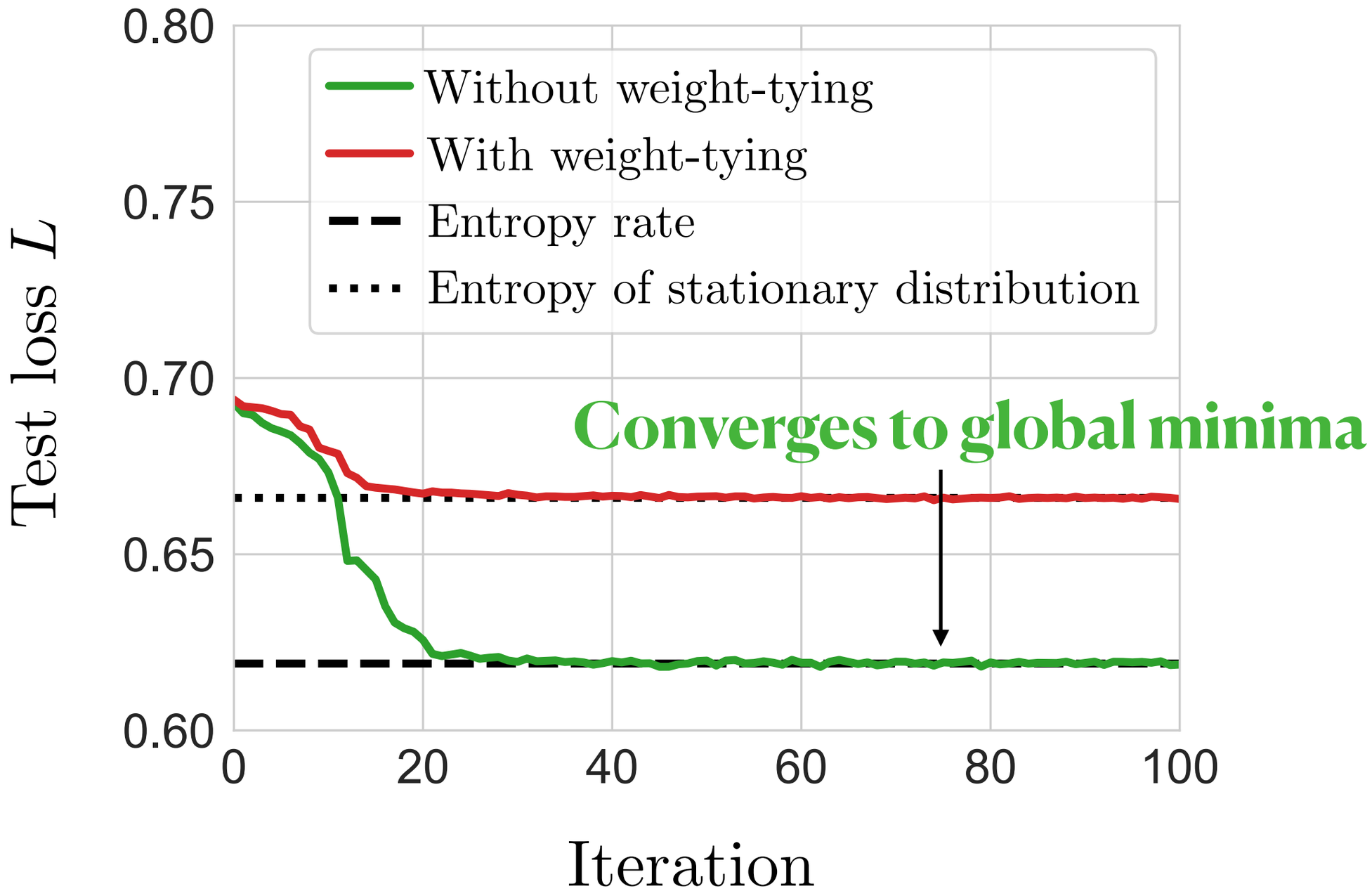
Single-layer transformers sometimes fail to learn even first-order Markov chains! *

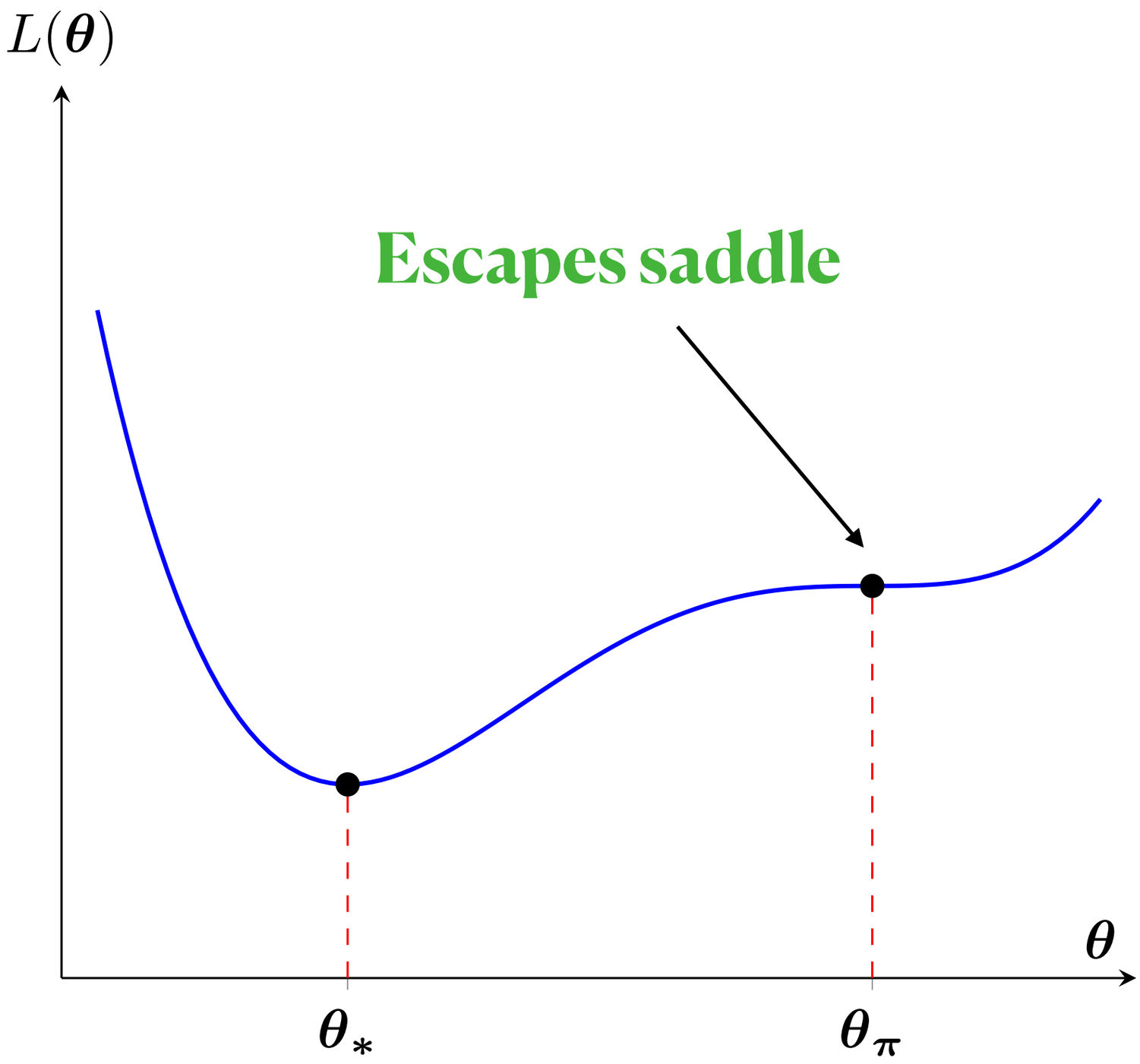
Weight tying




$L(\boldsymbol{\theta})$
Gets stuck at local minima!
θ
θ_*
θ_π

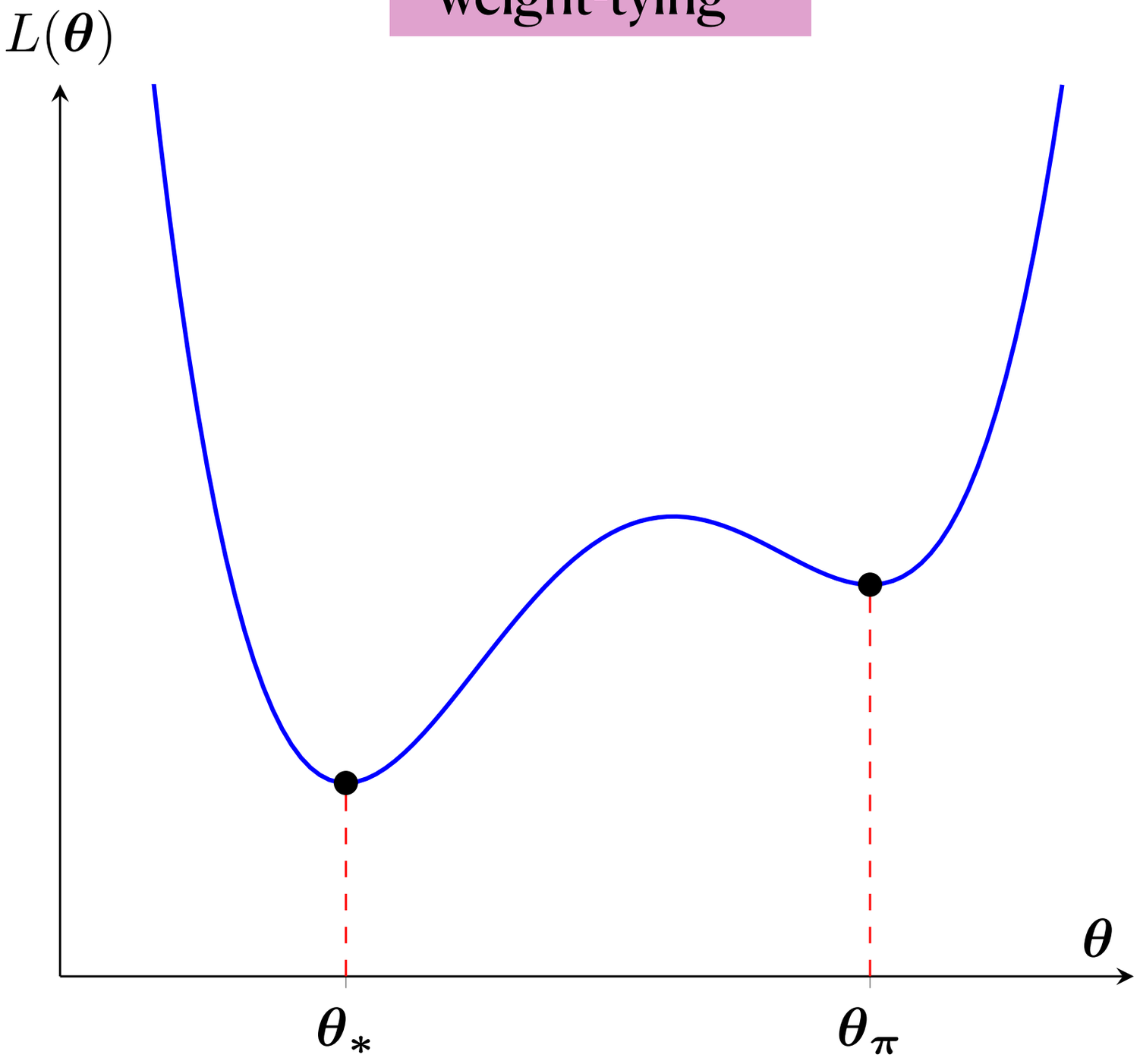
Without weight tying




$L(\boldsymbol{\theta})$
Escapes saddle
θ
θ_*
θ_π

What's happening

weight-tying



weight-tying

$L(\boldsymbol{\theta})$

$f_{\boldsymbol{\theta}_*}(x_1^n) = \mathbb{P}(x_{n+1} = 1 \mid x_n)$

all p, q

global minima

$\boldsymbol{\theta}$

$\boldsymbol{\theta}_*$

$\boldsymbol{\theta}_\pi$

weight-tying

$L(\boldsymbol{\theta})$

$f_{\boldsymbol{\theta}_*}(x_1^n) = \mathbb{P}(x_{n+1} = 1 \mid x_n)$

$f_{\boldsymbol{\theta}_{\boldsymbol{\pi}}}(x_1^n) = \mathbb{P}(x_{n+1} = 1)$

$p + q > 1$

bad local-minima

all p, q

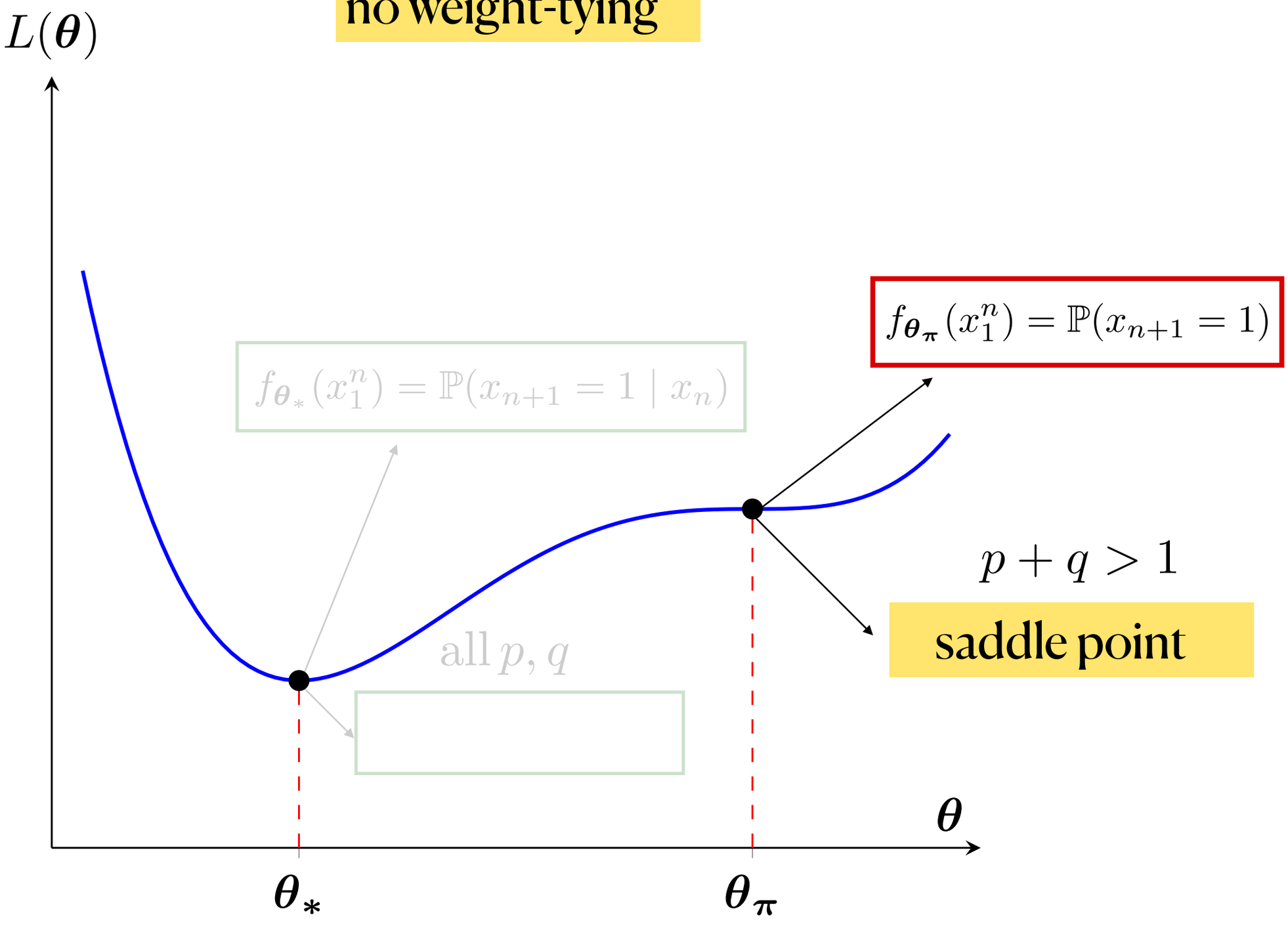
$\boldsymbol{\theta}$

$\boldsymbol{\theta}_*$

$\boldsymbol{\theta}_{\boldsymbol{\pi}}$

Interestingly...

no weight-tying



Theoretical results

Theoretical results

Bad local minima (weight tying)

If $p + q > 1$ and the weights are tied, there exists a $\boldsymbol{\theta_\pi}$ with explicit construction such that:

(i) $\boldsymbol{\theta_\pi}$ is a bad local minima for $L(\cdot)$ with $L(\boldsymbol{\theta_\pi}) > L(\boldsymbol{\theta}_*)$

(ii) $f_{\boldsymbol{\theta_\pi}}(x_1^n) = \mathbb{P}(x_{n+1} = 1)$, the marginal distribution

(iii) $L(\boldsymbol{\theta_\pi}) = H(\boldsymbol{\pi})$, the entropy of the stationary distribution

(iv) $\nabla L(\boldsymbol{\theta_\pi}) = 0$, i.e. $\boldsymbol{\theta_\pi}$ is a stationary point

Theoretical results

If $p + q > 1$ and the weights are tied, there exists a $\boldsymbol{\theta_\pi}$ with explicit construction such that:

(i) $\boldsymbol{\theta_\pi}$ is a bad local minima for $L(\cdot)$ with $L(\boldsymbol{\theta_\pi}) > L(\boldsymbol{\theta}_*)$

(ii) $f_{\boldsymbol{\theta_\pi}}(x_1^n) = \mathbb{P}(x_{n+1} = 1)$, the marginal distribution

(iii) $L(\boldsymbol{\theta_\pi}) = H(\boldsymbol{\pi})$, the entropy of the stationary distribution

(iv) $\nabla L(\boldsymbol{\theta_\pi}) = 0$, i.e. $\boldsymbol{\theta_\pi}$ is a stationary point

Saddle point (no weight tying)

Under the same setting as above with the weights not tied, $\boldsymbol{\theta_\pi}$ becomes a saddle point. It satisfies the same properties.

Theoretical results

Bad local minima (weight tying)

If $p + q > 1$ and the weights are tied, there exists a $\boldsymbol{\theta_\pi}$ with explicit construction such that:

(i) $\boldsymbol{\theta_\pi}$ is a bad local minima for $L(\cdot)$ with $L(\boldsymbol{\theta_\pi}) > L(\boldsymbol{\theta}_*)$

(ii) $f_{\boldsymbol{\theta_\pi}}(x_1^n) = \mathbb{P}(x_{n+1} = 1)$, the marginal distribution

(iii) $L(\boldsymbol{\theta_\pi}) = H(\boldsymbol{\pi})$, the entropy of the stationary distribution

(iv) $\nabla L(\boldsymbol{\theta_\pi}) = 0$, i.e. $\boldsymbol{\theta_\pi}$ is a stationary point

Saddle point (no weight tying)

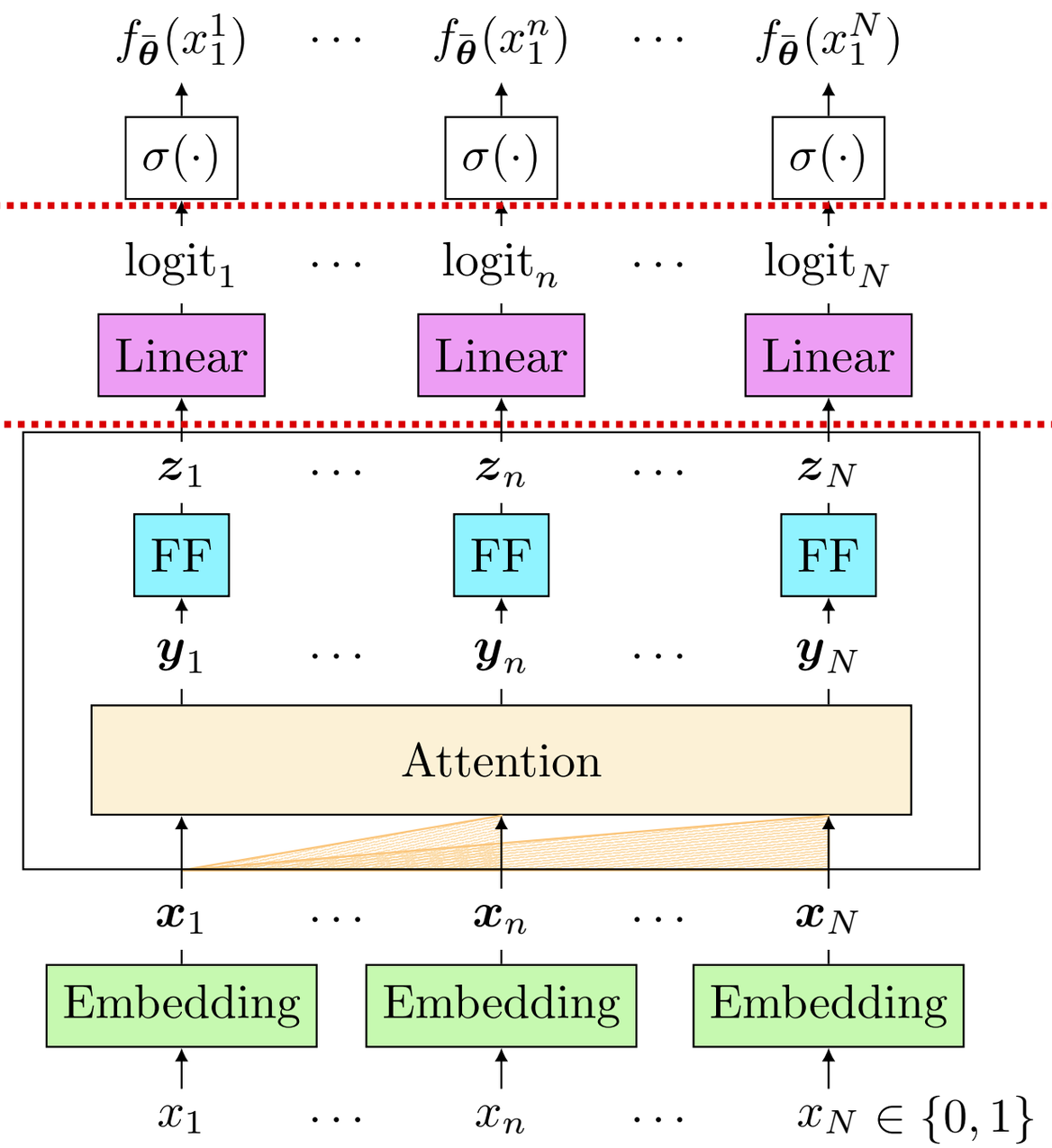
Under the same setting as above with the weights not tied,
$\boldsymbol{\theta_\pi}$ becomes a saddle point. It satisfies the same properties.

Intuition

$$f_{\boldsymbol{\theta}}(x_1^n) = \sigma(b) = \pi_1$$

$\boldsymbol{a} = 0$

$$\text{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$



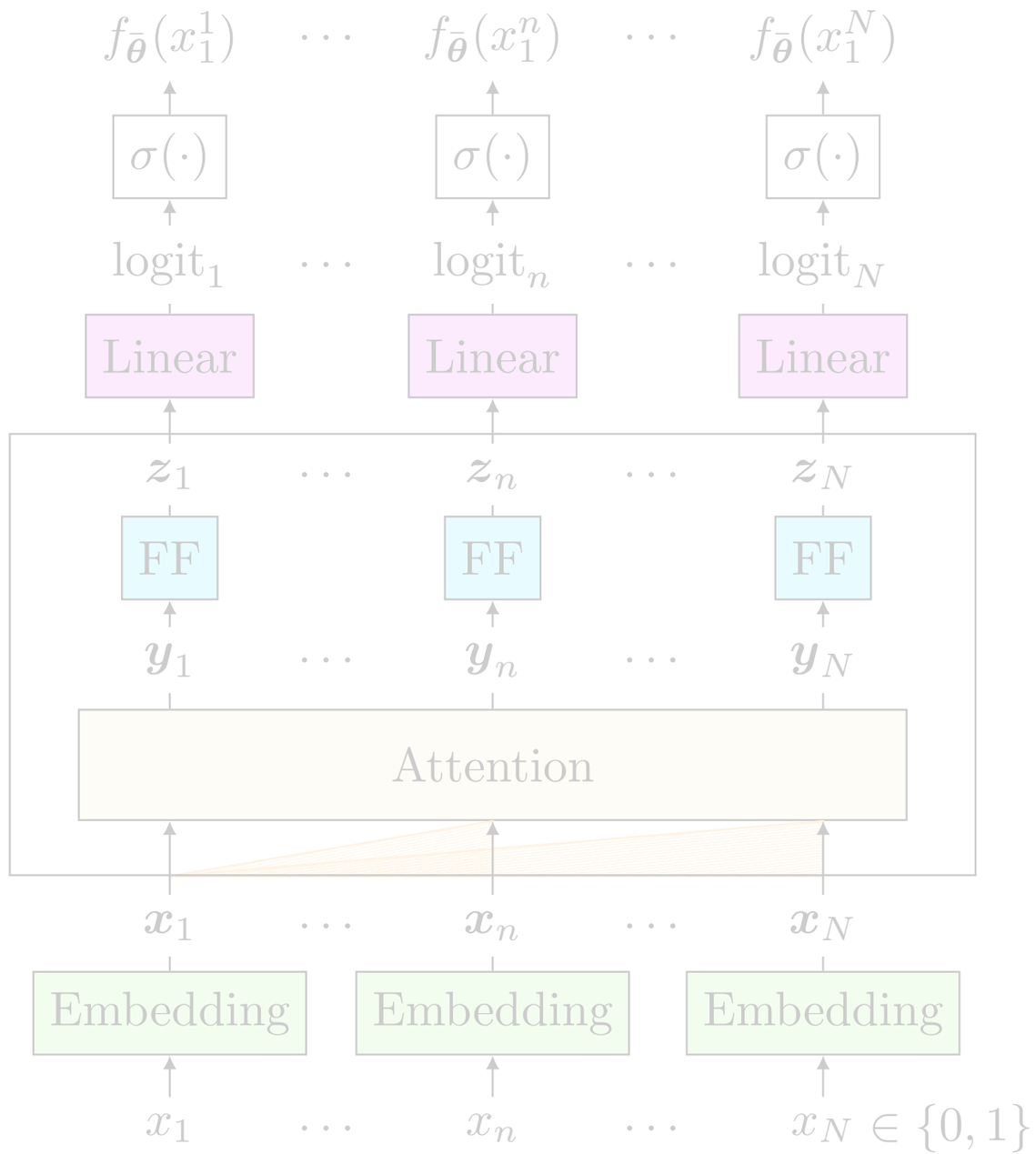
Intuition

weight-tying

Hessian is (almost) positive-definite

no weight-tying

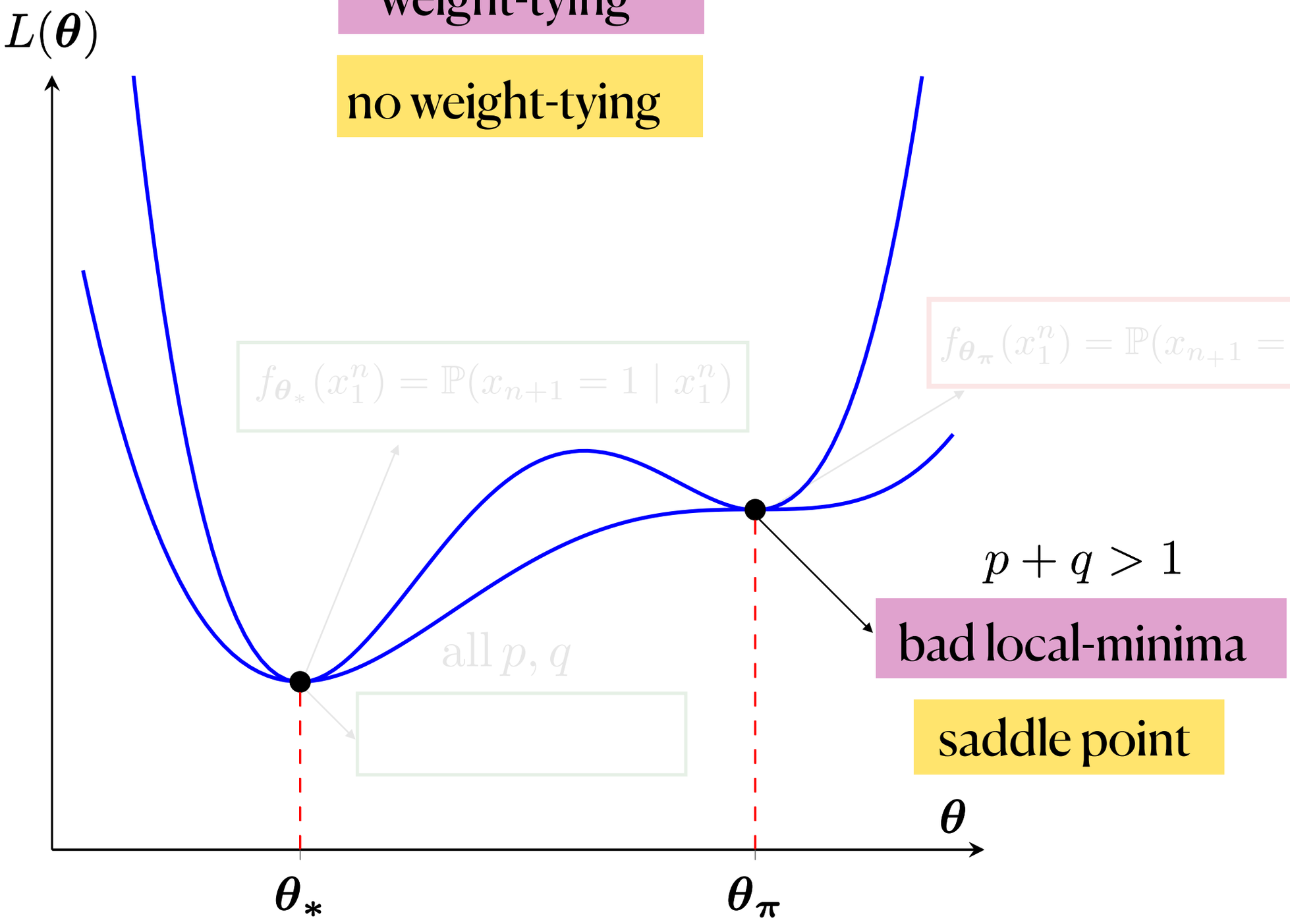
Hessian is indefinite



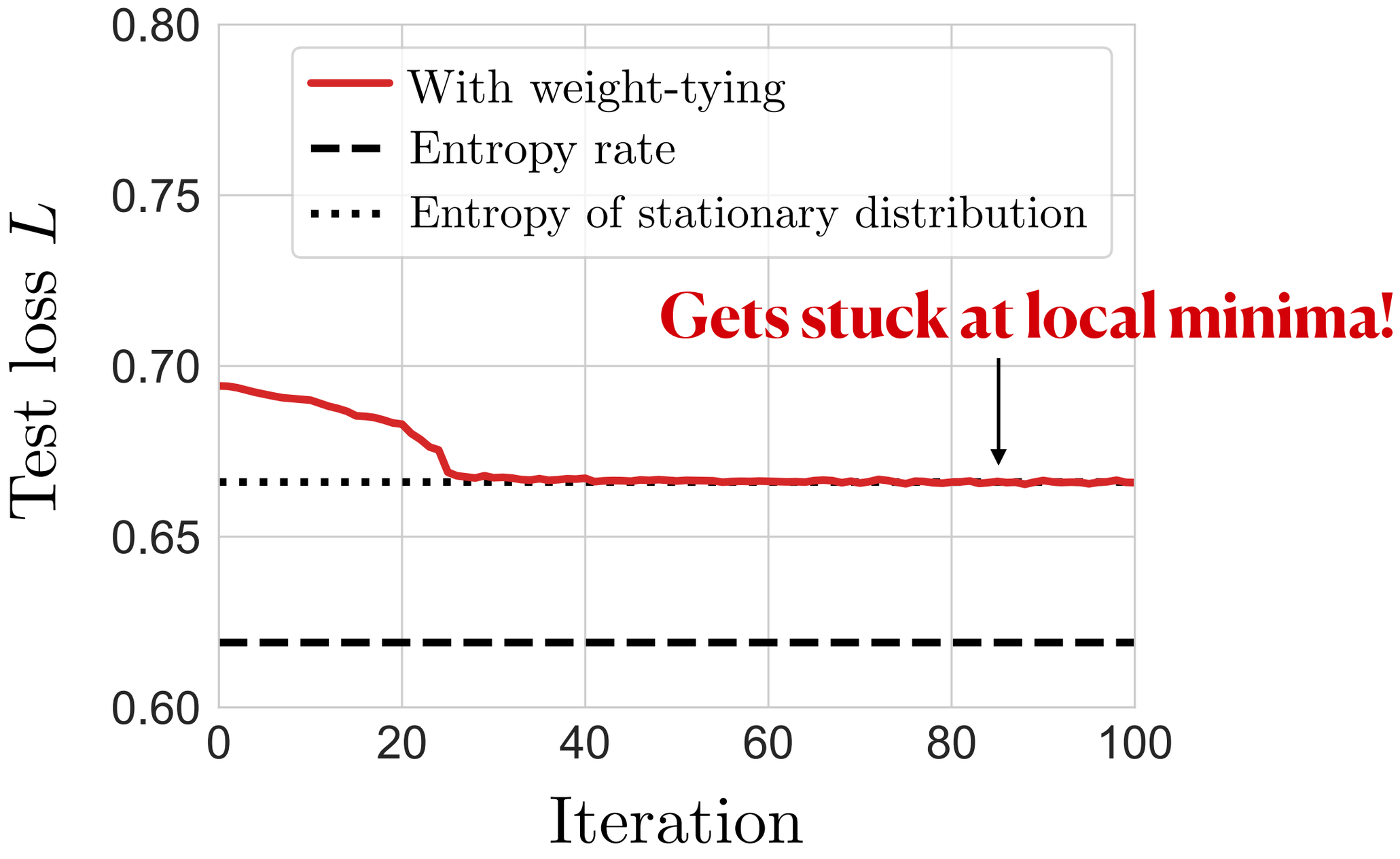
Main results

weight-tying

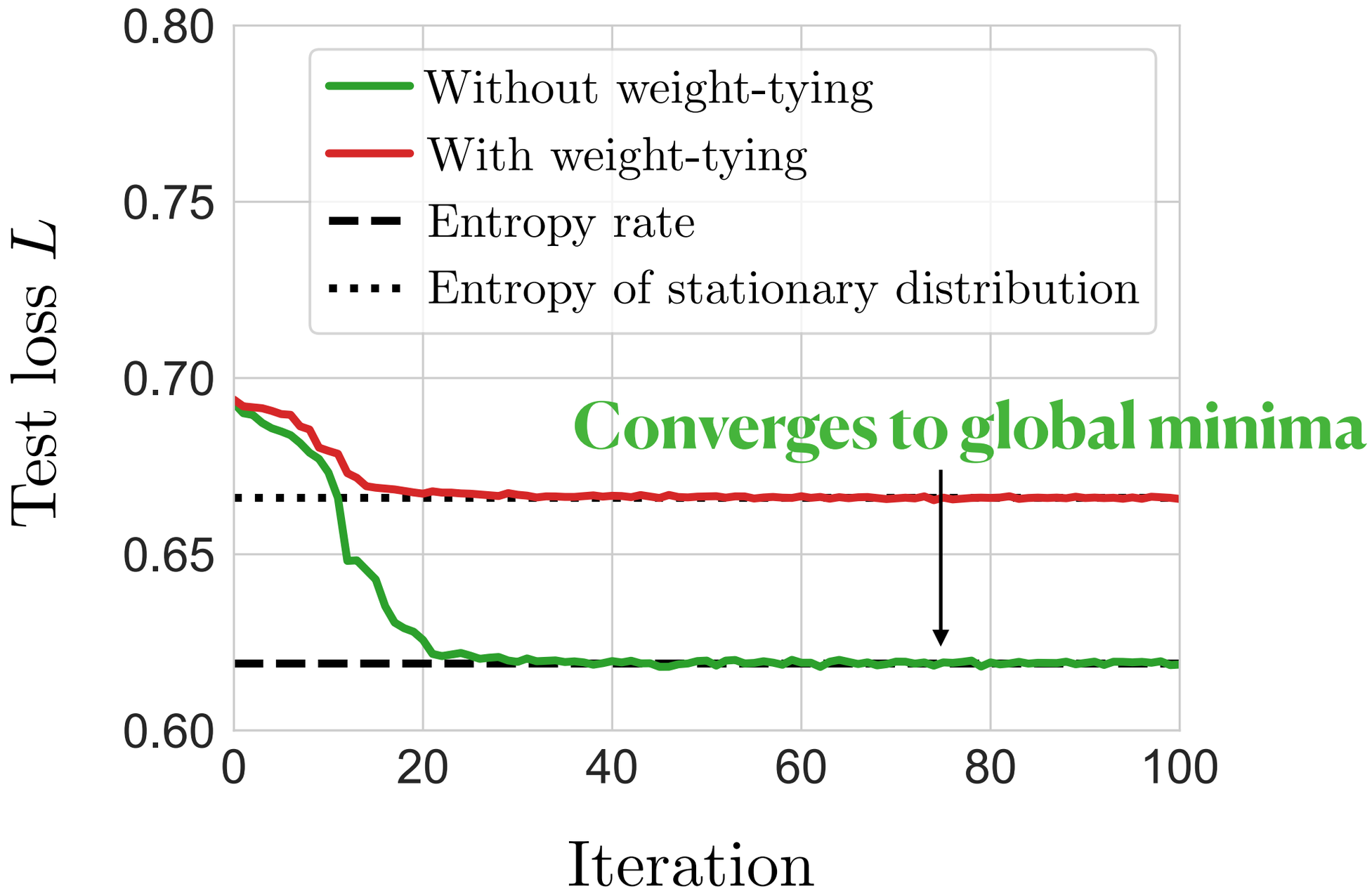
no weight-tying



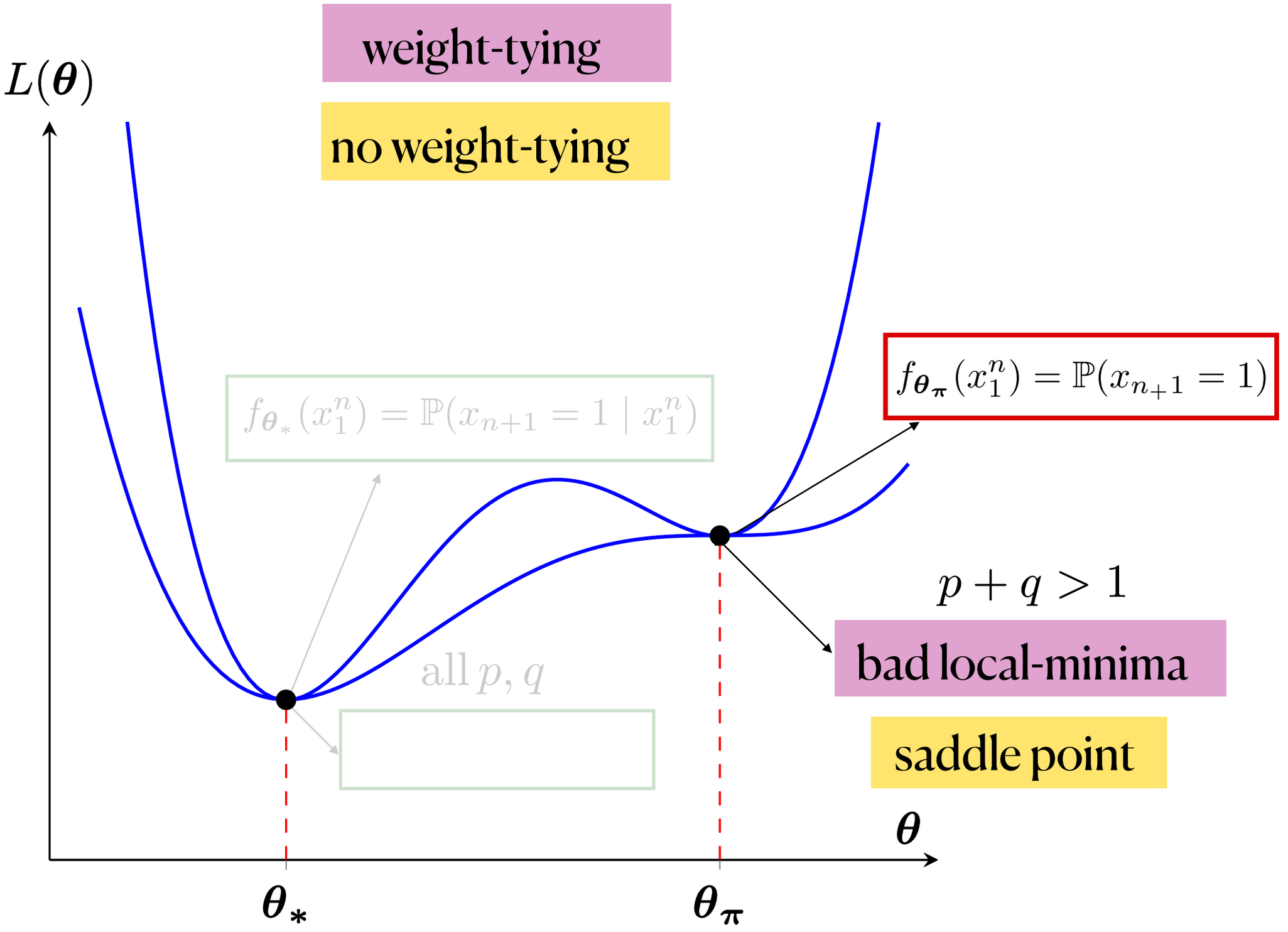
Weight tying



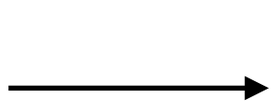
Without weight tying



Main results



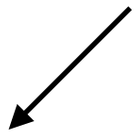
Markovian inputs



Transformers

Memory $= 1$

Depth $= 1$



What do they learn?

Markovian inputs

Transformers

Memory $= 1$

Depth $= 1$

How do they learn?

Memory = 1

Depth = 1

How do they learn?

Learning dynamics

Gradient-flow

Gradient flow

Gradient flow

$$\frac{\mathrm{d}\boldsymbol{\theta}_t}{\mathrm{d}t} = -\nabla L(\boldsymbol{\theta}_t)$$

Gradient flow

Transformer parameters

$$\frac{\mathrm{d}\boldsymbol{\theta}_t}{\mathrm{d}t} = -\nabla L(\boldsymbol{\theta}_t)$$

Next-token prediction loss

Recall

$$f_{\boldsymbol{\theta}}(x_1^n) = \mathbb{P}_{\boldsymbol{\theta}}(x_{n+1} = 1 \mid x_1^n) = \sigma(\text{logit}_n)$$

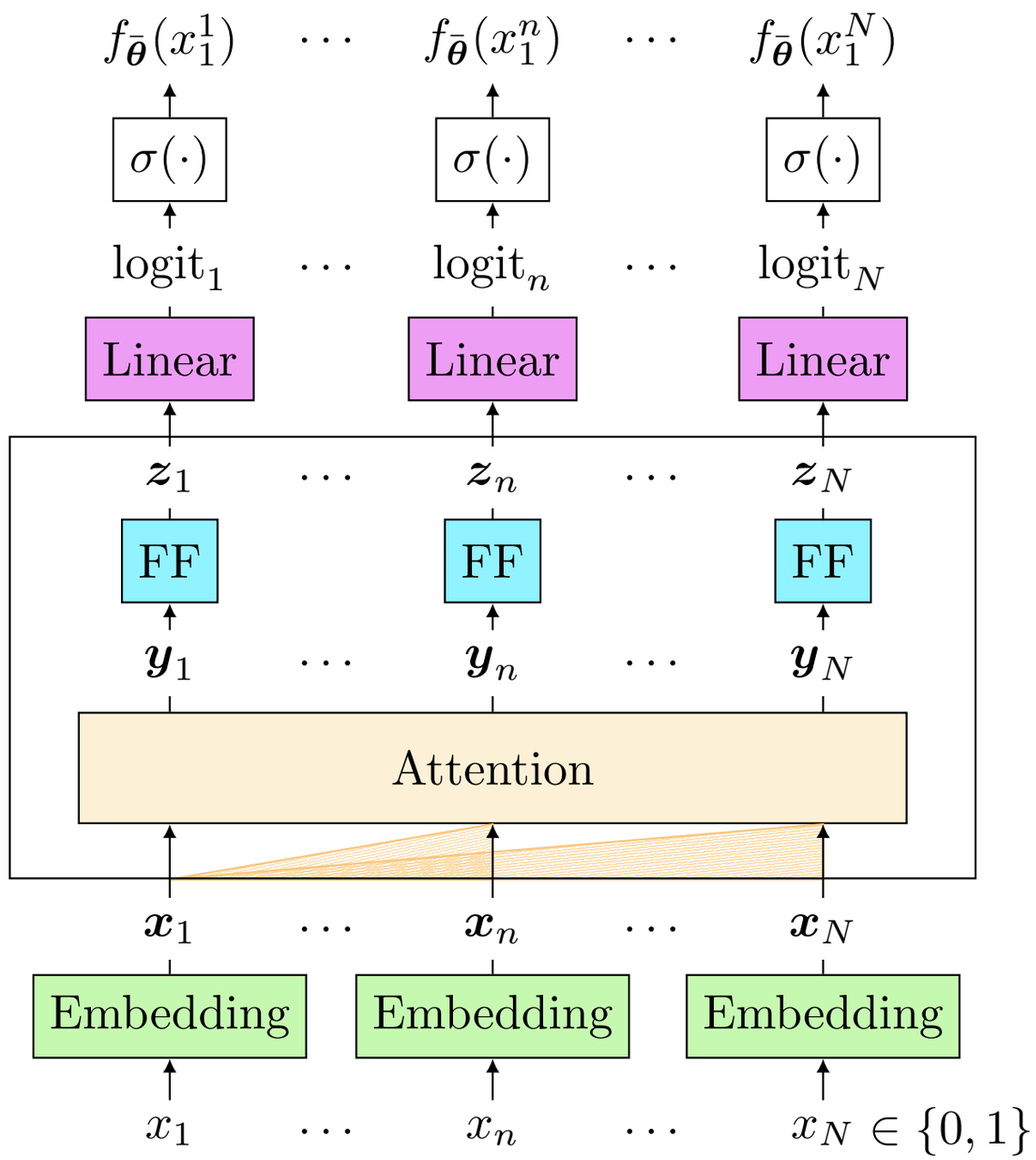
$$\text{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

$$\boldsymbol{z}_n = \boldsymbol{y}_n + \boldsymbol{W}_2 \,\text{ReLU}(\boldsymbol{W}_1 \boldsymbol{y}_n)$$

$$\boldsymbol{y}_n = \boldsymbol{x}_n + \boldsymbol{W}_O \sum_{i=1}^{n} \text{att}_{n,i} \cdot \boldsymbol{W}_V \boldsymbol{x}_i$$

$$\boldsymbol{x}_n = x_n \cdot \boldsymbol{e} + \boldsymbol{p}_n$$

$\boldsymbol{\theta}$



Low-rank structure

$$f_{\boldsymbol{\theta}}(x_1^n) = \mathbb{P}_{\boldsymbol{\theta}}(x_{n+1} = 1 \mid x_1^n) = \sigma(\text{logit}_n)$$

$$\text{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

$$\boldsymbol{z}_n = \boldsymbol{y}_n + \boldsymbol{W}_2 \,\text{ReLU}(\boldsymbol{W}_1 \boldsymbol{y}_n)$$

$$\boldsymbol{y}_n = \boldsymbol{x}_n + \boldsymbol{W}_O \sum_{i=1}^{n} \text{att}_{n,i} \cdot \boldsymbol{W}_V \boldsymbol{x}_i$$

$$\boldsymbol{x}_n = x_n \cdot \boldsymbol{e} + \boldsymbol{p}_n$$

Reparametrization

$$f_{\boldsymbol{\theta}}(x_1^n) = \mathbb{P}_{\boldsymbol{\theta}}(x_{n+1} = 1 \mid x_1^n) = \sigma(\text{logit}_n)$$

$$\text{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

w

$$\boldsymbol{z}_n = \boldsymbol{y}_n + \boldsymbol{W}_2 \,\text{ReLU}(\boldsymbol{W}_1 \boldsymbol{y}_n)$$

$$\boldsymbol{y}_n = \boldsymbol{x}_n + \boldsymbol{W}_O \sum_{i=1}^{n} \text{att}_{n,i} \cdot \boldsymbol{W}_V \boldsymbol{x}_i$$

a

$$\boldsymbol{x}_n = x_n \cdot \boldsymbol{e} + \boldsymbol{p}_n$$

e

Reparametrization

$$f_{\boldsymbol{\theta}}(x_1^n) = \mathbb{P}_{\boldsymbol{\theta}}(x_{n+1} = 1 \mid x_1^n) = \sigma(\text{logit}_n)$$

$$\text{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

w

$$\boldsymbol{z}_n = \boldsymbol{y}_n + \boldsymbol{W}_2 \,\text{ReLU}(\boldsymbol{W}_1 \boldsymbol{y}_n)$$

$$\boldsymbol{y}_n = \boldsymbol{x}_n + \boldsymbol{W}_O \sum_{i=1}^{n} \text{att}_{n,i} \cdot \boldsymbol{W}_V \boldsymbol{x}_i$$

a

$$\boldsymbol{x}_n = x_n \cdot \boldsymbol{e} + \boldsymbol{p}_n$$

e

$$\boldsymbol{\theta} = (e, w, a) \in \mathbb{R}^3$$

Gradient flow

$$\frac{\mathrm{d}\boldsymbol{\theta}_t}{\mathrm{d}t} = -\nabla L(\boldsymbol{\theta}_t), \quad \boldsymbol{\theta}_t = (e_t, w_t, a_t) \in \mathbb{R}^3$$

$$\frac{\mathrm{d}\boldsymbol{\theta}_t}{\mathrm{d}t} = -\nabla L(\boldsymbol{\theta}_t), \quad \boldsymbol{\theta}_t = (e_t, w_t, a_t) \in \mathbb{R}^3$$

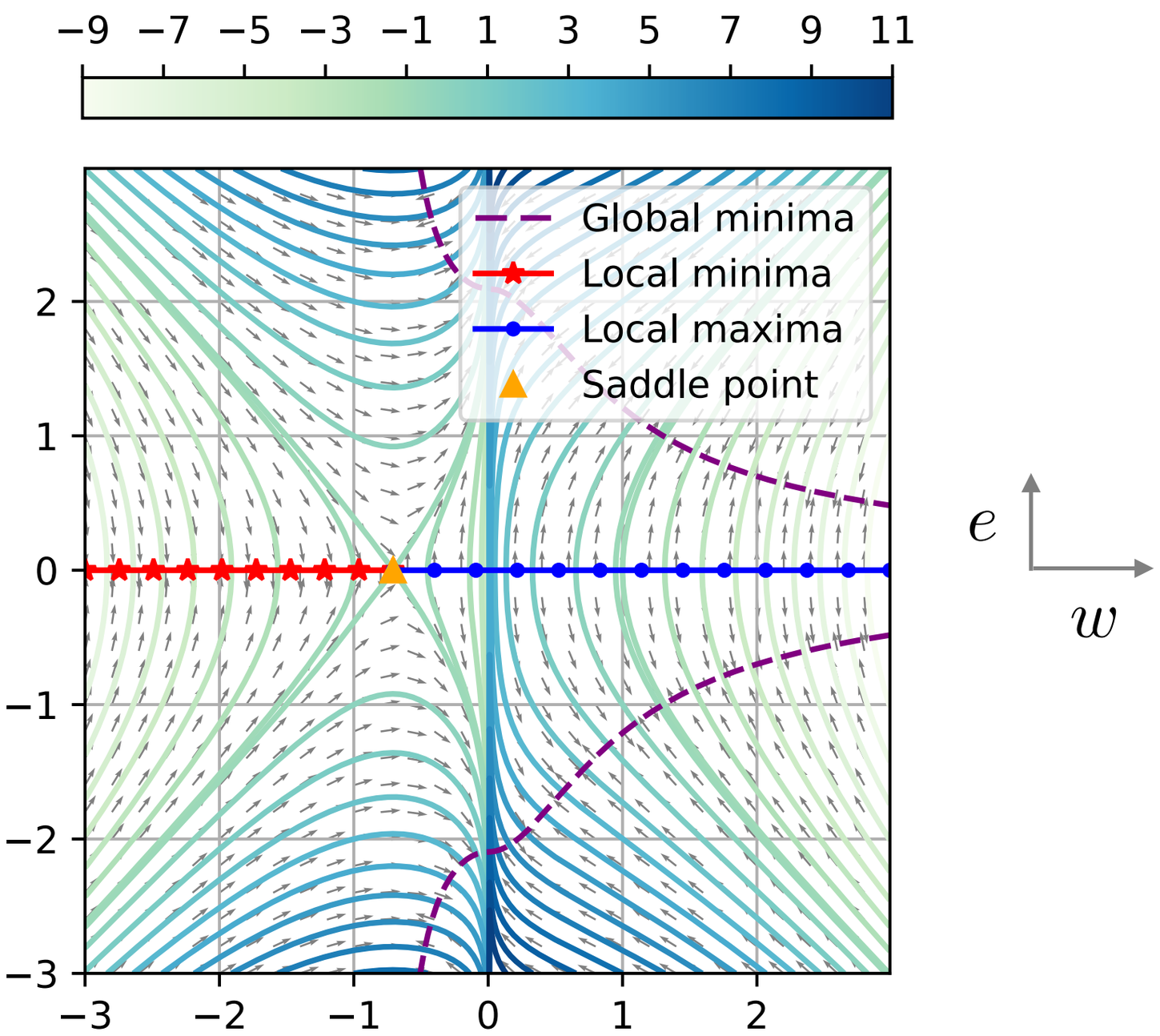
How does the flow look like?

$$\frac{\mathrm{d}\boldsymbol{\theta}_t}{\mathrm{d}t} = -\nabla L(\boldsymbol{\theta}_t), \quad \boldsymbol{\theta}_t = (e_t, w_t, a_t) \in \mathbb{R}^3$$

How does the flow look like?

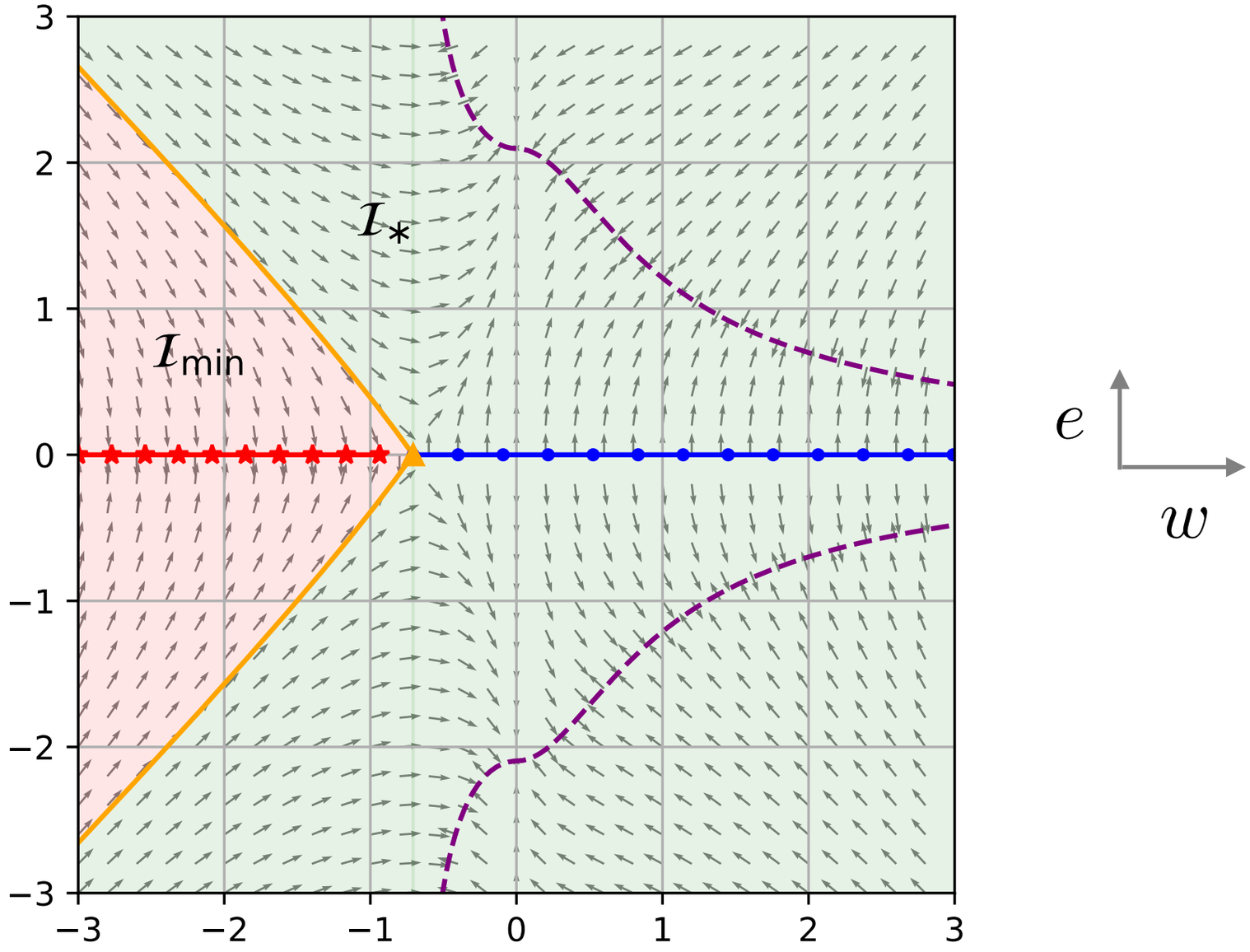
$$a = 0 \rightarrow \boldsymbol{\theta} = (e, w)$$

$p+q<1$


−9 −7 −5 −3 −1 1 3 5 7 9 11
Global minima
Local minima
Local maxima
Saddle point
2
1
0
−1
−2
−3
−3 −2 −1 0 1 2
e
w

p+q<1

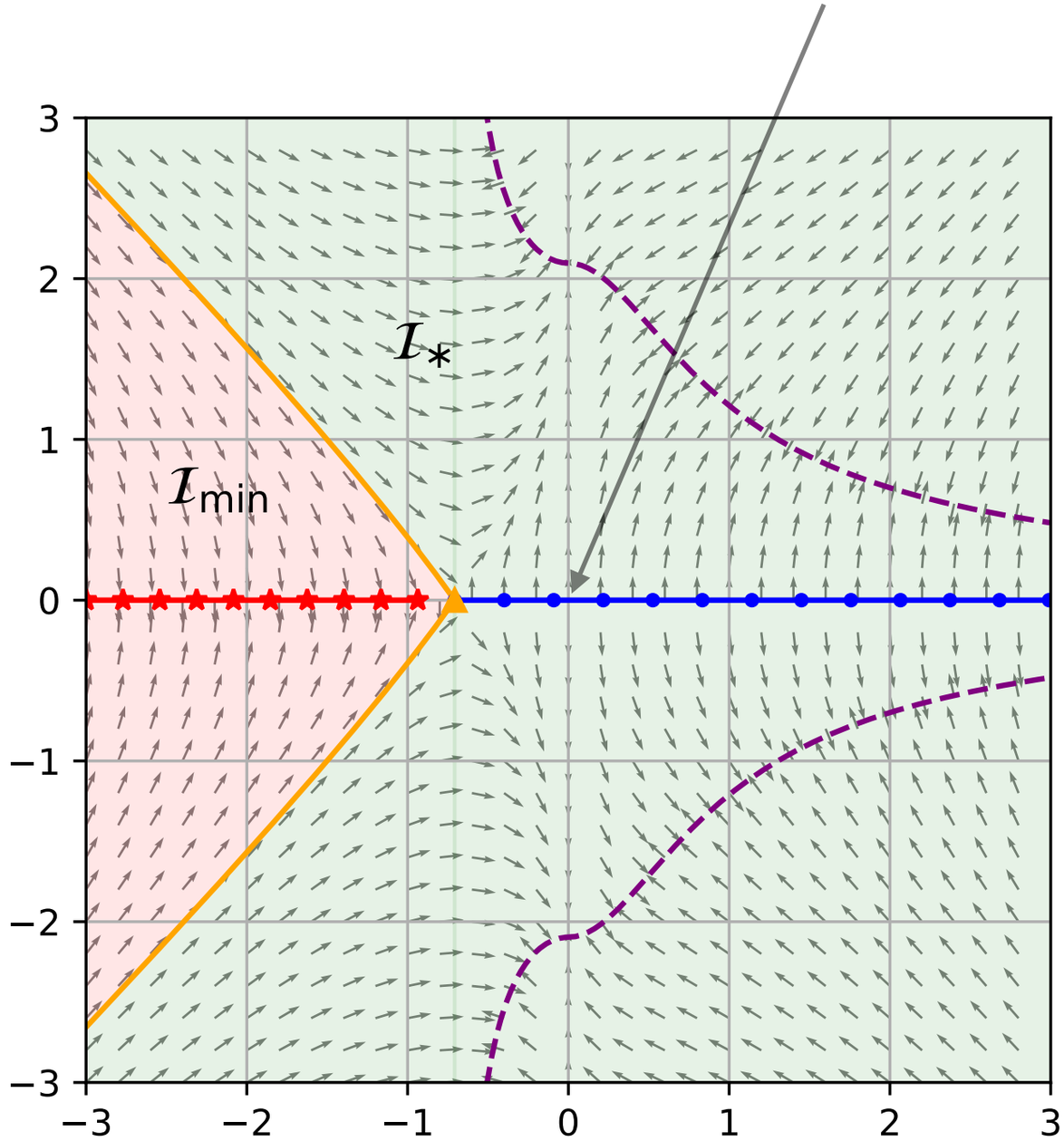
−9 −7 −5 −3 −1 1 3 5 7 9 11
Global minima
Local minima
Local maxima
Saddle point
2
1
0
−1
−2
−3
−3 −2 −1 0 1 2


$\mathcal{I}_*$
$\mathcal{I}_{\min}$
3
2
1
0
−1
−2
−3
−3 −2 −1 0 1 2 3

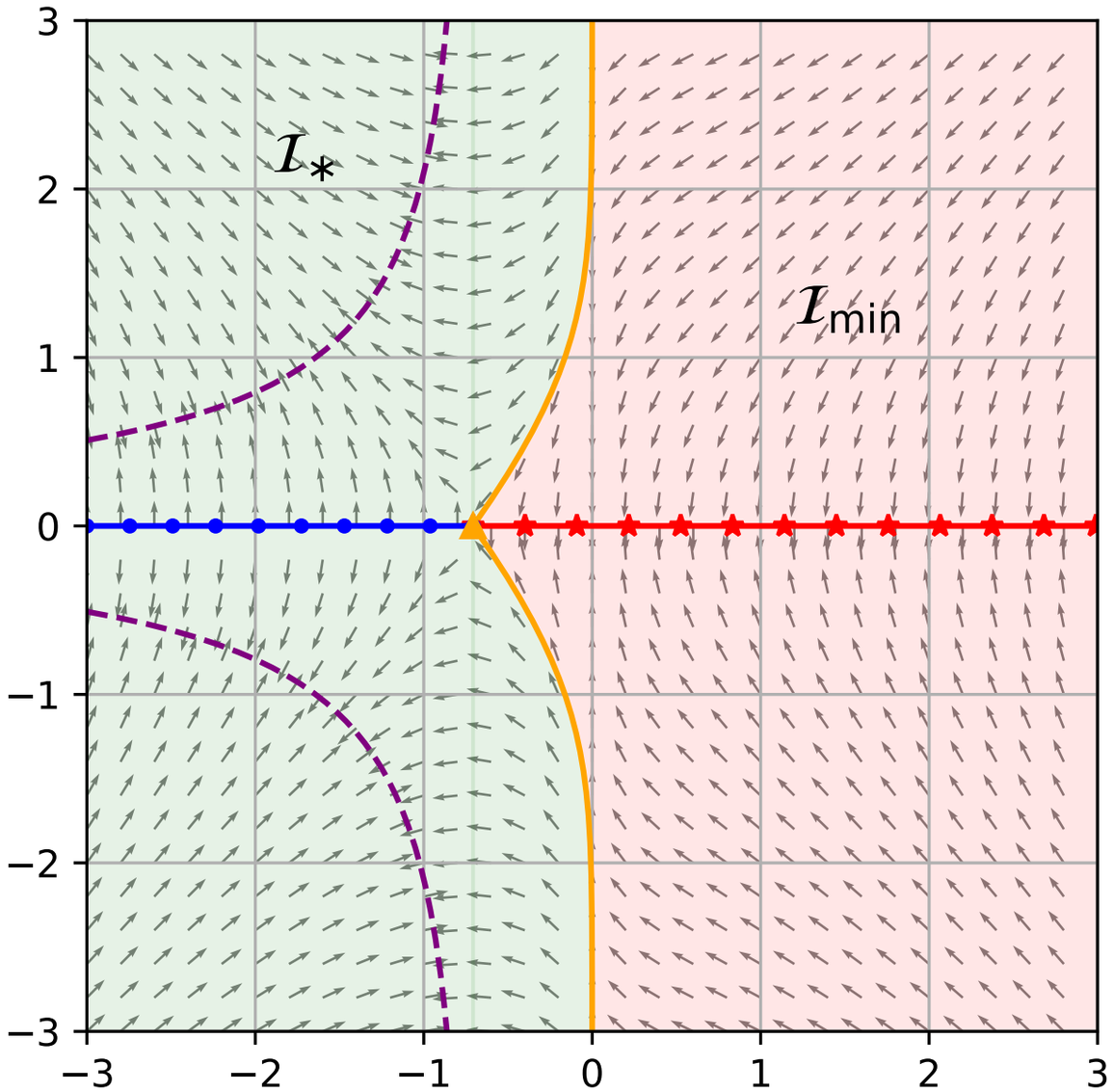
e
w

$p+q<1$

Gaussian init. converges to global minima

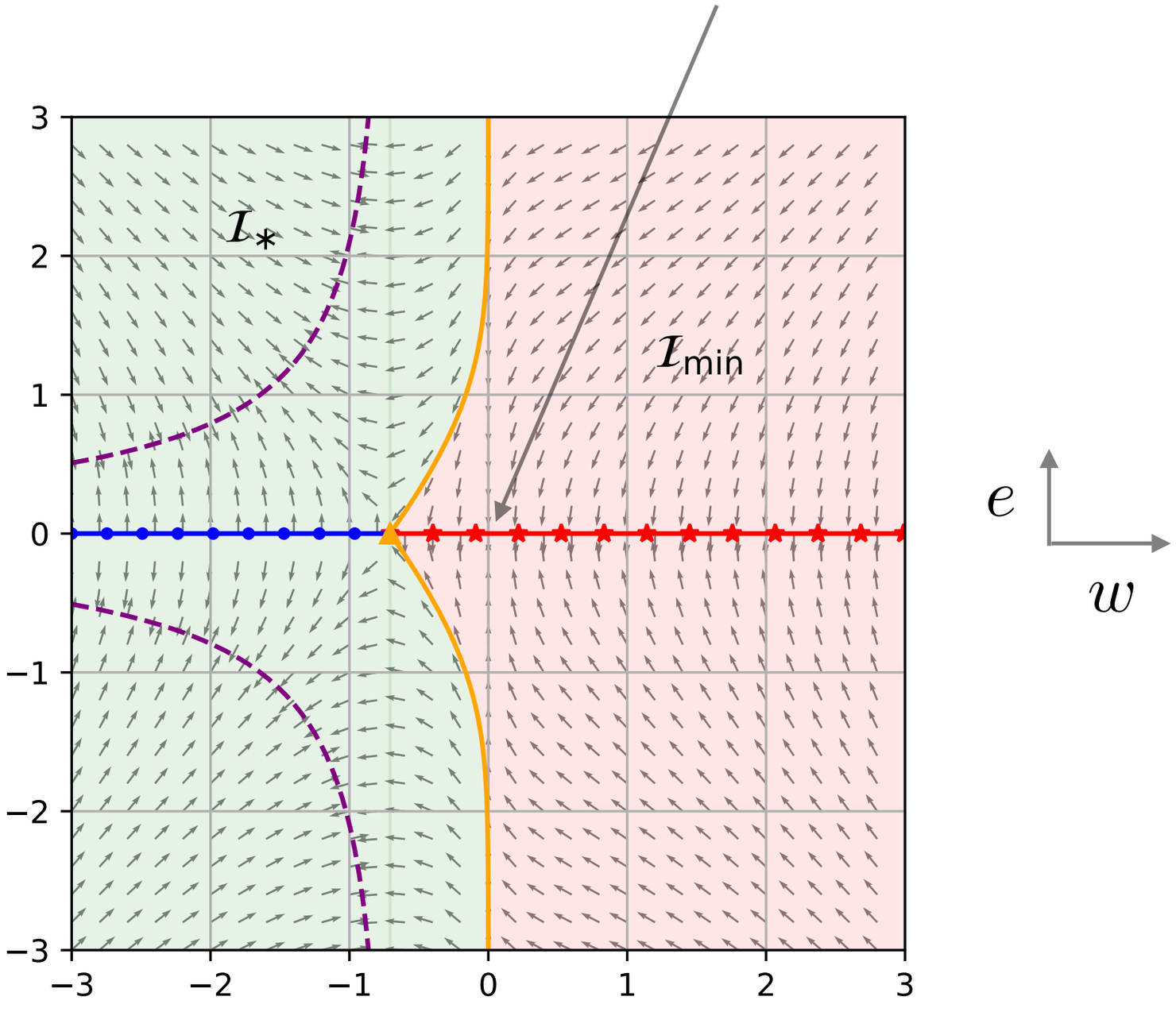


$p+q>1$

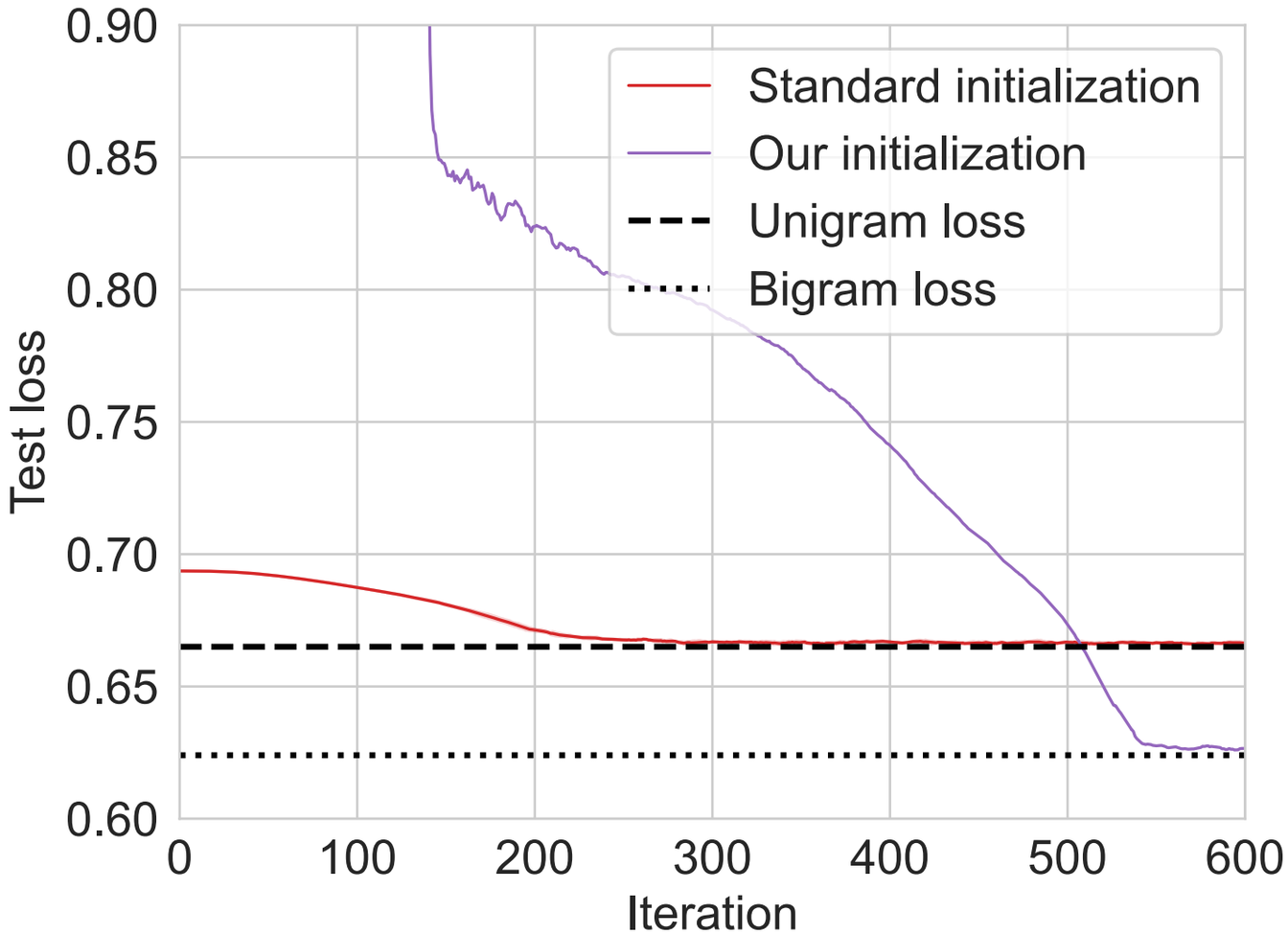


$\mathbf{p+q>1}$

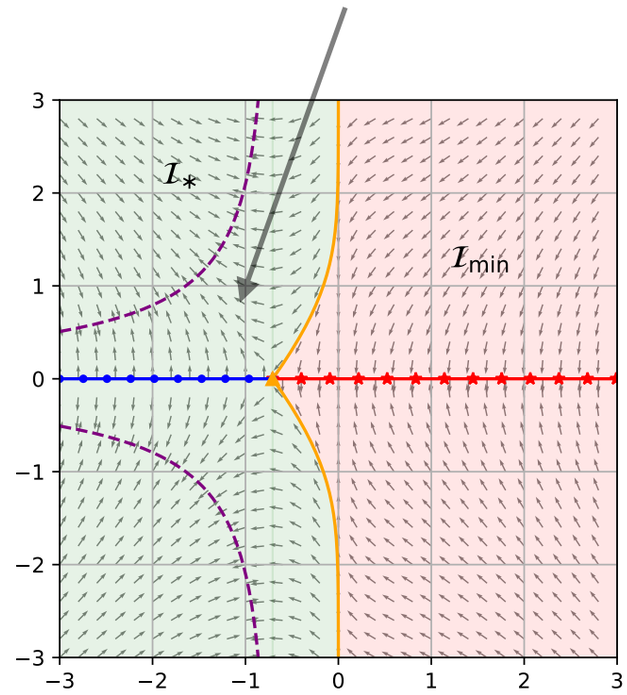
Gets stuck at local minima!



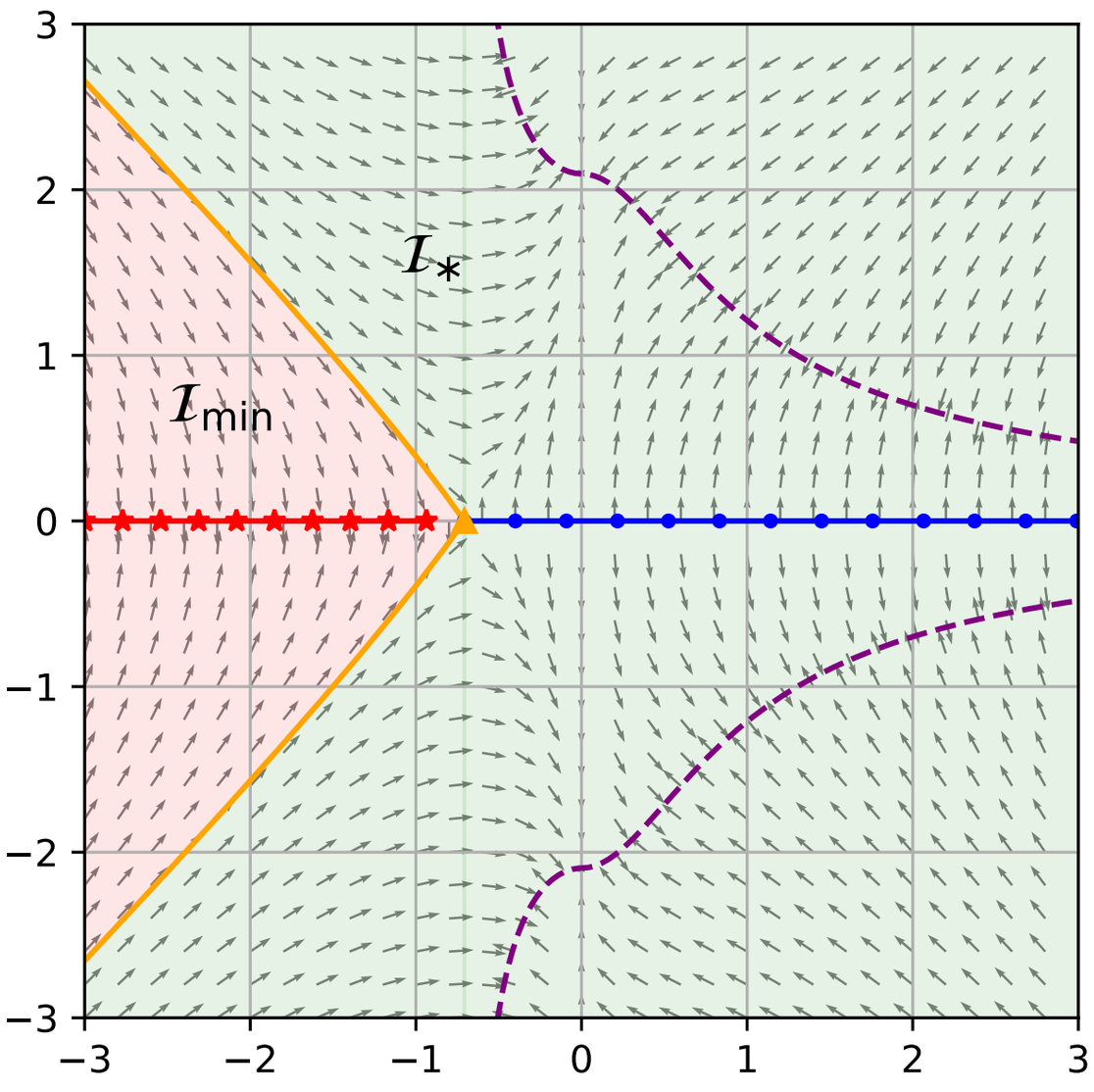
Can we escape it?



Initialise here

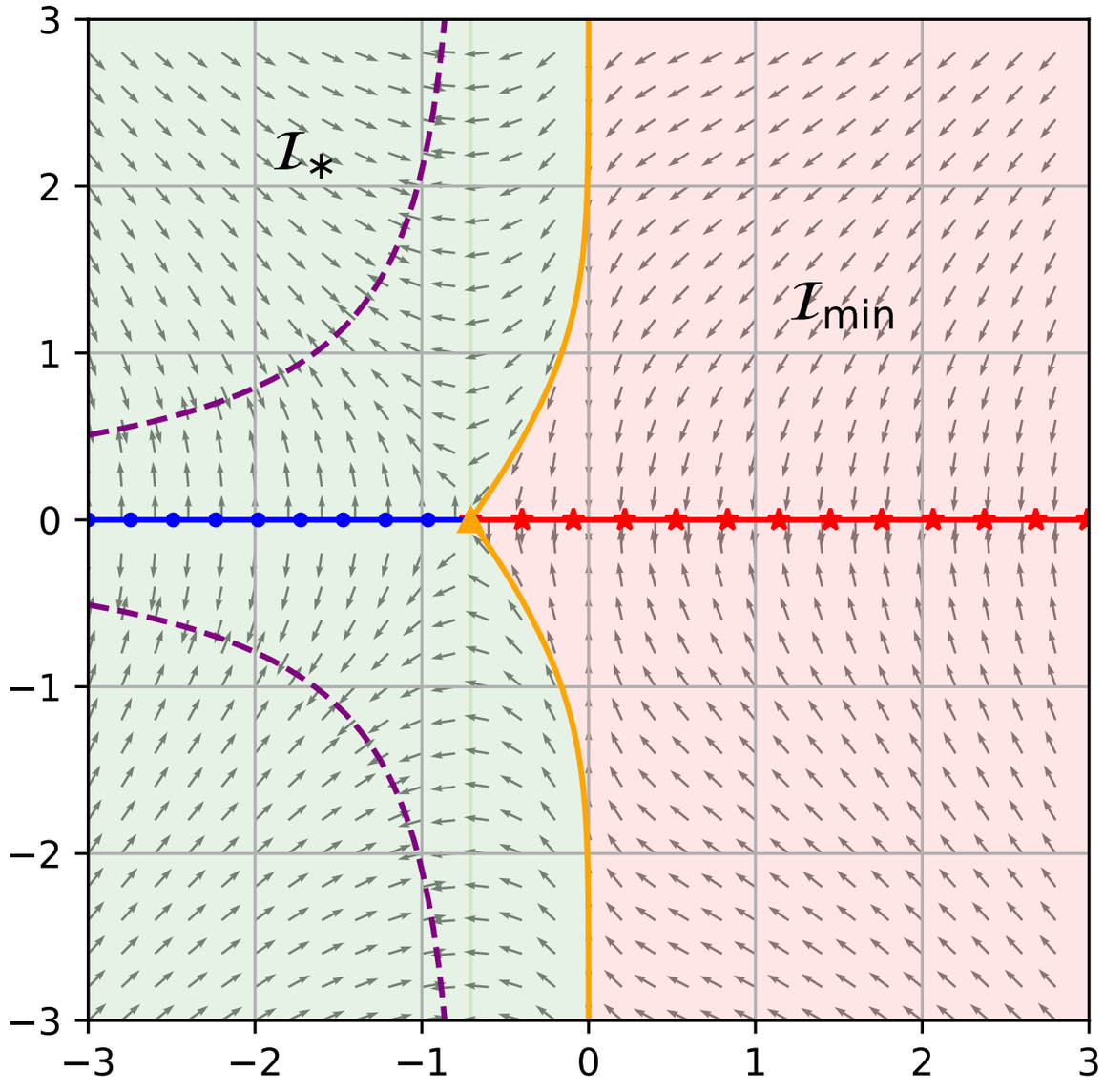


Converges to global minima

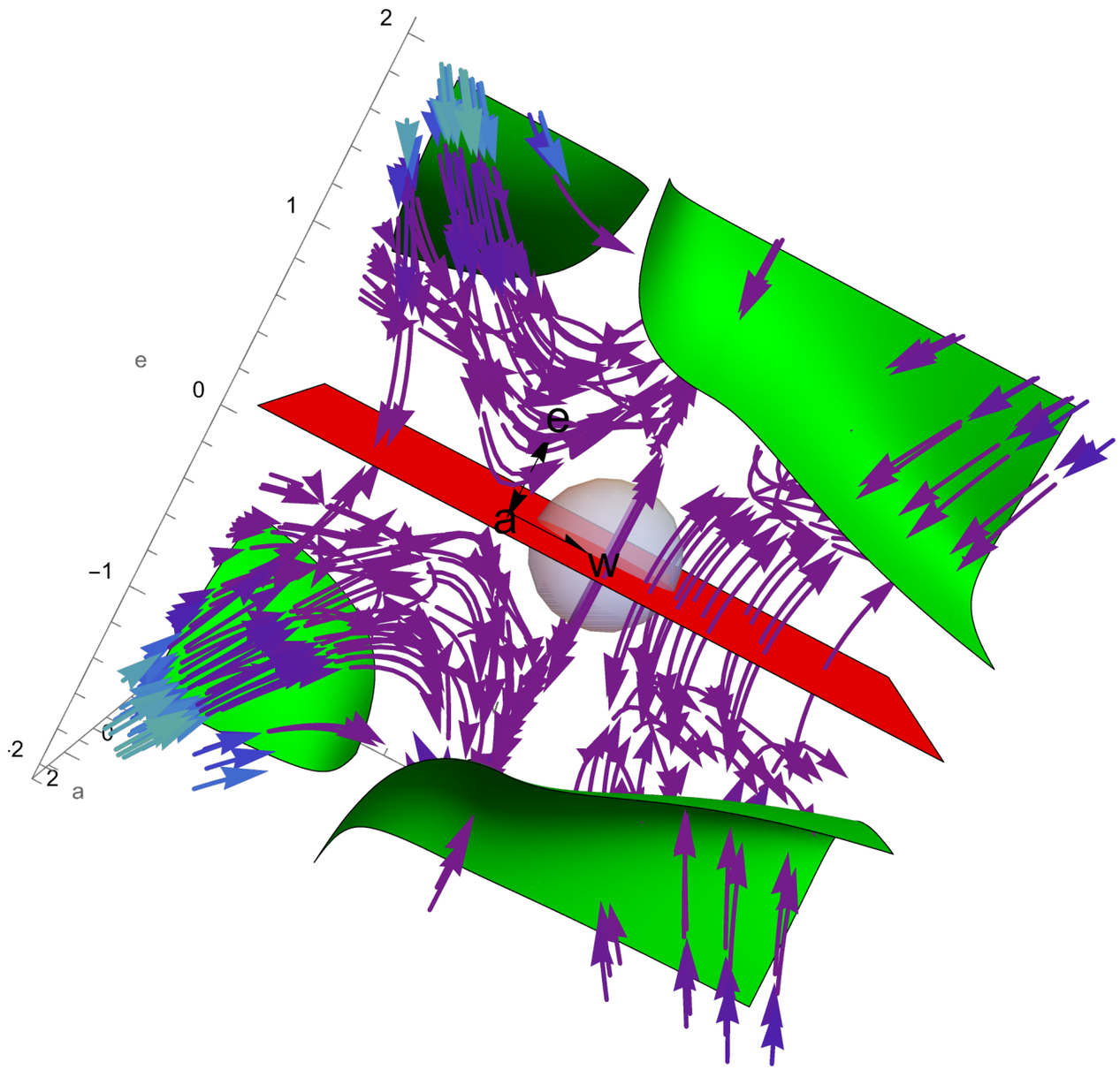


$p + q < 1$

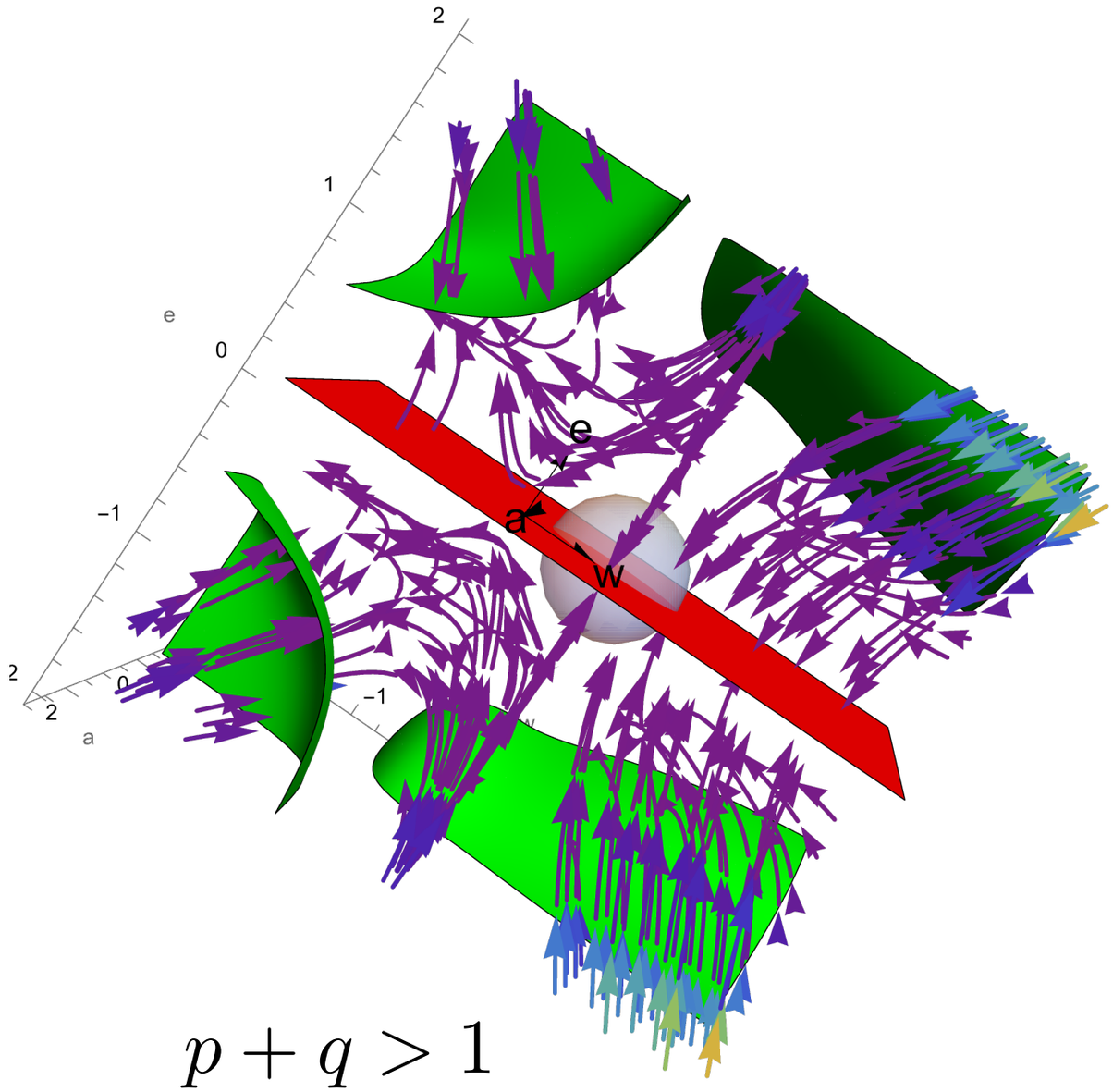
Gets stuck at local minima!



$p + q > 1$



$p + q < 1$

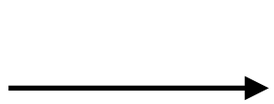


$p + q > 1$

Global minima Local minima Ball around origin

Part I.a

Markovian inputs



Transformers

Memory $= 1$

Depth $= 1$

What do they learn?

How do they learn?

Single-layer transformers sometimes fail to learn even first-order Markov chains!

Memory = 1

Depth = 1

Markovian switching and initialization play a key role in the learning dynamics

Part I.b

Markovian inputs

Transformers

Memory

Depth > 1

Memory

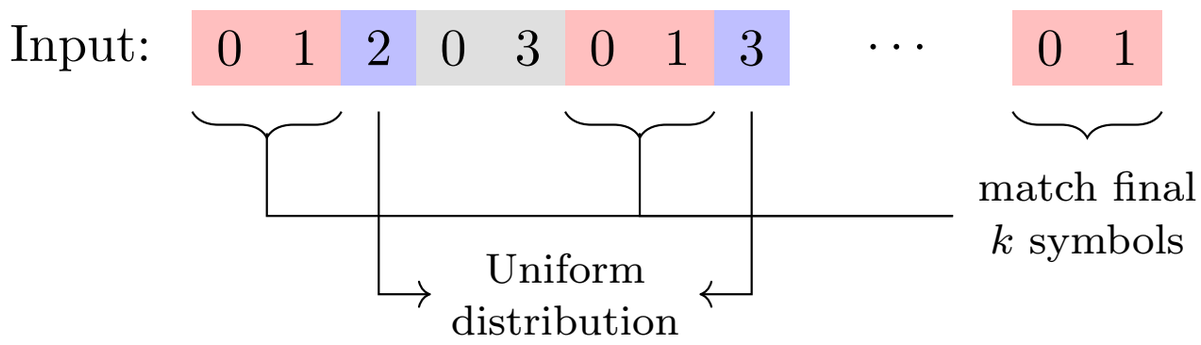
Depth > 1

In-context learning (ICL) emerges!

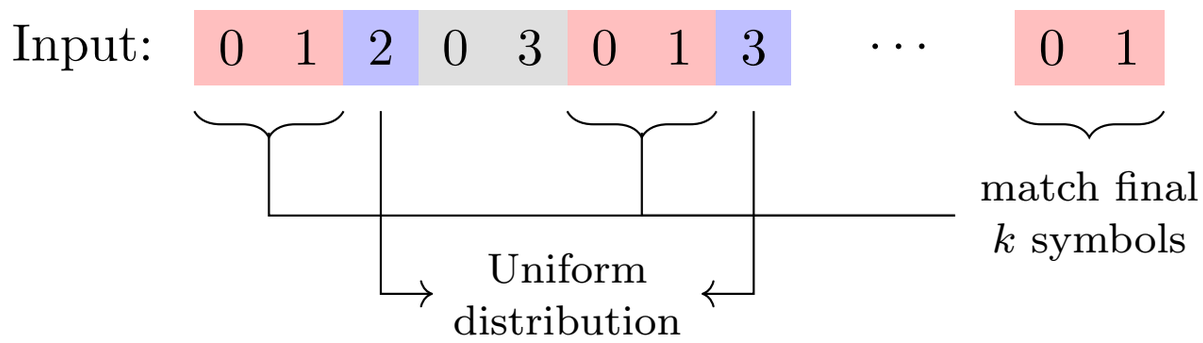
ICL

Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Mr Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large moustache. Mrs Durs___

ICL for Markov inputs



ICL for kth-order Markov



Main idea: Use historical tokens of same context as x_t to predict x_{t+1}

ICL for kth-order Markov

x_t x_{t+1}

In-context estimator:

$$\widehat{\Pr}_k(x \mid x_1, \cdots, x_t) \triangleq \frac{\sum_{i=k+1}^{t} \mathbb{I}(x_i = x, x_{i-1} = x_t, \ldots, x_{i-k} = x_{t-k+1})}{\sum_{i=k+1}^{t} \mathbb{I}(x_{i-1} = x_n, \ldots, x_{i-k} = x_{t-k+1})}$$

ICL for kth-order Markov

x_t x_{t+1}

In-context estimator:

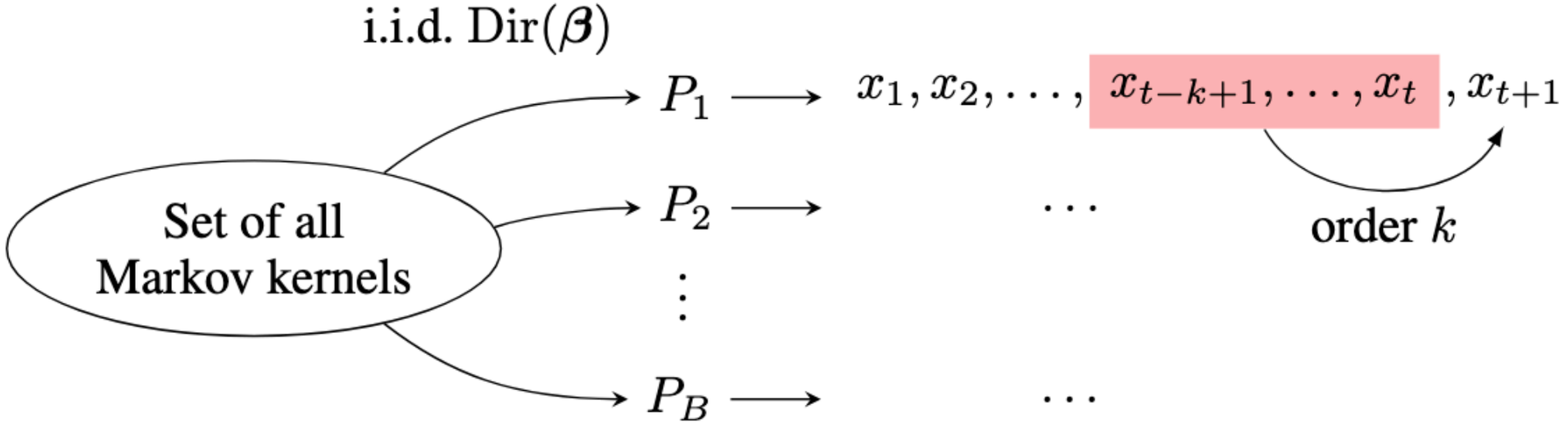
Context-matching

$$\widehat{\Pr}_k(x \mid x_1, \cdots, x_t) \triangleq \frac{\sum_{i=k+1}^{t} \mathbb{I}(x_i = x, x_{i-1} = x_t, \ldots, x_{i-k} = x_{t-k+1})}{\sum_{i=k+1}^{t} \mathbb{I}(x_{i-1} = x_n, \ldots, x_{i-k} = x_{t-k+1})}$$

In-context Markov chains

Memory $= k$

Depth



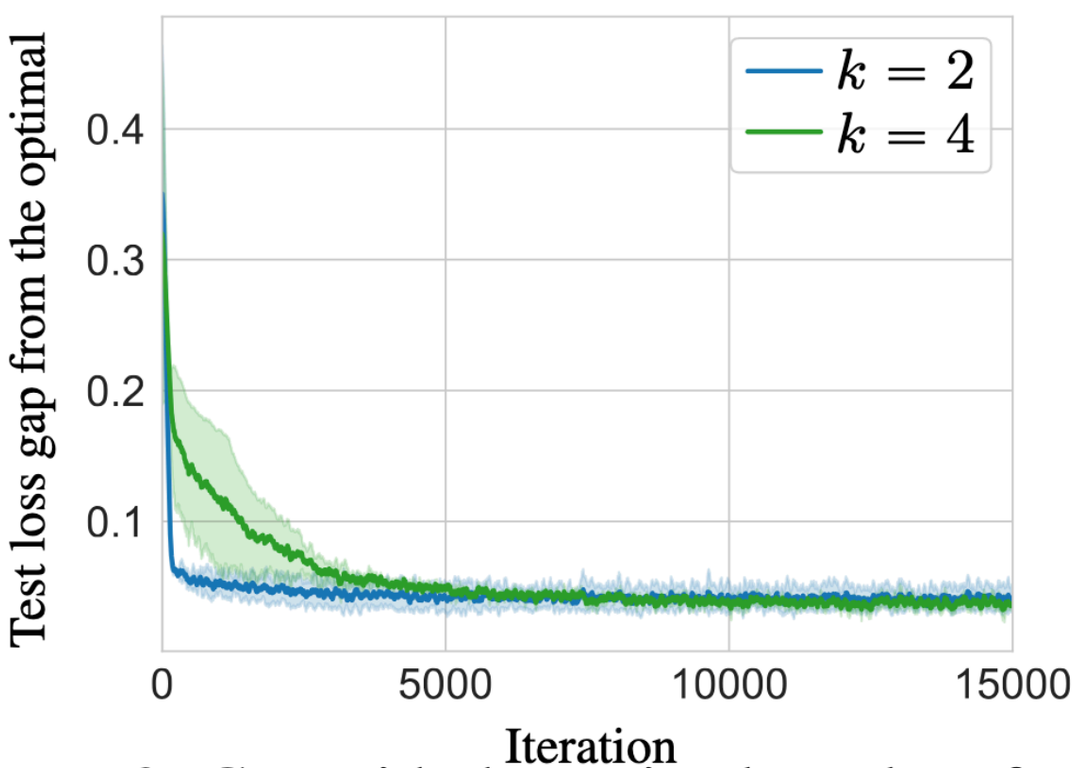
Memory $= k$

Depth?

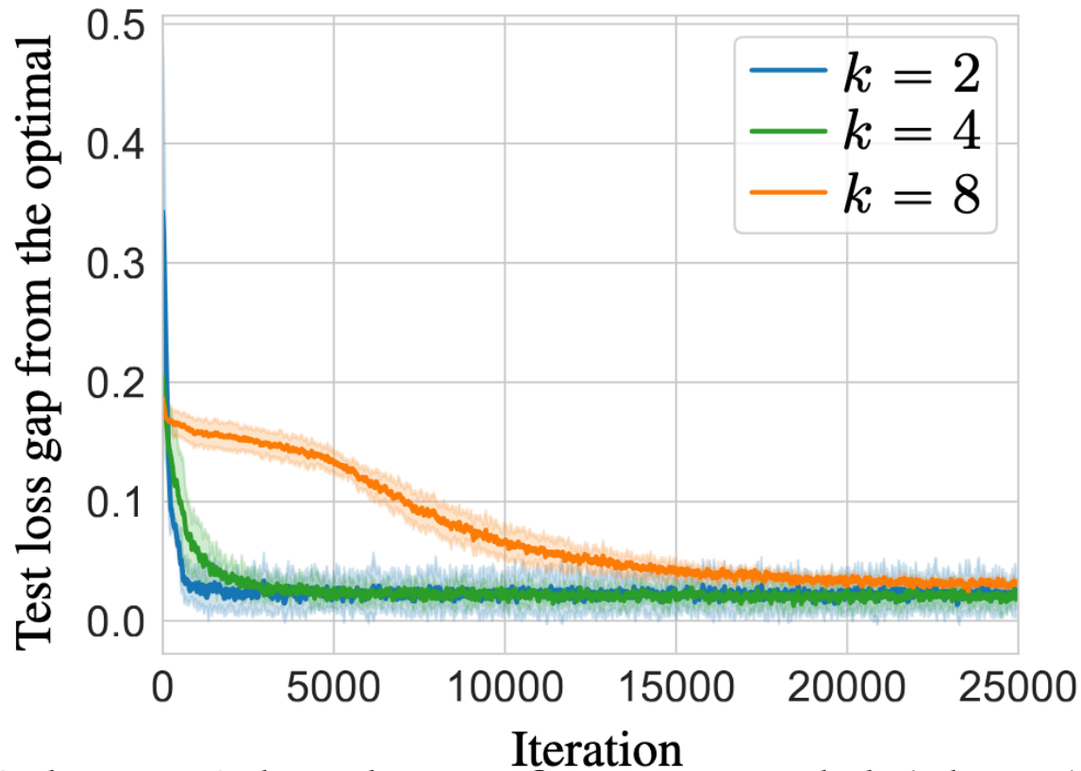
What we know so far

- (Edelman et al., 2024 & Nichani et al., 2024): Number of heads/layers should scale with k

But...



2-layer transformer



3-layer transformer

What's happening

What's happening

Main result (Constant depth suffices)

Any order k in-context estimator can be represented by a transformer with 3 layers, 1 head per layer, relative positional encodings and layer norm

What's happening

Main result (Constant depth suffices)

Any order k in-context estimator can be represented by a transformer with 3 layers, 1 head per layer, relative positional encodings and layer norm

Without layer norm, you need logarithmic depth

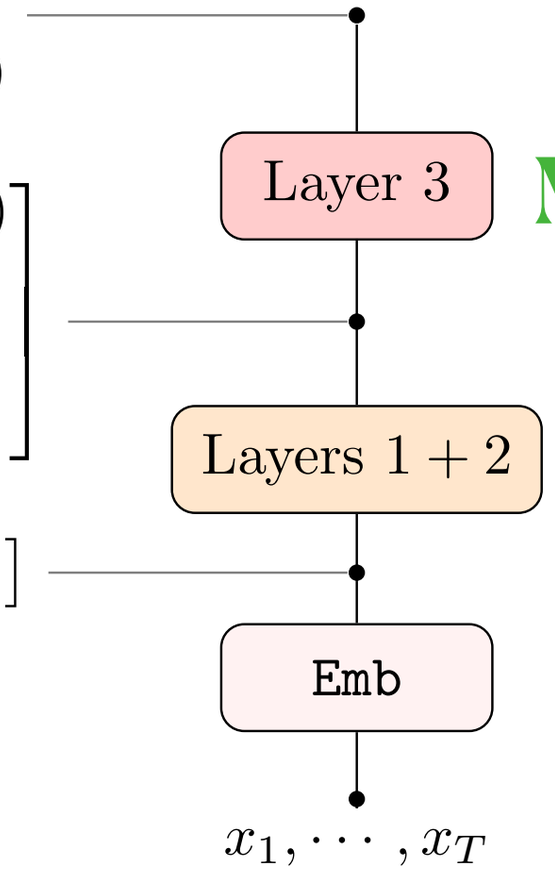
Intuition

$$\mathrm{att}_{T,n} \propto \exp\left(\kappa \frac{\langle \boldsymbol{v}_n, \boldsymbol{u}_T \rangle}{\|\boldsymbol{v}_n\|_2 \|\boldsymbol{u}_T\|_2}\right)$$

(realizes a k^{th}-order induction head)

$$\begin{bmatrix} \cdots & \mathtt{Emb}(x_n) & \cdots & \mathtt{Emb}(x_T) \\ & \frac{\boldsymbol{u}_n}{\|\boldsymbol{u}_n\|_2} & & \frac{\boldsymbol{u}_T}{\|\boldsymbol{u}_T\|_2} \\ & \frac{\boldsymbol{v}_n}{\|\boldsymbol{v}_n\|_2} & & \frac{\boldsymbol{v}_T}{\|\boldsymbol{v}_T\|_2} \end{bmatrix}$$

$$\begin{bmatrix} \cdots & \mathtt{Emb}(x_n) & \cdots & \mathtt{Emb}(x_T) \end{bmatrix}$$



Match contexts & count

Capture context

Intuition

2 layers suffice!

[Ekbote et al. 2025]

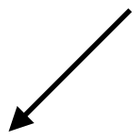
Part I.b

Markovian inputs

Transformers

Memory

Depth > 1



What do they learn?

Part I.b

Markovian inputs

Transformers

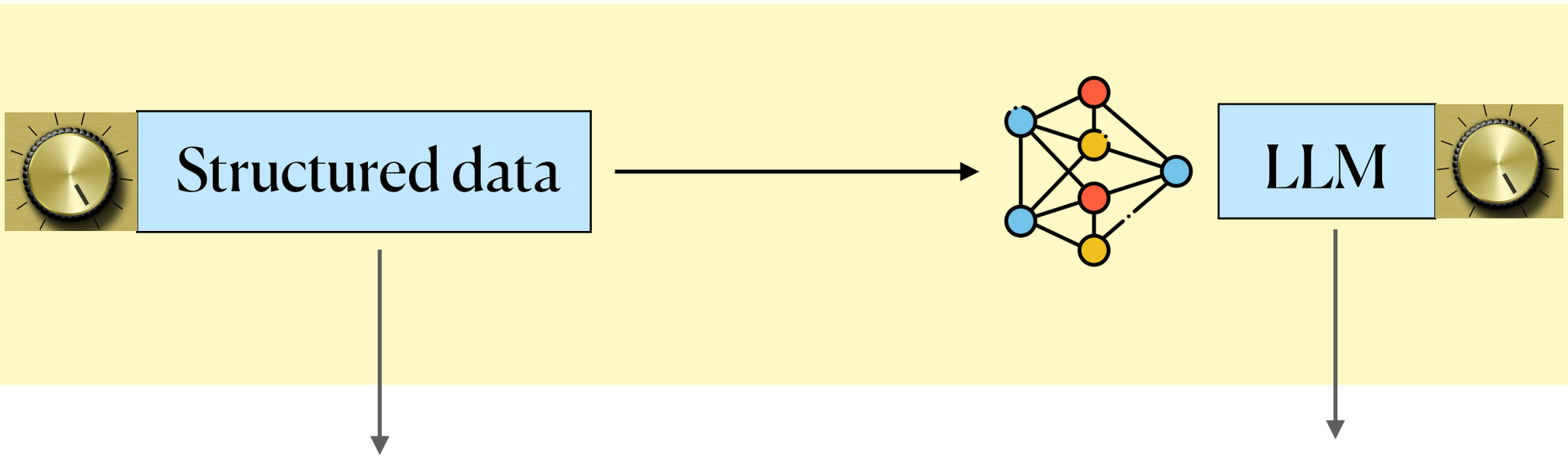
Memory

Depth > 1

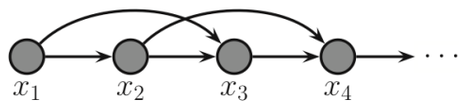
How do they learn?

Ongoing..

Part I

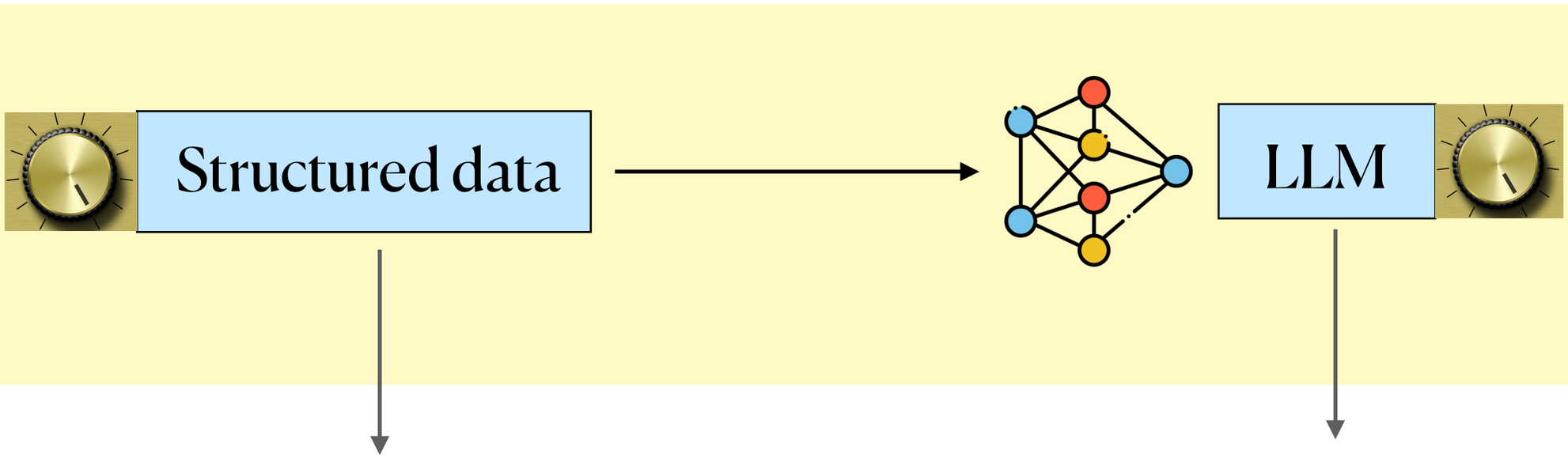


Markov processes

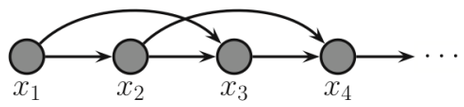


Transformers

Part II



Markov processes

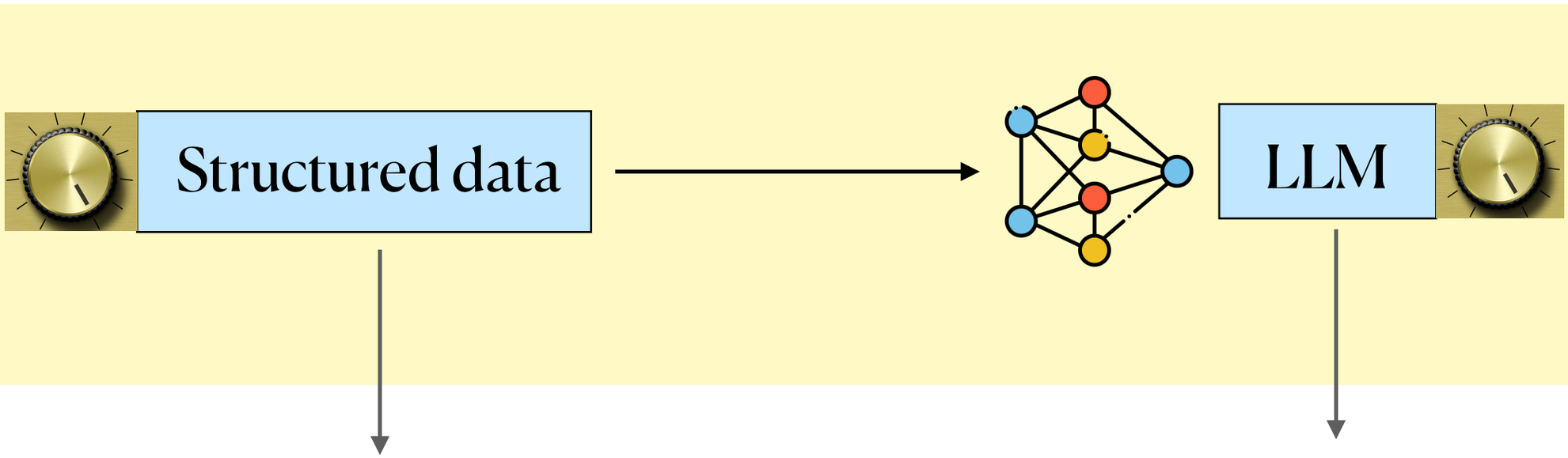


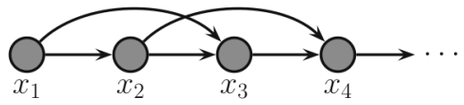
State-Space Models (Mamba)

Even a single-layer Mamba learns the optimal Laplacian smoothing estimator!

Convolution plays a critical role, more so than gating and non-linearities

Part II



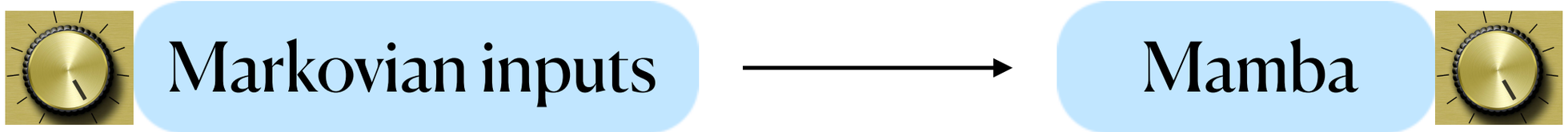
Markov processes 

State-Space Models (Mamba)

From Markov to Laplace: How Mamba In-Context Learns Markov chains

ICLR 2026

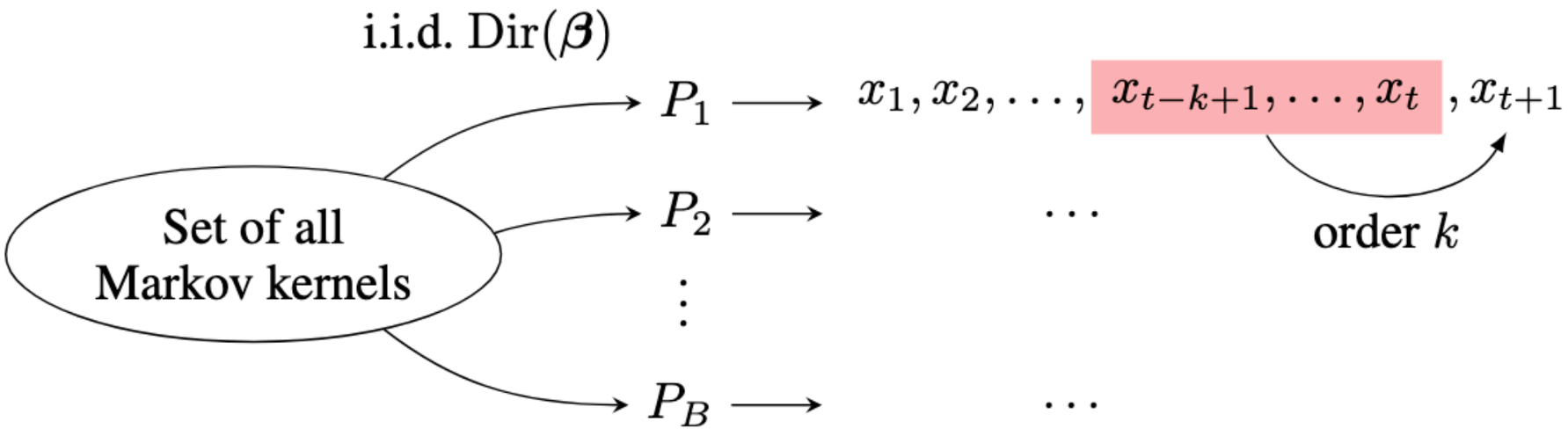
Mamba with Markov



In-context Markov chains

Markovian inputs

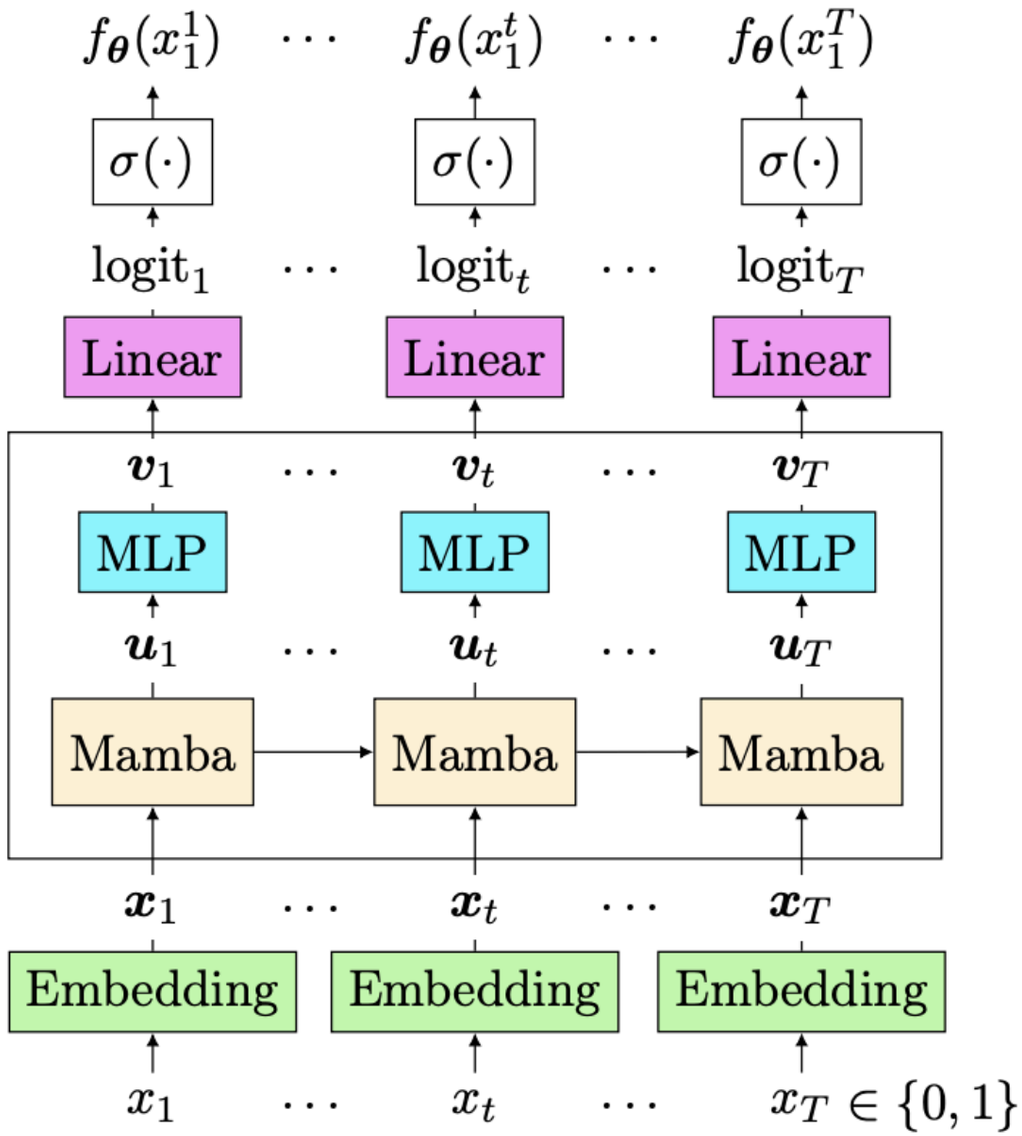
Memory $= k$



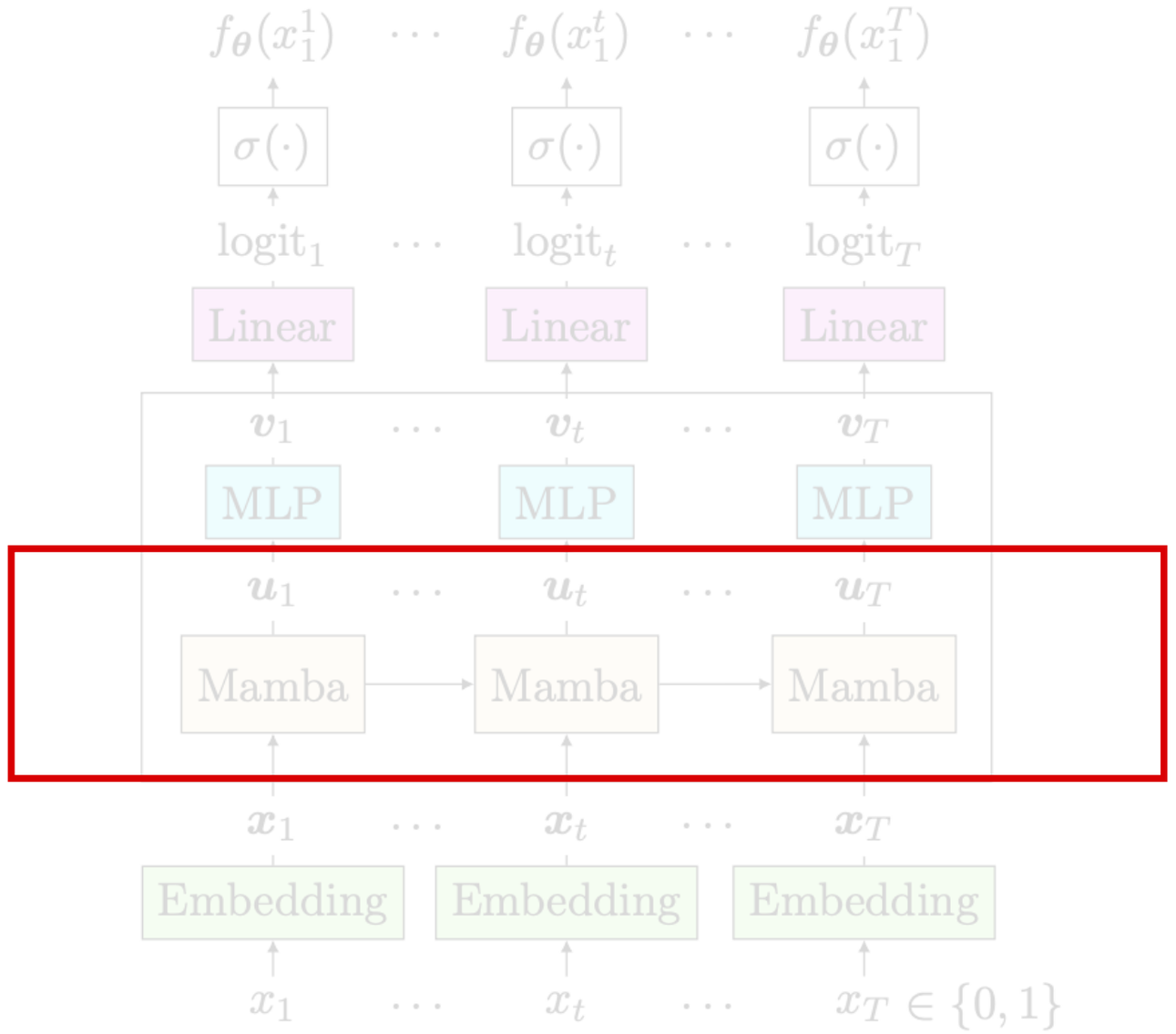
Mamba

$\text{Memory} = k$

Mamba-based language model



Mamba-based language model



RNN

Mamba

$$H_t = a_t\, H_{t-1} + \tilde{\boldsymbol{x}}_t\, \boldsymbol{b}_t^\top \in \mathbb{R}^{ed \times N},$$
$$\boldsymbol{y}_t = H_t\, \boldsymbol{c}_t \in \mathbb{R}^{ed},$$
$$\boldsymbol{z}_t = \boldsymbol{y}_t \odot \mathrm{ReLU}(W_z\, \boldsymbol{x}_t) \in \mathbb{R}^{ed},$$
$$\boldsymbol{o}_t = W_o\, \boldsymbol{z}_t \in \mathbb{R}^{d},$$

Input-selective recurrence

$$\text{Mamba} : \boldsymbol{x} = (\boldsymbol{x}_t)_{t=1}^T \in \mathbb{R}^{d \times T} \mapsto \boldsymbol{o} = (\boldsymbol{o}_t)_{t=1}^T \in \mathbb{R}^{d \times T}$$

Mamba

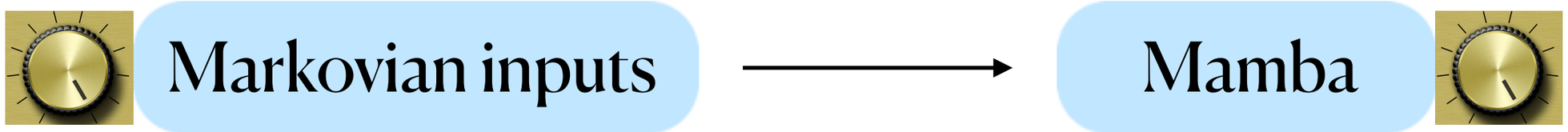
$$H_t = a_t H_{t-1} + \widetilde{\boldsymbol{x}}_t \boldsymbol{b}_t^\top \in \mathbb{R}^{ed \times N},$$
$$\boldsymbol{y}_t = H_t \boldsymbol{c}_t \in \mathbb{R}^{ed},$$
$$\boldsymbol{z}_t = \boldsymbol{y}_t \odot \mathrm{ReLU}(W_z \boldsymbol{x}_t) \in \mathbb{R}^{ed},$$
$$\boldsymbol{o}_t = W_o \boldsymbol{z}_t \in \mathbb{R}^d,$$

Temporal convolution

$$a_t \triangleq \exp(-a \cdot \Delta_t) \in (0,1),$$
$$\Delta_t \triangleq \mathrm{softplus}(\langle \boldsymbol{w}_\Delta, \boldsymbol{x}_t \rangle + \delta) \in \mathbb{R},$$
$$\widetilde{\boldsymbol{x}}_t \triangleq \mathrm{ReLU}(\mathrm{conv}_X(W_X \boldsymbol{x}_{t-w+1}^t)) \cdot \Delta_t,$$
$$\boldsymbol{b}_t \triangleq \mathrm{ReLU}(\mathrm{conv}_B(W_B \boldsymbol{x}_{t-w+1}^t)),$$
$$\boldsymbol{c}_t \triangleq \mathrm{ReLU}(\mathrm{conv}_C(W_C \boldsymbol{x}_{t-w+1}^t)),$$

$$: \boldsymbol{x} = (\boldsymbol{x}_t)_{t=1}^T \in \mathbb{R}^{d \times T} \mapsto \boldsymbol{o} = (\boldsymbol{o}_t)_{t=1}^T \in \mathbb{R}^{d \times T}$$

Mamba with Markov



Next-token prediction loss

$$L(\boldsymbol{\theta}) = -\frac{1}{T}\sum_{t\in[T]} \mathbb{E}_P \mathbb{E}_{x_1^{t+1}\sim P}[\log f_{\boldsymbol{\theta}}^{(x_{t+1})}(x_1^t)]$$

Cross-entropy loss between x_{t+1} and prediction probability $f_{\boldsymbol{\theta}}(x_1^t) = \mathbb{P}_{\boldsymbol{\theta}}(\cdot \mid x_1^t)$

Optimal estimator: Laplacian smoothing

$$\mathbb{P}_\beta(x_{t+1} = j \mid x_1^t) \triangleq \mathbb{E}_P[\mathbb{P}_\beta(x_{t+1} = j \mid x_1^t)] = \frac{n_j + \beta}{n + 2\beta}$$

Prediction probability

Add-β estimator

Optimal estimator: Laplacian smoothing

$$\mathbb{P}_\beta(x_{t+1} = j \mid x_1^t) \triangleq \mathbb{E}_P[\mathbb{P}_\beta(x_{t+1} = j \mid x_1^t)] = \frac{n_j + \beta}{n + 2\beta}$$

Prediction probability

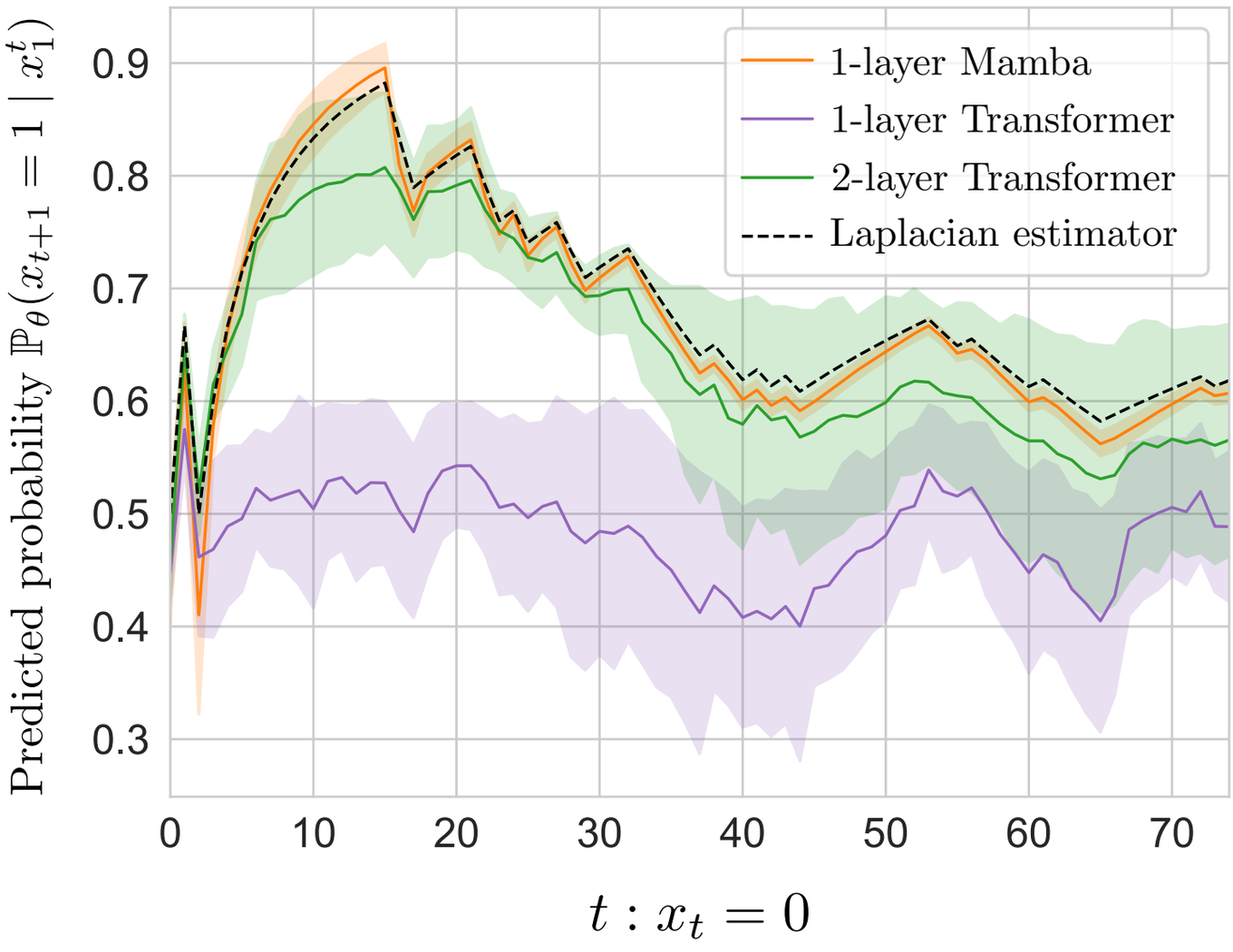
Add-β estimator

n_j : no. of times token j follows the context x_{t-k+1}^t

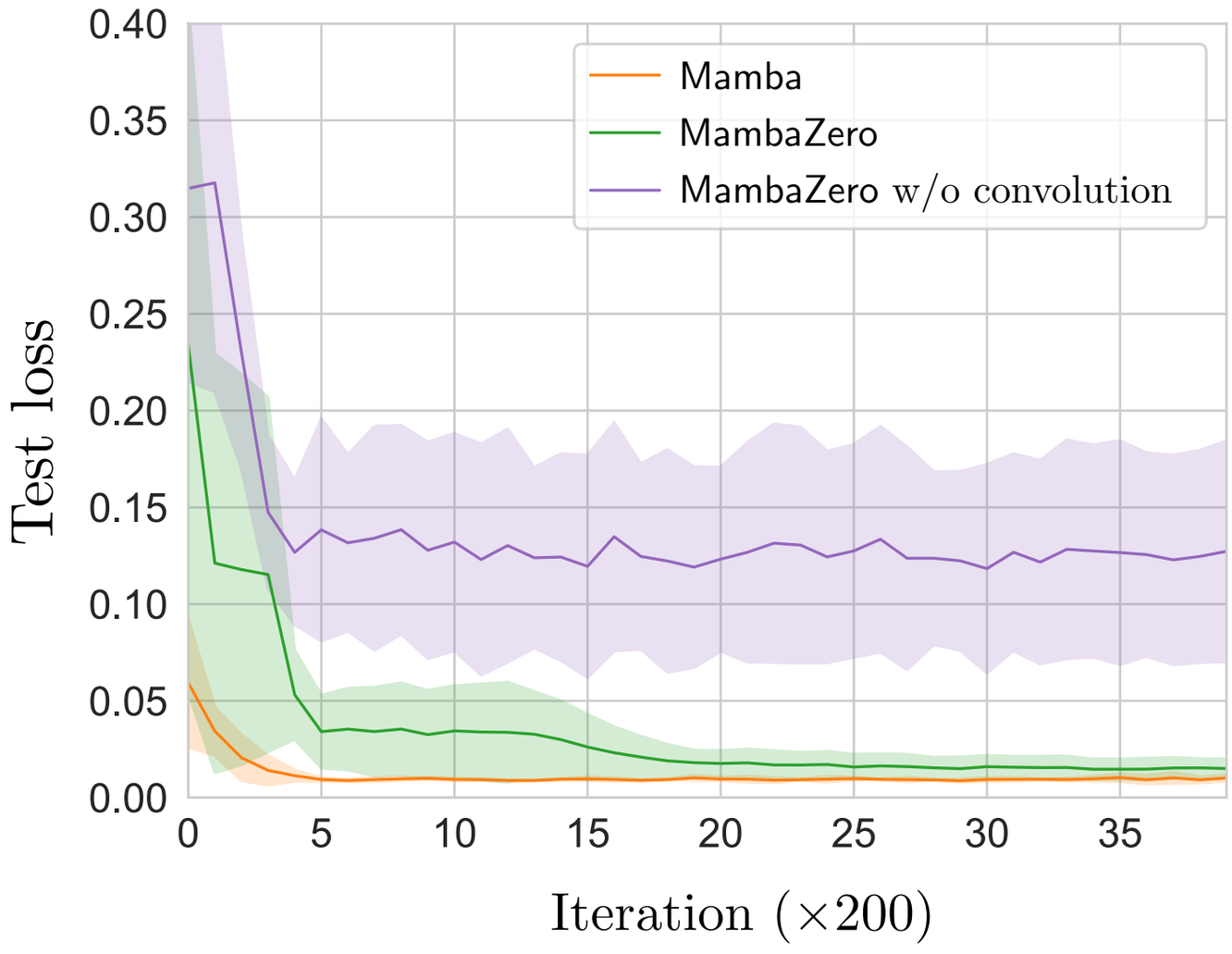
n : no. of times context x_{t-k+1}^t occurs

Can Mamba learn Laplacian smoothing?

Can Mamba learn Laplacian smoothing?



Convolution is the key



MambaZero = Mamba with just the convolution

Main result

MambaZero realizes Laplacian smoothing for first-order

For first-order Markov processes on any finite state-space, there exists a realization of MambaZero model such that for all sequences $(x_t)_{t\geq 1}$ and all $t \geq 1$:

$$D_{\mathrm{KL}}\left(\mathbb{P}_{\beta}(\cdot \mid x_1^t) || \mathbb{P}_{\boldsymbol{\theta}}(\cdot \mid x_1^t)\right) = 0.$$

Main result

MambaZero realizes Laplacian smoothing for first-order

For first-order Markov processes on any finite state-space, there exists a realization of MambaZero model such that for all sequences $(x_t)_{t\geq 1}$ and all $t \geq 1$:

$$D_{\mathrm{KL}}\left(\mathbb{P}_{\beta}(\cdot \mid x_1^t) || \mathbb{P}_{\boldsymbol{\theta}}(\cdot \mid x_1^t)\right) = 0.$$

Empirically true for higher-orders too

Intuition

$$
\begin{aligned}
H_t &= a_t\, H_{t-1} + \widetilde{\boldsymbol{x}}_t\, \boldsymbol{b}_t^\top \in \mathbb{R}^{ed \times N}, \\
\boldsymbol{y}_t &= H_t\, \boldsymbol{c}_t \in \mathbb{R}^{ed}, \\
\boldsymbol{z}_t &= \boldsymbol{y}_t \odot \mathrm{ReLU}(W_z\, \boldsymbol{x}_t) \in \mathbb{R}^{ed}, \\
\boldsymbol{o}_t &= W_o\, \boldsymbol{z}_t \in \mathbb{R}^{d},
\end{aligned}
$$

$$
\begin{aligned}
a_t &\triangleq \exp(-a \cdot \Delta_t) \in (0,1), \\
\Delta_t &\triangleq \mathrm{softplus}(\langle \boldsymbol{w}_\Delta, \boldsymbol{x}_t \rangle + \delta) \in \mathbb{R}, \\
\widetilde{\boldsymbol{x}}_t &\triangleq \mathrm{ReLU}(\mathrm{conv}_X(W_X\, \boldsymbol{x}_{t-w+1}^t)) \cdot \Delta_t, \\
\boldsymbol{b}_t &\triangleq \mathrm{ReLU}(\mathrm{conv}_B(W_B\, \boldsymbol{x}_{t-w+1}^t)), \\
\boldsymbol{c}_t &\triangleq \mathrm{ReLU}(\mathrm{conv}_C(W_C\, \boldsymbol{x}_{t-w+1}^t)),
\end{aligned}
$$

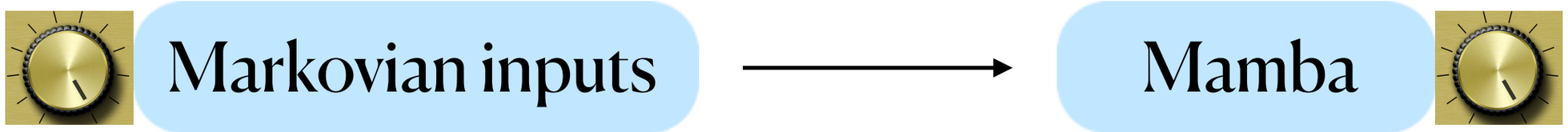
↓

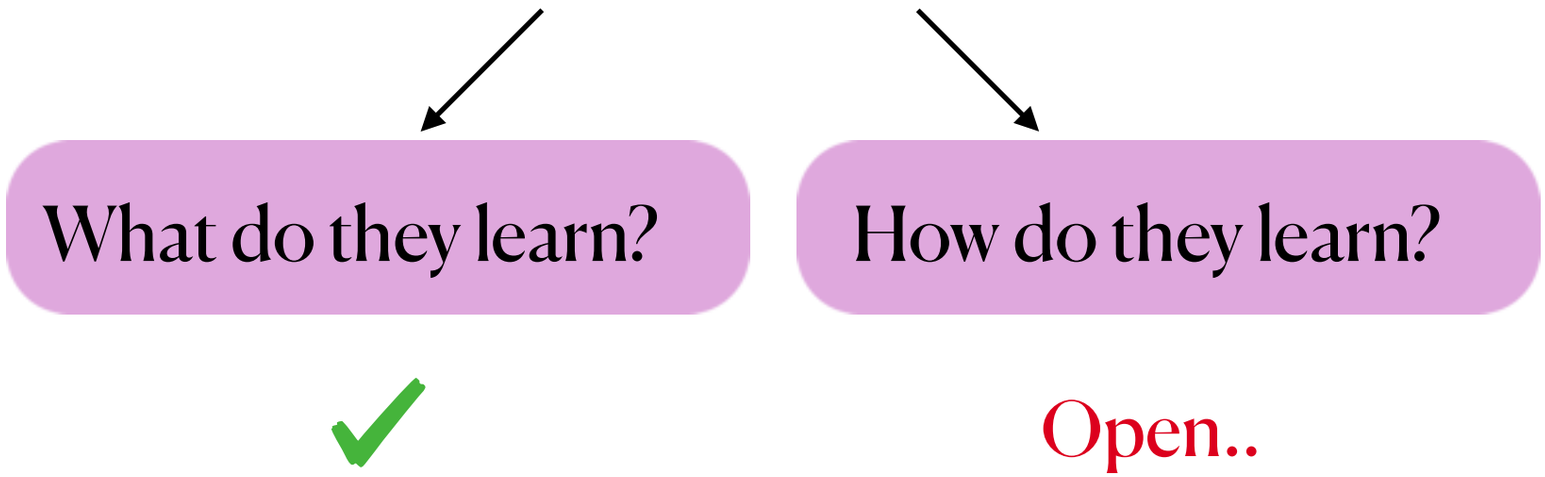
Capture only context-relevant counting

↓

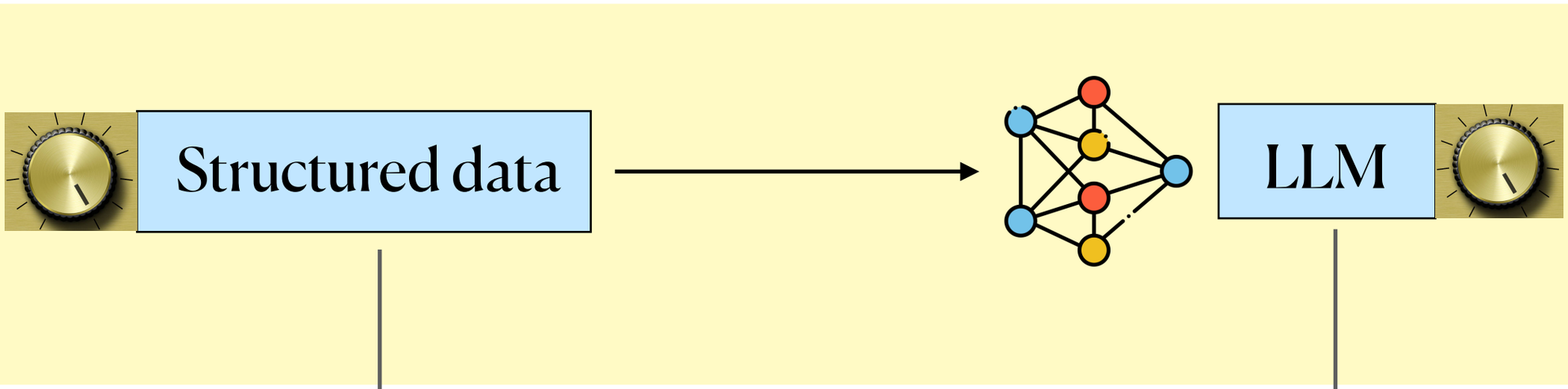
Appropriate convolution kernel and matrices

Mamba with Markov

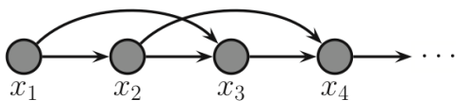




Summary



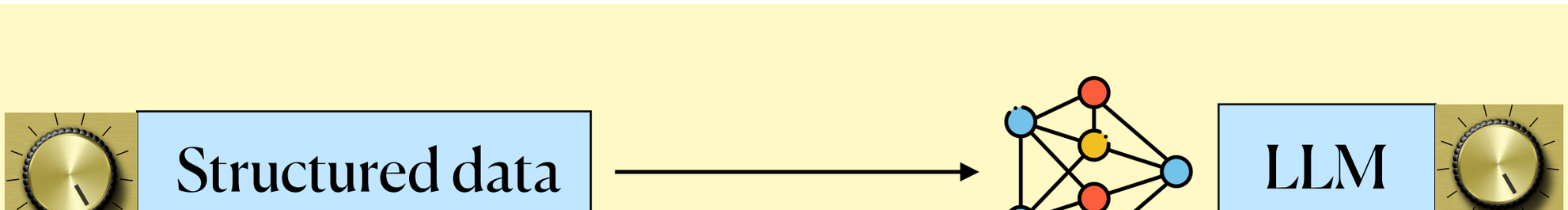
Markov processes



Transformers

State-Space Models

Summary



Markov processes

Transformers

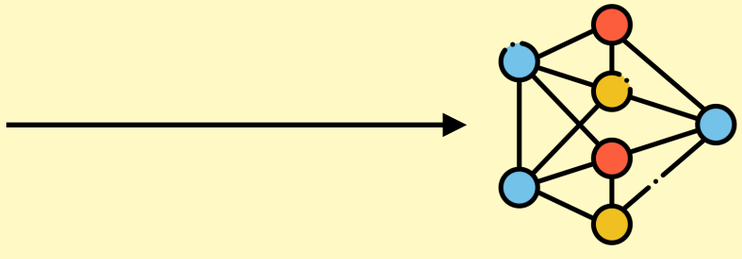
State-Space Models

Non-linearity

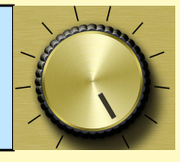
Convolution

Future directions

Structured data

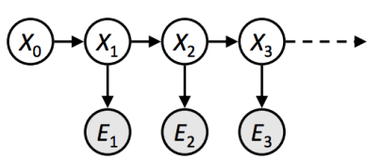


LLM



Multi-hop reasoning

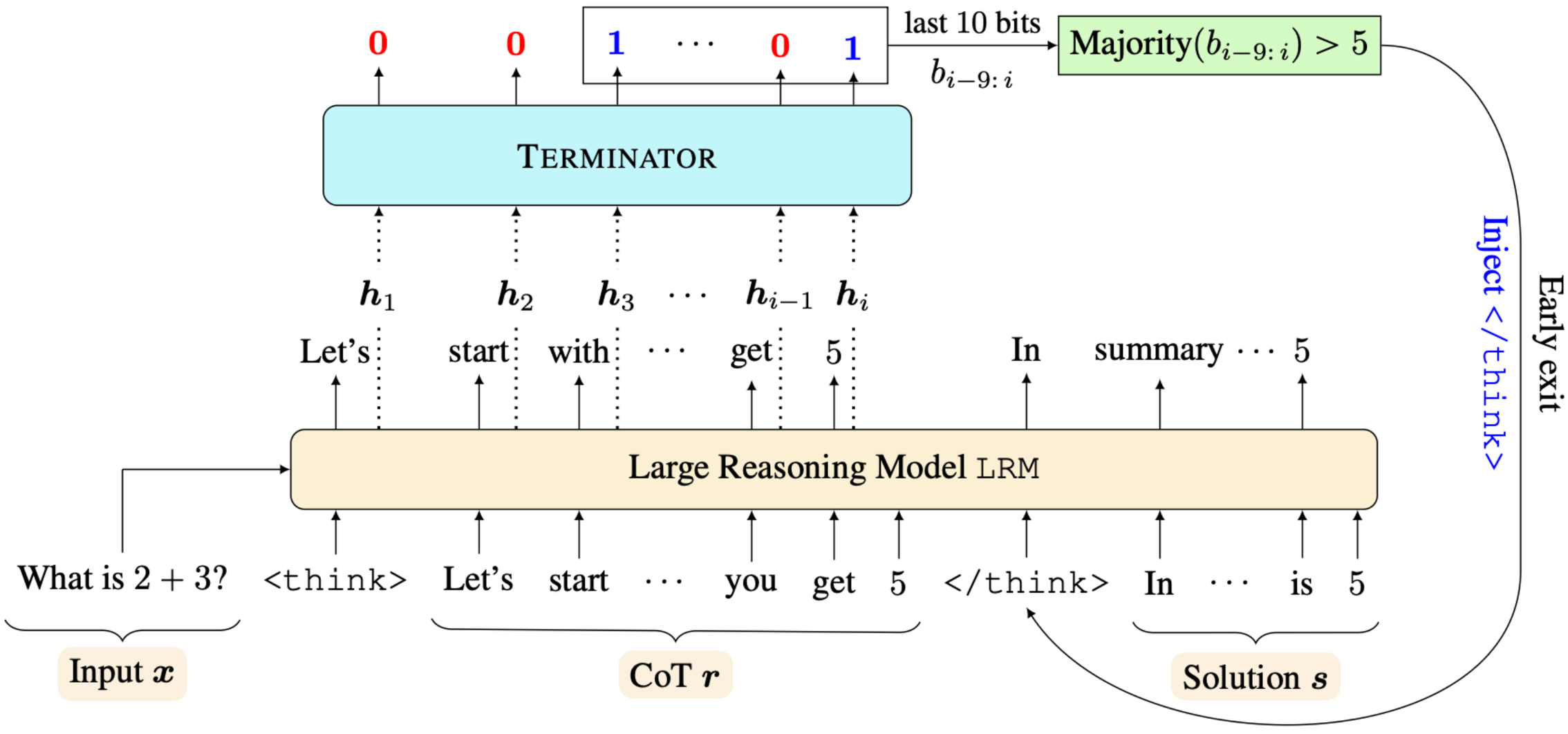
Hidden-Markov-Models

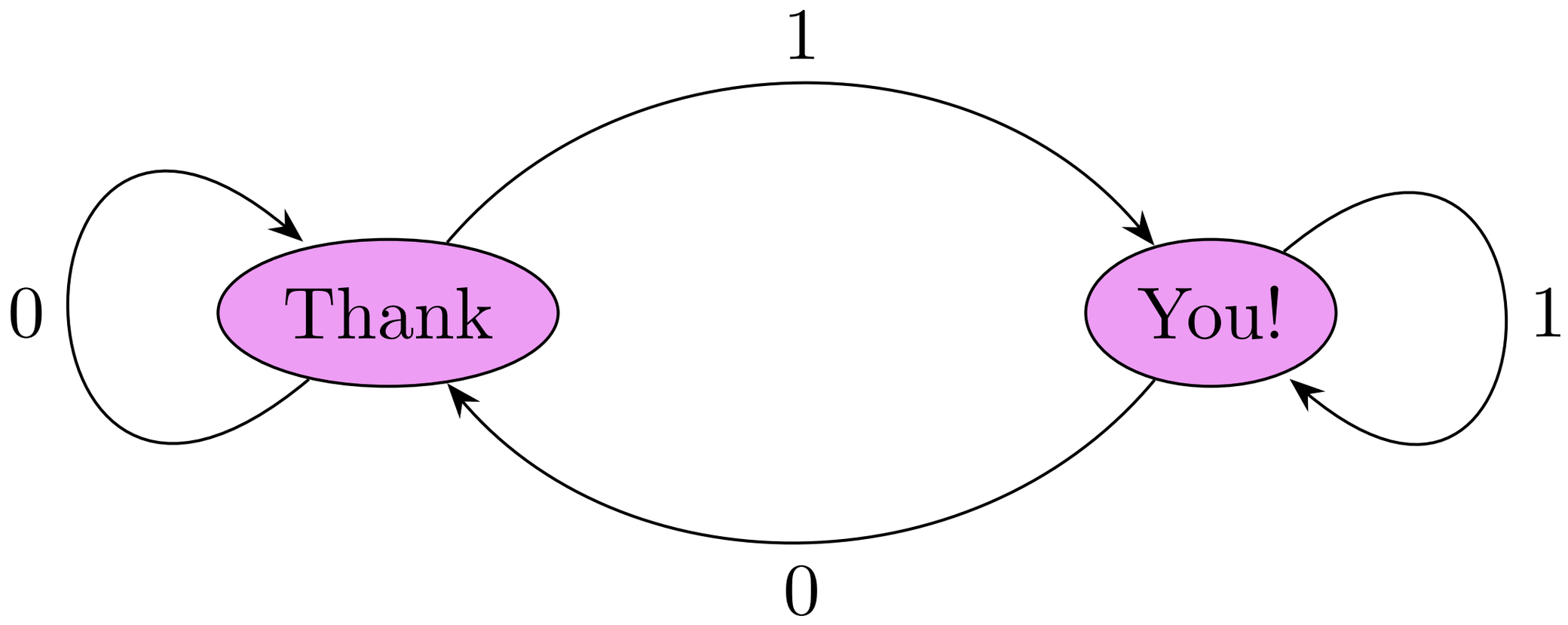


Diffusion Models

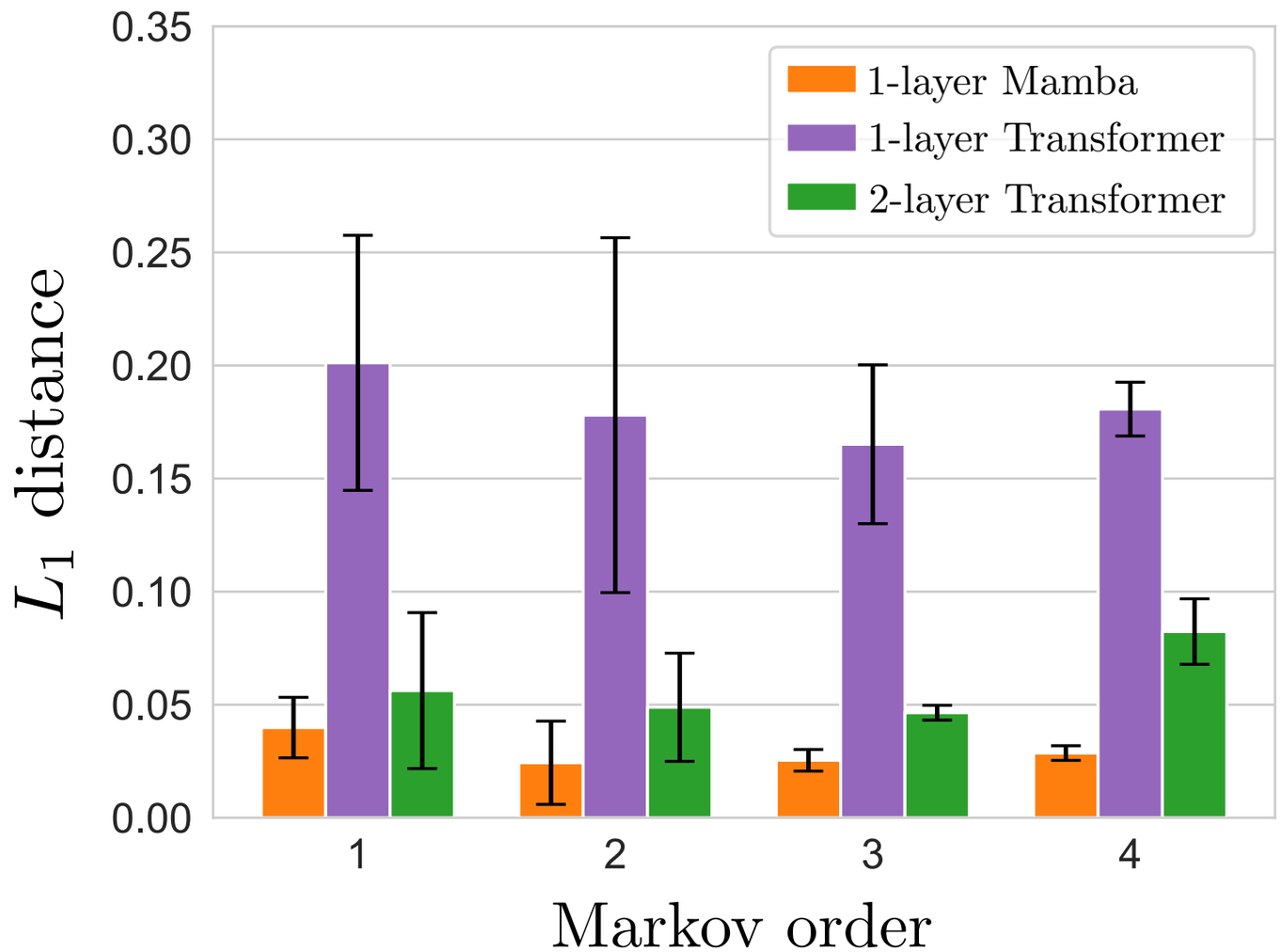
Efficient and hybrid architectures

Terminator: Efficient Reasoning




1
0
Thank
You!
1
0

All Markov orders



Convolution window vs. Markov order