Attention with Markov: A Markovian Tale of Transformers

Ashok Vardhan Makkuva

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

Berkeley
NEVER EXPOSE THE WRONGDOING OF THE
UNIVERSITY OF CALIFORNIA

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

TEXAS
The University of Texas at Austin

Attention with Markov

Attention with Markov

Attention with Markov

Attention

Attention

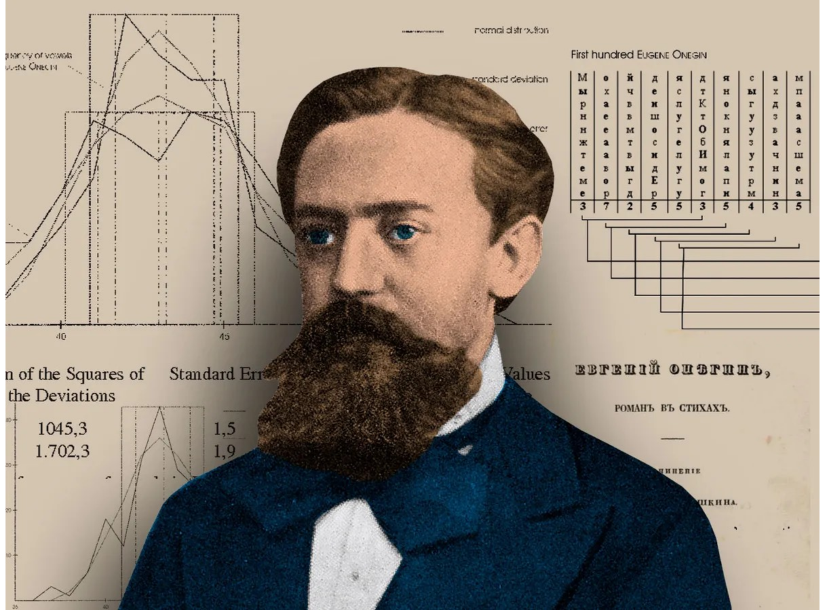
ChatGPT

GPT-4

Attention with Markov

ChatGPT

GPT-4

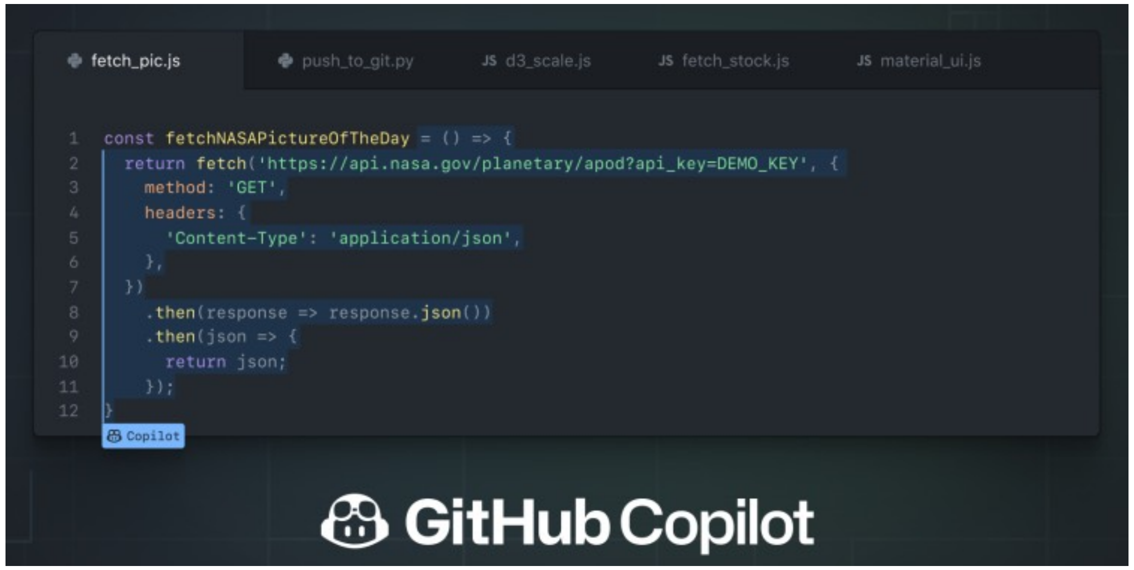


Attention with Markov

ChatGPT

GPT-4

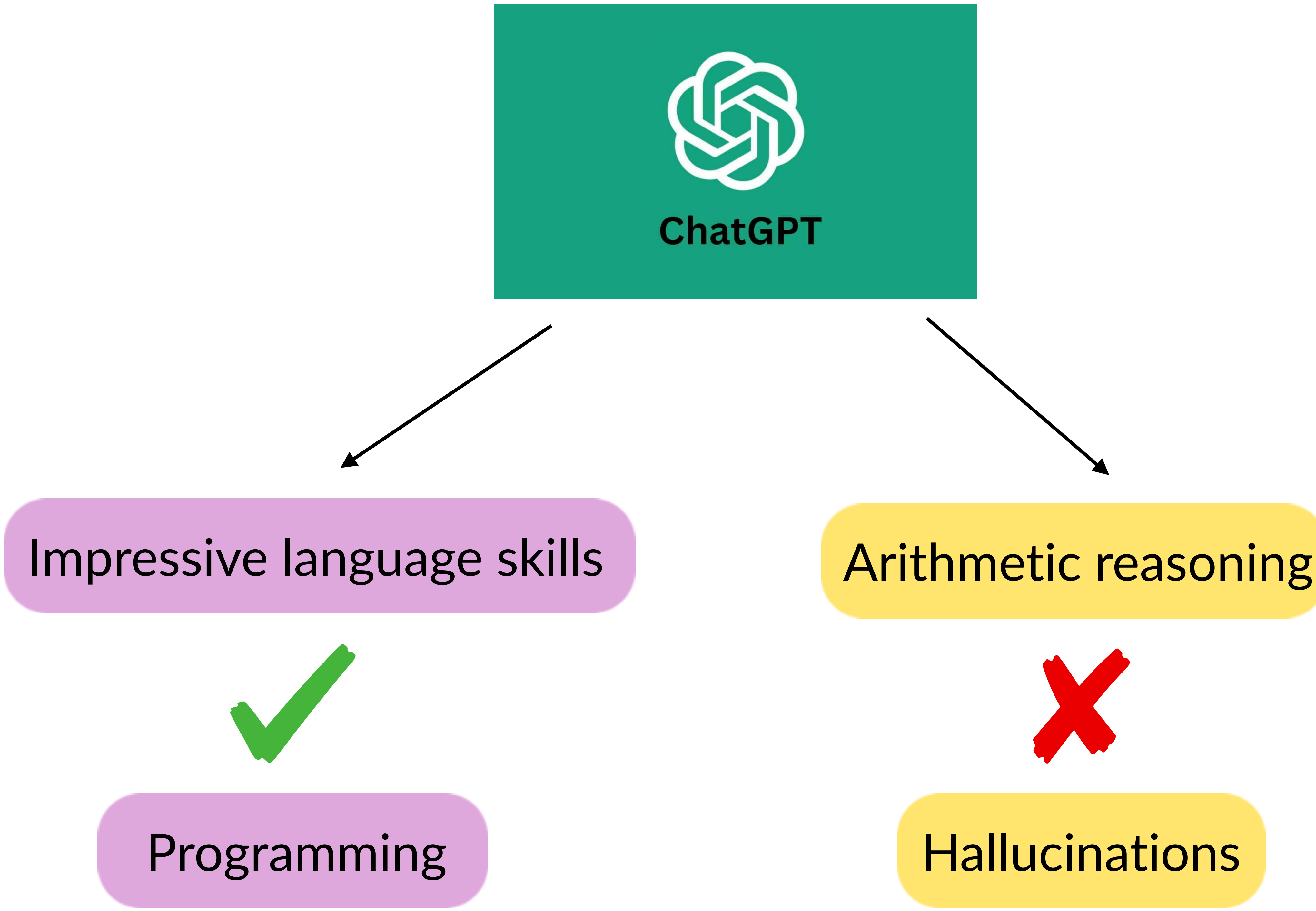
LLMs are part of daily lives



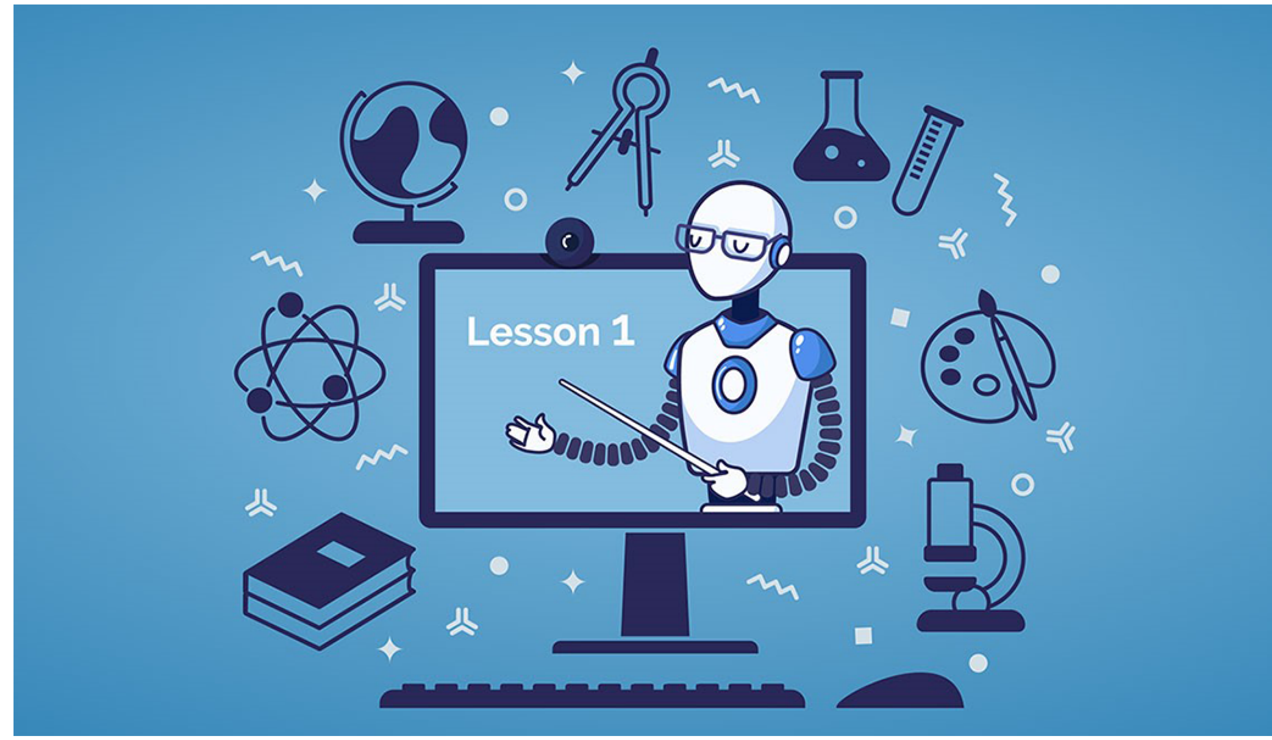
The good and the bad

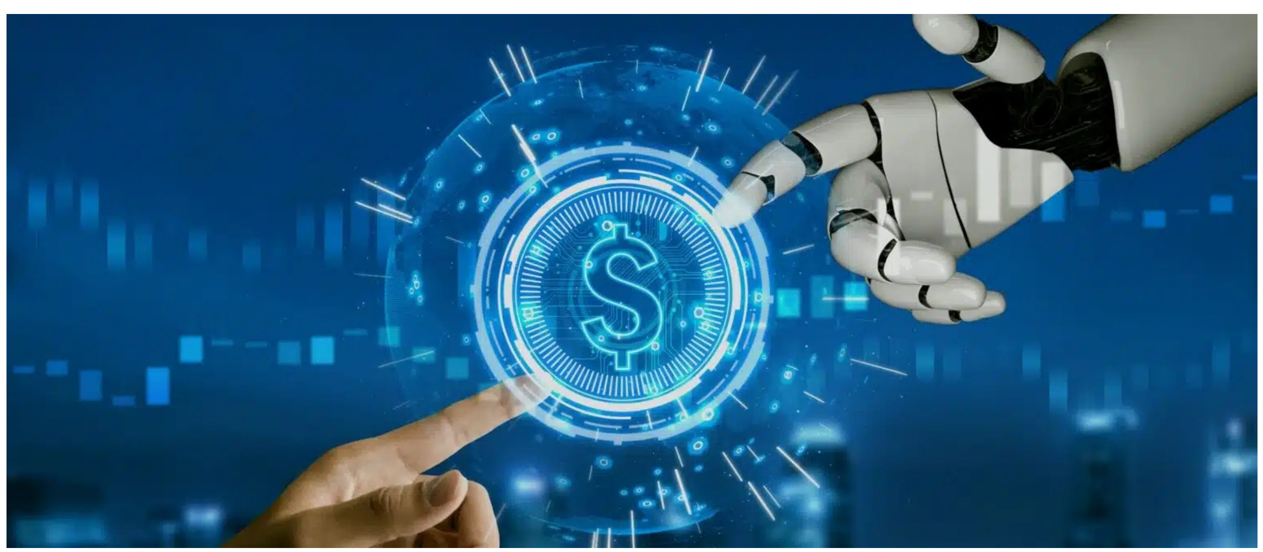
ChatGPT

The good and the bad



Critical domains





Need of the hour

ChatGPT

Fundamental understanding

ChatGPT

Fundamental understanding

What do they learn?

How do they learn?

ChatGPT

Principled frameworks and tools

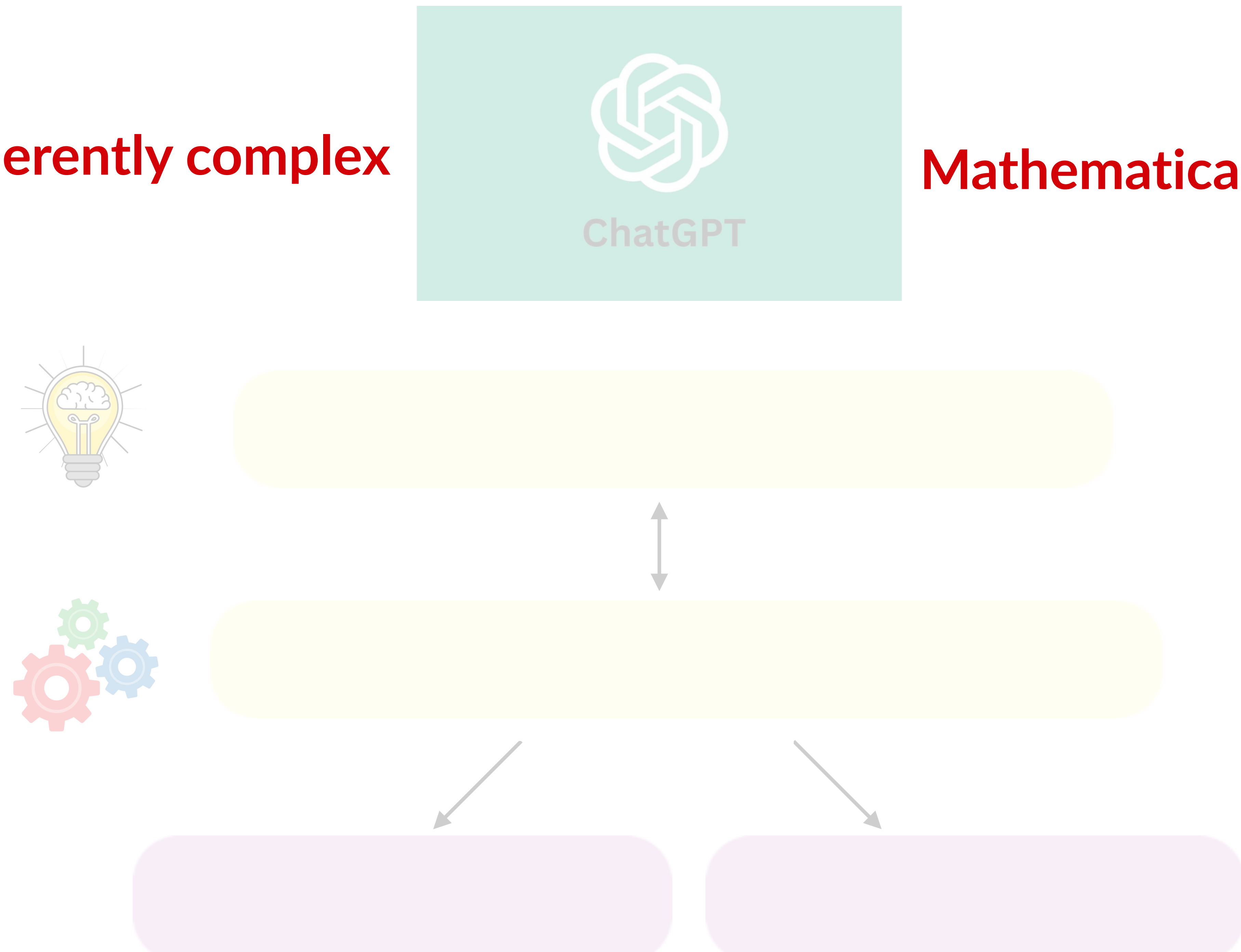
What do they learn?

How do they learn?

Challenges

Challenges

Inherently complex

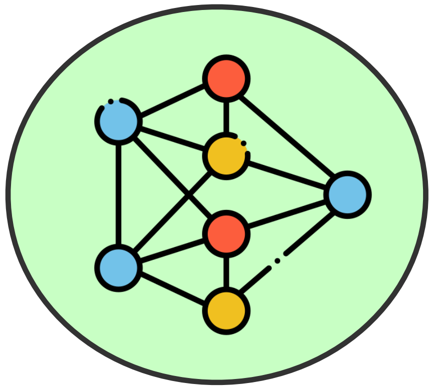


Mathematically intractable

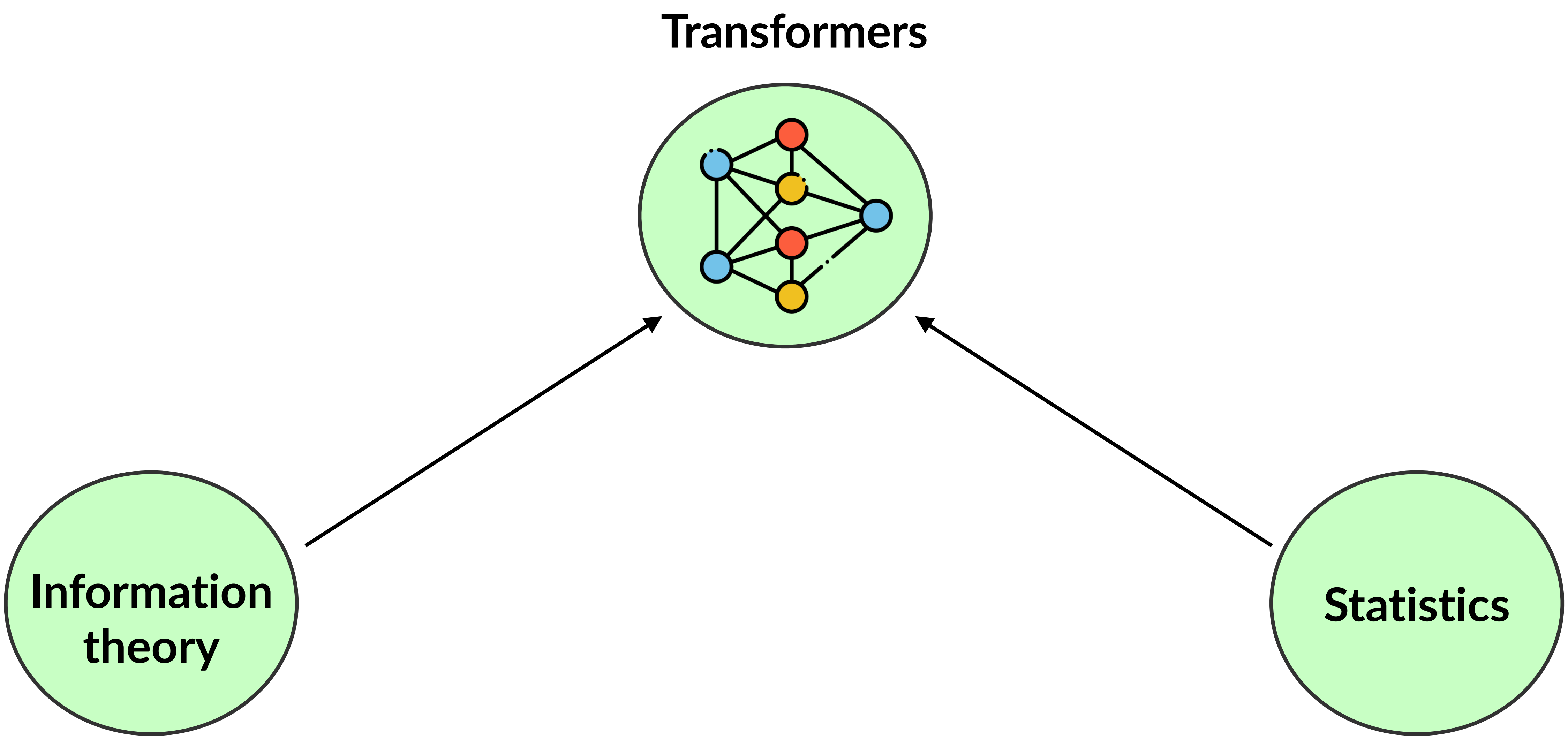
My research

My research

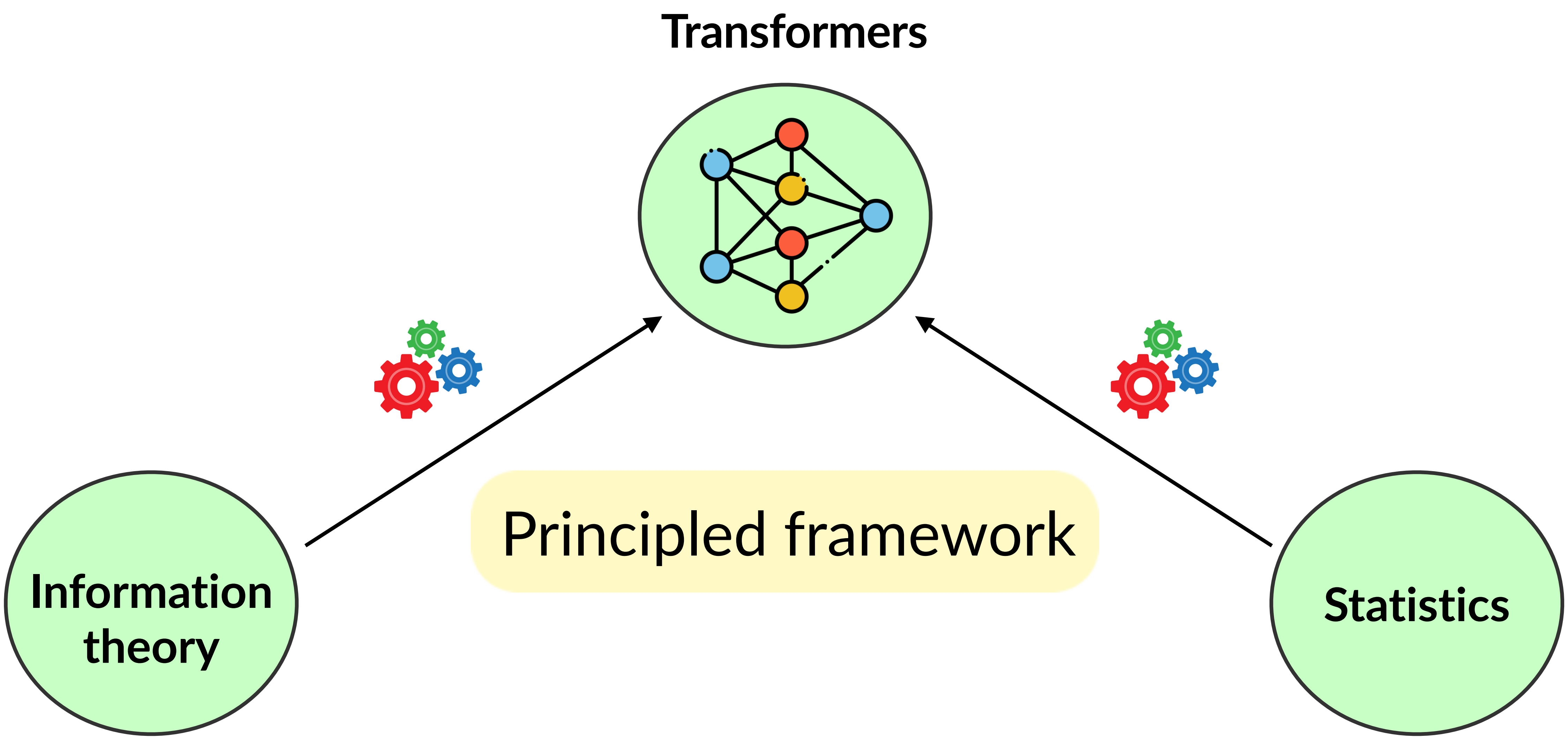
Transformers



My research



My research



Transformers
Principled framework
Information theory
Statistics


Attention with Markov

Attention with Markov

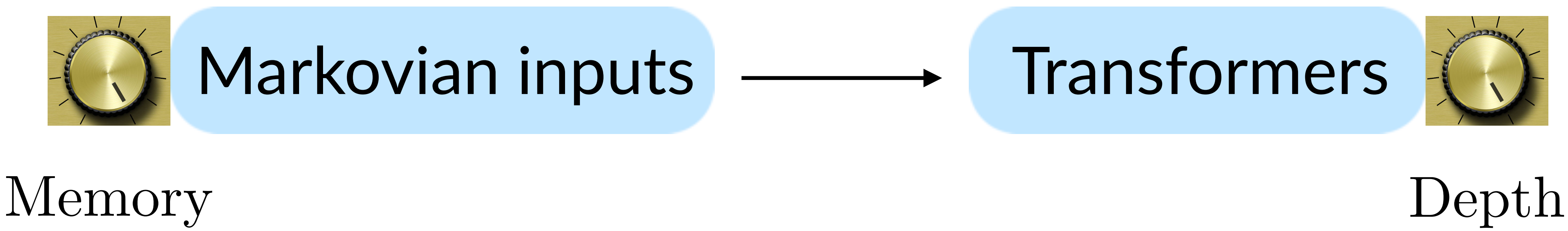
Attention with Markov

Sequential inputs $\longrightarrow$ Transformers

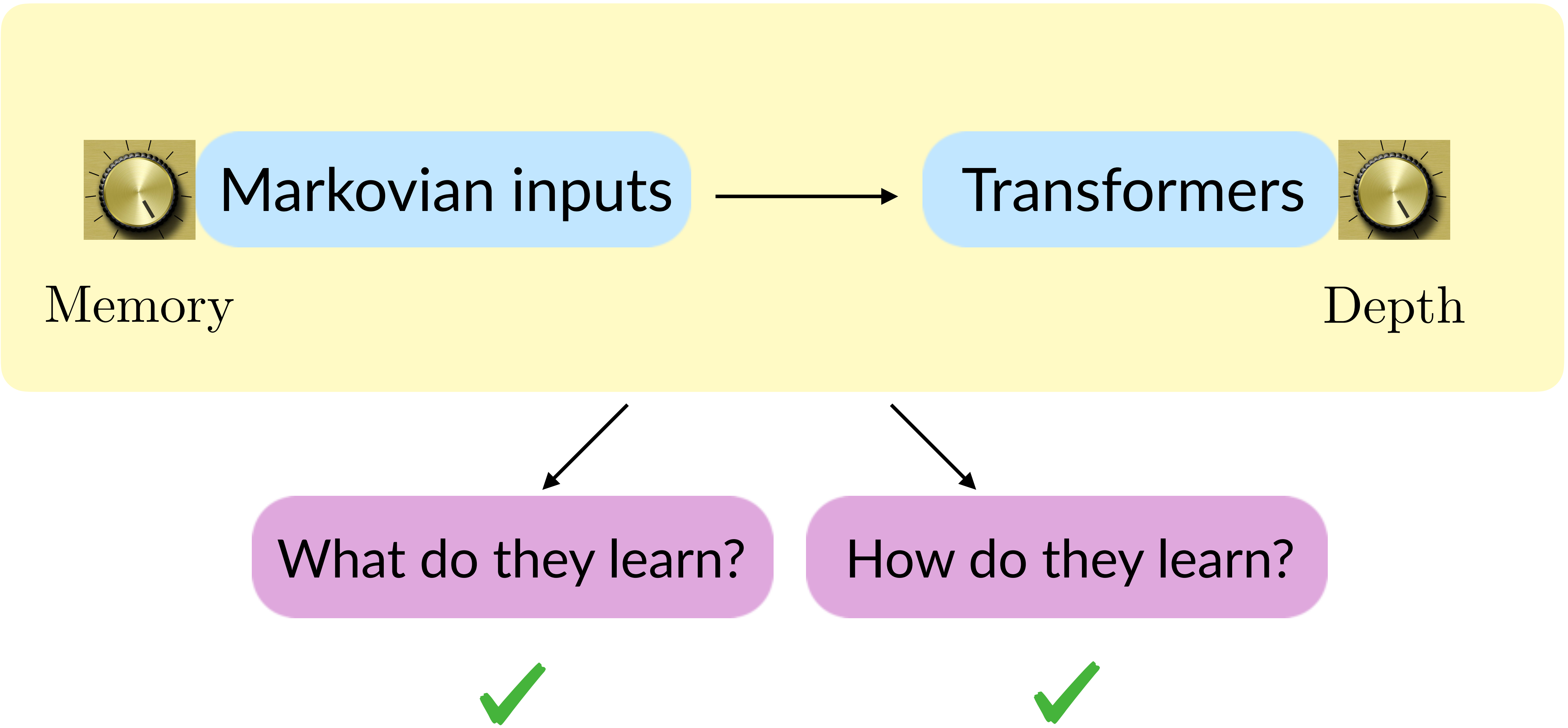
Attention with Markov

Markovian inputs → Transformers

Attention with Markov



Attention with Markov



Depth plays a crucial role in transformer functionality

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

3-layer transformer can learn Markov chains of all orders

This talk

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

NeurIPS 2024

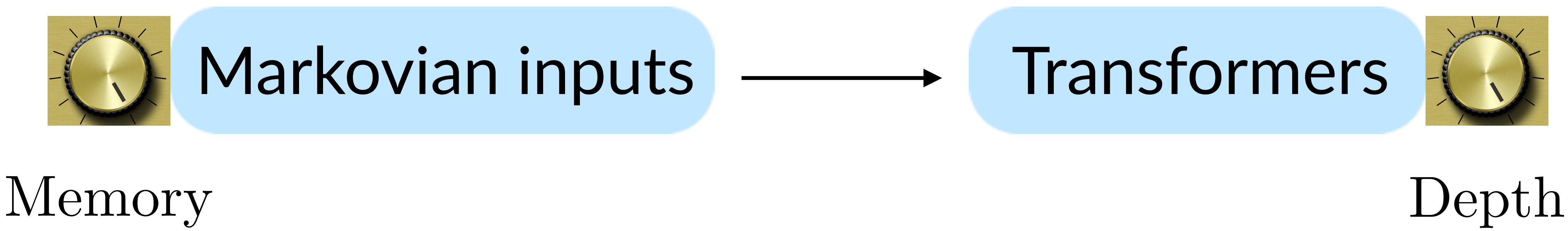
Transformers on Markov Data: Constant Depth Suffices

NeurIPS 2024

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

ICML 2024 Workshop, Under review ICLR 2025 (All hail Reviewer #2)

Attention with Markov



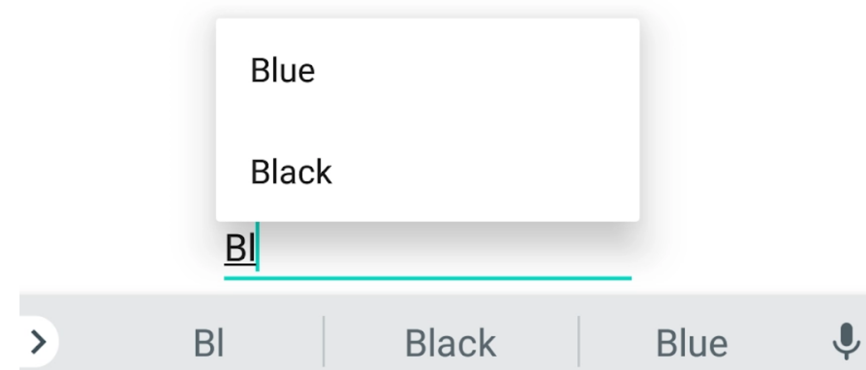
Why Markovian?

Shannon, 1948

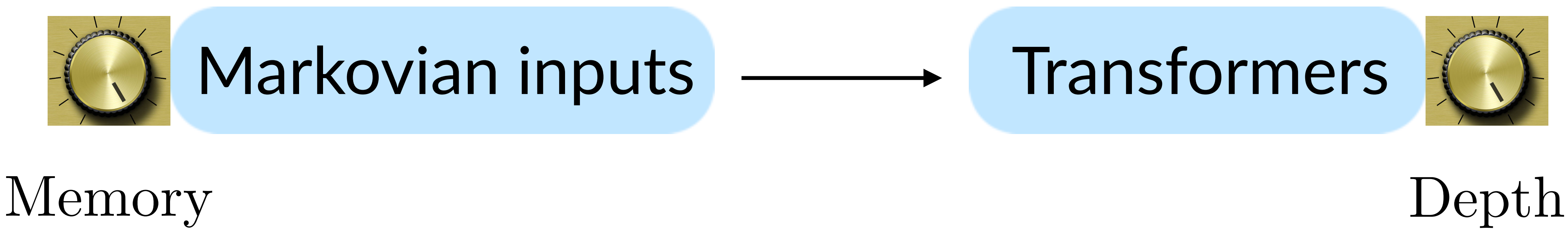
Grammar

Syntax

Markovian



Attention with Markov

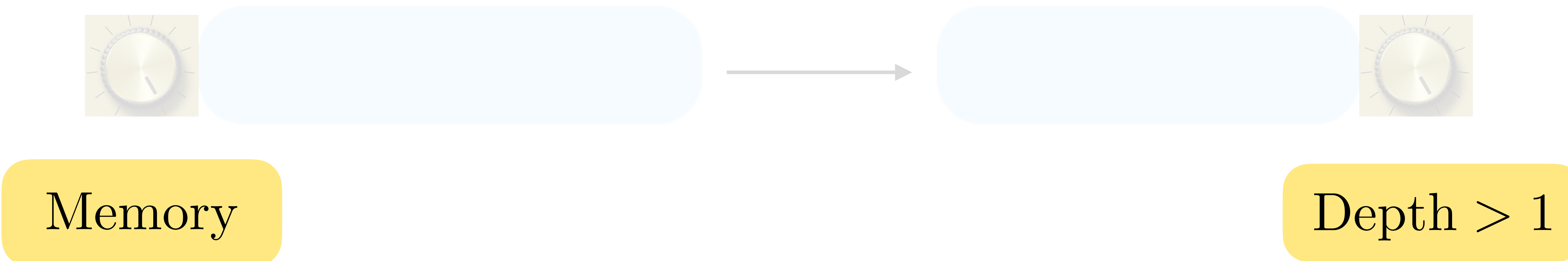


Part I

$\text{Memory} = 1$

$\text{Depth} = 1$

Part II



Part I

Memory $= 1$

Depth $= 1$

Part I

Markovian inputs

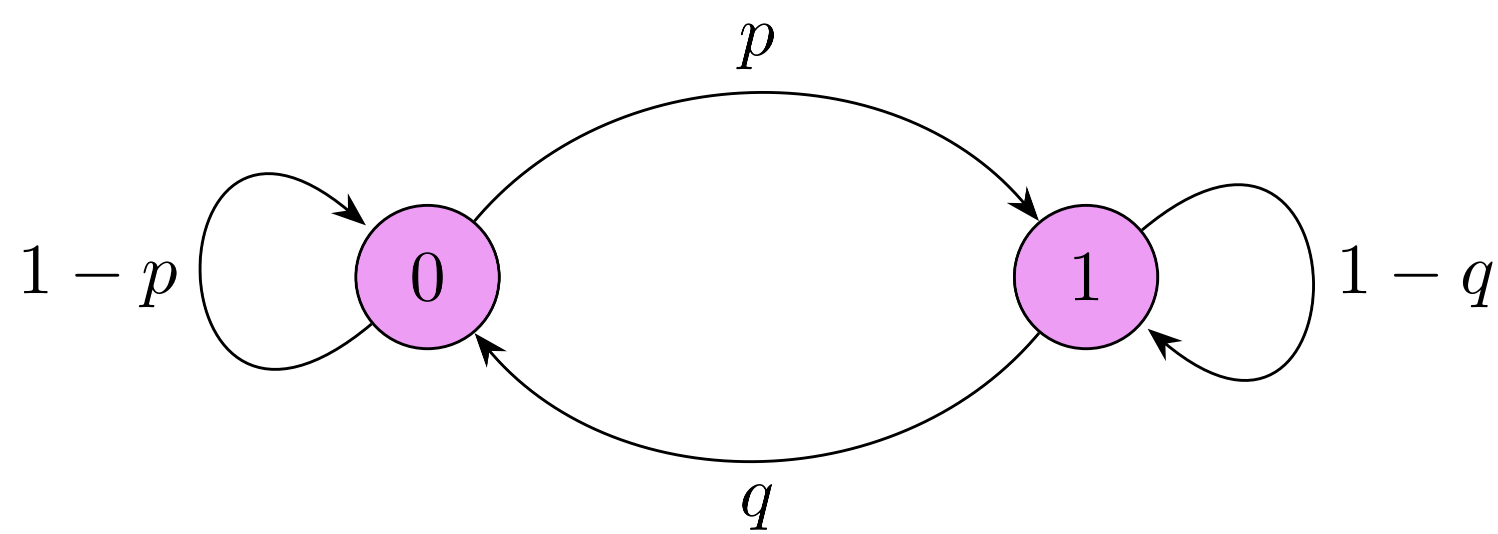
$\text{Memory} = 1$

$\text{Depth} = 1$

Input data: First-order Markov chain

Input data: First-order Markov chain

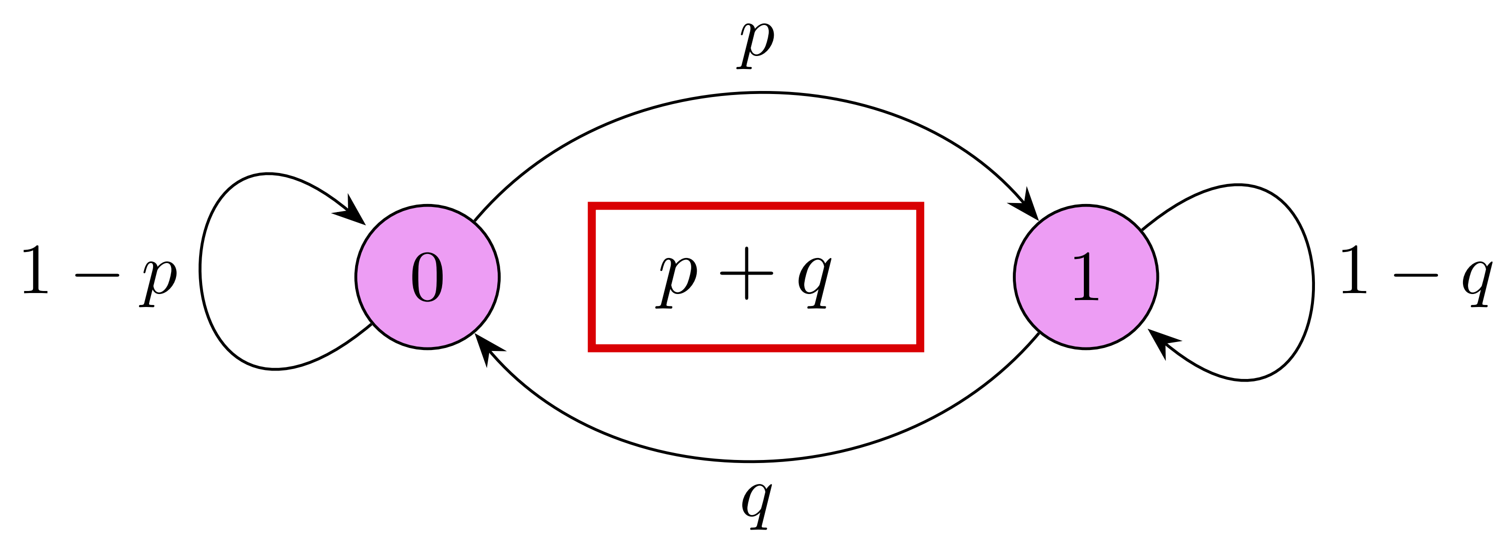
$$(x_n)_{n\geq 1} \sim (\boldsymbol{\pi}, \boldsymbol{P}) \quad \Leftrightarrow$$



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

Input data: First-order Markov chain

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



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

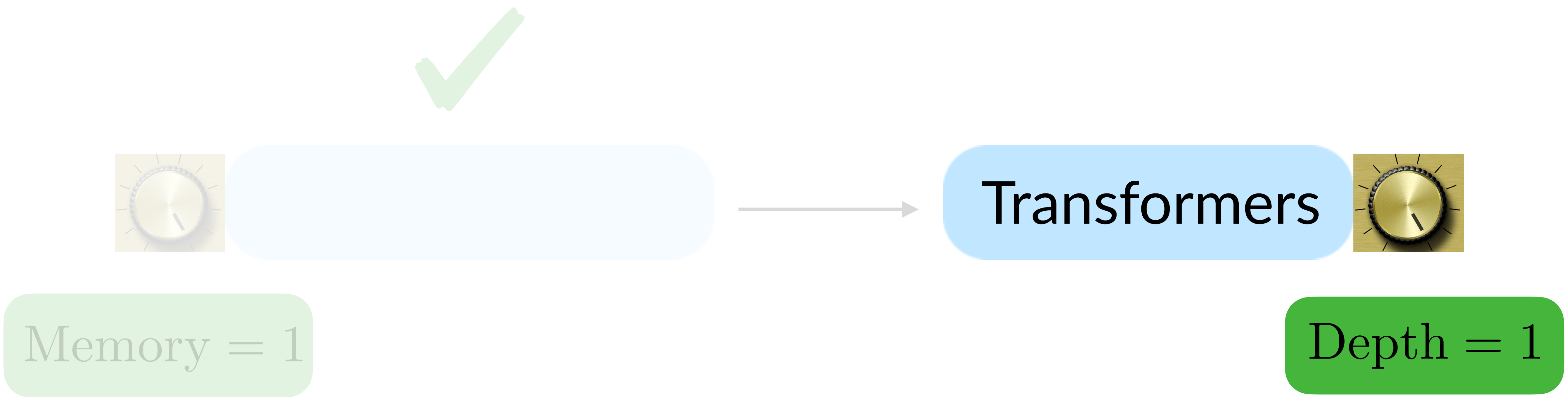
Part I

Markovian inputs

$\text{Memory} = 1$

$\text{Depth} = 1$

Part I

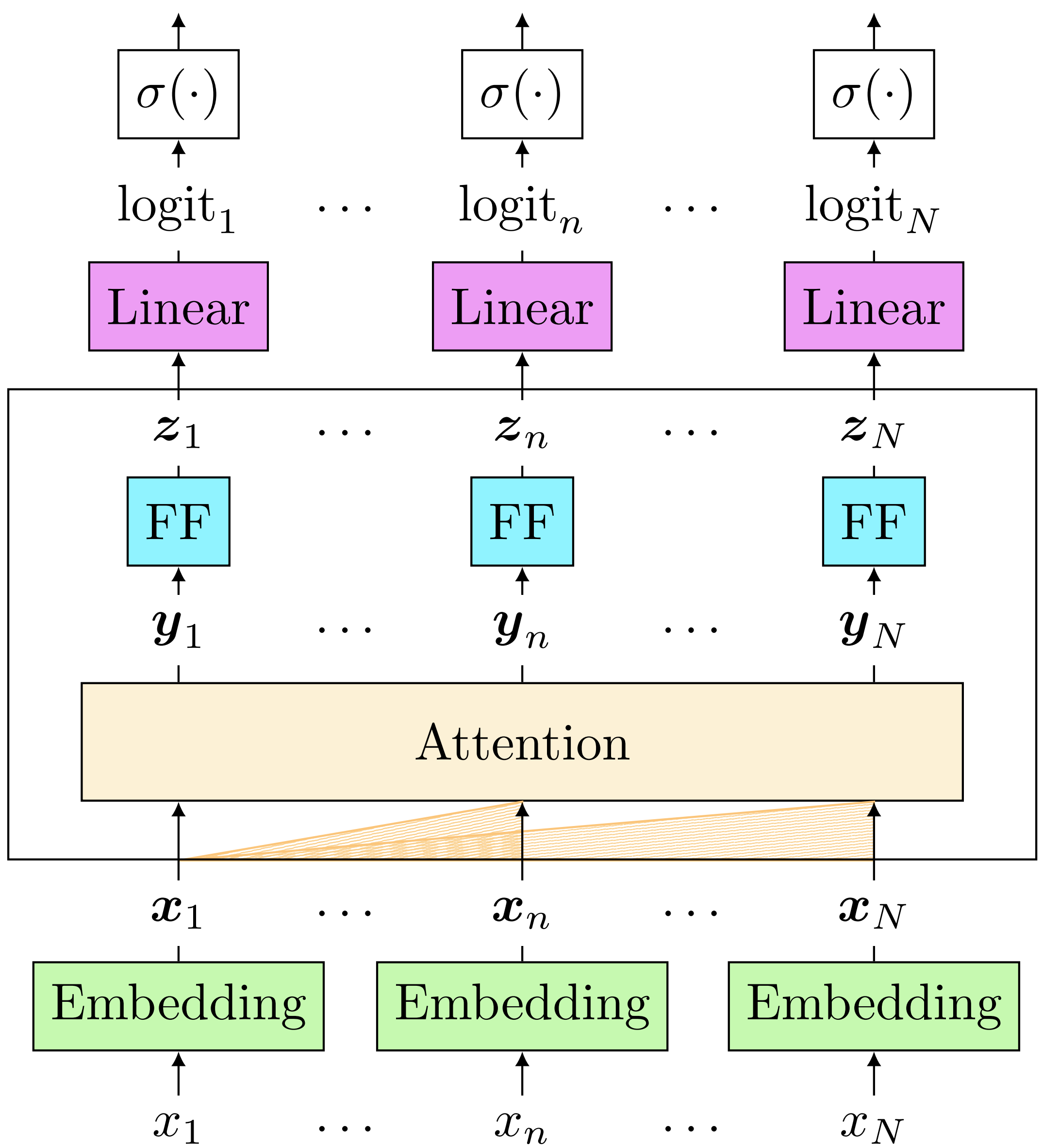


Transformers

Transformers

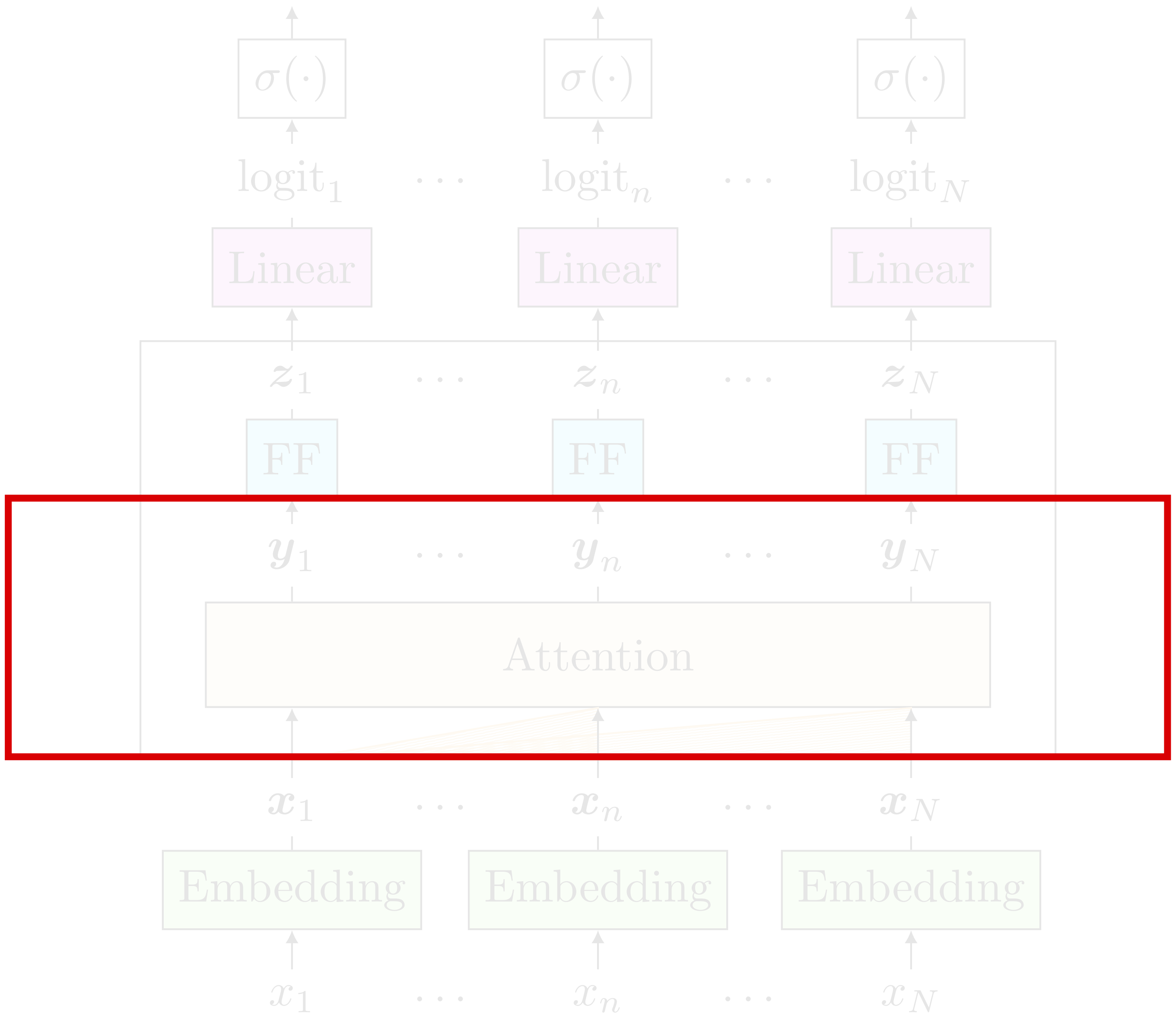
Transformers

Transformers

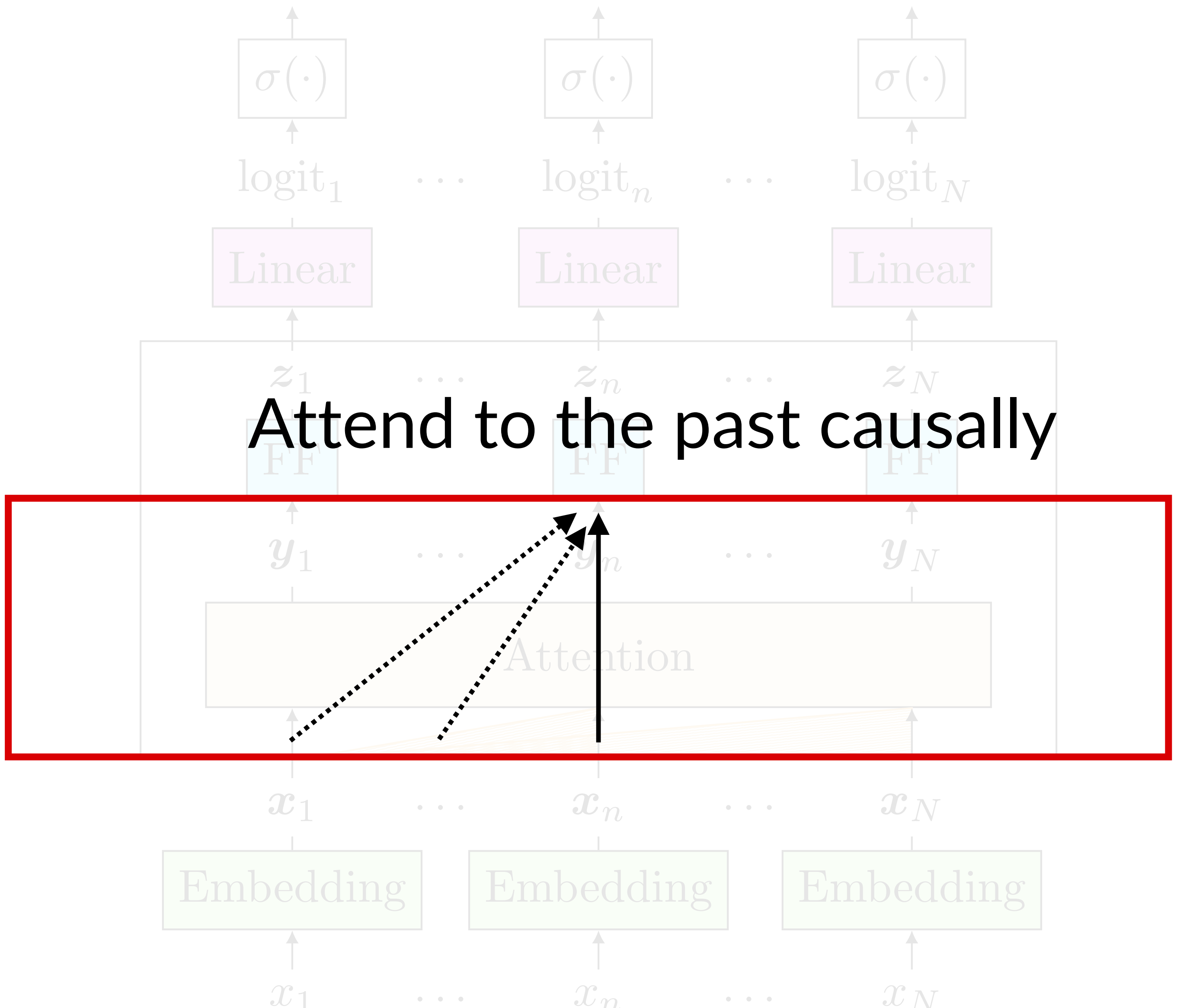


(Vaswani et. al., 2017)

Transformers



Transformers



Single-layer transformer

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

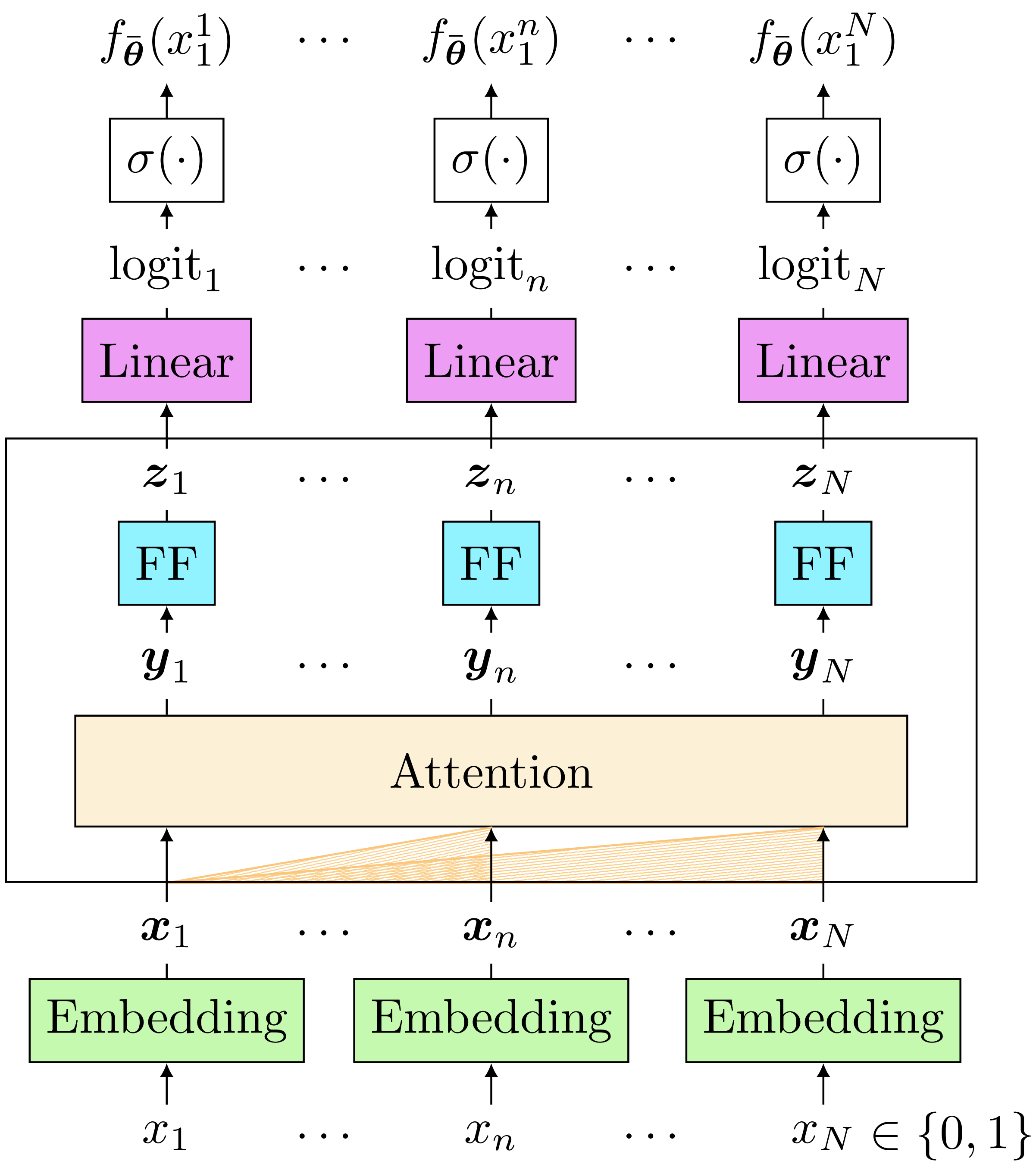
$$\mathrm{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

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

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

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

$\boldsymbol{\theta}$



Single-layer transformer

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

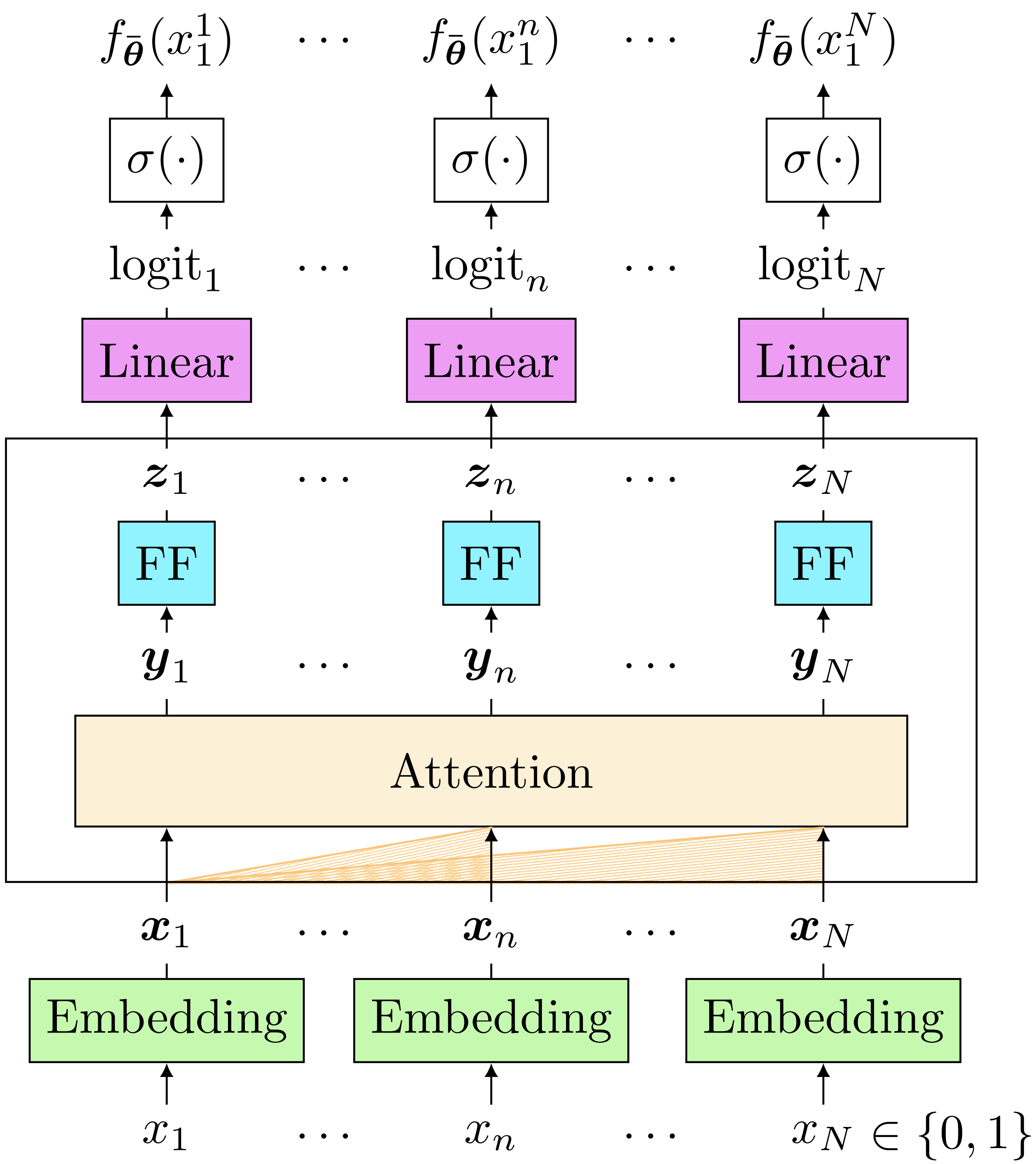
$$\mathrm{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

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

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

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

$\boldsymbol{\theta}$



Single-layer transformer

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

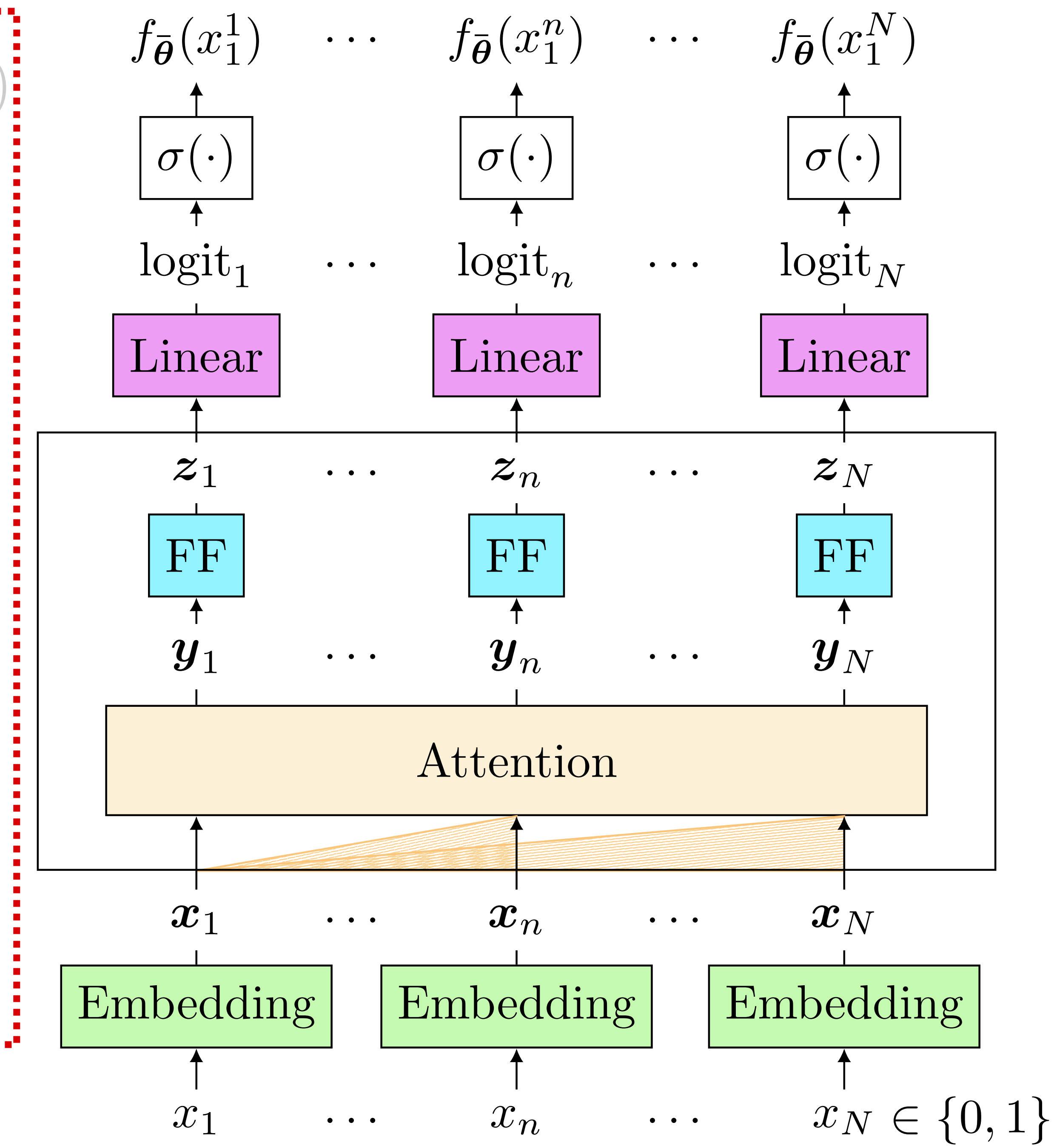
$$\text{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

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

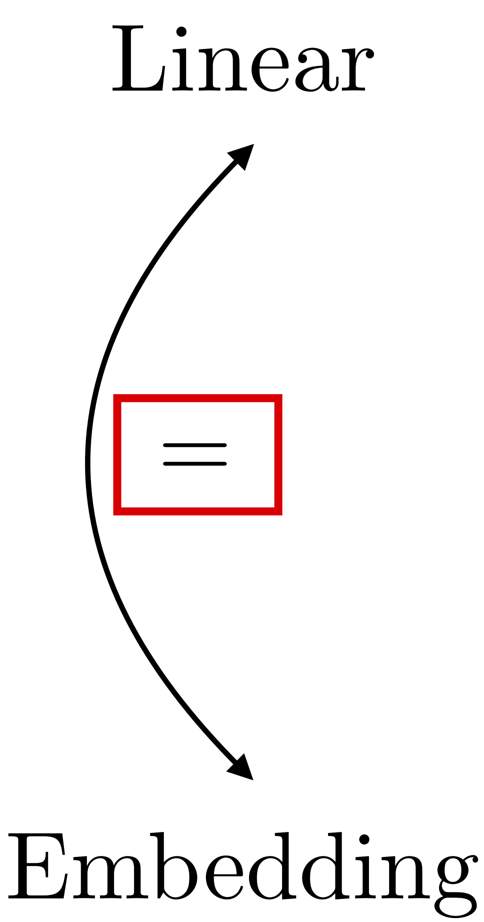
$$\boldsymbol{y}_n = \boldsymbol{x}_n + \boldsymbol{W}_O \sum_{i=1}^{n} \text{att}_{n,i} \cdot \boldsymbol{W}_V \boldsymbol{x}_i$$

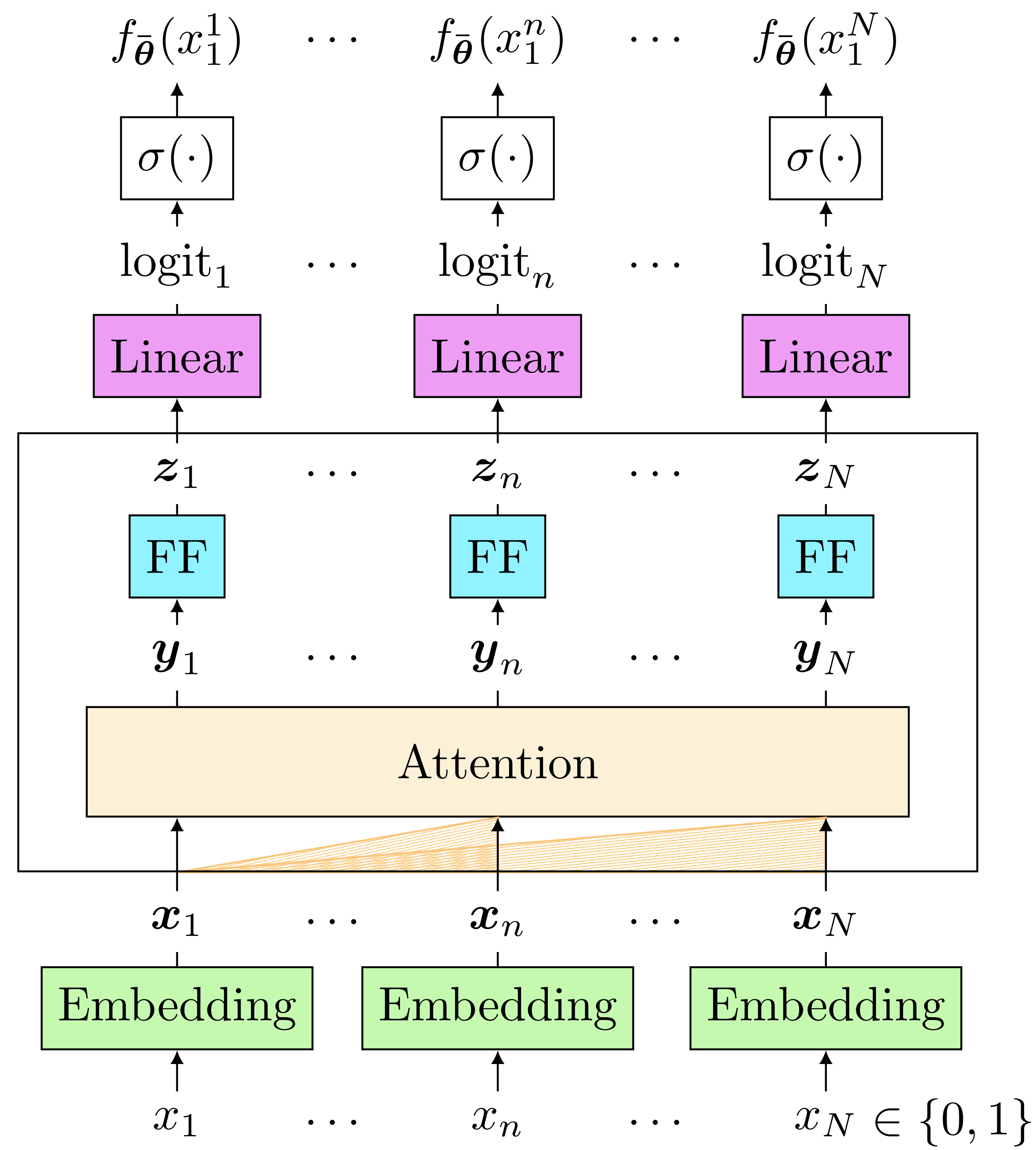
$$\boldsymbol{x}_n = x_n \cdot \boldsymbol{e} + \boldsymbol{p}_n$$

$\boldsymbol{\theta}$



Weight tying

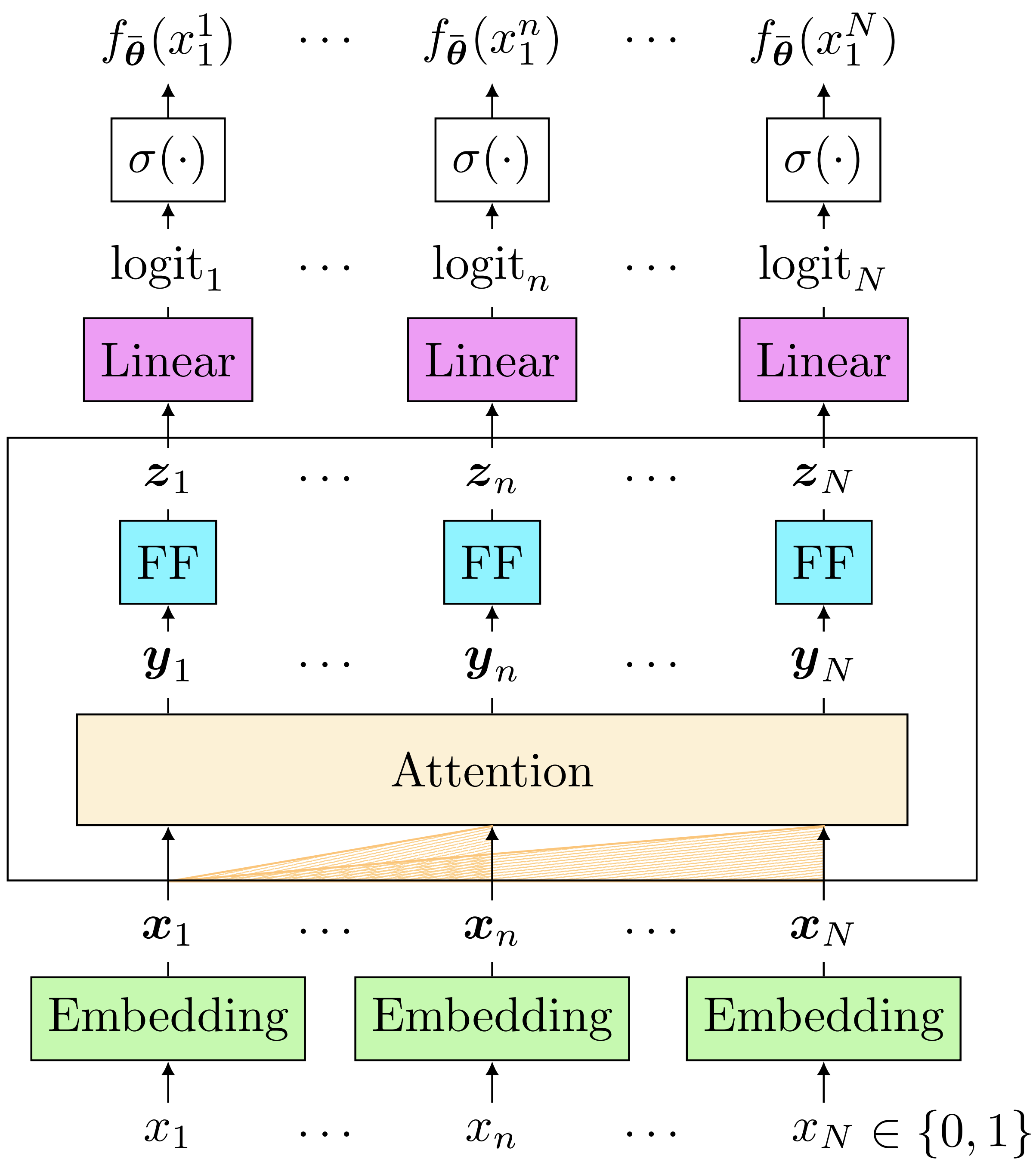




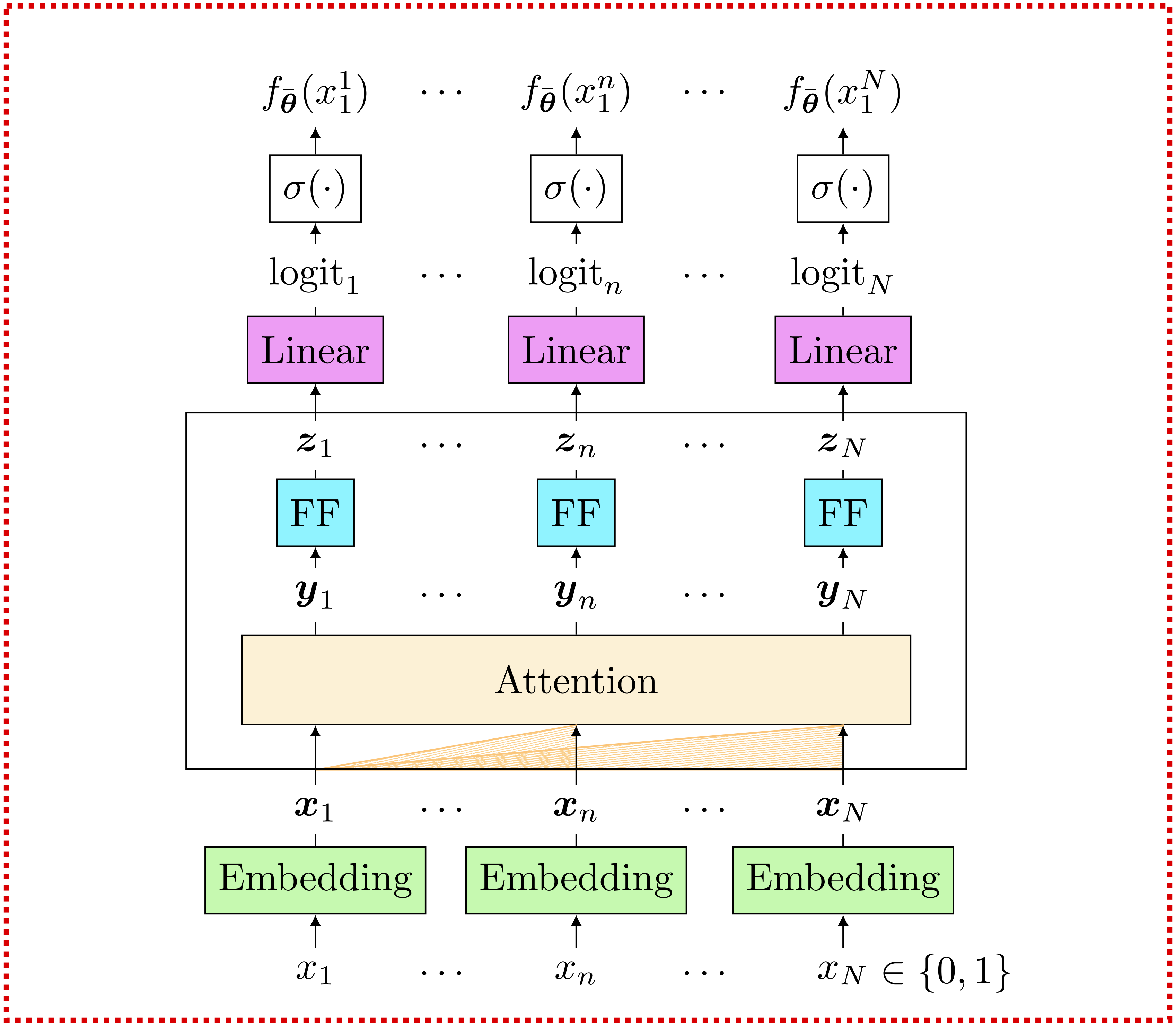
No weight tying

Linear

Embedding

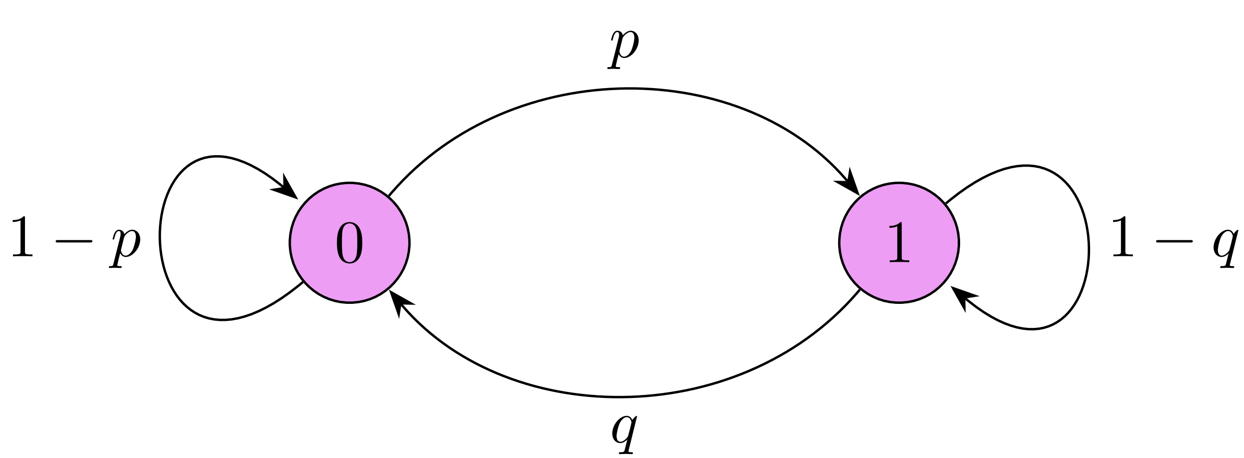


Single-layer transformer



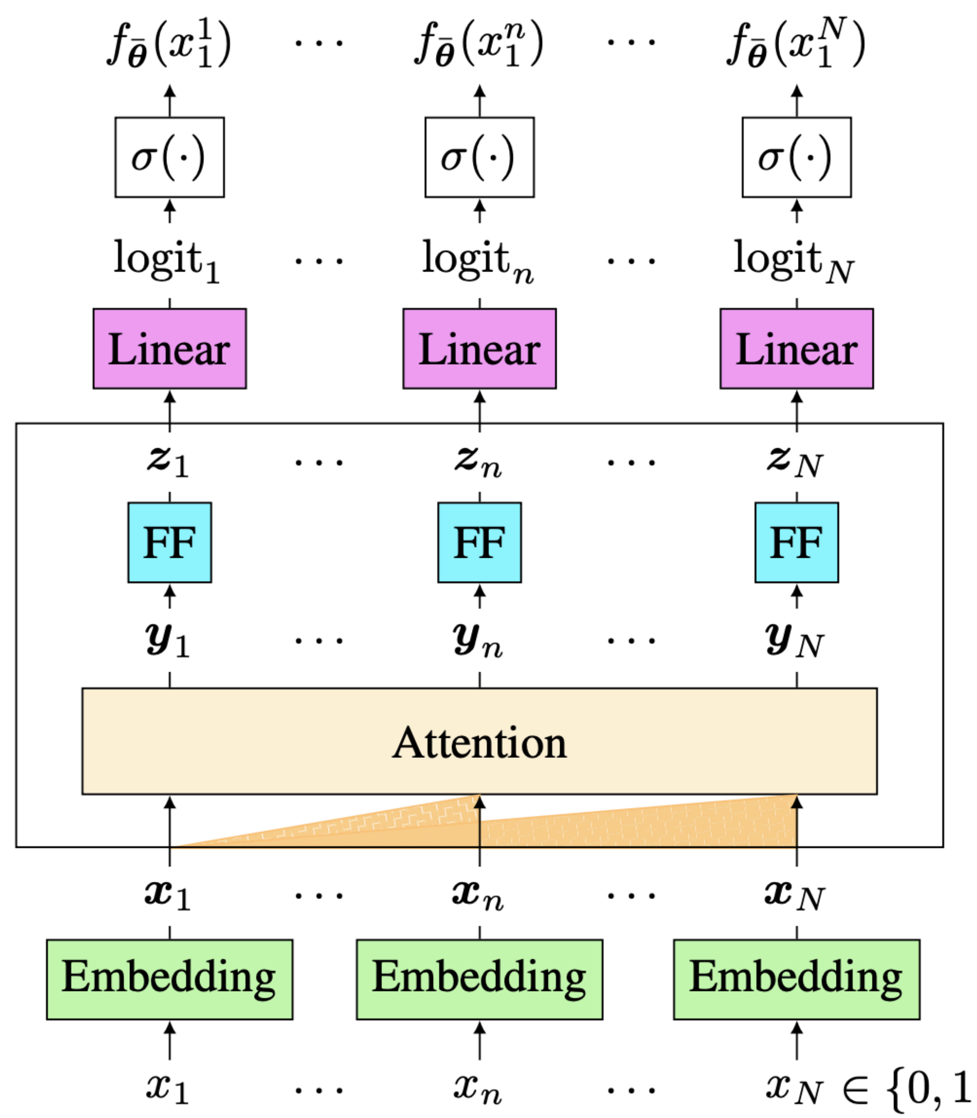
$\boldsymbol{\theta}$

First-order Markov chain + Single-layer transformer



$$\downarrow$$

$$(x_n)_{n\geq 1}$$



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

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

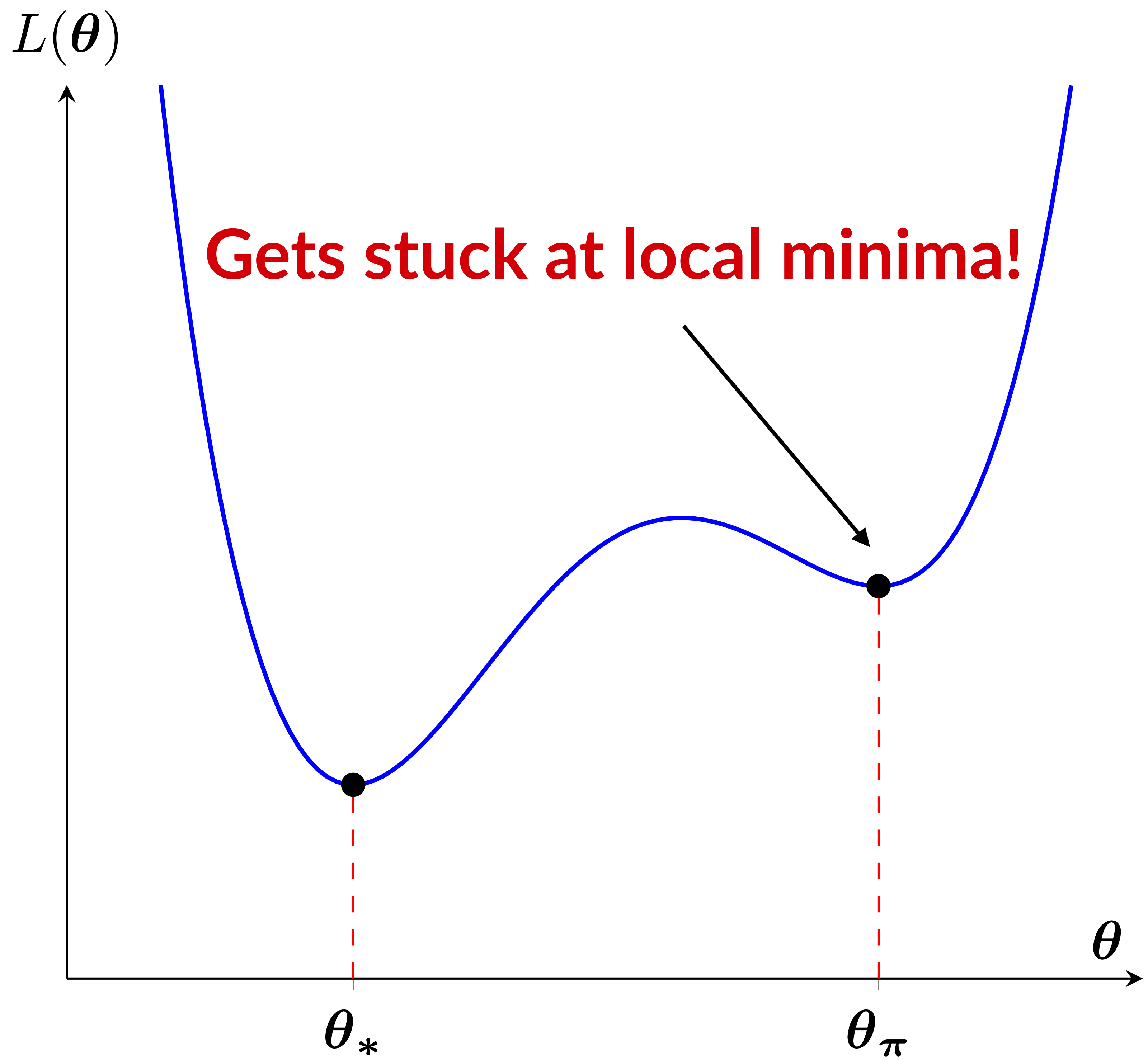
$$f_{\boldsymbol{\theta}}(x_1^n) \approx \mathbb{P}(x_{n+1}=1 \mid x_n)$$

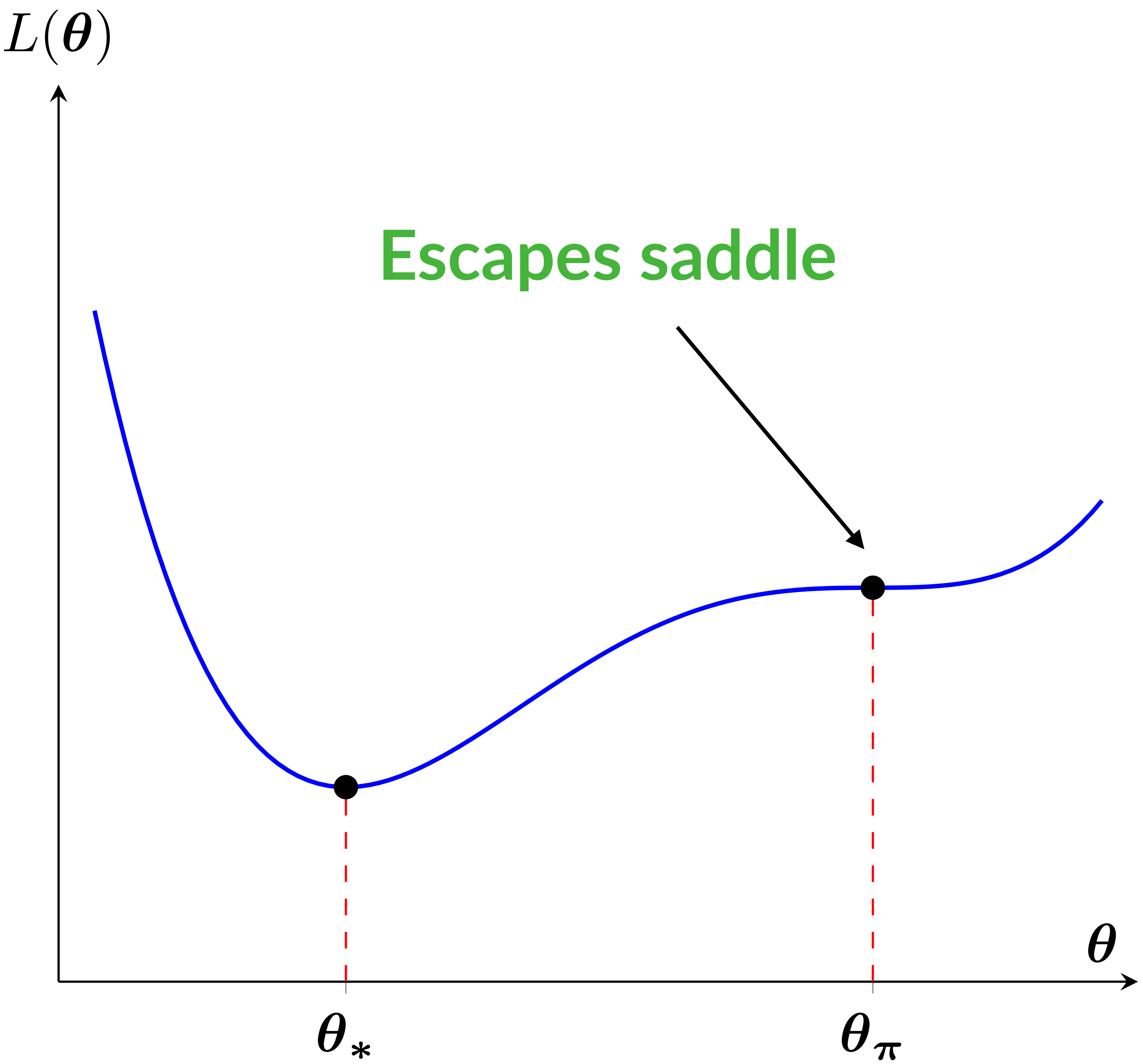
x_{n+1} $f_{\boldsymbol{\theta}}(x_1^n)$

But…

But…

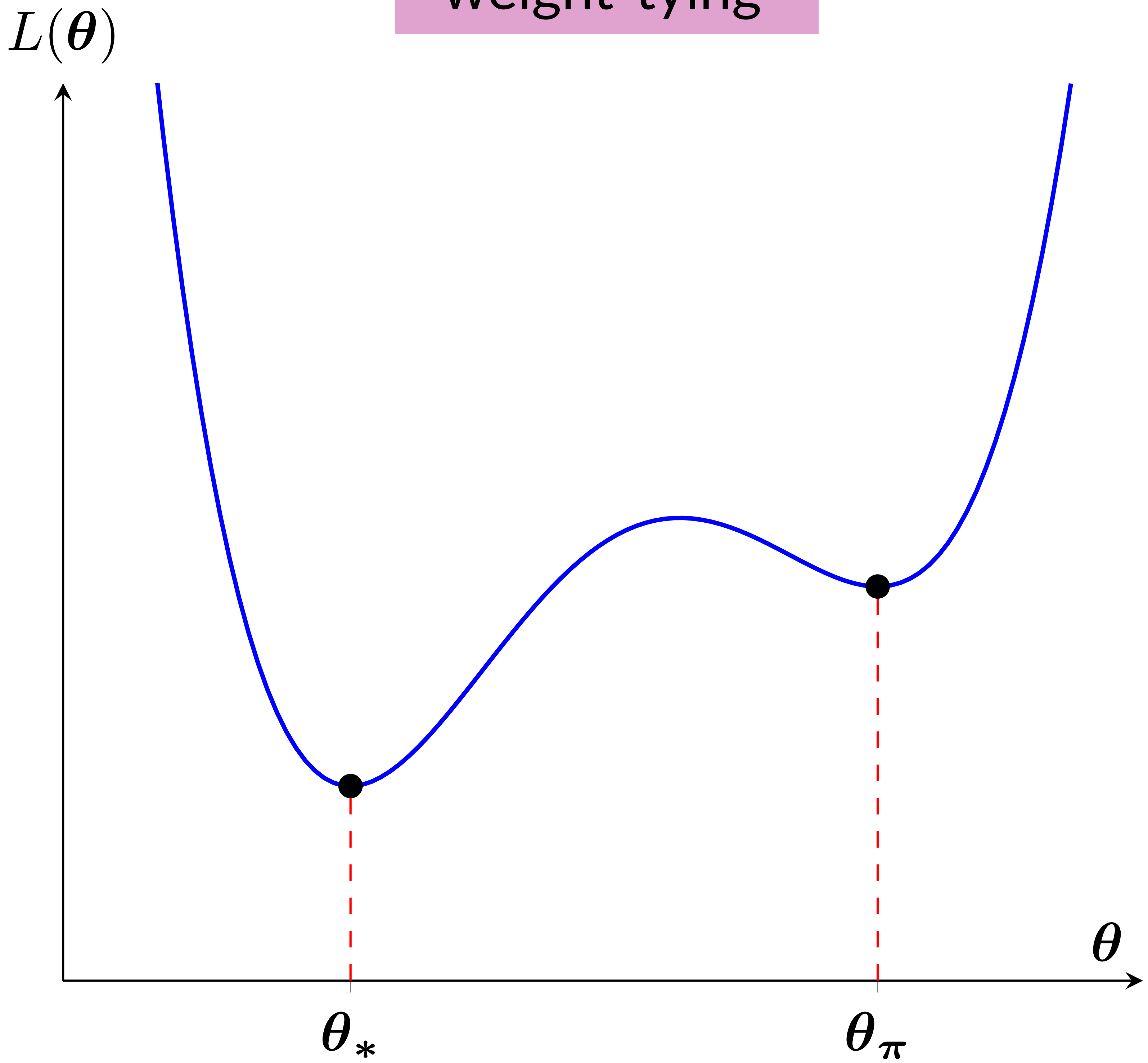
Single-layer transformers sometimes fail to learn even first-order Markov chains! *

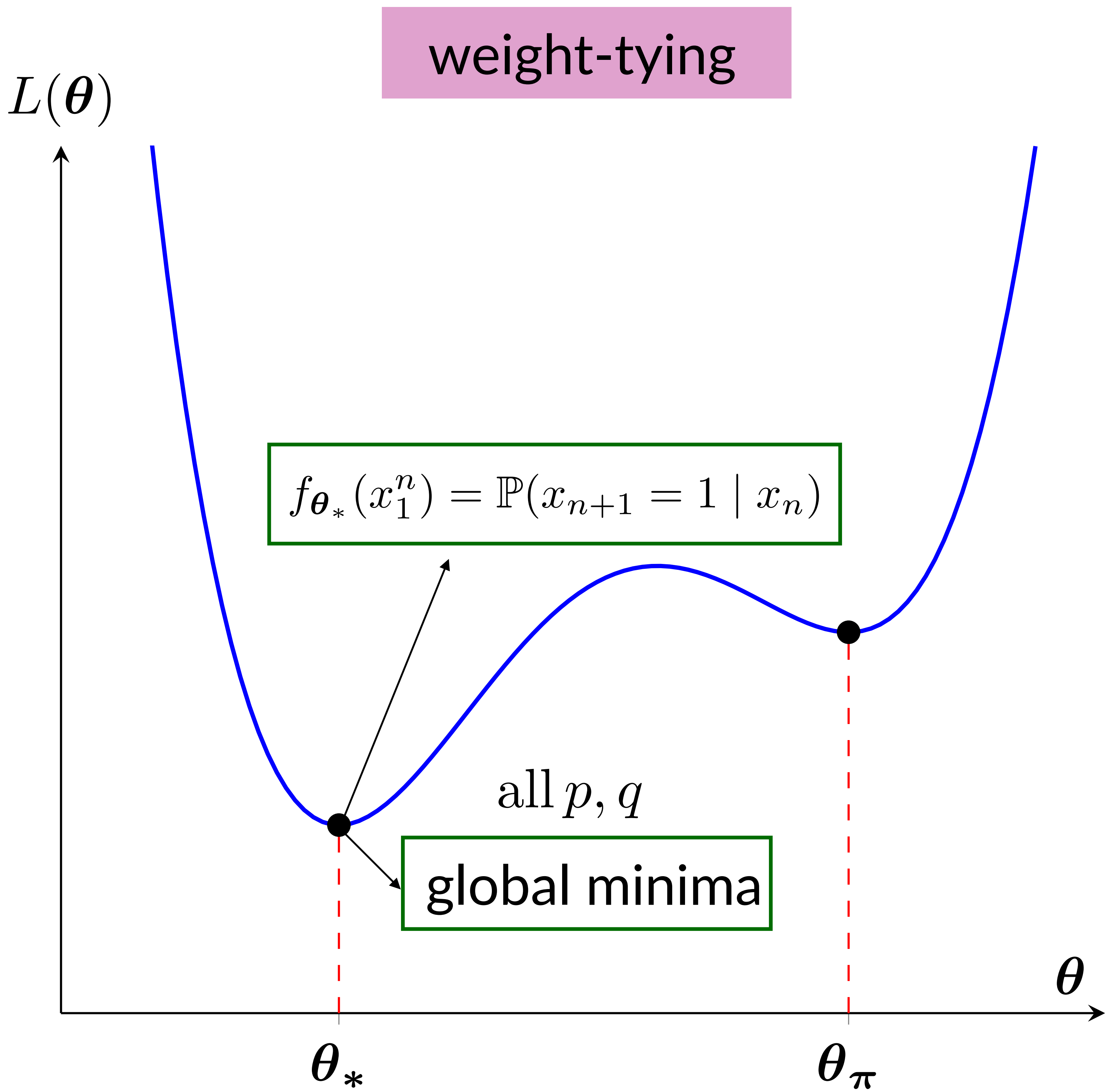

$L(\boldsymbol{\theta})$
Gets stuck at local minima!
$\boldsymbol{\theta}$
$\boldsymbol{\theta}_*$
$\boldsymbol{\theta}_{\boldsymbol{\pi}}$

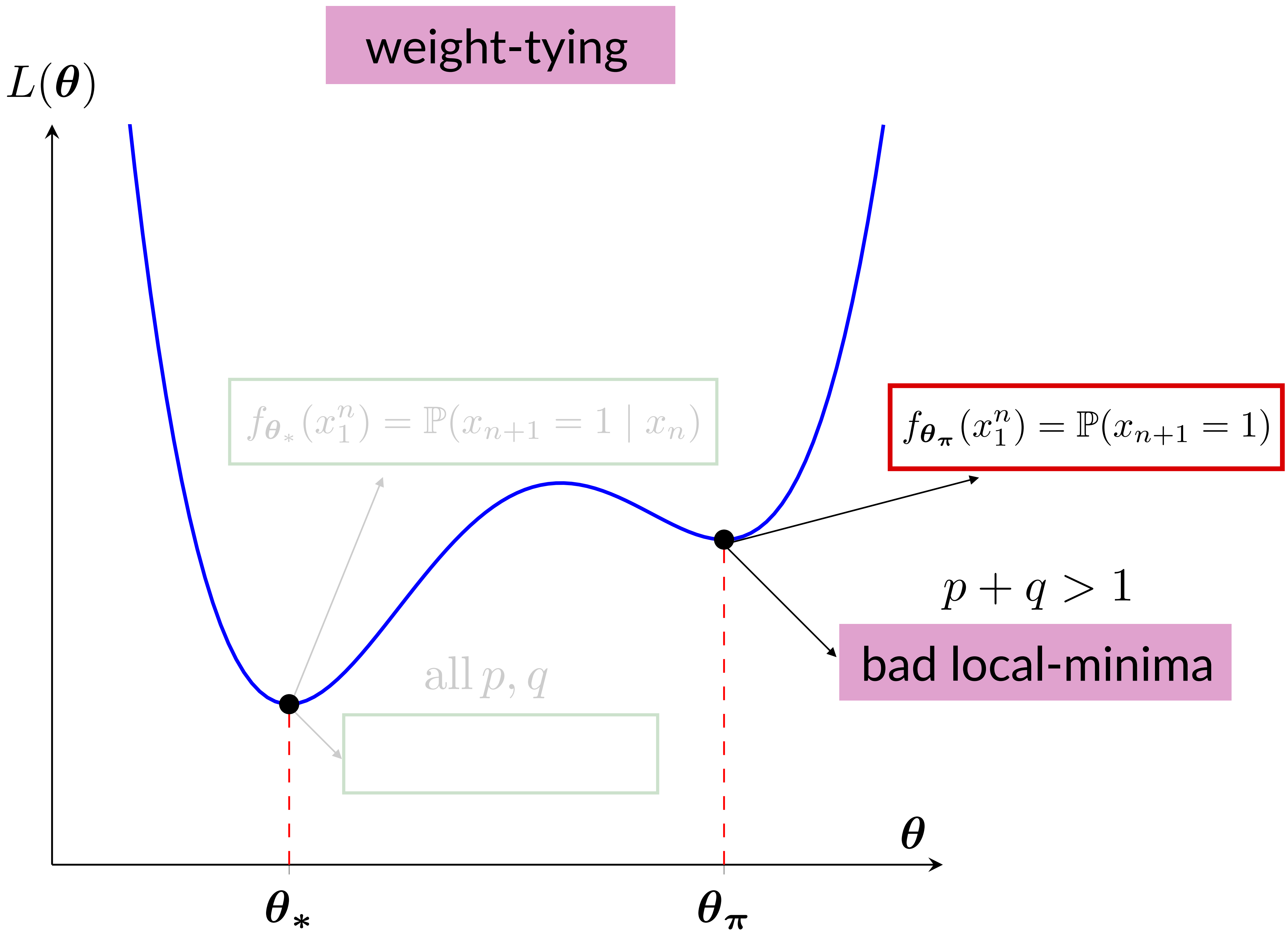

$L(\boldsymbol{\theta})$
Escapes saddle
$\boldsymbol{\theta}$
$\boldsymbol{\theta_*}$
$\boldsymbol{\theta_\pi}$

weight-tying

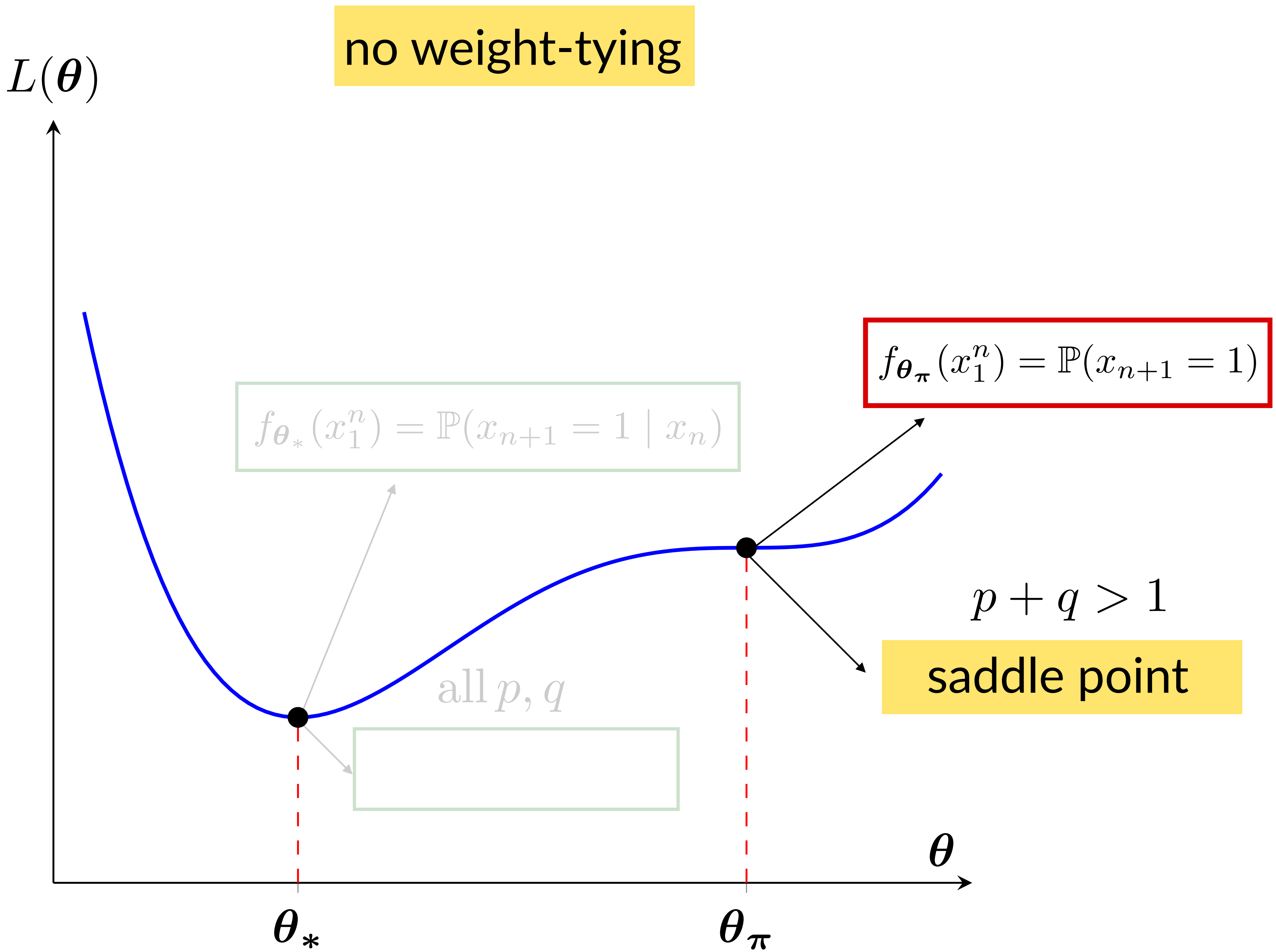
weight-tying




weight-tying
$L(\boldsymbol{\theta})$
$f_{\boldsymbol{\theta}_*}(x_1^n) = \mathbb{P}(x_{n+1} = 1 \mid x_n)$
$\text{all}\, p, q$
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...



Theoretical results

Theoretical results

Bad local minima (weight tying)

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

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

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

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

(iv) $\nabla L(\boldsymbol{\theta}_{\boldsymbol{\pi}}) = 0$, i.e. $\boldsymbol{\theta}_{\boldsymbol{\pi}}$ 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 $\boldsymbol{\theta}_{\boldsymbol{\pi}}$ with explicit construction such that:

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

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

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

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

Saddle point (no weight tying)

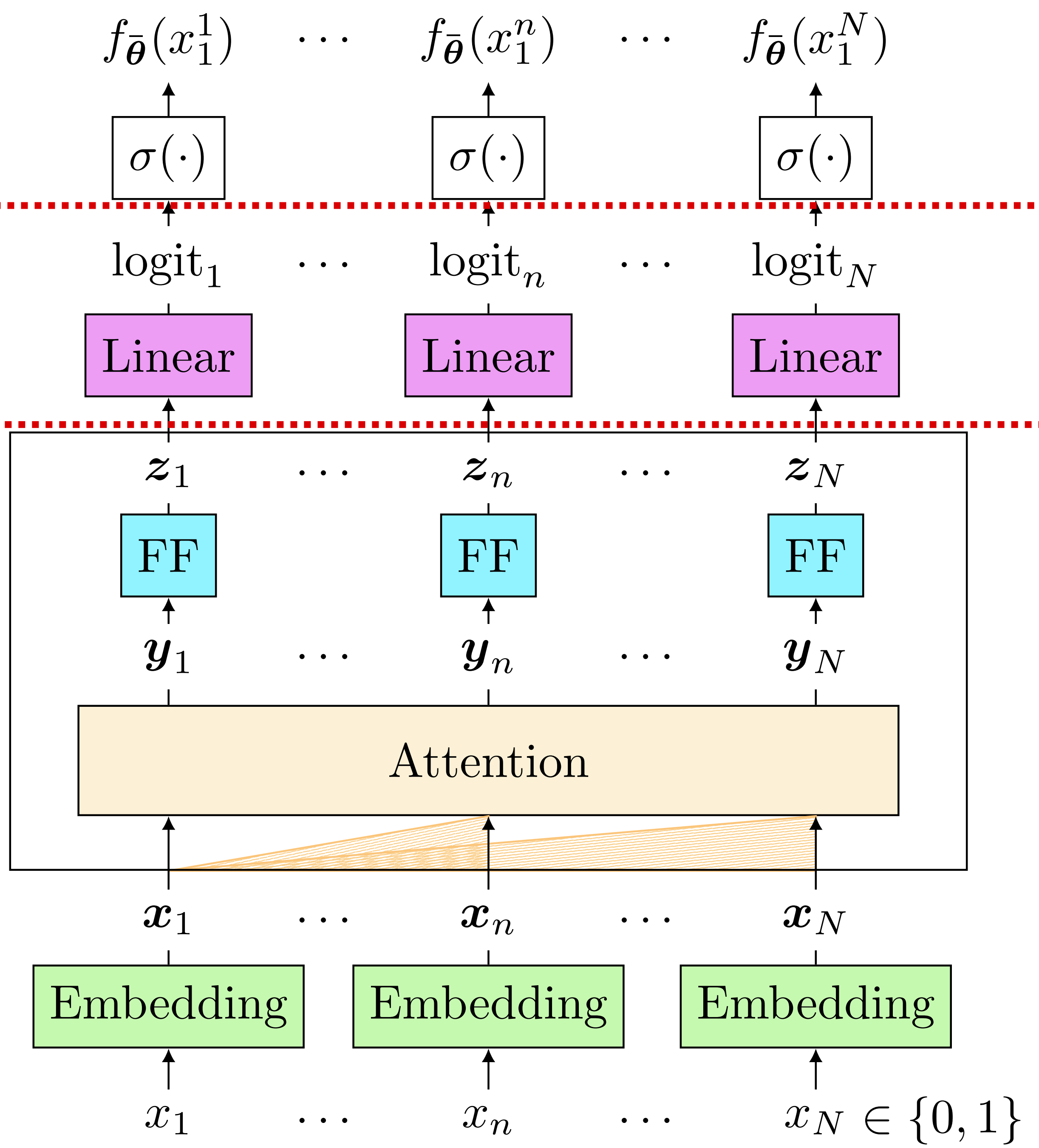
Under the same setting as above with the weights not tied, $\boldsymbol{\theta}_{\boldsymbol{\pi}}$ becomes a saddle point. It satisfies the same properties.

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

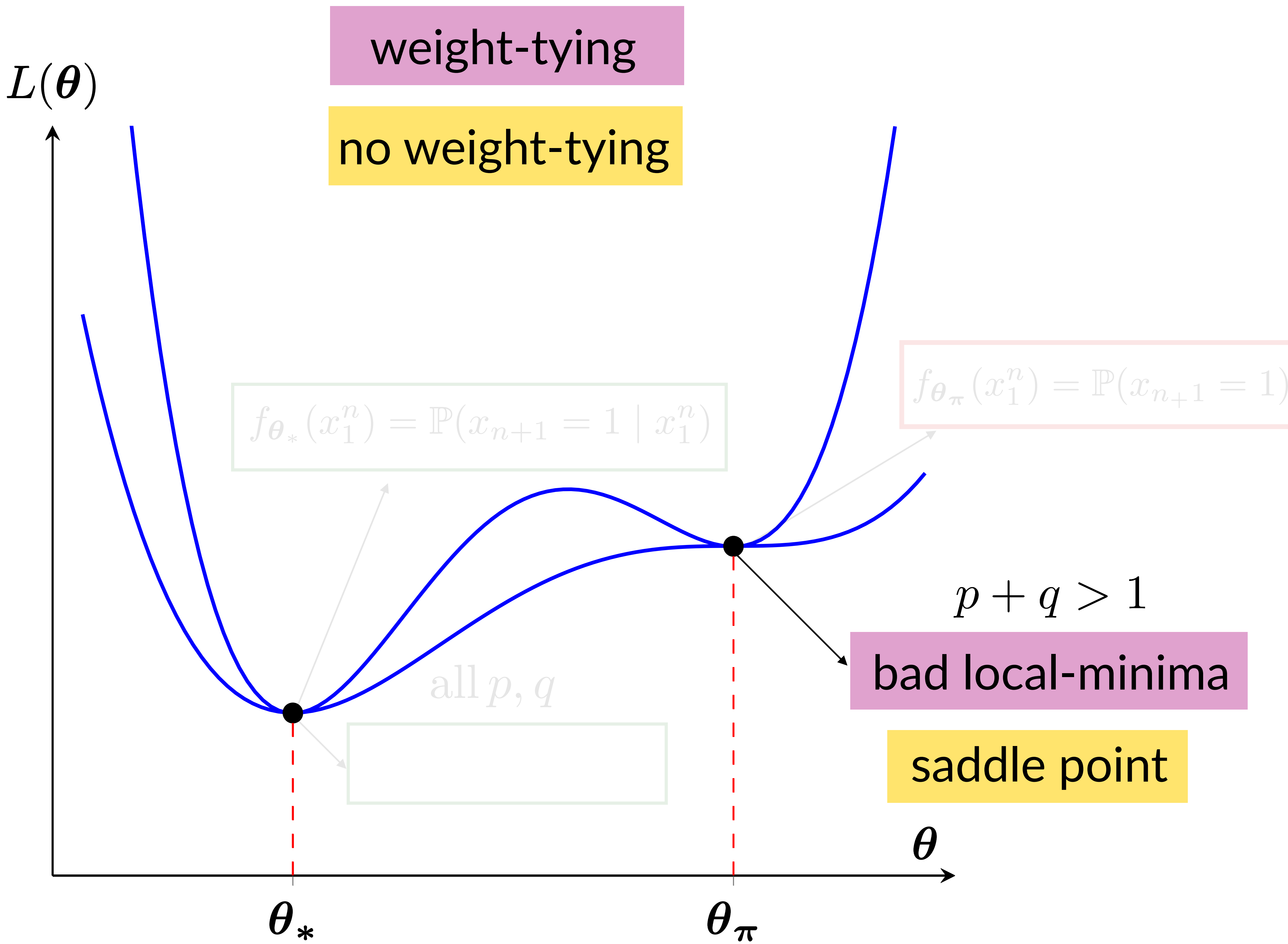
weight-tying

Hessian is (almost) positive-definite

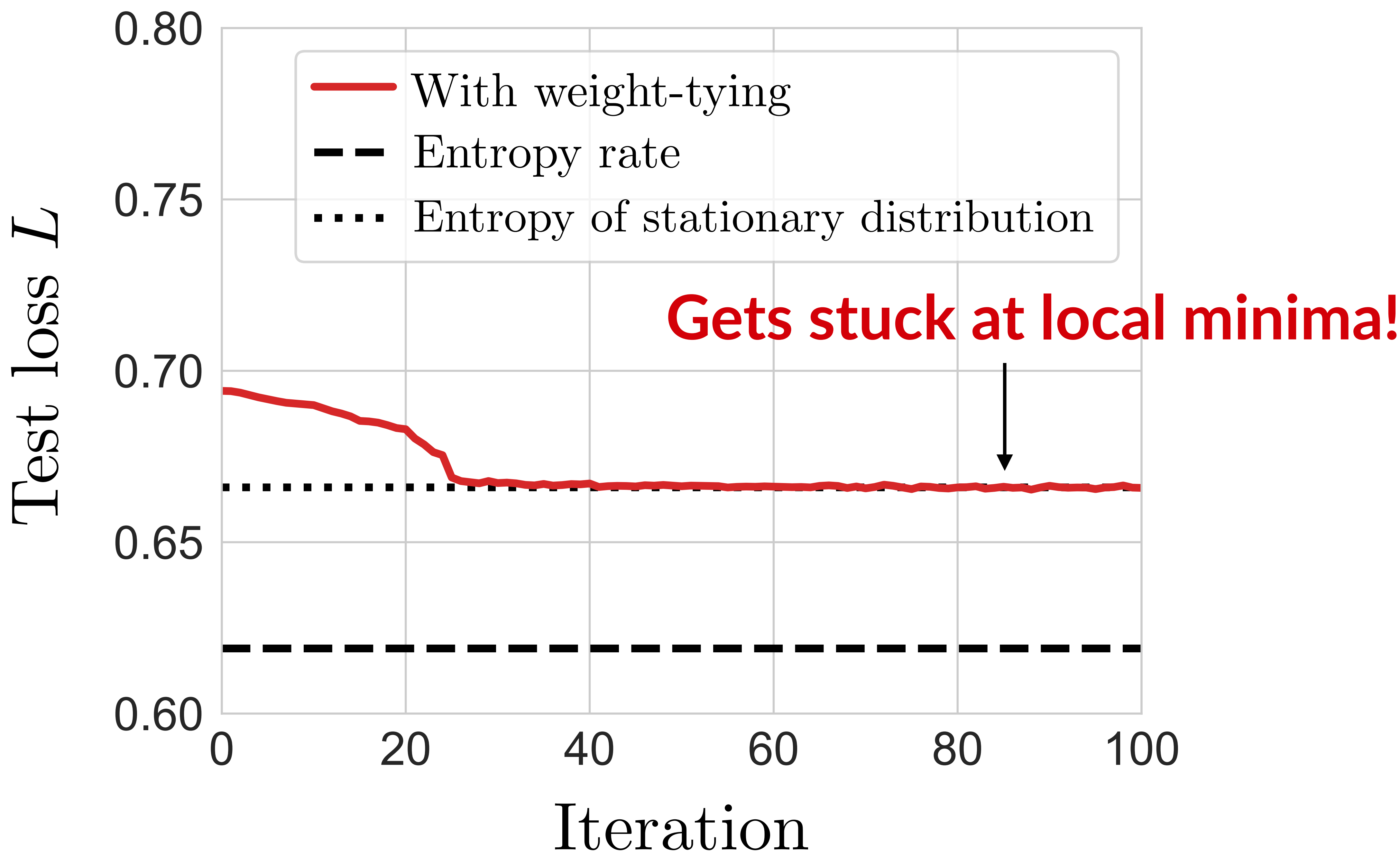
no weight-tying

Hessian is indefinite

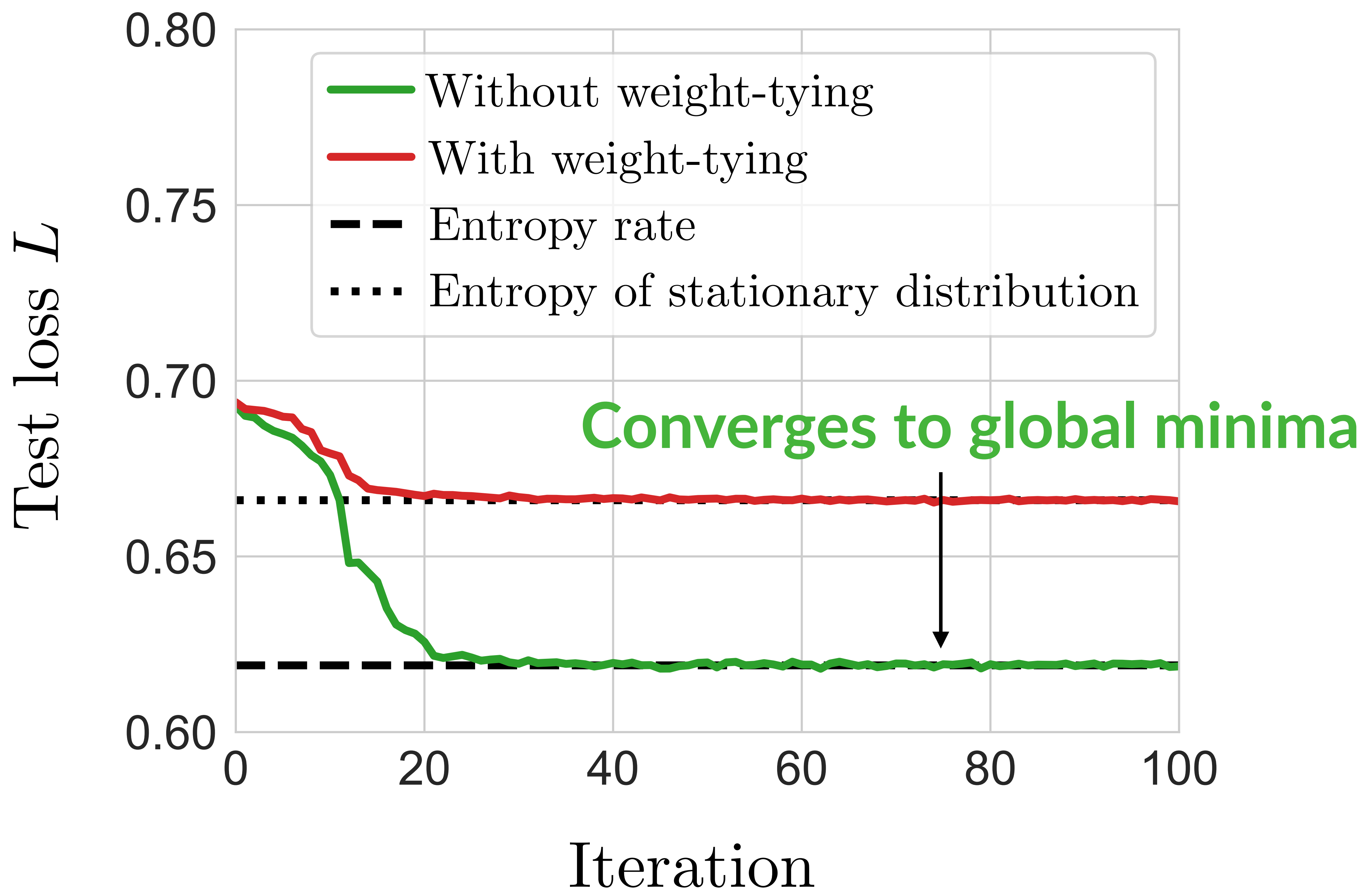
Main results



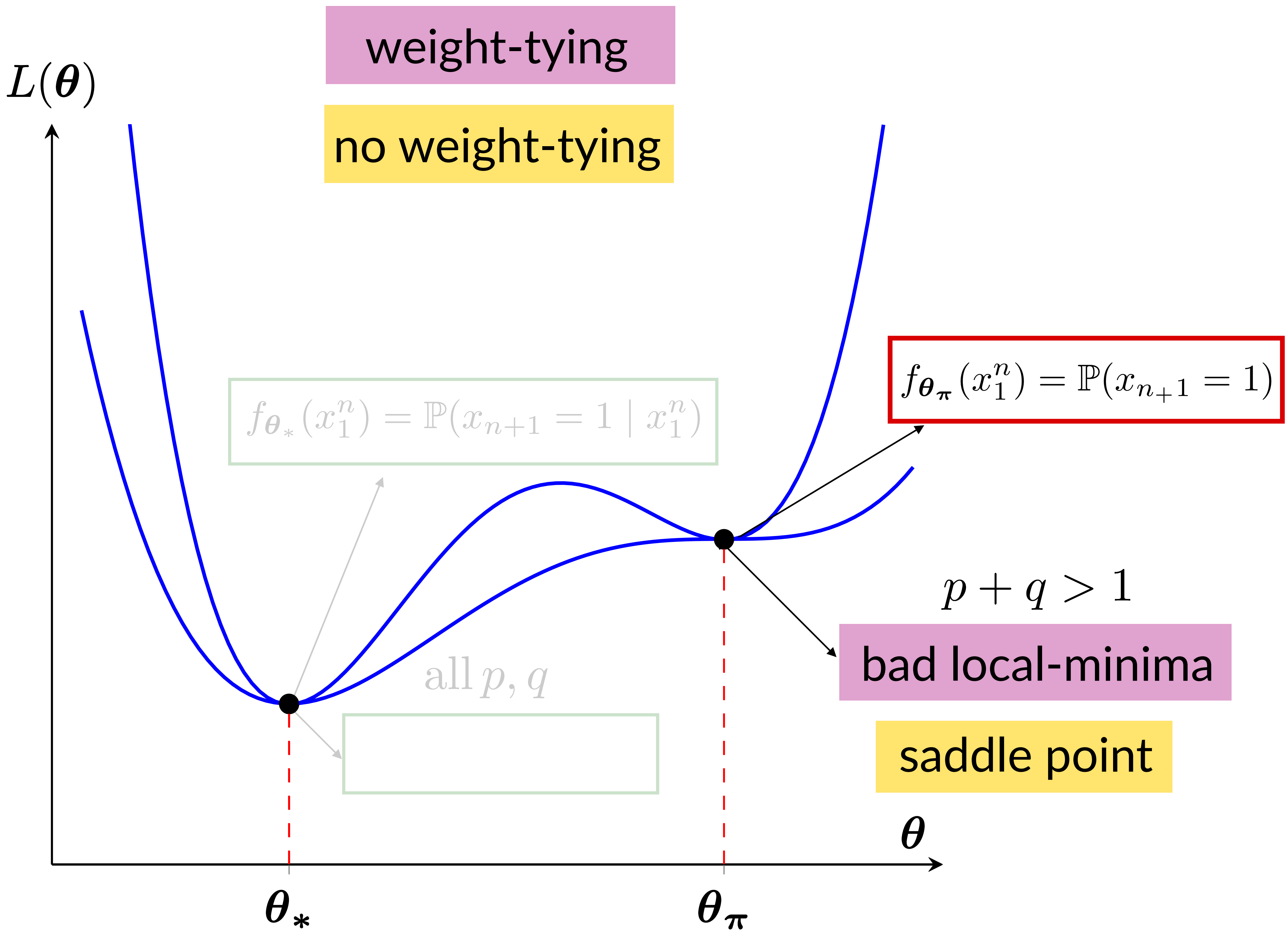
Weight tying

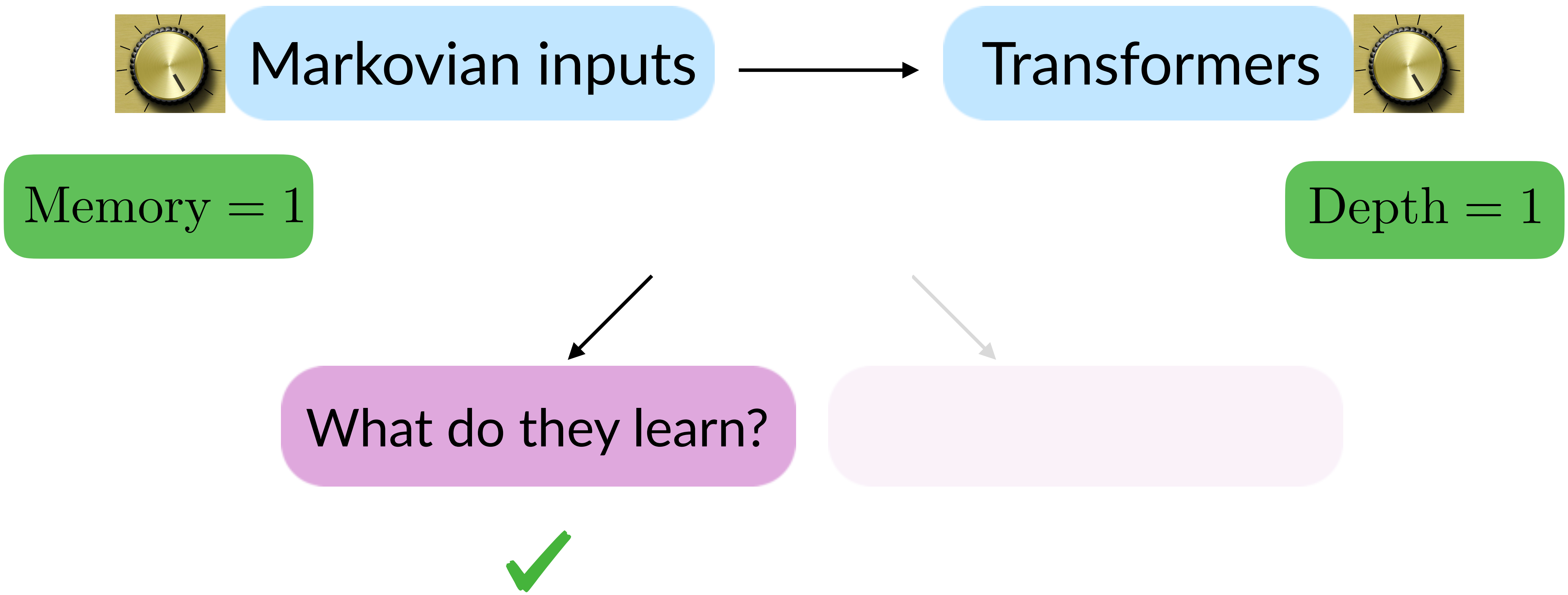


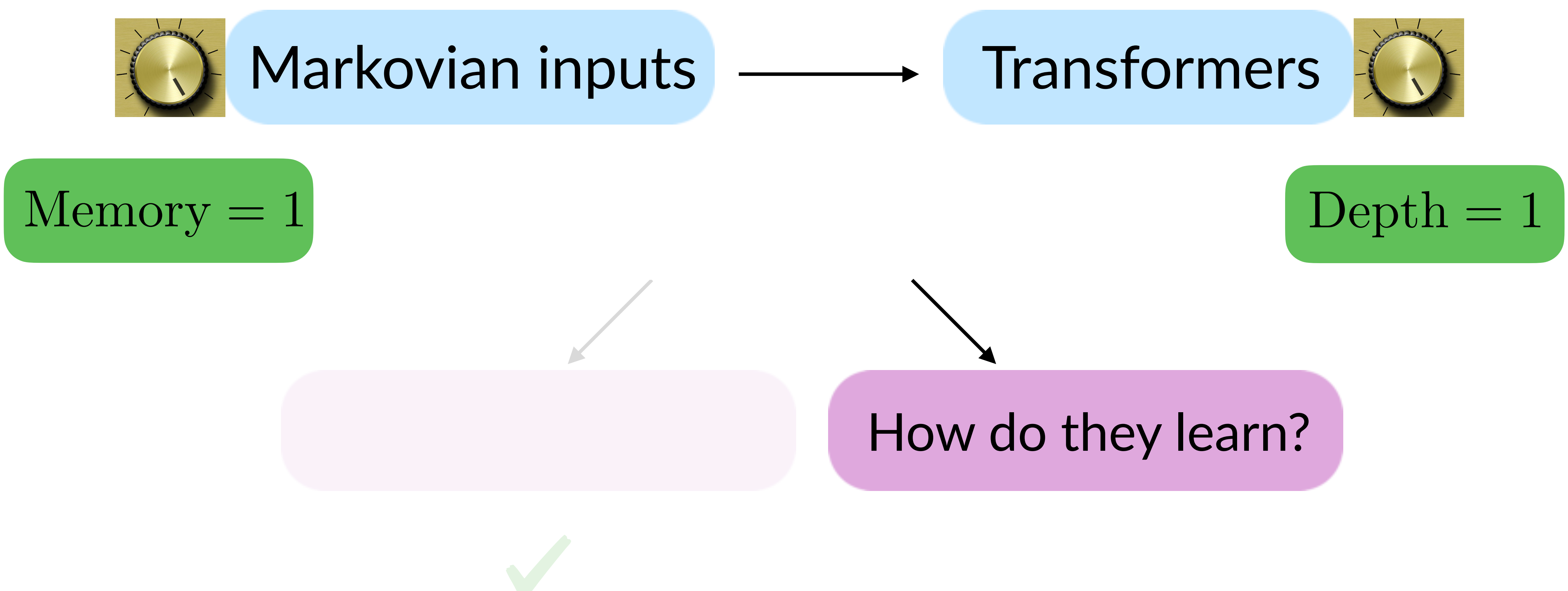
Without weight tying

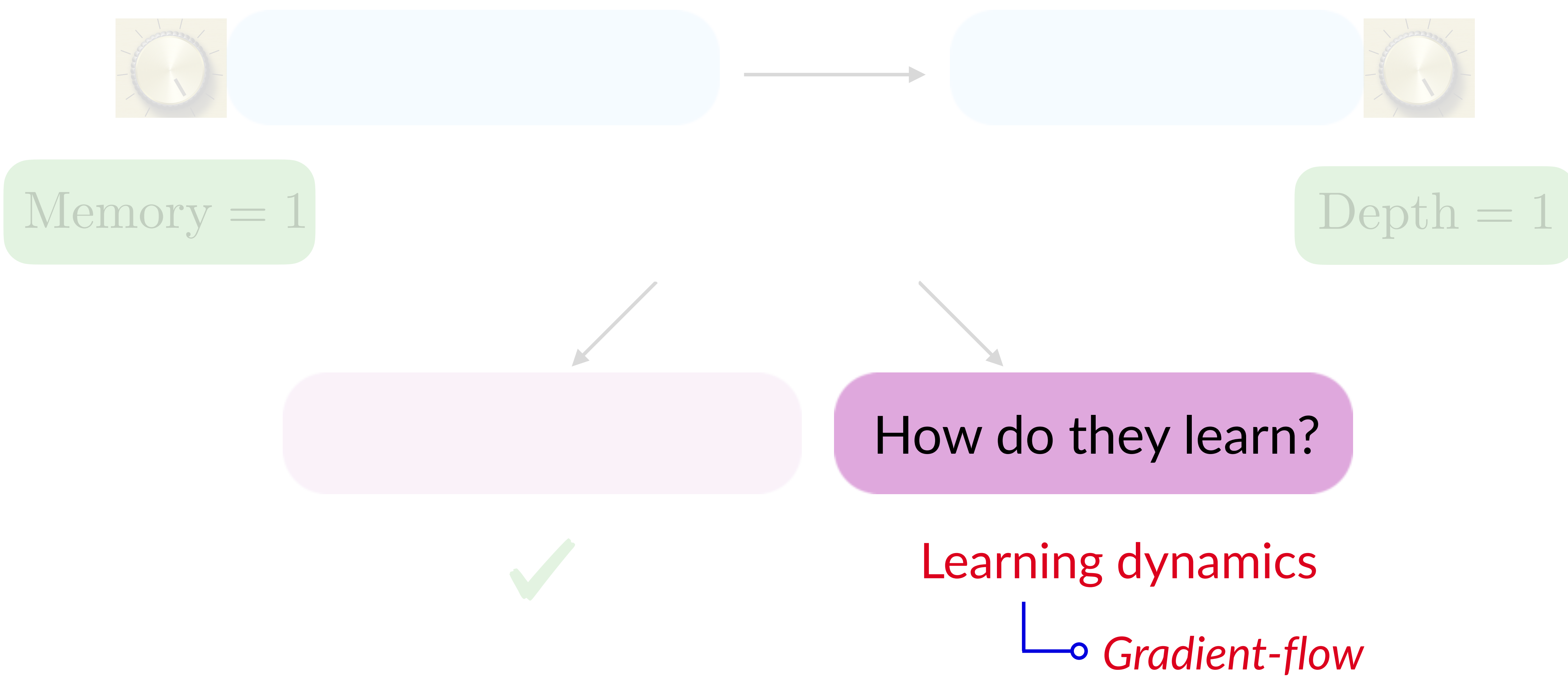


Main results




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

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

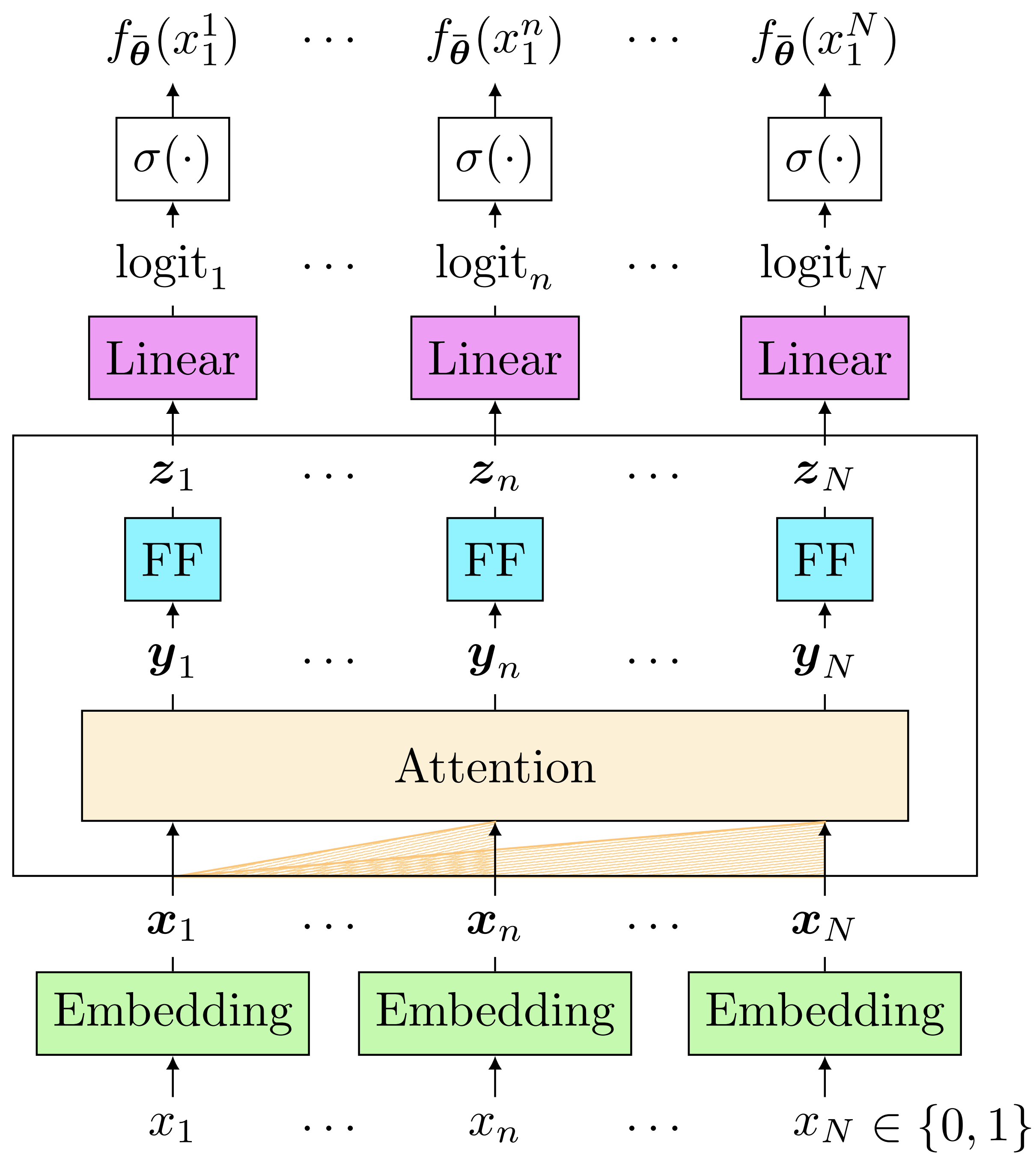
$$\mathrm{logit}_n = \langle \boldsymbol{a}, \boldsymbol{z}_n \rangle + b \in \mathbb{R}$$

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

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

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

$$\boldsymbol{\theta}$$



Low-rank structure

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

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

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

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

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

Reparametrization

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

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

w

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

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

a

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

Reparametrization

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

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

w

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

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

a

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

e

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

Gradient flow

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

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

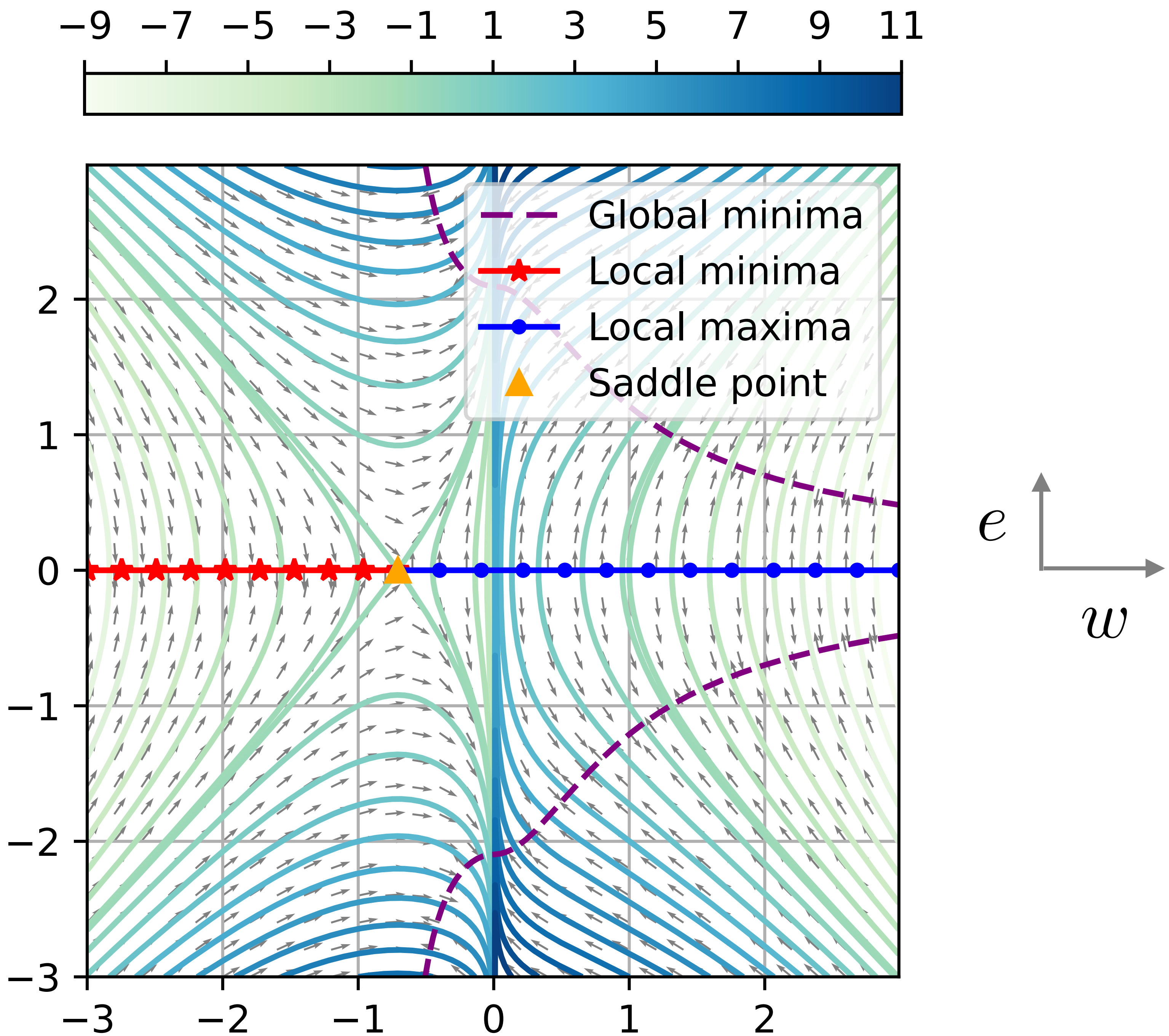
How does the flow look like?

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

How does the flow look like?

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

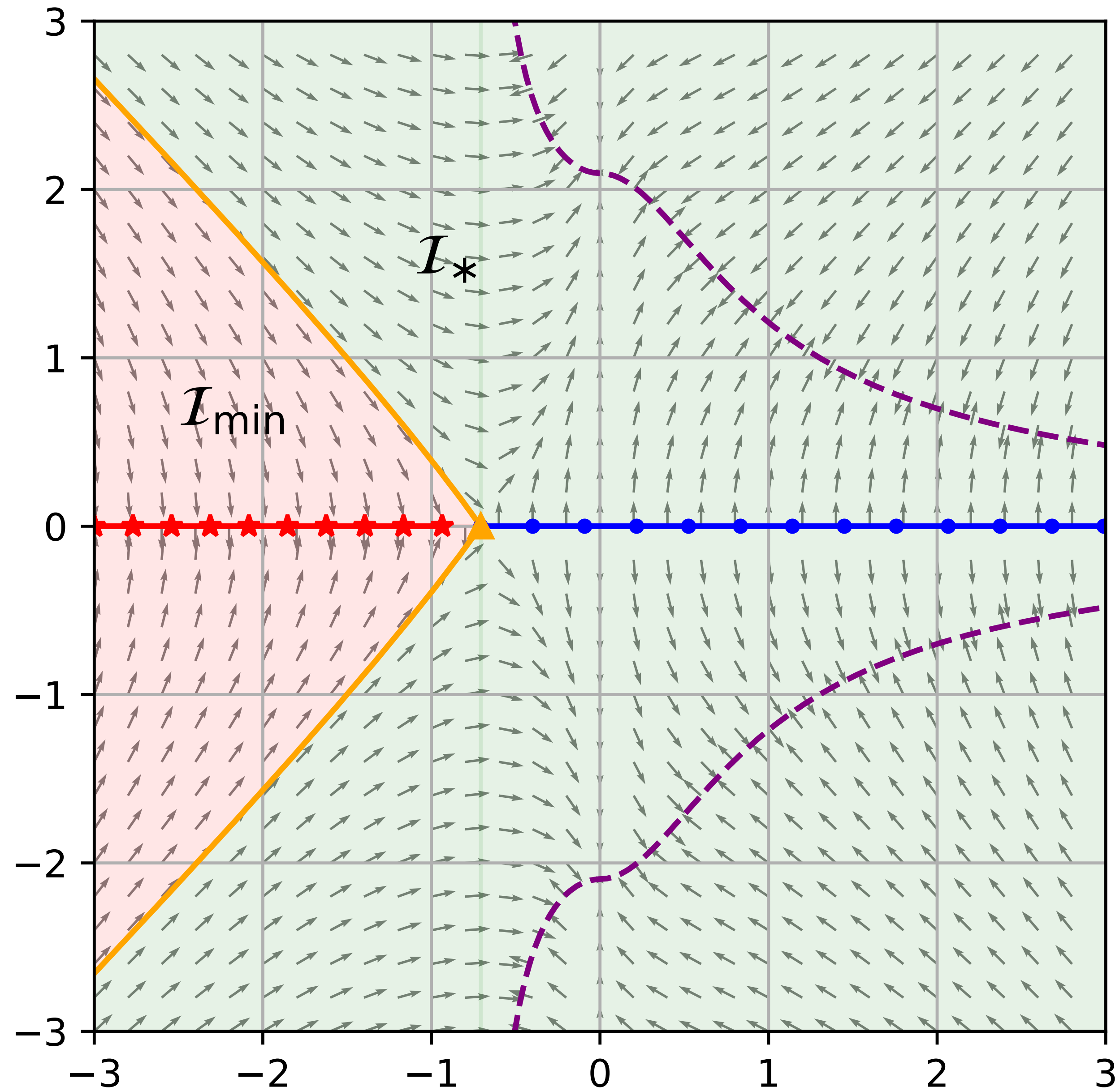
p+q < 1



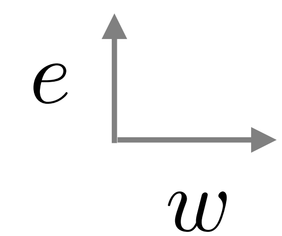
−9 −7 −5 −3 −1 1 3 5 7 9 11
Global minima
Local minima
Local maxima
Saddle point
2
1
0
−1
−2
−3
−3 −2 −1 0 1 2
e
w

p+q < 1

Global minima
Local minima
Local maxima
Saddle point



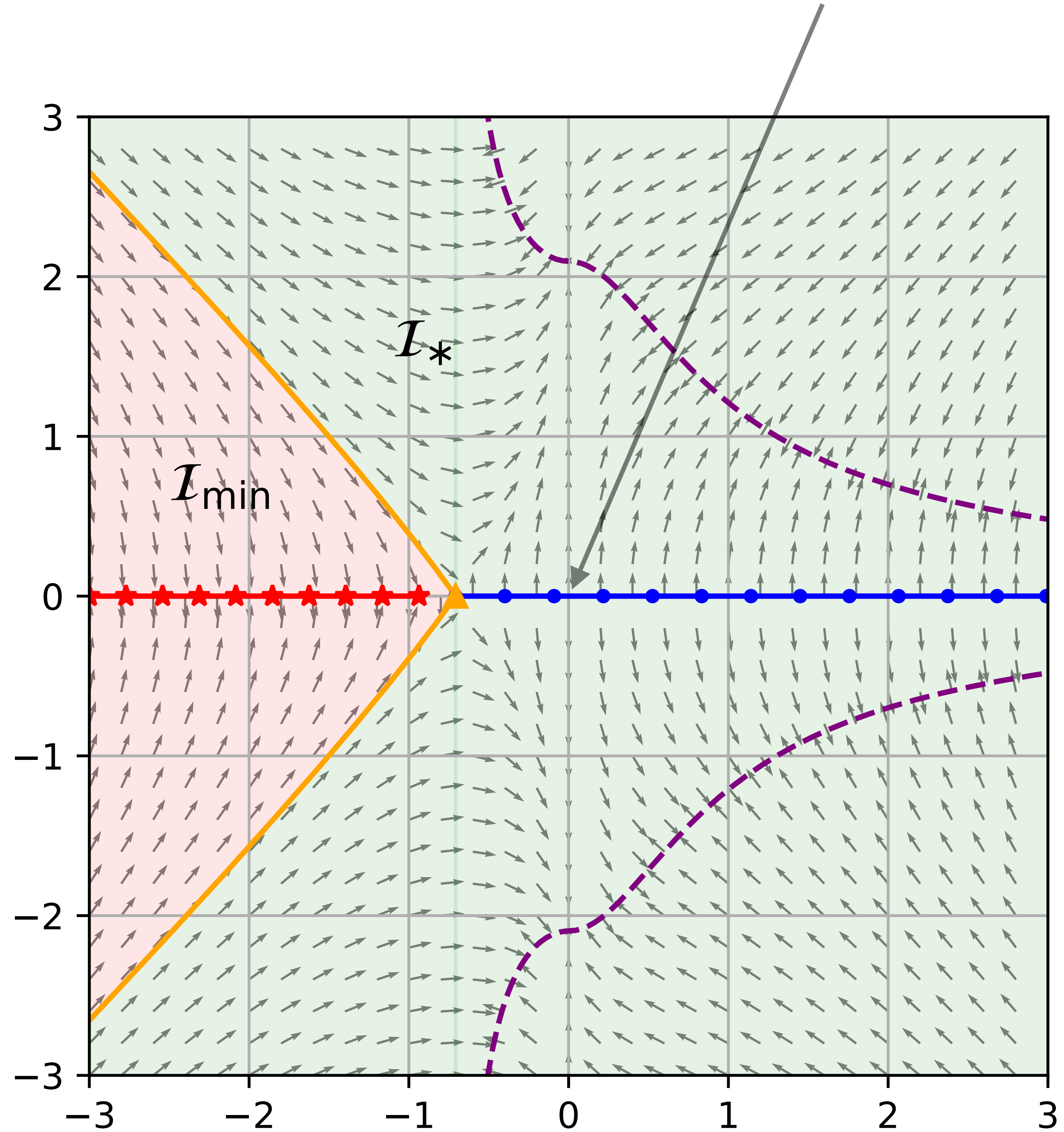
$\mathcal{I}_*$
$\mathcal{I}_{\text{min}}$



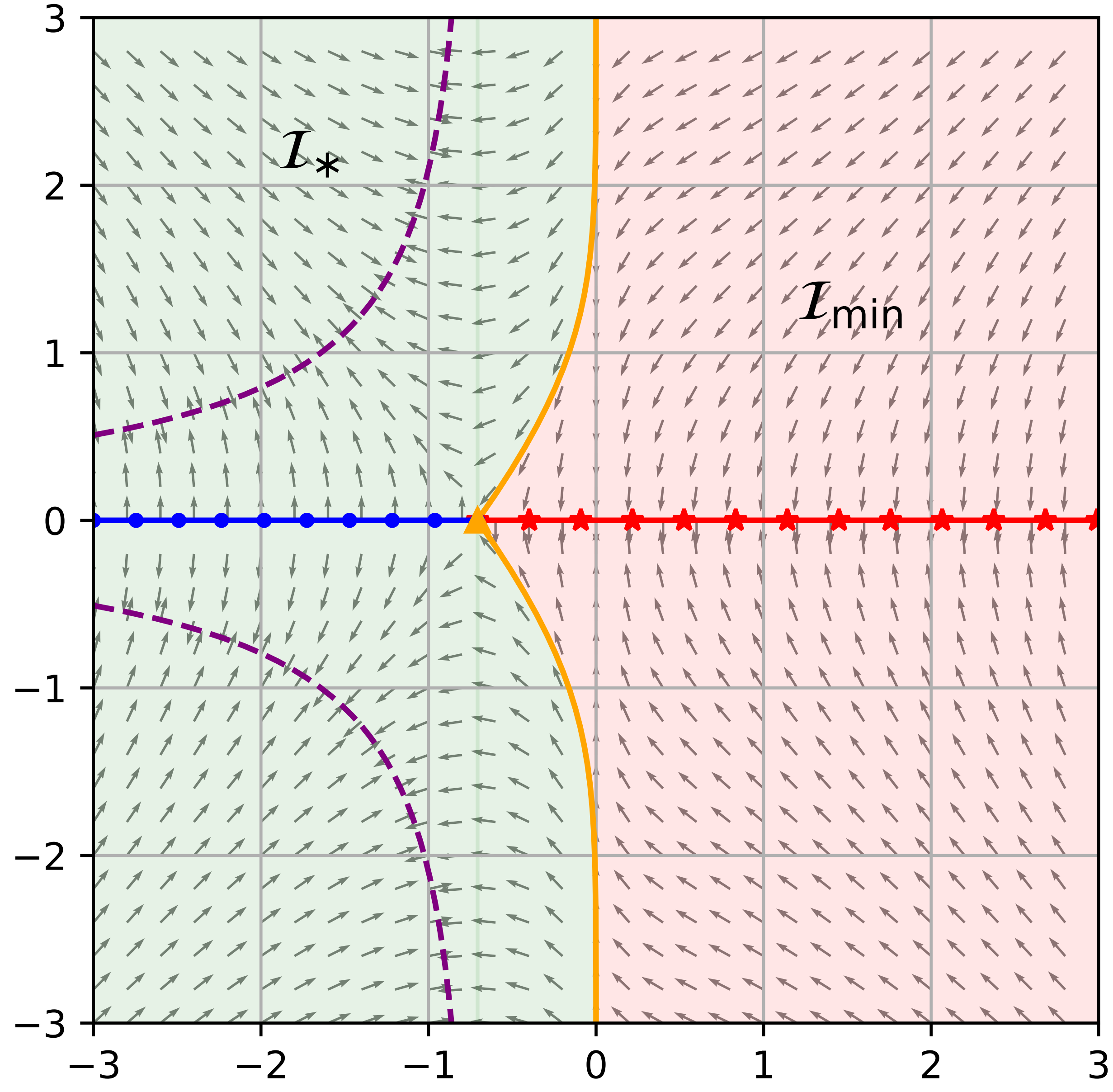
e
w

p+q < 1

Gaussian init. converges to global minima

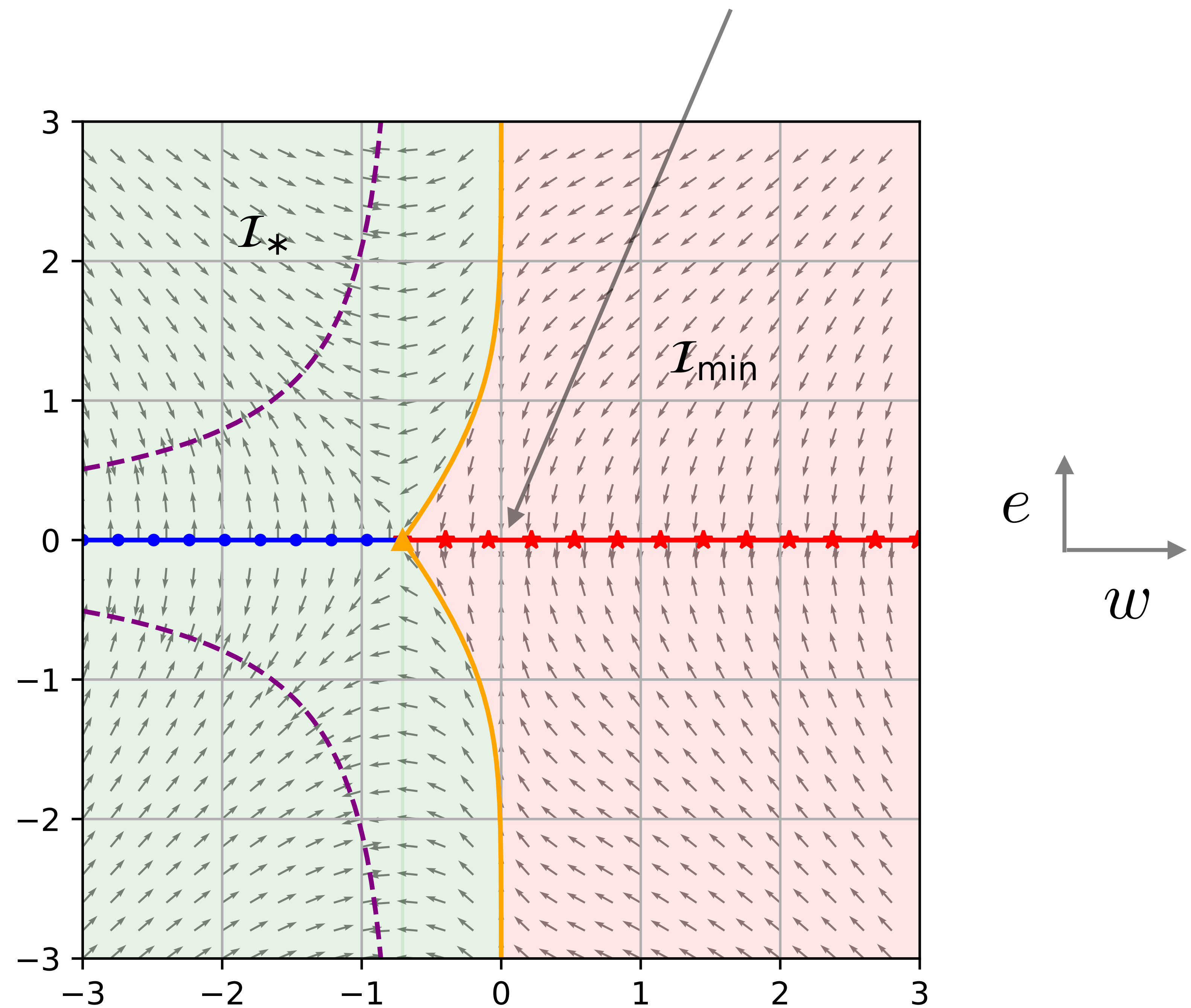


p+q > 1



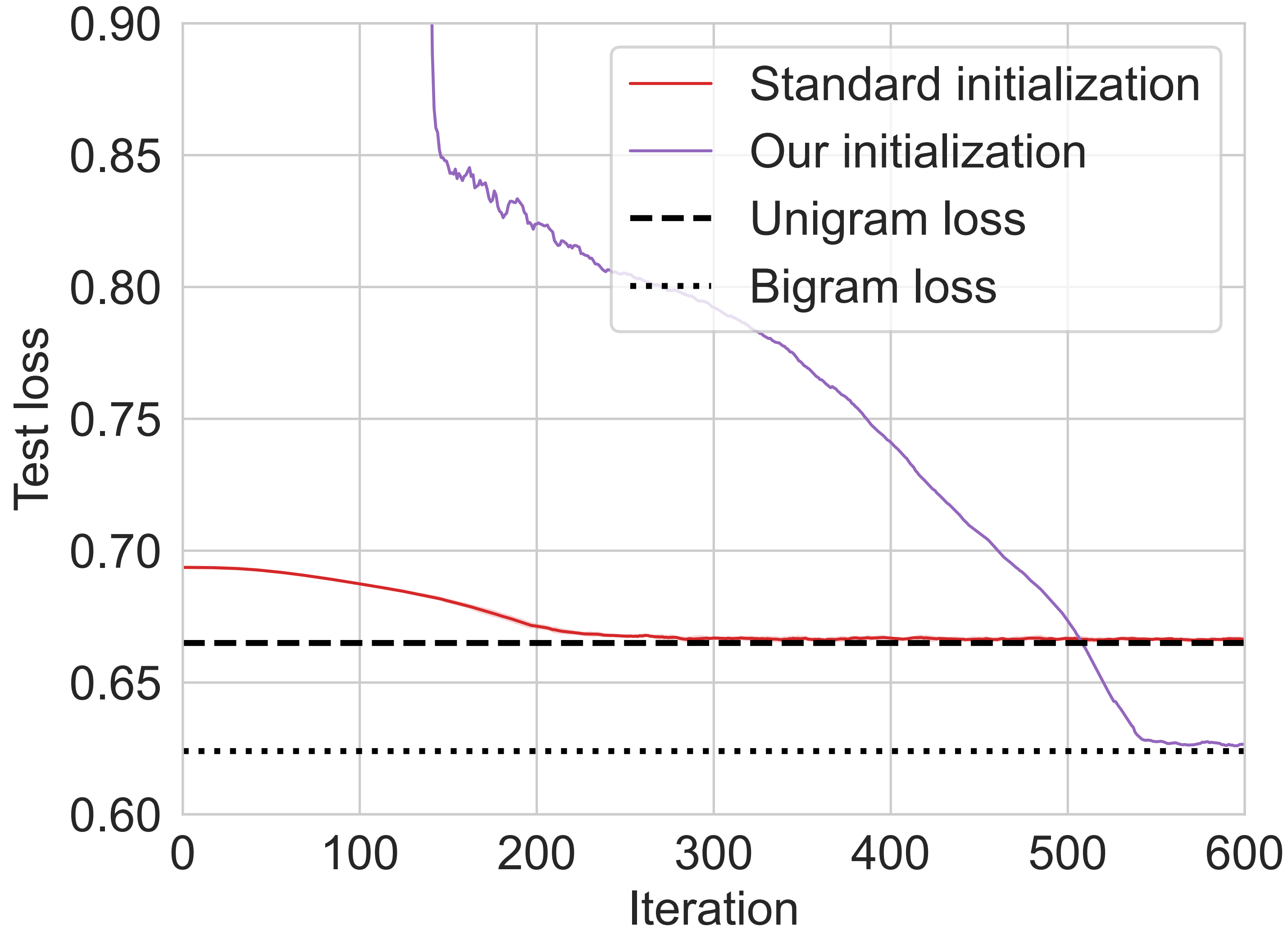
p+q > 1

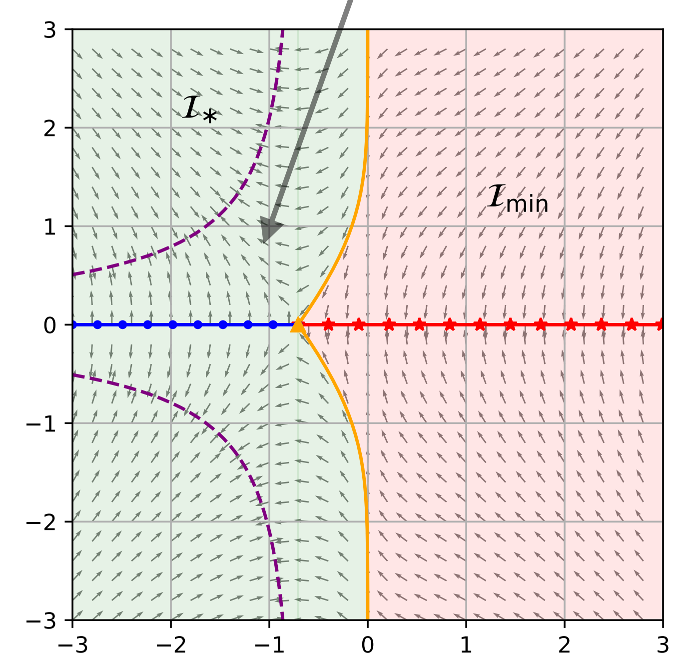
Gets stuck at local minima!



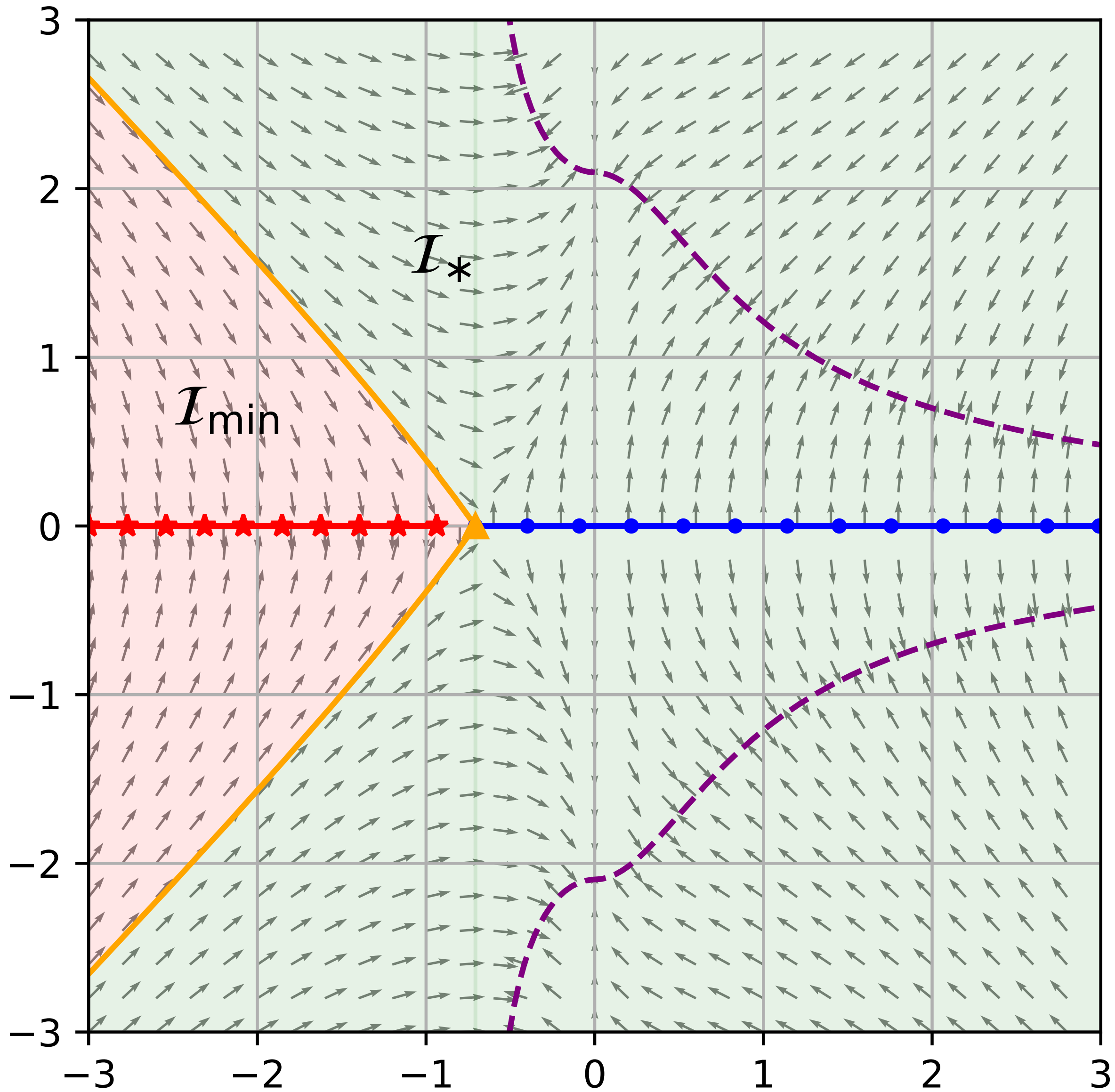
Can we escape it?

Initialise here

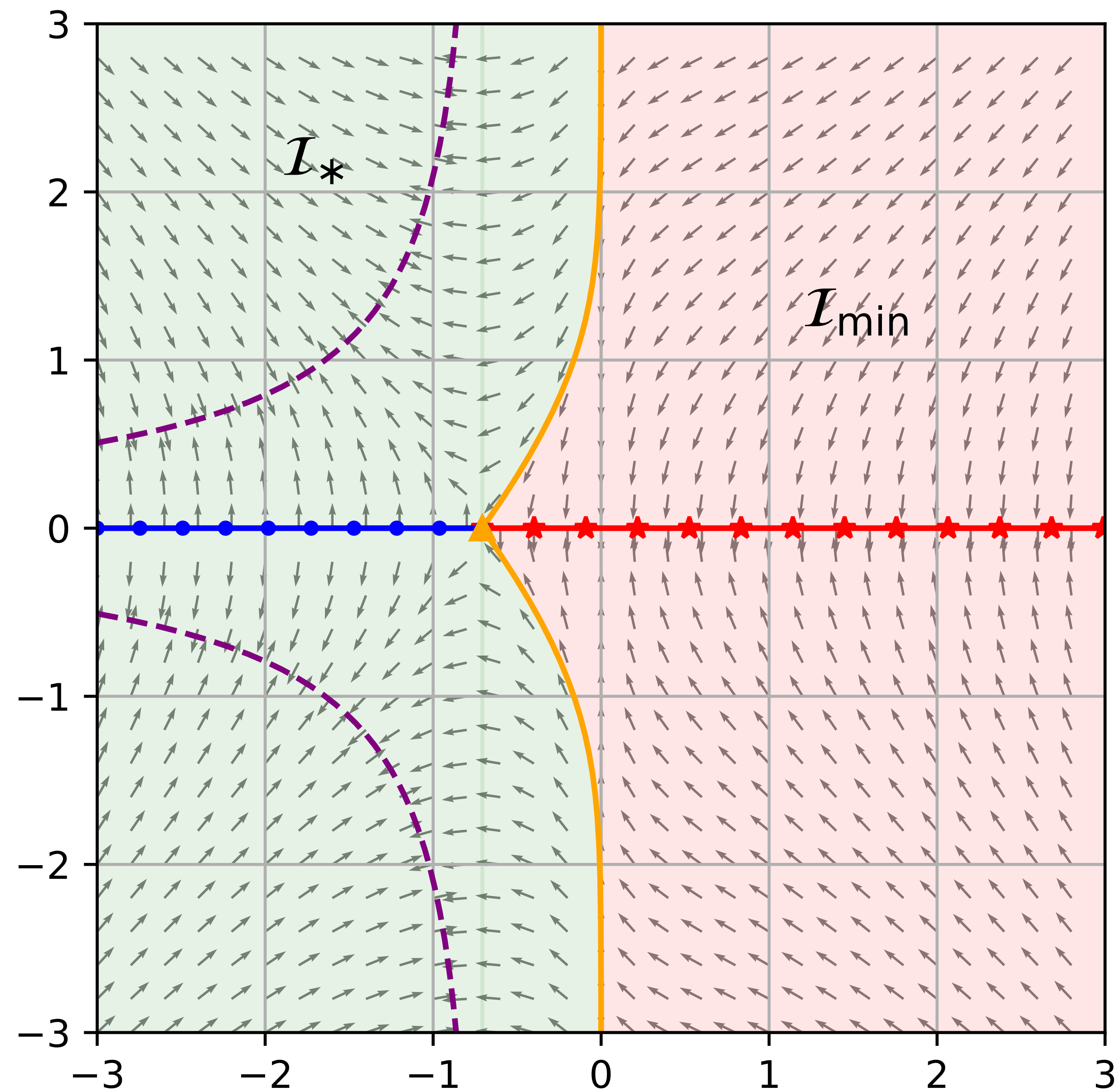




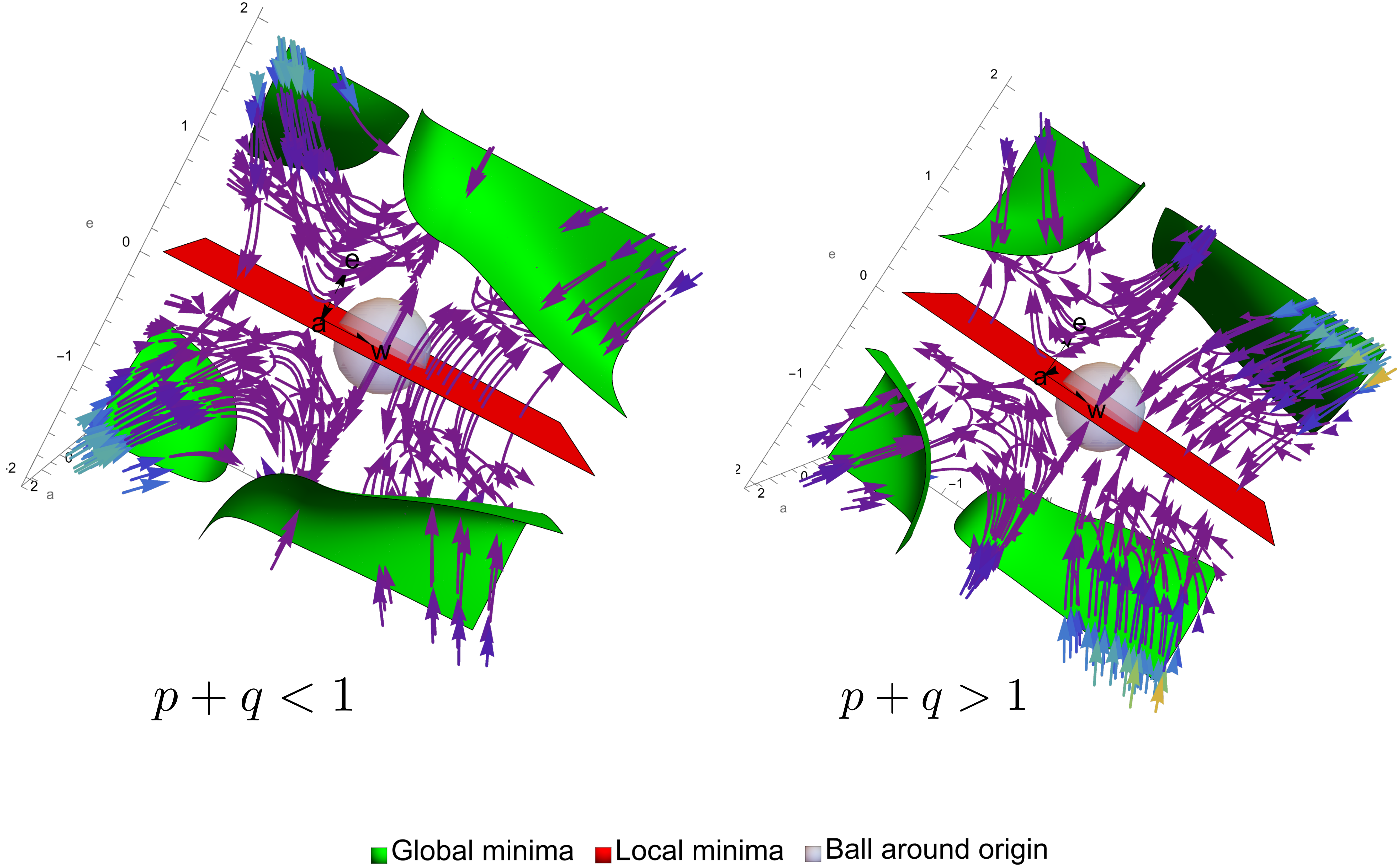
Converges to global minima | **Gets stuck at local minima!**



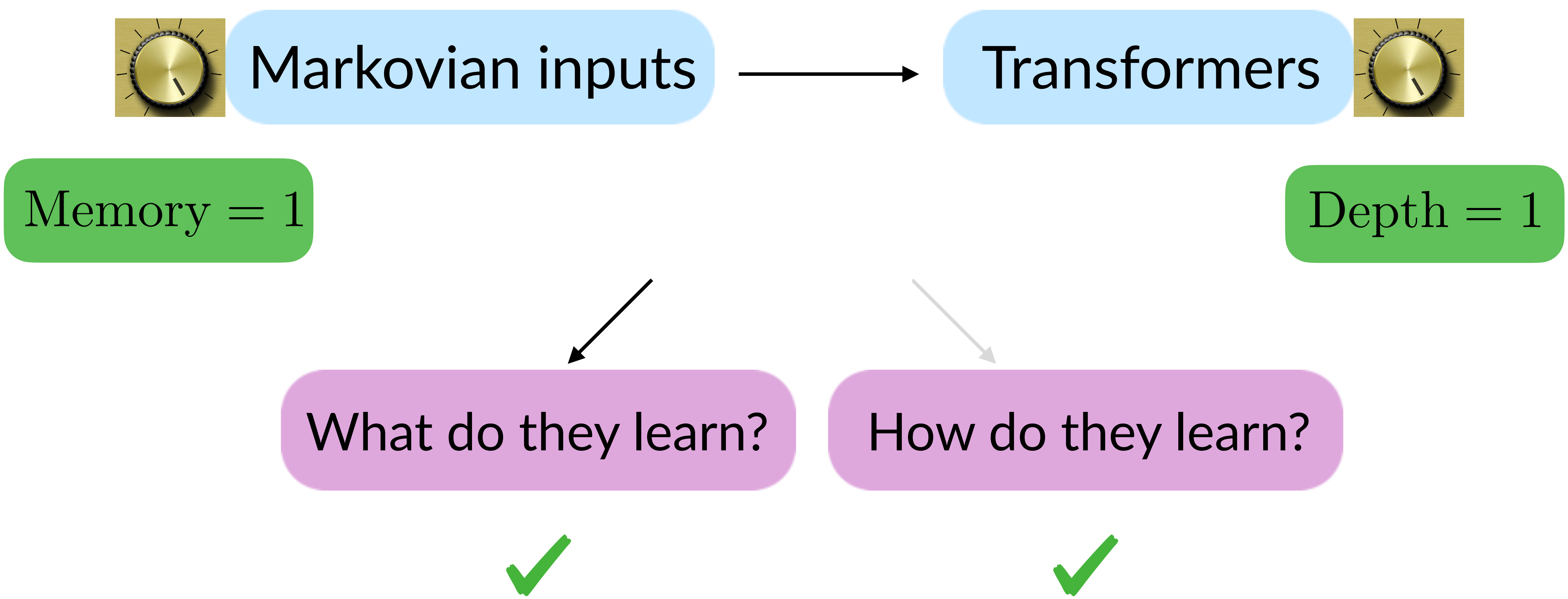
$$p + q < 1$$



$$p + q > 1$$



2
1
e
0
−1
-2
2
0
a
e
a
w
$p+q<1$
2
1
e
0
−1
2
2
0
−1
a
e
a
w
$p+q>1$
Global minima
Local minima
Ball around origin


Markovian inputs
Transformers
$\text{Memory} = 1$
$\text{Depth} = 1$
What do they learn?
How do they learn?
✓
✓

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

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

Part II

Markovian inputs $\longrightarrow$ Transformers

Memory

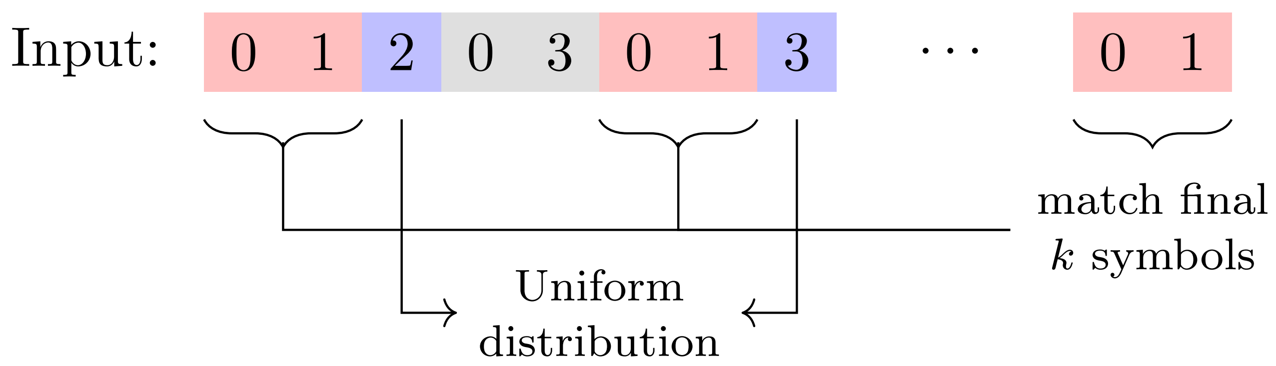
Depth > 1

In-context learning (ICL) emerges!

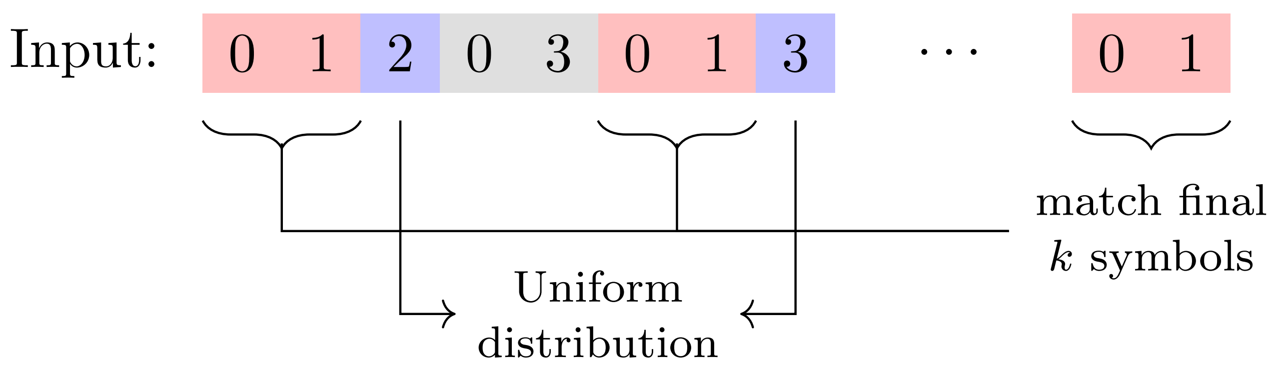
ICL

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

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

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

Memory $= k$

Depth

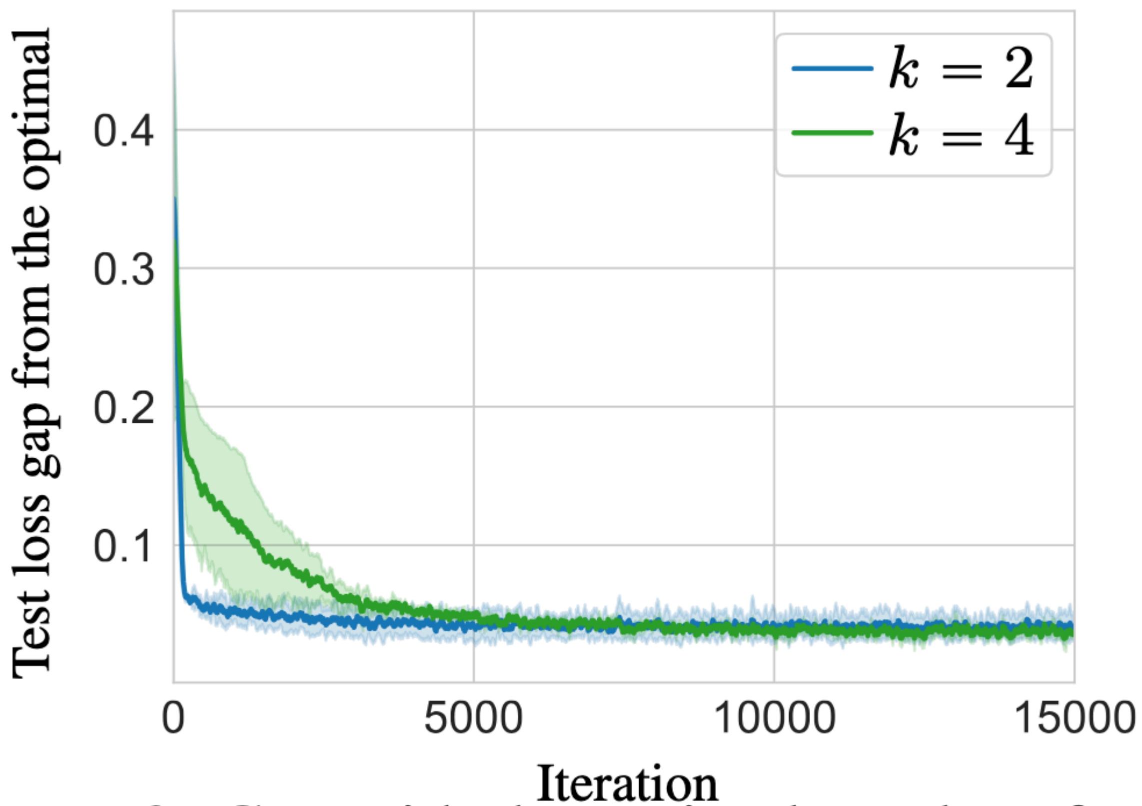
Memory $= k$

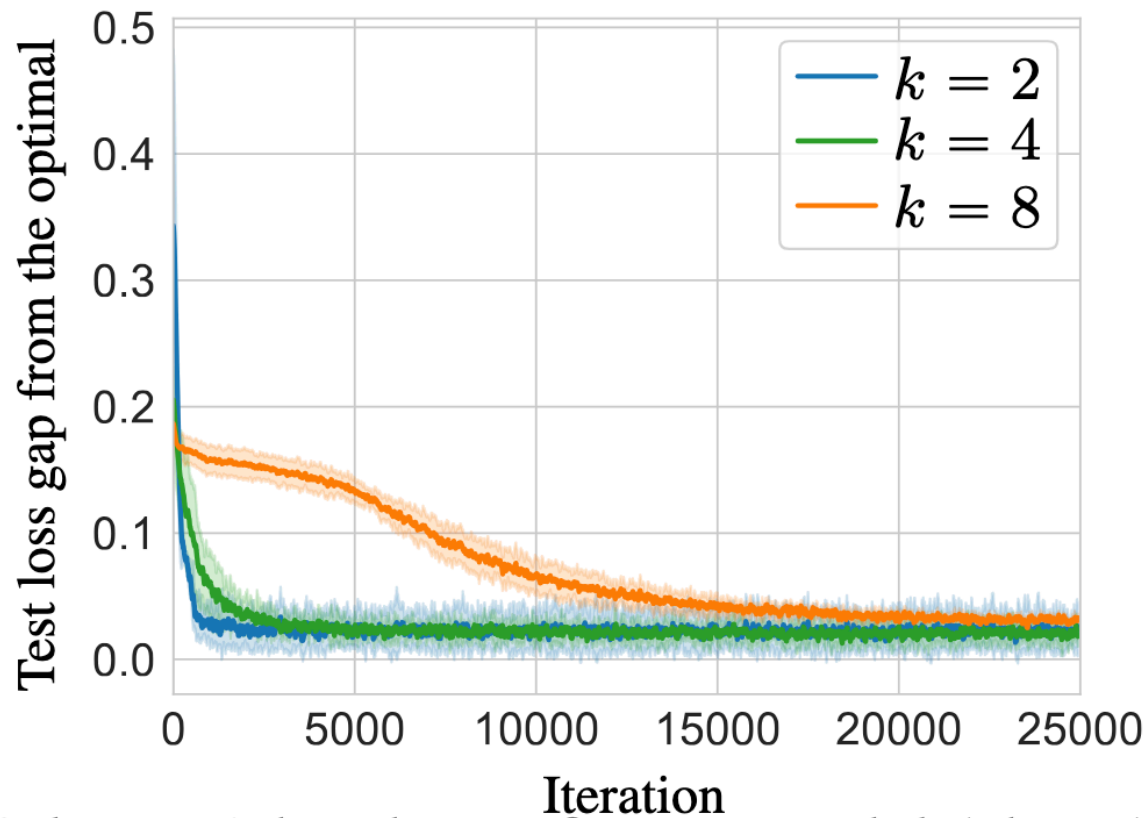
Depth?

What we know so far

- (Edelman et al., 2024 & Niching 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 non-linearities, you need logarithmic depth

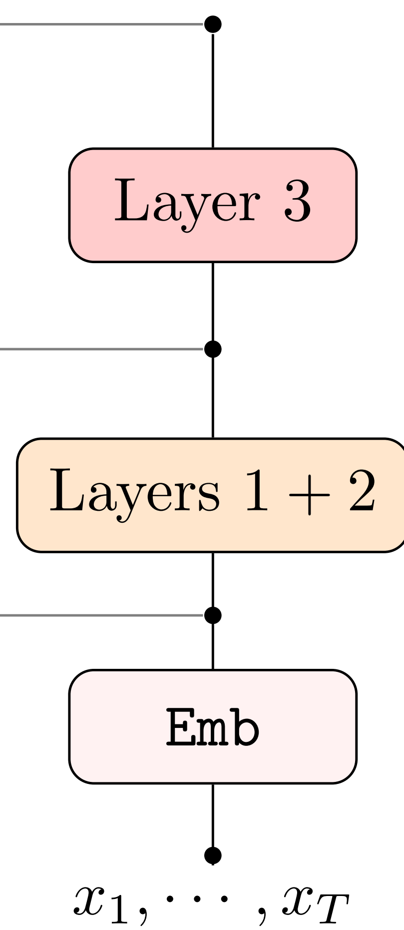
Intuition

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

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

$$\left[\begin{array}{c|c|c|c} \cdots & \begin{array}{c} \mathtt{Emb}(x_n) \\ \frac{\boldsymbol{u}_n}{\|\boldsymbol{u}_n\|_2} \\ \frac{\boldsymbol{v}_n}{\|\boldsymbol{v}_n\|_2} \end{array} & \cdots & \begin{array}{c} \mathtt{Emb}(x_T) \\ \frac{\boldsymbol{u}_T}{\|\boldsymbol{u}_T\|_2} \\ \frac{\boldsymbol{v}_T}{\|\boldsymbol{v}_T\|_2} \end{array} \end{array}\right]$$

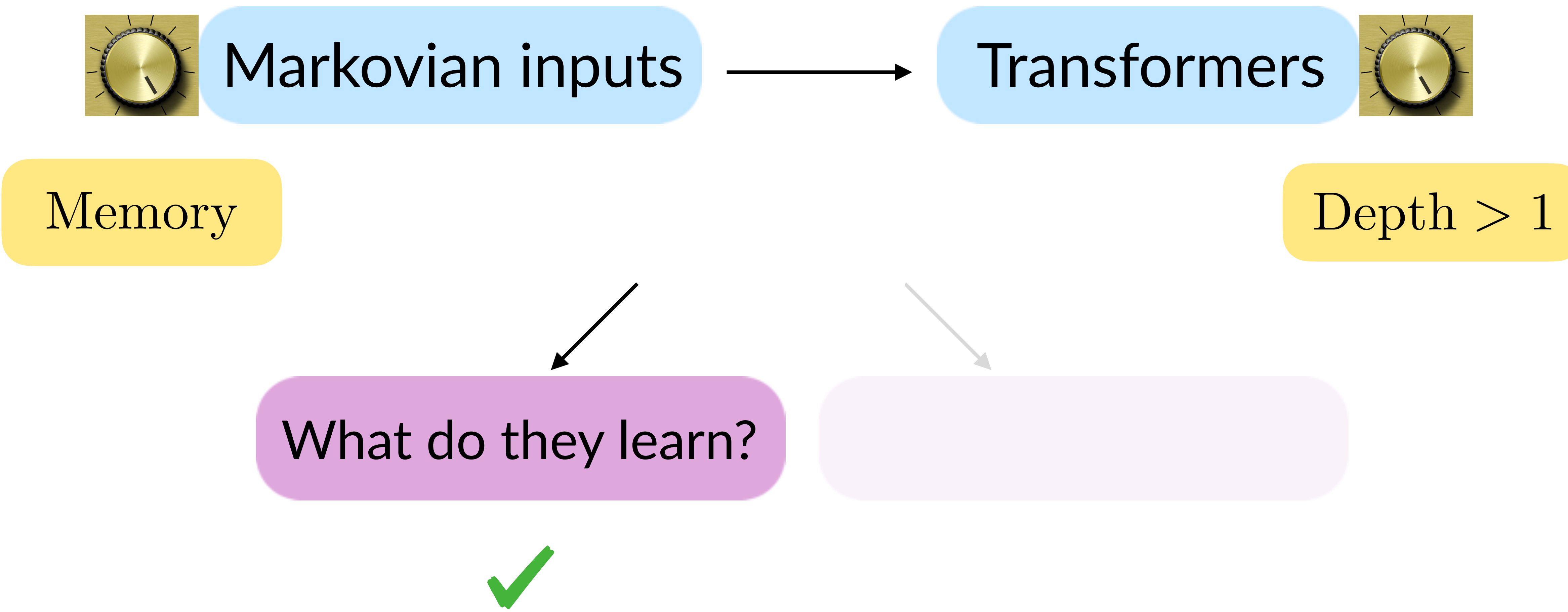
$$\left[\begin{array}{c|c|c|c} \cdots & \mathtt{Emb}(x_n) & \cdots & \mathtt{Emb}(x_T) \end{array}\right]$$



Match contexts & count

Capture context

Summary



Markovian inputs
Transformers
Memory
Depth > 1
What do they learn?

Summary

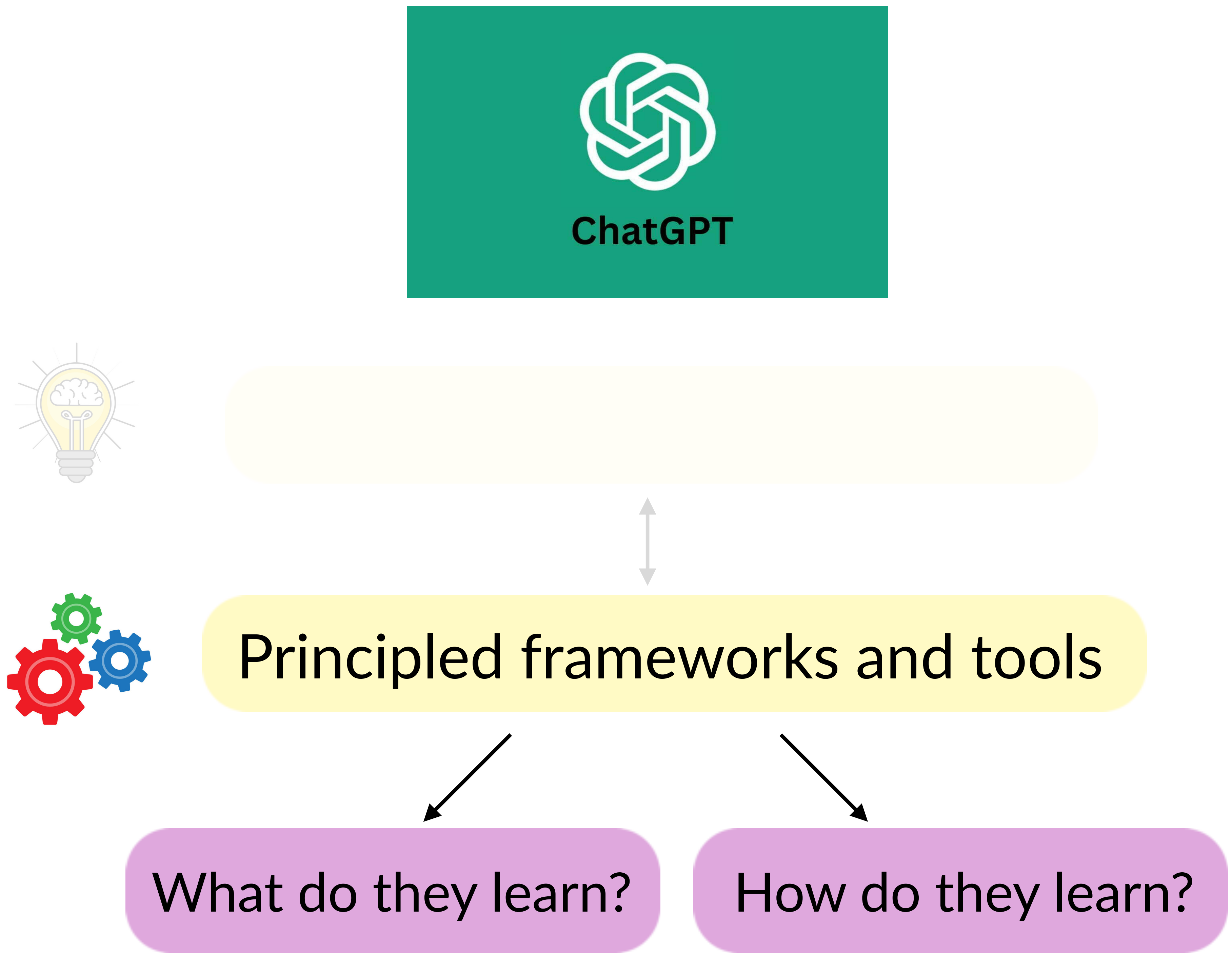
 Markovian inputs → Transformers 

Memory

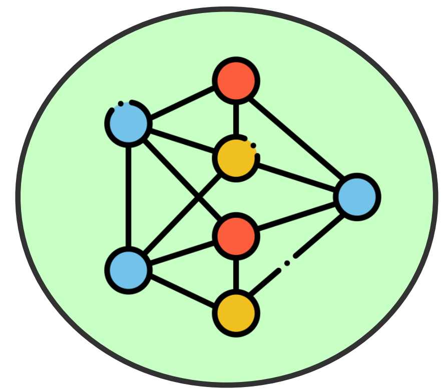
$\text{Depth} > 1$

How do they learn?

Ongoing..


ChatGPT
Principled frameworks and tools
What do they learn?
How do they learn?

Transformers



Attention with Markov

What do they learn? ✓

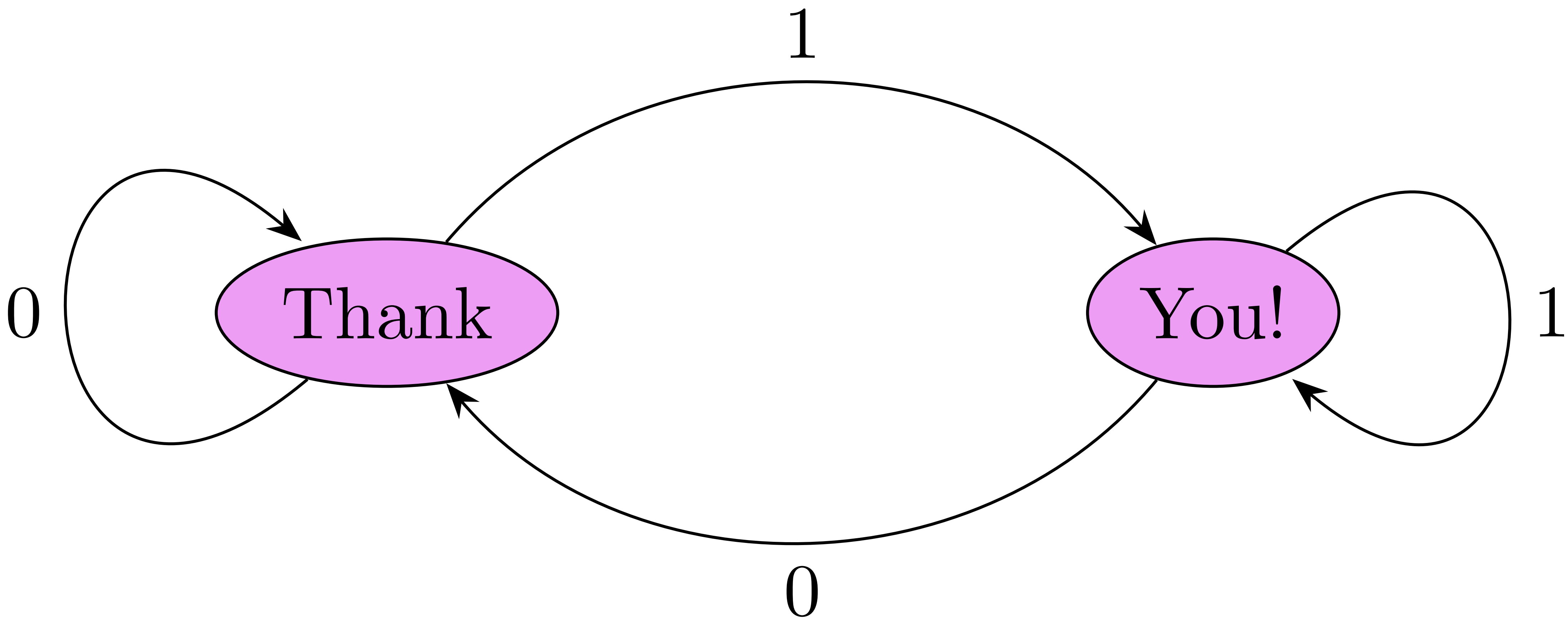
How do they learn? ✓

Future directions

- More realistic data models: Hidden-Markov Models

- State-space models

- More practical guidelines for training LLMs


1
0
Thank
You!
1
0