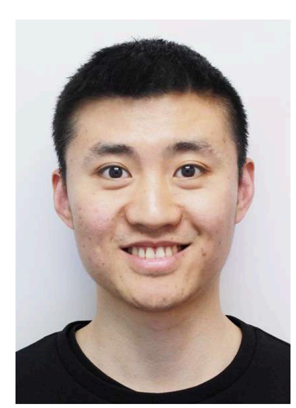
Discreteness
in
Neural Natural Language Processing

Lili Mou[a] Hao Zhou[b] Lei Li[b]

[a]Alberta Machine Intelligence Institute (Amii), University of Alberta
[b]ByteDance AI Lab
doublepower.mou@gmail.com
{zhouhao.nlp,lileilab}@bytedance.com

EMNLP-IJCNLP 2019 Tutorial

Part III: Discrete Latent Space

Roadmap

- Definitions & Examples
- General techniques
  - Maximum likelihood estimation
  - Reinforcement learning
  - Gumbel-softmax
  - Step-by-step Attention
- Case studies
  - Weakly supervised semantic parsing
  - Unsupervised syntactic parsing

Latent Variable

- Consider a probabilistic model on (x, y, z)
  - x: Discriminative (conditional)
  - y: Generative (joint)
  - z: Unknown during both training and prediction

- Their relations depend on applications.
- The definition here is based on the **model** $p(z, y \mid x)$, instead of the **task**

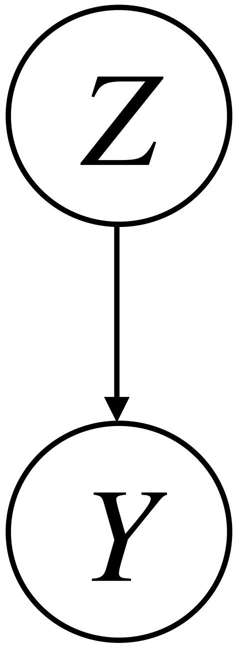
Latent Variable

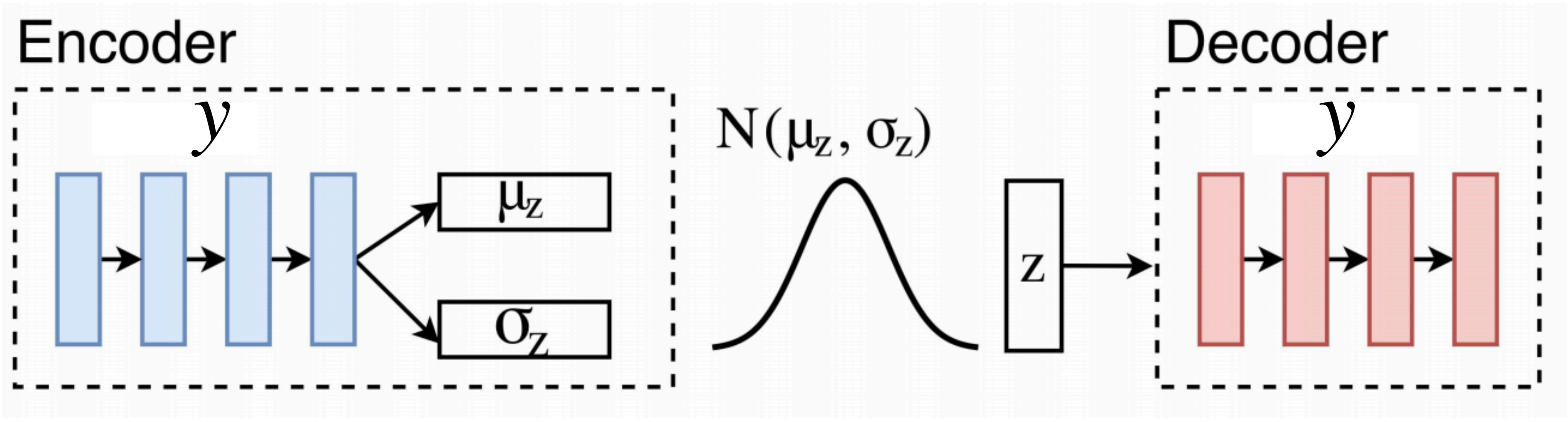
- Consider a probabilistic model on (x, y, z)
  - x: Discriminative (conditional)
  - y: Generative (joint)
  - z: Unknown during both training and prediction

- Their relations depend on applications.
- The definition here is based on the **model** $p(z, y \mid x)$, instead of the **task**

Examples

- Continuous latent variable
  - **Variational autoencoder (VAE)**
  - A data point y is subject to some latent variable y
  - Encoder: recognizing z from y
  - Decoder: generating y from z

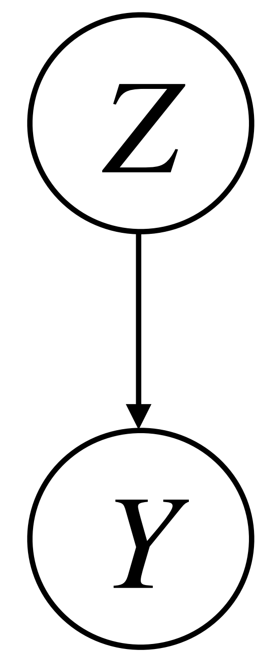


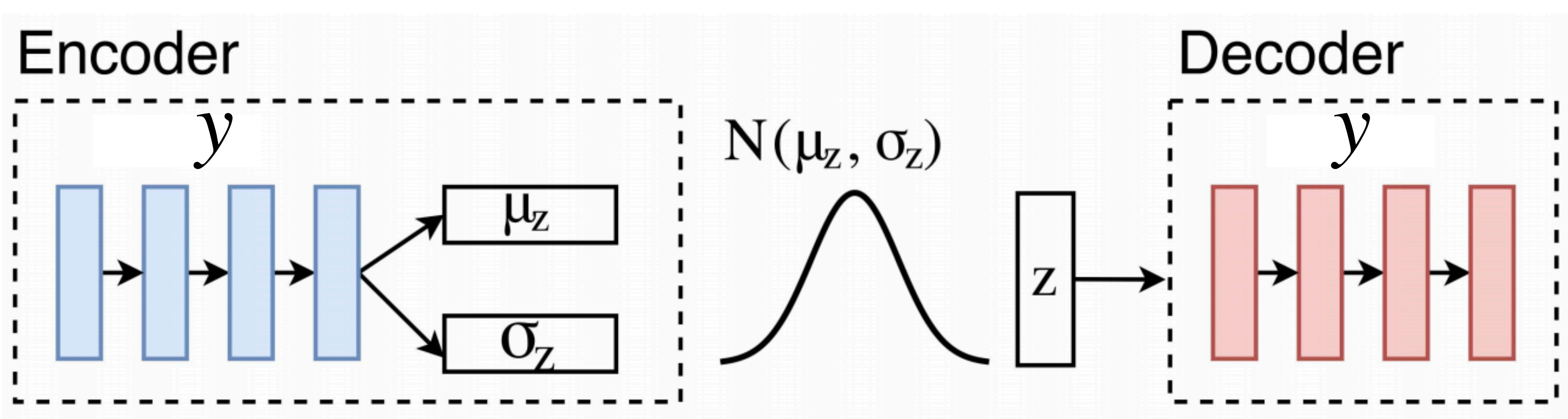


Kingma DP, Welling M. Auto-encoding variational Bayes. In *ICLR*, 2014.

Examples: VAE

- Continuous latent variable
  - **Variational autoencoder (VAE)**
  - A data point y is subject to some latent variable y
  - Encoder: recognizing z from y
  - Decoder: generating y from z



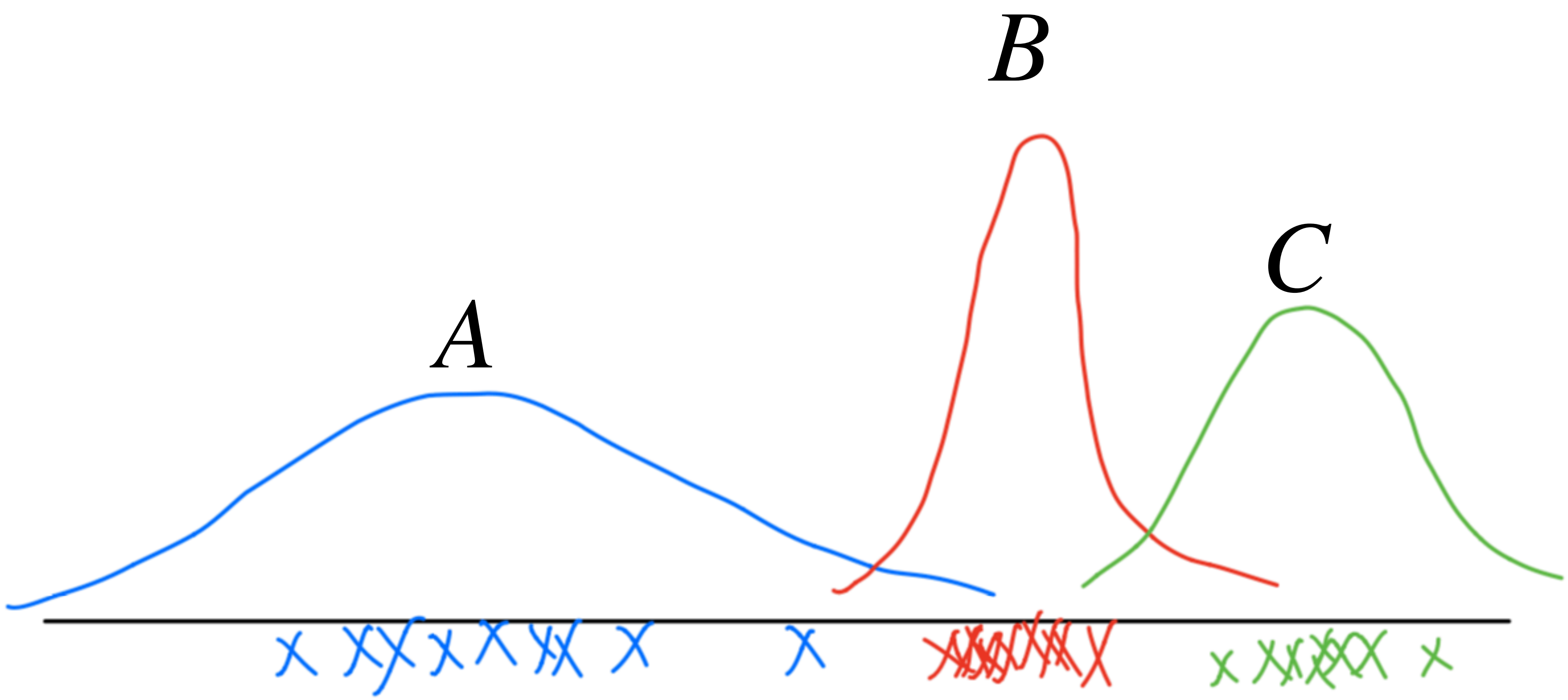


Kingma DP, Welling M. Auto-encoding variational Bayes. In *ICLR*, 2014.

Examples: GMM

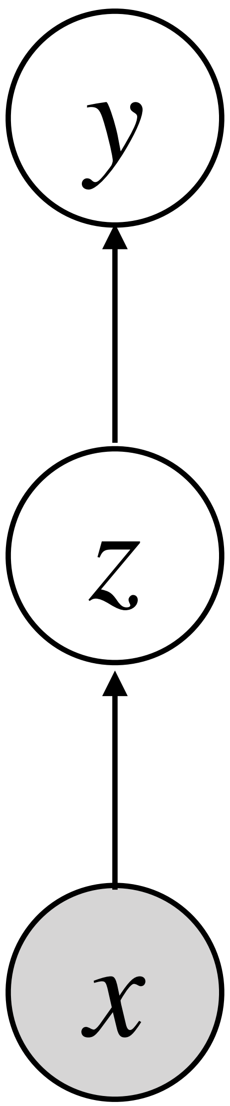
- Discrete latent variable: Clustering with Gaussian mixtures

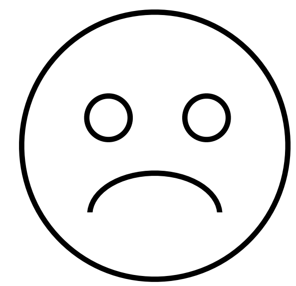
$$z \in \{A, B, C\}$$

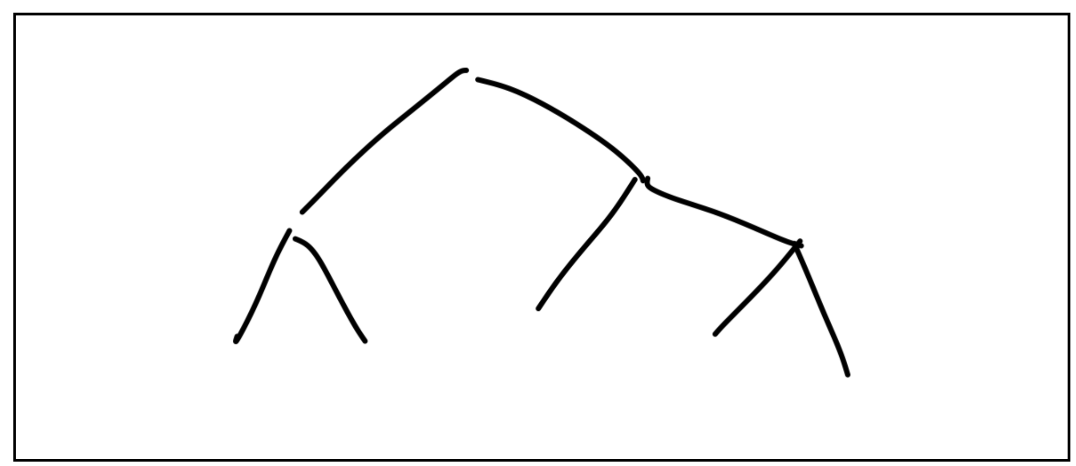


Examples: Latent Tree Induction

- Discrete latent variable: Syntactic parse trees







`The  tutorial is  very  boring`

Latent variables may play a role in discriminative models

General Criteria for Latent Variables

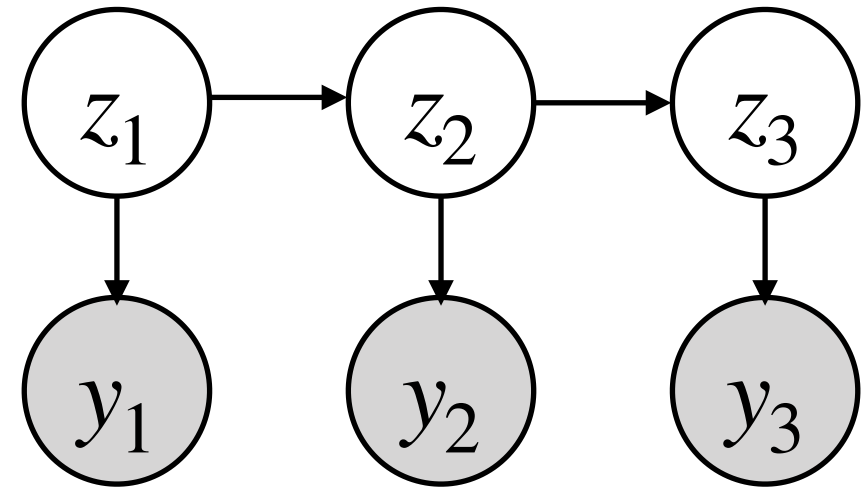
- Training
  - Marginalization
    - Something of $\mathbb{E}$
    - $\mathbb{E}$ of something
    - All sorts of approx. for $\mathbb{E}$
- Inference (depending on applications)
  - Target prediction: Predict y by marginalizing z
  - Latent variable prediction: predict z
    - Max *a posteriori*
    - Sampling

General Criteria for Latent Variables

- Training
  - Marginalization
    - Something of $\mathbb{E}$
    - $\mathbb{E}$ of something
    - All sorts of approx. for $\mathbb{E}$
- Inference (depending on applications)
  - Target prediction: Predict y by marginalizing z
  - Latent variable prediction: predict z
    - Max *a posteriori*
    - Sampling

Maximum Likelihood Estimation

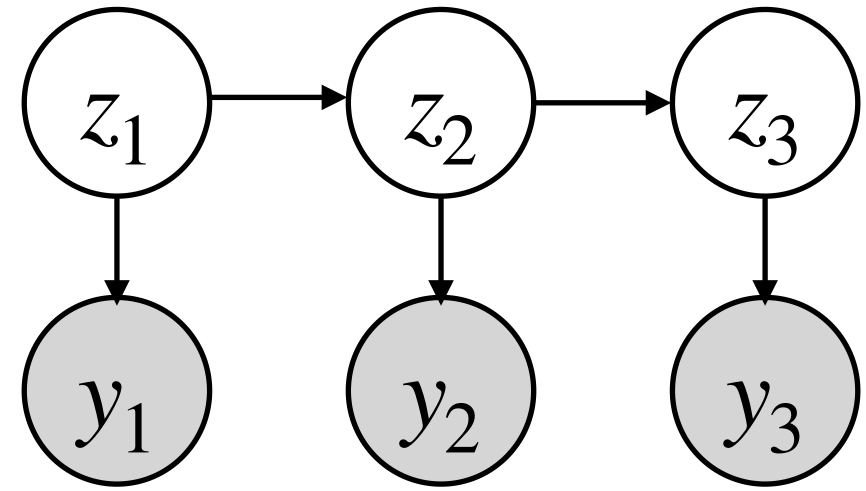
Hidden Markov Models



- Observed tokens: $y_1, y_2, \cdots, y_T$
- Latent states: $z_1, \cdots, z_T$
- Generative procedure
  - Choose z_1 (omitted here)
  - For every step t:
    - Pick $z_t \sim p(z_t \mid z_{t-1})$
    - Emit $y_t \sim p(y_t \mid z_t)$
  - Suppose both parametrized by probability tables
- Example
  - $y_1, y_2, \cdots, y_T$: a sequence of words
  - $z_1, z_2, \cdots, z_T$: POS tags

Rabiner LR, Juang BH. An introduction to hidden Markov models. *IEEE ASSP Magazine,* 1986.

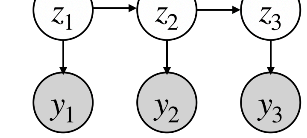
Hidden Markov Models



- Observed tokens: $y_1, y_2, \cdots, y_T$
- Latent states: $z_1, \cdots, z_T$
- Generative procedure
  - Choose z_1 (omitted here)
  - For every step t:
    - Pick $z_t \sim p(z_t \mid z_{t-1})$
    - Emit $y_t \sim p(y_t \mid z_t)$
  - Suppose both parametrized by probability tables
- Example
  - $y_1, y_2, \cdots, y_T$: a sequence of words
  - $z_1, z_2, \cdots, z_T$: POS tags

Rabiner LR, Juang BH. An introduction to hidden Markov models. *IEEE ASSP Magazine,* 1986.

Hidden Markov Models



- **E-step** (expectation for sufficient statistics)
  - Expectation of a state, that is, $\gamma_t(i) \stackrel{\Delta}{=} \mathbb{E}[z_t = i \mid \cdot\,]$
  - Expectation of two consecutive states, that is, $\xi_t(i, j) \stackrel{\Delta}{=} \mathbb{E}[z_t = i, z_{t+1} = j \mid \cdot\,]$
  - Computed by

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{p(\boldsymbol{Y})} \qquad \xi_t(i, j) = \frac{\alpha_t(i) p_{\boldsymbol{\theta}}(x_t \mid z_n = i) p_{\boldsymbol{\theta}}(z_t = j \mid z_{t-1} = i)\beta_t(j)}{p(\boldsymbol{Y})}$$

where

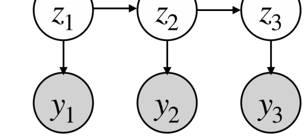
$$\alpha_t(i) \stackrel{\Delta}{=} p(\boldsymbol{y}_{1:t}, z_t = i)$$

and

$$\beta_t(i) \stackrel{\Delta}{=} p(\boldsymbol{y}_{t+1:T} \mid z_t = i)$$

are given by dynamic programming

Hidden Markov Models



- **E-step** (expectation for sufficient statistics)
  - Expectation of a state, that is, $\gamma_t(i) \triangleq \mathbb{E}[z_t = i \,|\, \cdot\,]$
  - Expectation of two consecutive states, that is, $\xi_t(i,j) \triangleq \mathbb{E}[z_t = i, z_{t+1} = j \,|\, \cdot\,]$
- **M-step** (MLE by soft counting)

$$p(z_t = j \,|\, z_{t-1} = i) = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$p(x \,|\, z_t = j) = \frac{\sum_{t=1}^{T} \gamma_t(j) \mathbb{1}\{X_t = x\}}{\sum_{t=1}^{T} \gamma_t(j)}$$

EM as MLE

$$\ell(\boldsymbol{\theta}_{t+1}) = \sum_i \log p(\boldsymbol{y}_i; \boldsymbol{\theta}_{t+1})$$

$$= \sum_i \log \left(\sum_z p(\boldsymbol{y}_i, z; \boldsymbol{\theta}_{t+1}) \right)$$

$$\geq \sum_i \sum_z q_t(z \,|\, \boldsymbol{y}_i) \log \frac{p(\boldsymbol{y}_i, z; \boldsymbol{\theta}_{t+1})}{q_t(z \,|\, \boldsymbol{y}_i)}$$

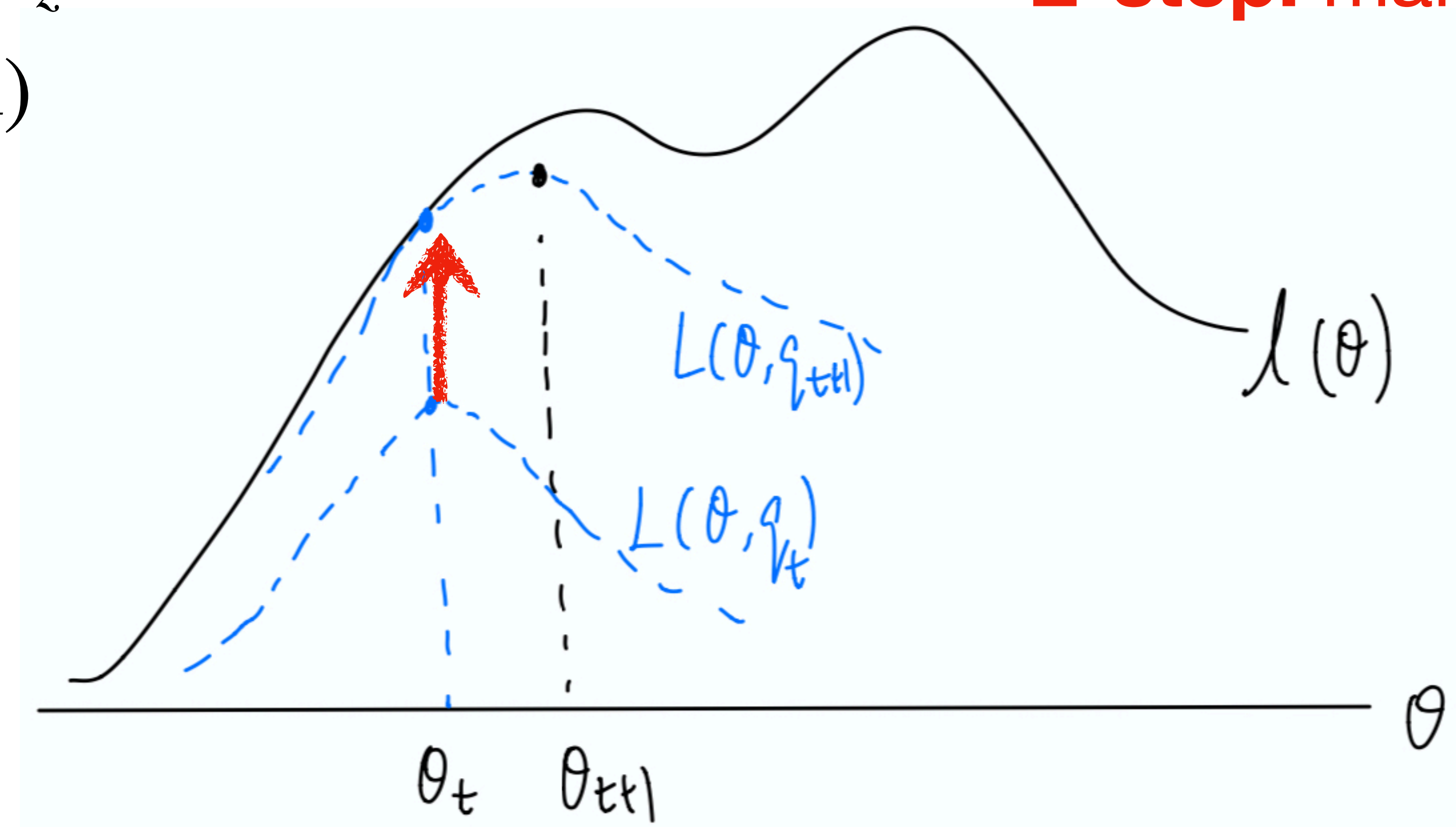
$$\geq \sum_i \sum_z q_t(z \,|\, \boldsymbol{y}_i) \log \frac{p(\boldsymbol{y}_i, z; \boldsymbol{\theta}_t)}{q_t(z \,|\, \boldsymbol{y}_i)}$$

$$= \ell(\boldsymbol{\theta}_t)$$

[Lower bound holds for any q_t]

M-step: $\boldsymbol{\theta}_{t+1} = \arg\max\{\,\cdot\,\}$

E-step: make lower bound tight



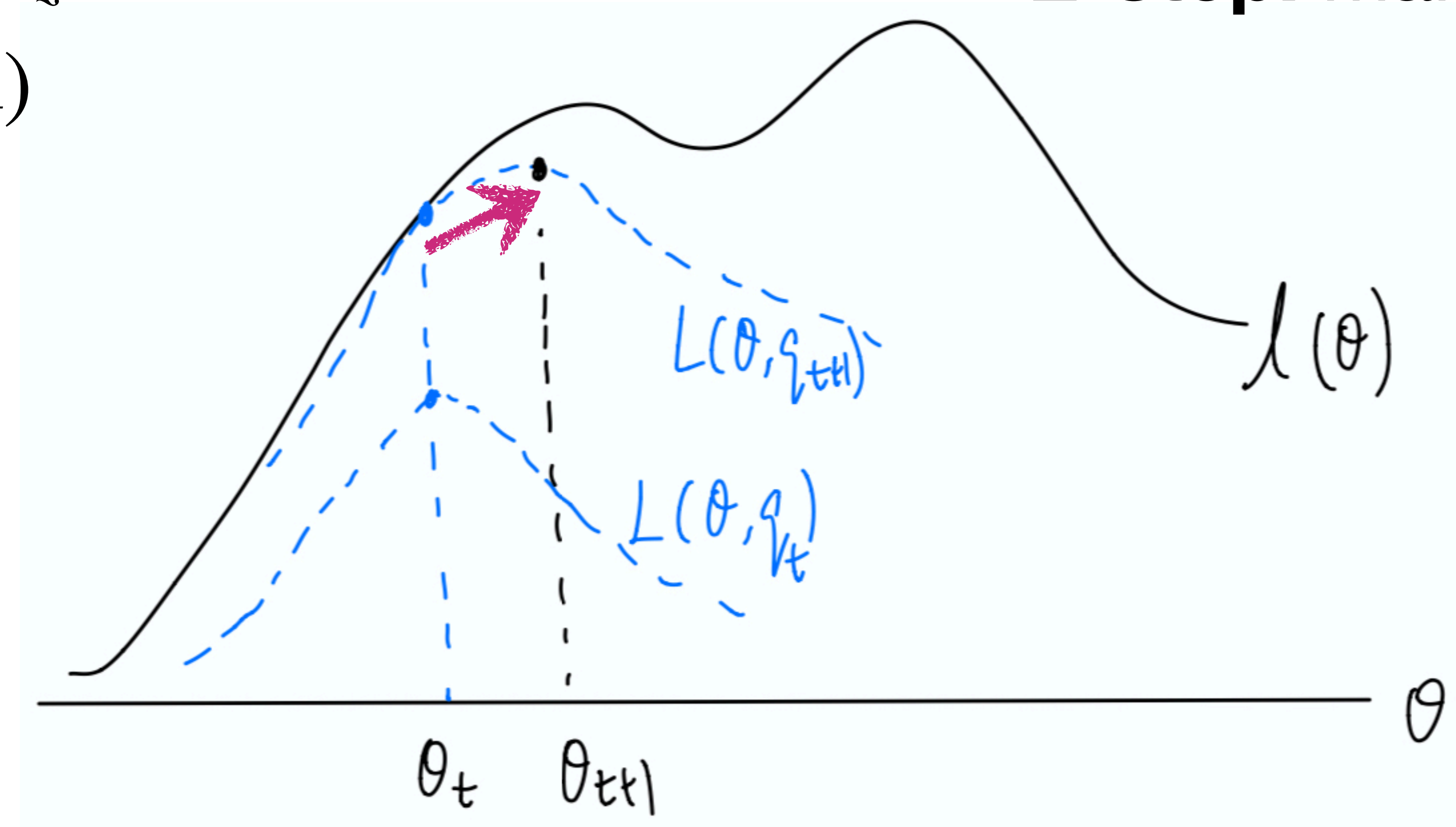
EM as MLE

$$
\begin{aligned}
\ell(\boldsymbol{\theta}_{t+1}) &= \sum_i \log p(\boldsymbol{y}_i; \boldsymbol{\theta}_{t+1}) \\
&= \sum_i \log \left(\sum\nolimits_z p(\boldsymbol{y}_i, z; \boldsymbol{\theta}_{t+1}) \right) \\
&\geq \sum_i \sum_z q_t(z \,|\, \boldsymbol{y}_i) \log \frac{p(\boldsymbol{y}_i, z; \boldsymbol{\theta}_{t+1})}{q_t(z \,|\, \boldsymbol{y}_i)} \\
&\geq \sum_i \sum_z q_t(z \,|\, \boldsymbol{y}_i) \log \frac{p(\boldsymbol{y}_i, z; \boldsymbol{\theta}_{t})}{q_t(z \,|\, \boldsymbol{y}_i)} \\
&= \ell(\boldsymbol{\theta}_t)
\end{aligned}
$$

[Lower bound holds for any q_t]

M-step: $\boldsymbol{\theta}_{t+1} = \arg\max\{\,\cdot\,\}$

E-step: make lower bound tight



Back Propagation

$$\log p(\boldsymbol{Y} \mid \boldsymbol{\theta}) = \log\left(\sum_z p(\boldsymbol{Y}, z \mid \boldsymbol{\theta})\right)$$

- Perplexity of BP = $\mathcal{O}$(Perplexity of FP)
- EM is BP

$$p(y, z \mid x) = \frac{1}{Z}\exp\{\sum_i \theta_i f_i\}$$

$$\frac{\partial}{\partial \theta_i}\log p(y, z \mid x) = \mathbb{E}_{z \sim p(z \mid x, y)}[f_i] - \mathbb{E}_{y, z \sim p(y, z \mid x)}[f_i]$$

Eisner, Jason. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, 2016.

Back Propagation

$$\log p(\boldsymbol{Y} \mid \boldsymbol{\theta}) = \log\left(\sum_z p(\boldsymbol{Y}, z \mid \boldsymbol{\theta})\right)$$

- Perplexity of BP = $\mathcal{O}$(Perplexity of FP)
- EM is BP

$$p(y, z \mid x) = \frac{1}{Z}\exp\{\sum_i \theta_i f_i\}$$

$$\frac{\partial}{\partial \theta_i}\log p(y, z \mid x) = \mathbb{E}_{z \sim p(z \mid x, y)}[f_i] - \mathbb{E}_{y, z \sim p(y, z \mid x)}[f_i]$$

Eisner, Jason. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, 2016.

Other Treatments

$$\log p(\boldsymbol{Y} \,|\, \boldsymbol{\theta}) = \log\left(\boxed{\sum_z} p(\boldsymbol{Y}, z \,|\, \boldsymbol{\theta})\right)$$

- Exact marginalization (enumeration as in GMM, DP as in HMM)
- Hard-EM: Choose the single best z
  - E.g., K-means clustering
- Choose top-N latent variables
  - Beam search
- Sampling

Other Treatments

$$\log p(\boldsymbol{Y} \mid \boldsymbol{\theta}) = \log\left(\boxed{\sum_z} p(\boldsymbol{Y}, z \mid \boldsymbol{\theta})\right)$$

- Exact marginalization (enumeration as in GMM, DP as in HMM)
- Hard-EM: Choose the single best z
  - E.g., K-means clustering
- Choose top-N latent variables
  - Beam search
- Sampling

Latent Variables in Discriminative Model

- In GMM and HMM
  - We model the joint probability $p(z, y)$
- Sometimes we have discriminative variables
  - We predict y from x with z being a latent variable

$$\log p_{\boldsymbol{\theta}}(y \mid x) = \log \left(\sum_z p_{\boldsymbol{\theta}}(y, z \mid x) \right)$$

Massage

maximize $$\log\left(\boxed{\sum_z p(z)}\, p(\boldsymbol{Y}\,|\,z,\boldsymbol{\theta})\right)$$

maximize $$\sum_z p(z)\,\log\left(p(\boldsymbol{Y}\,|\,z,\boldsymbol{\theta})\right)$$

$\Downarrow$ generalize

maximize $$\sum_z p(z)\,R(Y\,|\,z,\boldsymbol{\theta})$$

Reinforcement Learning

Markov Decision Process

- In a time series, $t = 1,\ 2, \cdots,\ T$
  - We are in some states, $s_1, s_2, \cdots, s_T$
  - We take action $a_1, a_2, \cdots, a_T$
  - We have reward $r_1, r_2, \cdots, r_T$

Sutton RS, Barto AG. Introduction to Reinforcement Learning. 1998.

Markov Decision Process

- In a time series, $t = 1,\ 2, \cdots,\ T$
  - We are in some states, $s_1, s_2, \cdots, s_T$
  - We take action $a_1, a_2, \cdots, a_T$
  - We have reward $r_1, r_2, \cdots, r_T$
- Formally, MDP: $\langle S, A, P, R, \gamma \rangle$

  S : Set of states

  A : Set of actions

  $P^a_{ss'} = \mathbb{P}[S_{t+1} = s' \,|\, S_t = s, A_t = a]$

  R^a_s : Reward at state s with action a

  γ : Discount factor in $[0,1]$

MDP in NLP

- Consider a text generation task

  (we assume latent)

- Formally, MDP: $\langle S, A, P, R, \gamma \rangle$

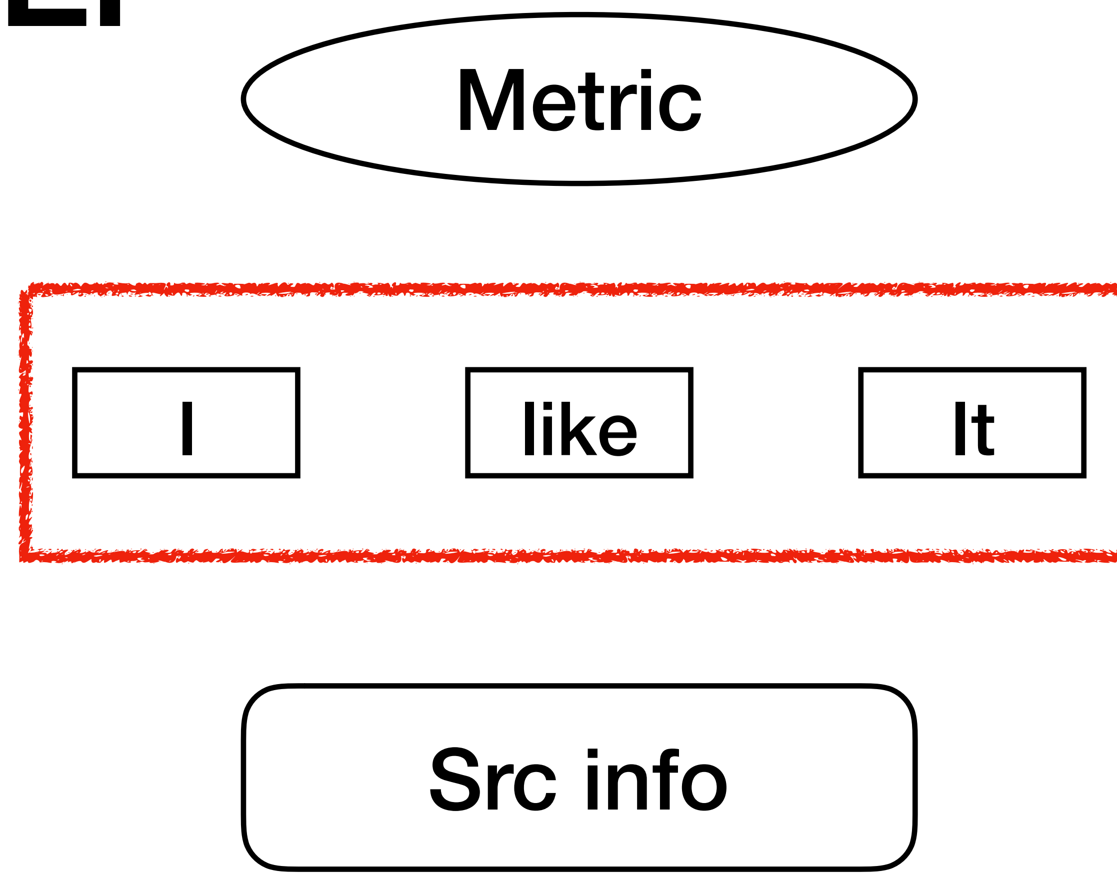
  S : Set of states

  A : Set of actions

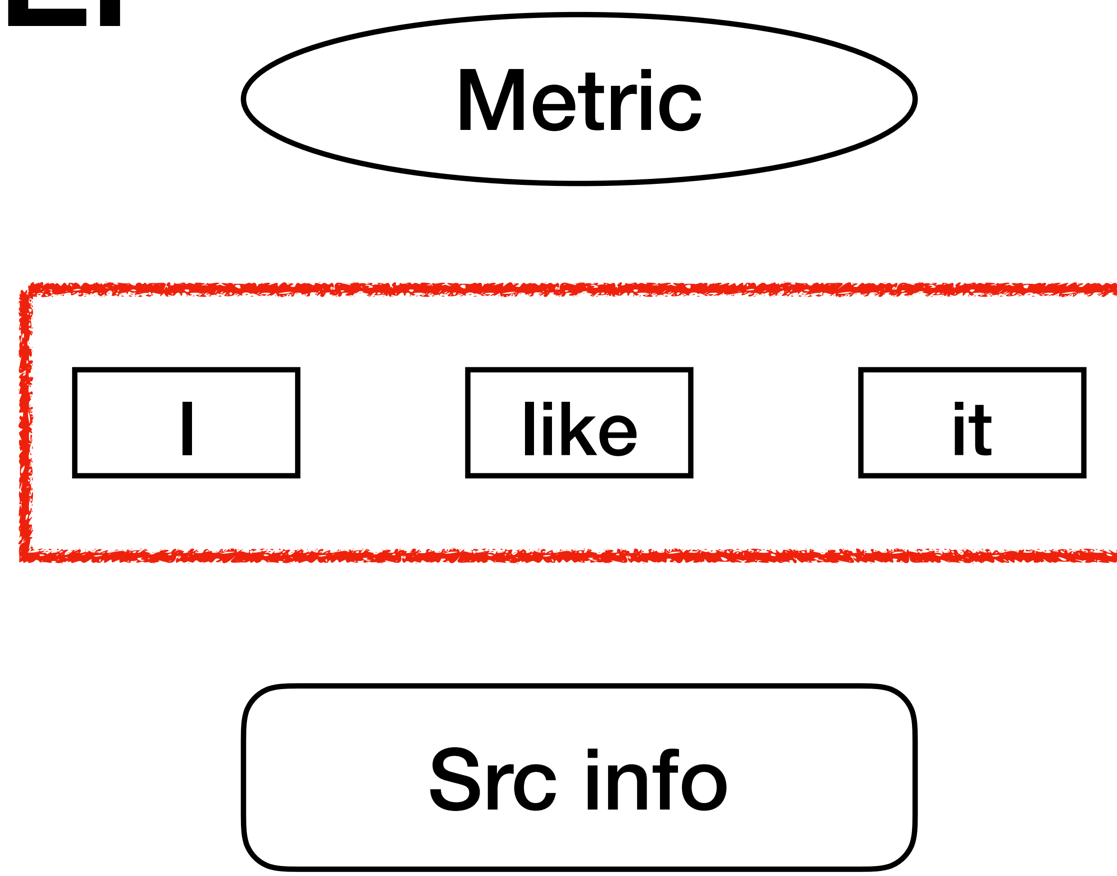
  $P_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \,|\, S_t = s, A_t = a]$

  R_s^a : Reward at state s with action a

  γ : Discount factor in $[0,1]$



MDP in NLP



- Consider a text generation task

  (we assume latent)

- Formally, MDP: $\langle S, A, P, R, \gamma \rangle$

  S : Set of states

  States: Src & generated words
  Usually approximated by NN

  $a]$

  R_s^a : Reward at state s with action a

  γ : Discount factor in $[0,1]$

MDP in NLP

- Consider a text generation task

  (we assume latent)

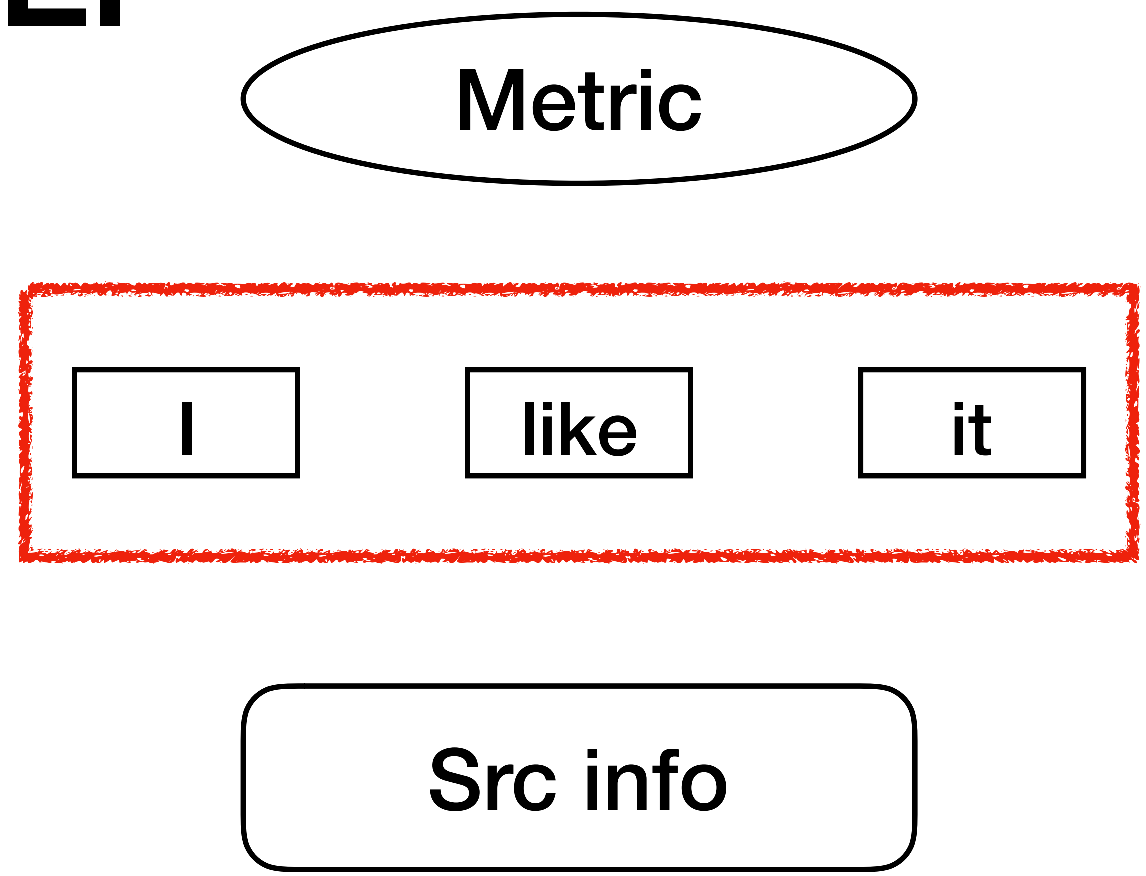
- Formally, MDP: $\langle S, A, P, R, \gamma \rangle$

  S : Set of states

  A : Set of actions

Actions: all words in vocabulary, usually very large

γ : Discount factor in $[0,1]$



MDP in NLP

- Consider a text generation task

  (we assume latent)

- Formally, MDP: $\langle S, A, P, R, \gamma \rangle$

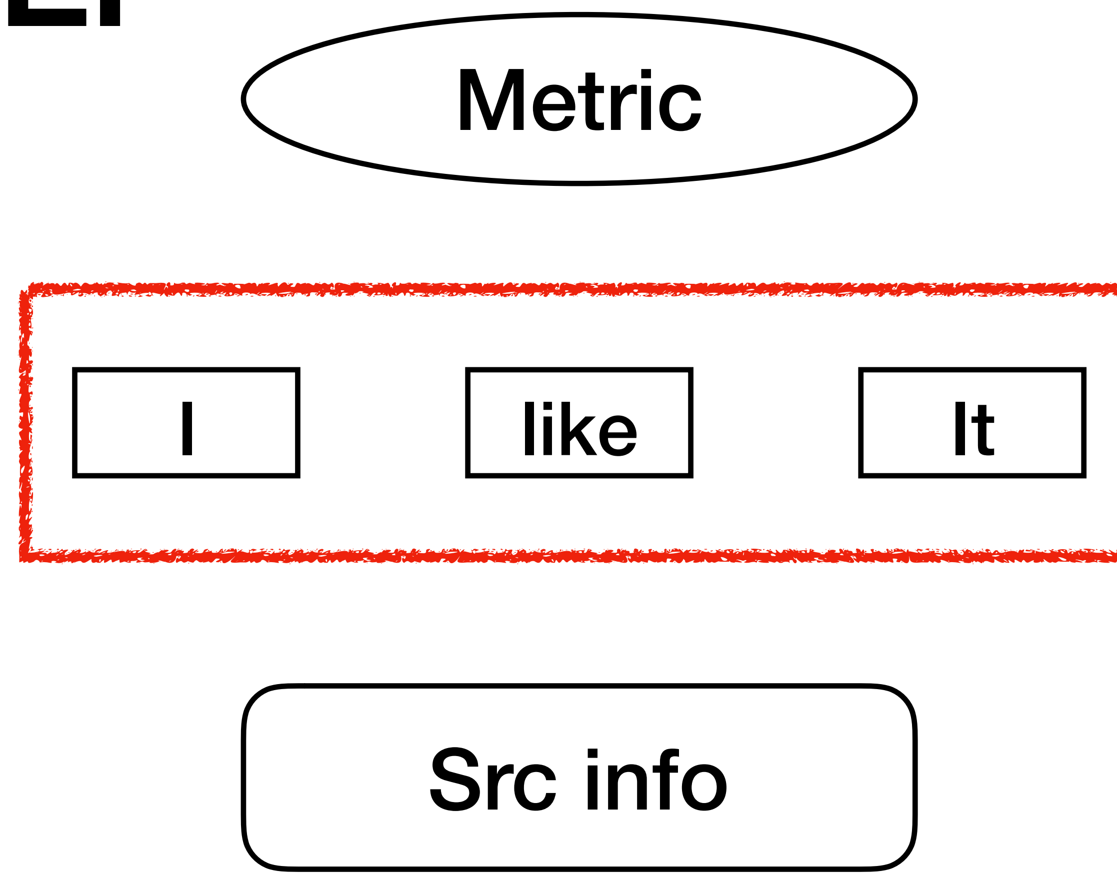
  S : Set of states

  A : Set of actions

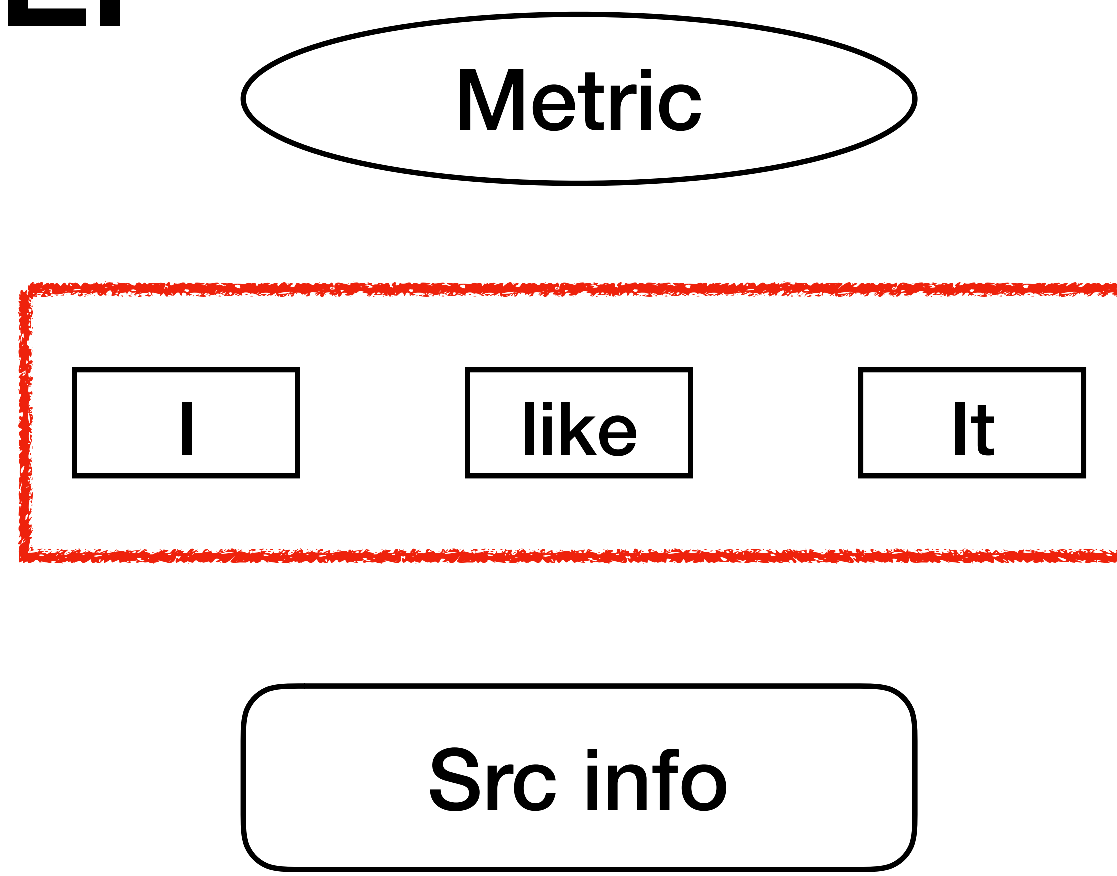
  $$P^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

  action a

Transition: deterministic



MDP in NLP



- Consider a text generation task

  (we assume latent)

- Formally, MDP: $\langle S, A, P, R, \gamma \rangle$

  S : Set of states

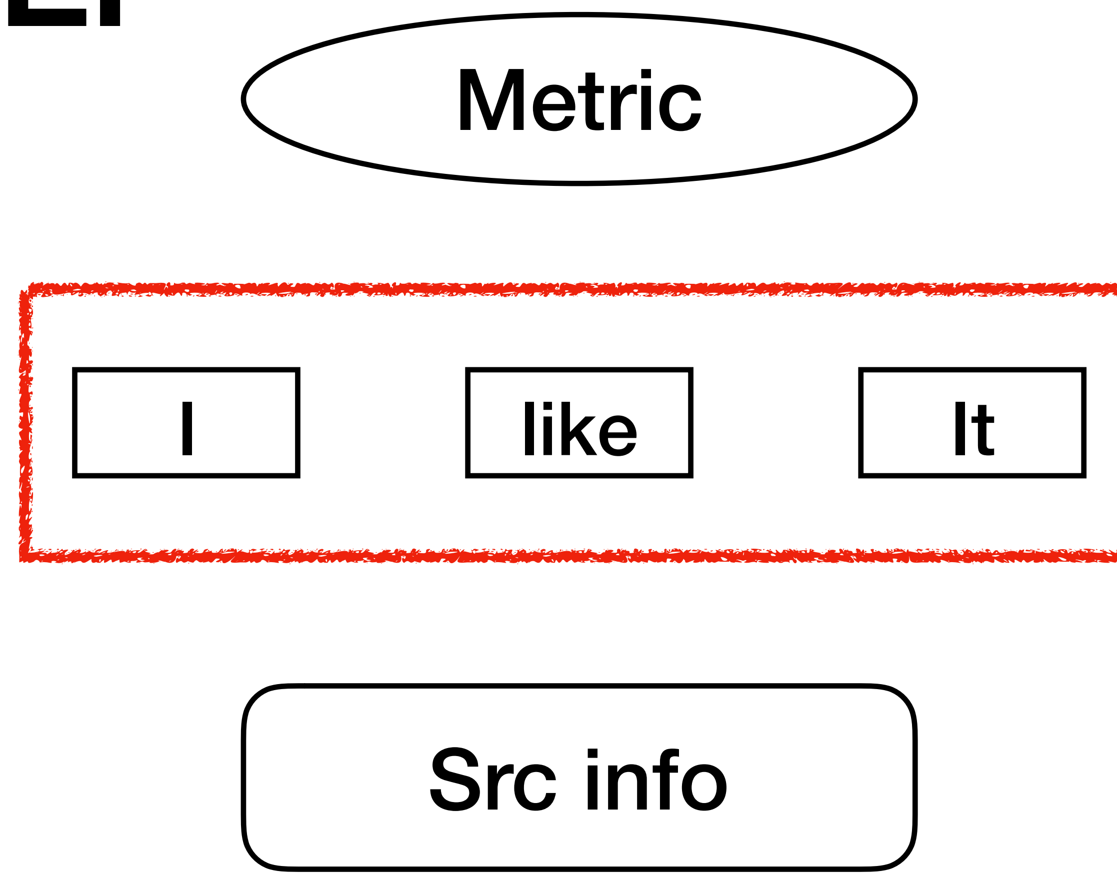
  A : Set of actions

  $P^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$

  R^a_s : Reward at state s with action a

Reward: measure of success, usually very sparse

MDP in NLP



- Consider a text generation task

  (we assume latent)

- Formally, MDP: $\langle S, A, P, R, \gamma \rangle$

  S : Set of states

  A : Set of actions

  $S_t = s, A_t = a]$

  s with action a

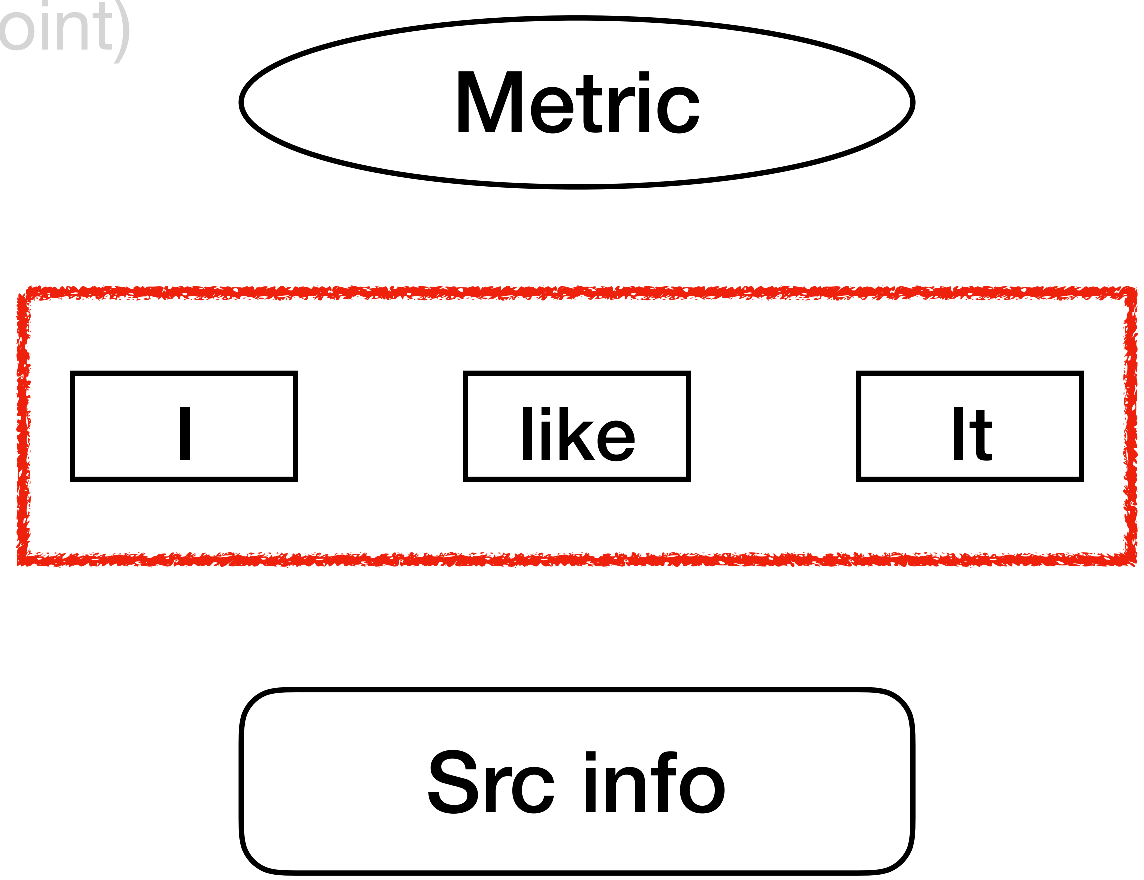
  Discount: doesn't matter too much

  γ : discount factor in $[0,1]$

REINFORCE

- Stochastic policy
  - Action given state (called policy) modeled by probability
  - Model $p(action \mid \cdot)$
  - Action is our latent variable, called z
- Monte Carlo sampling
  - Sampling until the end of episode (data point)
  - No bootstrapping
- Goal is to maximize

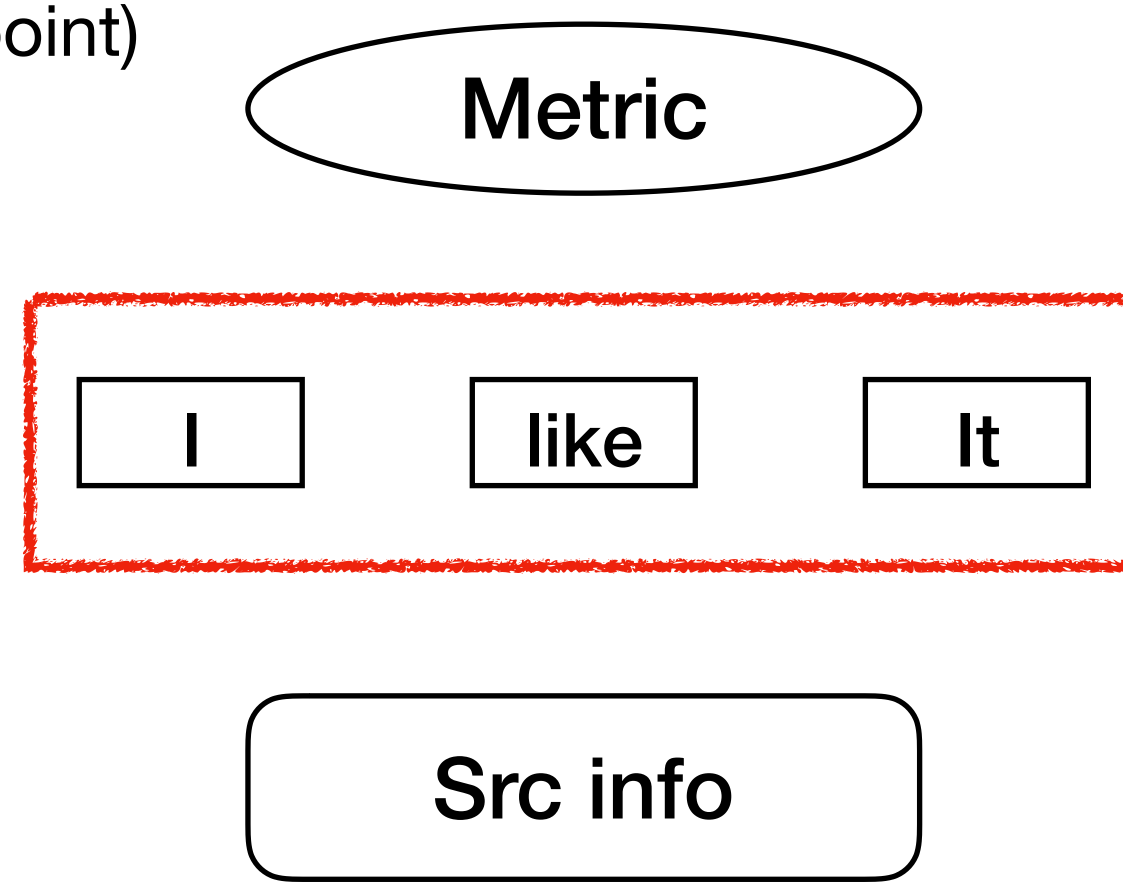
$$\mathbb{E}_z\, R(Y \mid z; \boldsymbol{\theta})$$



REINFORCE

- Stochastic policy
  - Action given state (called policy) modeled by probability
  - Model $p(action \mid \cdot\,)$
  - Action is our latent variable, called z
- Monte Carlo sampling
  - Sampling until the end of episode (data point)
  - No bootstrapping
- Goal is to maximize

$$\mathbb{E}_z \; R(Y \mid z; \boldsymbol{\theta})$$



For simplicity, we here only consider the reward at the end of the sequence

REINFORCE: MC Policy Gradient

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathop{\mathbb{E}}_{z_1,\cdots,z_T \sim p_{\boldsymbol{\theta}}} \Big[-R(y_1, \cdots, y_n \,|\, z_1, \cdots, z_T) \Big]$$

Statisticians seem to be pessimistic creatures who think in terms of losses. Decision theorists in economics and business talk instead in terms of gains (utility).

James O. Berger (1985). *Statistical Decision Theory and Bayesian Analysis*.

REINFORCE: MC Policy Gradient

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathop{\mathbb{E}}_{z_1,\cdots,z_T \sim p_{\boldsymbol{\theta}}} \left[-R(y_1, \cdots, y_n \,|\, z_1, \cdots, z_T) \right]$$

Suppose we only have final reward

Otherwise, z_t is contributing to $R_t, \cdots, R_T$

$$\nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{z_1,\cdots,z_T} \left[-R \right]$$

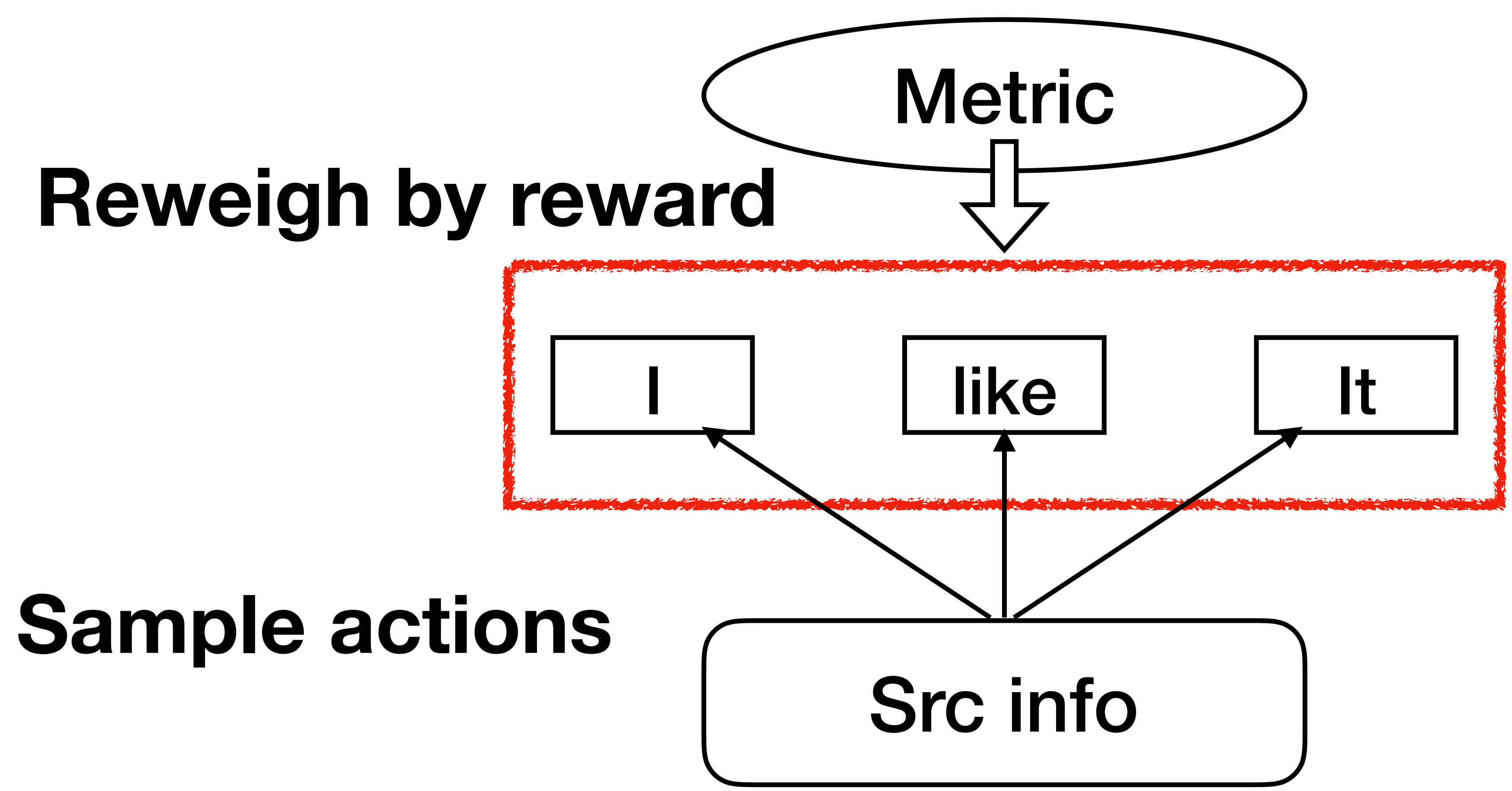
$$= \sum_{z_1,\cdots,z_T} \underline{\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(z_1, \cdots, z_T)} \cdot (-R)$$

$$= \underbrace{\sum_{z_1,\cdots,z_T} p_{\boldsymbol{\theta}}(z_1, \cdots, z_T)}_{\mathbb{E}} \underline{\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(z_1, \cdots, z_T)} \cdot (-R)$$

REINFORCE vs Supervised

- Sample a few sequences of actions
- Pretend that they are groundtrueh
- But reweigh it by (minus) reward

$$-\mathbb{E}_{\boldsymbol{z}}\left[R \cdot \nabla_{\boldsymbol{\theta}} \log p_{\theta}(\boldsymbol{z})\right]$$



High Variance of REINFORCE

$$-\mathbb{E}_{\boldsymbol{z}}\left[\boxed{R} \cdot \nabla_{\boldsymbol{\theta}} \log p_{\theta}(\boldsymbol{z})\right]$$

$$(R - B)$$

Baseline

- Mean
- Per-data mean
- $\hat{V}(s)$
  - Critic, which can be learned by $(R - V(s))^2$

RL vs MLE

Method	**Approximation of $E_q\left[\cdot\right]$**	**Exploration strategy**	**Gradient weight $q(\mathbf{z})$**
REINFORCE	Monte Carlo integration	independent sampling	$p_\theta(\mathbf{z} \mid x)$
BS-MML	numerical integration	beam search	$p_\theta(\mathbf{z} \mid x, R(\mathbf{z}) \neq 0)$
RANDOMER	numerical integration	randomized beam search	$q_\beta(\mathbf{z})$

Guu K, Pasupat P, Liu EZ, Liang P. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *ACL*, 2017.

Massage

maximize $$\log\left(\boxed{\sum_z p(z)} p(\boldsymbol{Y}\mid z,\boldsymbol{\theta})\right)$$

$\Downarrow$ generalize

maximize $$\underbrace{\sum_z p(z)}_{\mathbb{E}}\ R(Y(z))$$

$$\mathop{\mathbb{E}}_{z\sim p_{\boldsymbol{\theta}}(z)} R(Y(z))$$

$\Downarrow$ reparametrize

maximize $$\mathop{\mathbb{E}}_{\epsilon\in p(\epsilon)} J(Y(\ z_{\boldsymbol{\theta}}(\epsilon)\ ))$$

Gumbel-softmax

Reparametrization Trick

- **If** $z \sim p_{\boldsymbol{\theta}}(z) \iff \epsilon \sim p(\epsilon),\ z = f_{\boldsymbol{\theta}}(\epsilon)$
- And **if** f is a differentiable function w.r.t $\boldsymbol{\theta}$
- **Then** life would be much easier

Reparametrization Trick

- **If** $z \sim p_{\boldsymbol{\theta}}(z) \iff \epsilon \sim p(\epsilon),\ z = f_{\boldsymbol{\theta}}(\epsilon)$
- And **if** f is a differentiable function w.r.t $\boldsymbol{\theta}$
- **Then** life would be much easier

- Gaussian distribution

$$z \sim \mathcal{N}(\mu, \sigma) \iff \epsilon \sim \mathcal{N}(0,1),\ z = f_{\mu,\sigma}(\epsilon) = \mu + \sigma \cdot \epsilon$$

Reparametrization Trick

- **If** $z \sim p_{\boldsymbol{\theta}}(z) \iff \epsilon \sim p(\epsilon),\, z = f_{\boldsymbol{\theta}}(\epsilon)$
- And **if** f is a differentiable function w.r.t $\boldsymbol{\theta}$
- **Then** life would be much easier

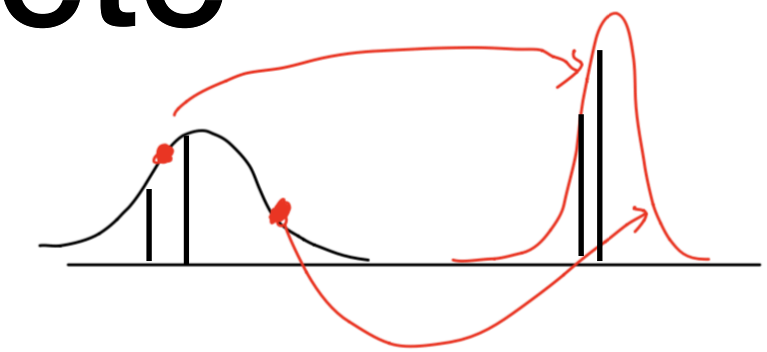
- Gaussian distribution

$$z \sim \mathcal{N}(\mu, \sigma) \iff \epsilon \sim \mathcal{N}(0,1),\, z = f_{\mu,\sigma}(\epsilon) = \mu + \sigma \cdot \epsilon$$

- This doesn't happen in the **discrete** case

Kingma DP, Welling M. Auto-encoding variational Bayes.
In *ICLR*, 2014.

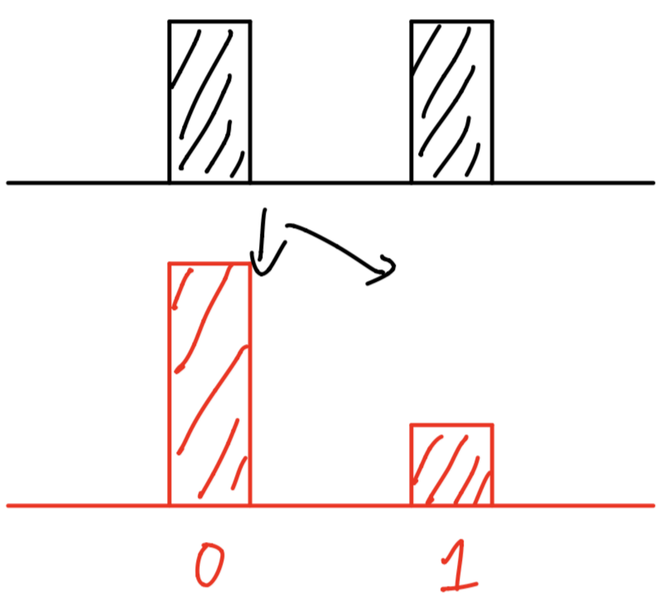
Continuous vs Discrete



- Closer look at continuous reparametrization

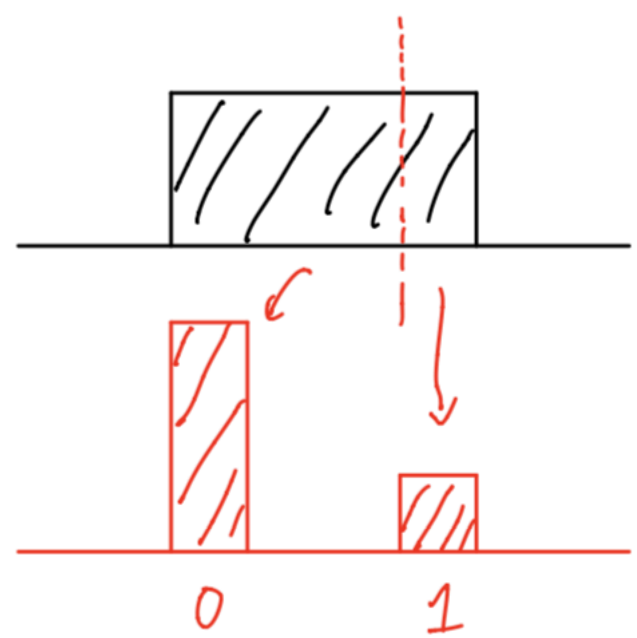
$$z \sim \mathcal{N}(\mu, \sigma) \iff \epsilon \sim \mathcal{N}(0,1),\ z = f_{\mu,\sigma}(\epsilon) = \mu + \sigma \cdot \epsilon$$

- Discrete $\longrightarrow$ Discrete



Infeasible in general

- Continuous $\longrightarrow$ Discrete



f =CDF^{-1} **not differentiable**

Reparametrization is still feasible

- Gumbel-max

$$z \sim \text{one\_hot}[\text{Cat}(\pi_1, \pi_2, \cdots, \pi_n)]$$

$$\Updownarrow$$

$$z = \text{one\_hot}\left[\ \underset{i \in \{1,\cdots,n\}}{\arg\max}\{g_i + \log \pi_i\}\ \right]$$

$$g_i \sim \text{Gumbel}(0,1) \quad \Leftrightarrow \quad g = -\log(-\log(u)), u \sim U(0,1)$$

Gumbel EJ. Statistical theory of extreme values and some practical applications: a series of lectures. US Government Printing Office; 1948.

Reparametrization is still feasible

- Gumbel-max

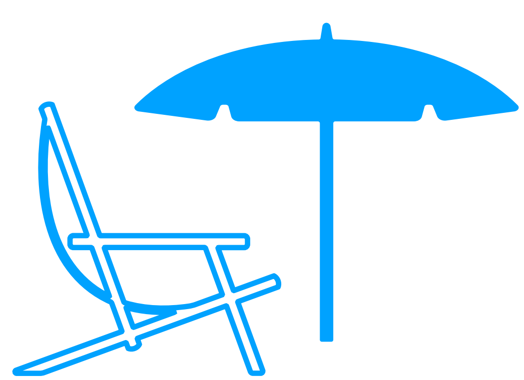
$$z \sim \text{one\_hot}[\text{Cat}(\pi_1, \pi_2, \cdots, \pi_n)]$$

$$\Updownarrow$$

$$z = \text{one\_hot}\left[\underset{i \in \{1,\cdots,n\}}{\arg\max} \{g_i + \log \pi_i\} \right]$$

$$g_i \sim \text{Gumbel}(0,1) \quad \Leftrightarrow \quad g = -\log(-\log(u)), u \sim U(0,1)$$

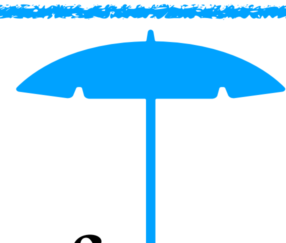
- Gumbel-max itself doesn't help much
- But we can **relax**



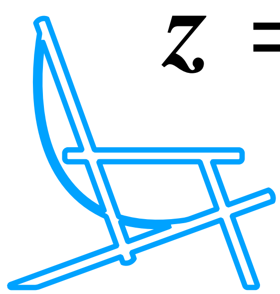
Gumbel-Softmax

$$g = -\log(-\log(u)), u \sim U(0,1)$$

$$z = \text{one\_hot}\left[\underset{i \in \{1,\cdots,n\}}{\arg\max} \{g_i + \log \pi_i\} \right]$$



$$\tilde{z} = \underset{i \in \{1,\cdots,n\}}{\text{softmax}} \{(g_i + \log \pi_i)/\tau\}$$

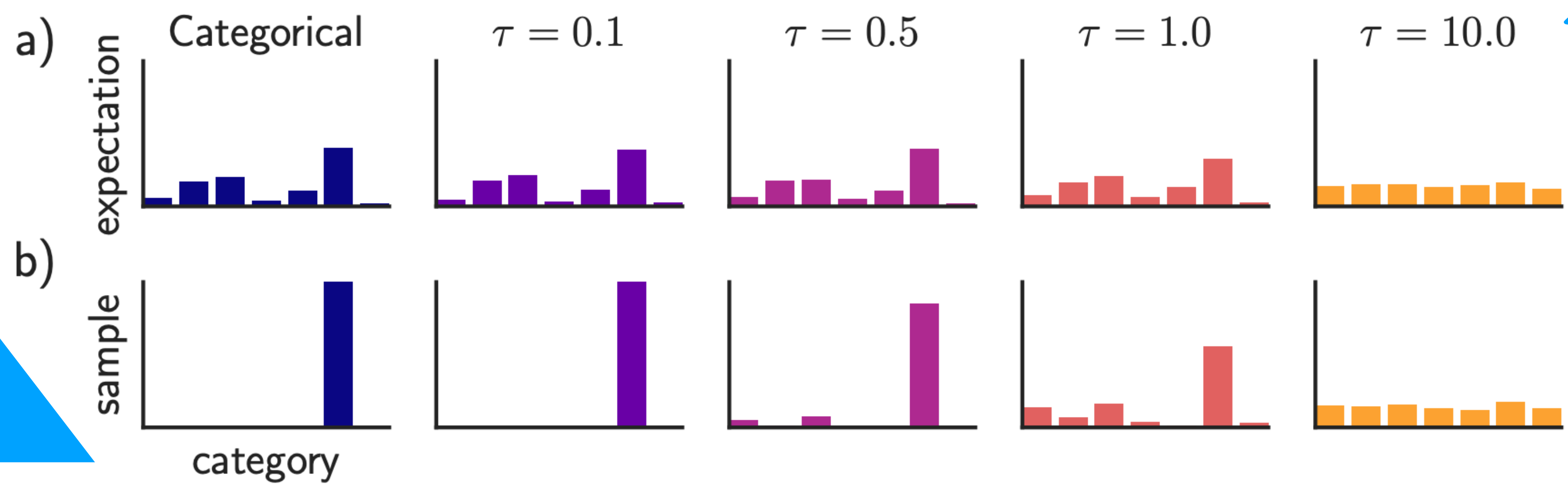


Jang E, Gu S, Poole B. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017.

Gumbel-Softmax

$$z = \text{one\_hot}\left[\ \underset{i\in\{1,\cdots,n\}}{\arg\max}\{g_i + \log \pi_i\}\ \right]$$

$$\tilde{z} = \underset{i\in\{1,\cdots,n\}}{\text{softmax}}\{g_i + \log \pi_i\}$$



- Interpolation between one-hot **sample** and uniform
- Interpolation considers distribution info

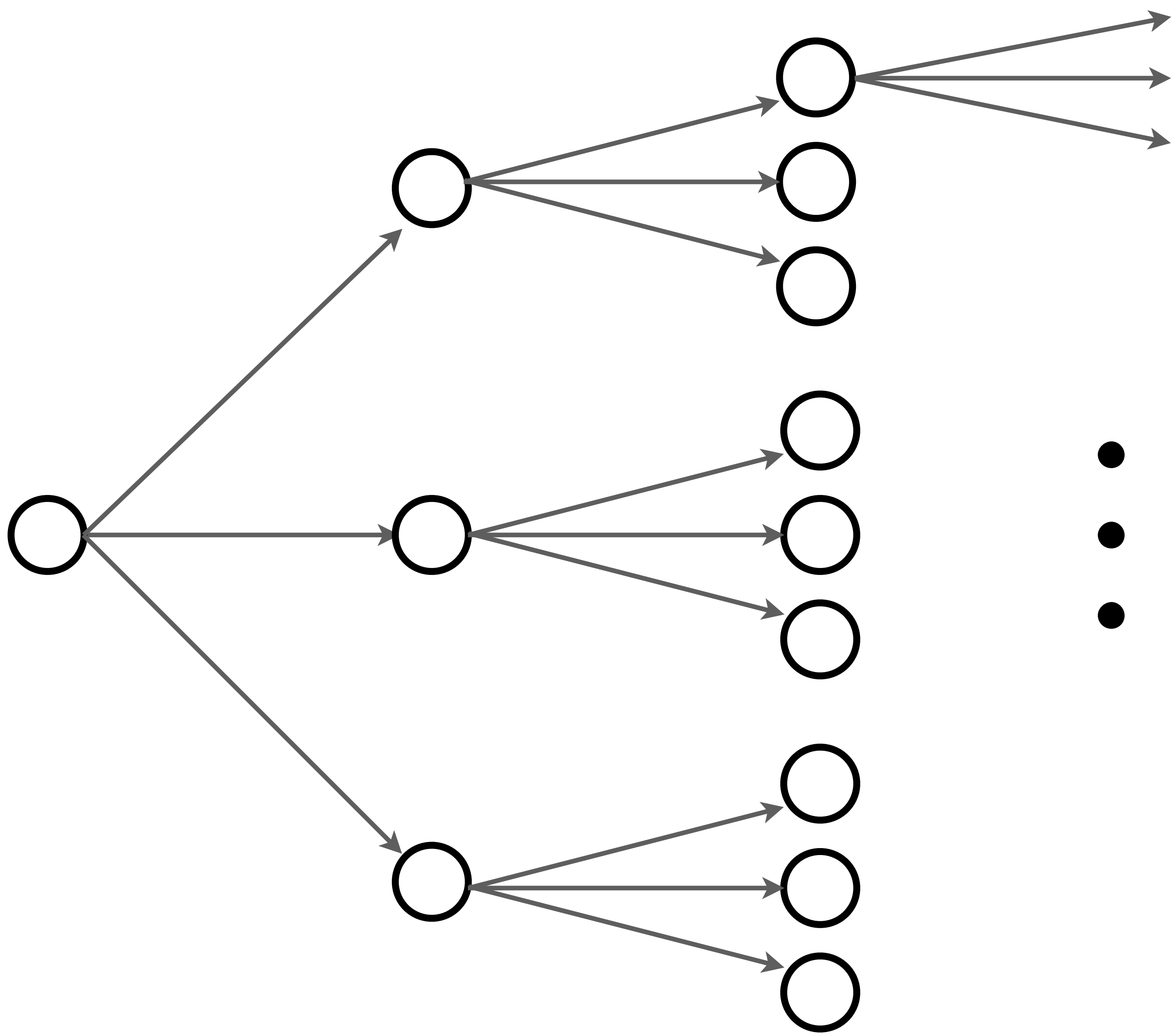
Gumbel-Softmax in NN

$$z = \text{one\_hot}\left[\underset{i \in \{1,\cdots,n\}}{\arg\max} \{ g_i + \log \pi_i \} \right]$$

$$\tilde{z} = \underset{i \in \{1,\cdots,n\}}{\text{softmax}} \{ g_i + \log \pi_i \}$$

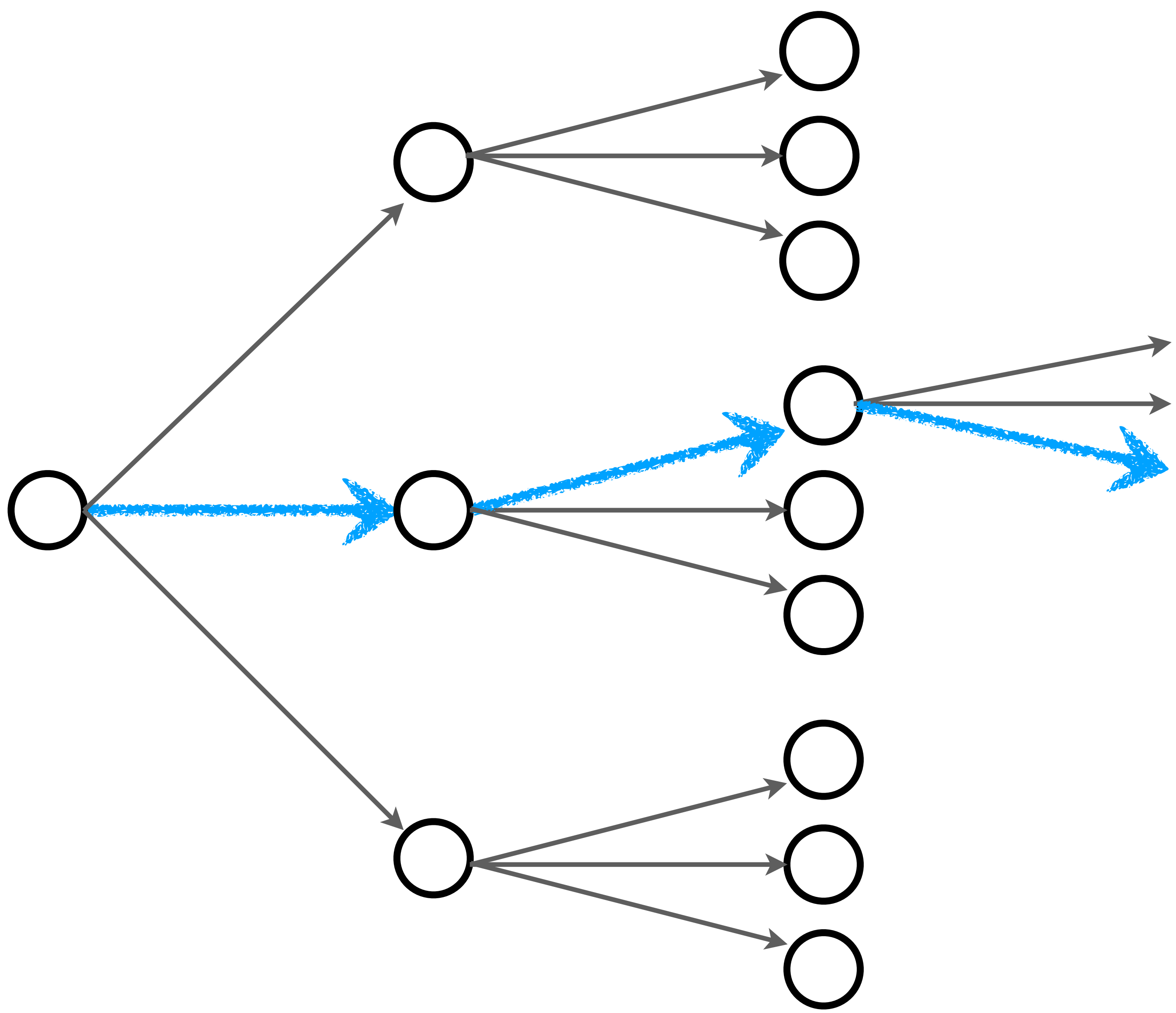
- **Straight-through Gumbel-softmax**
  - Forward prop: Sample **one** action
  - Backward prop: Relax by $\tilde{z}$
- **Gumbel-softmax**
  - Both forward/backprop relaxed by $\tilde{z}$

Exponential Search Space



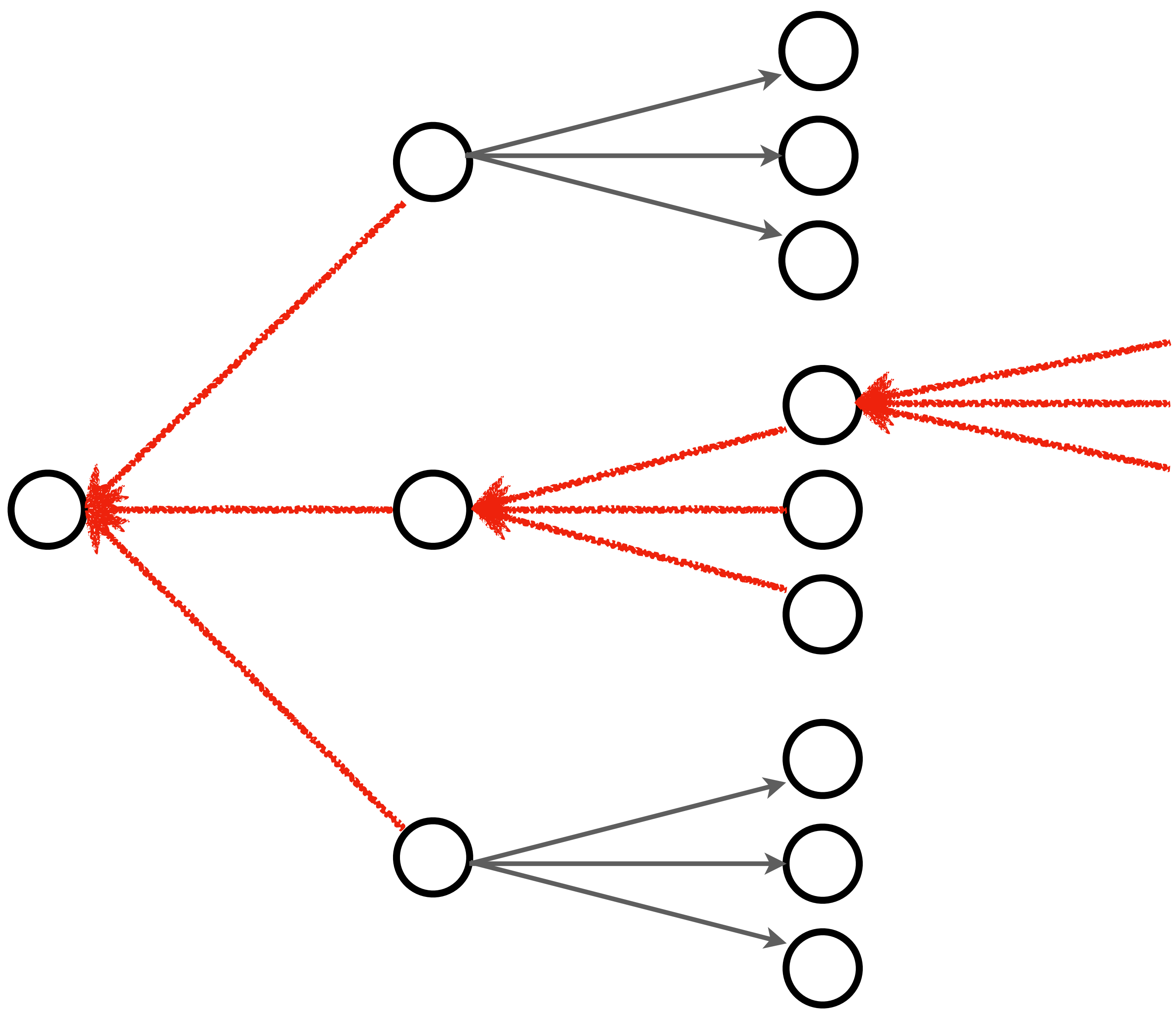
- Single discrete variable is not too bad
- But, space $\propto \exp(\text{ step })$

Exponential Search Space



- Gumbel-softmax straight-through (ST)
  - Forward: sample one action
  - Backward: Relax by Gumbel-softmax

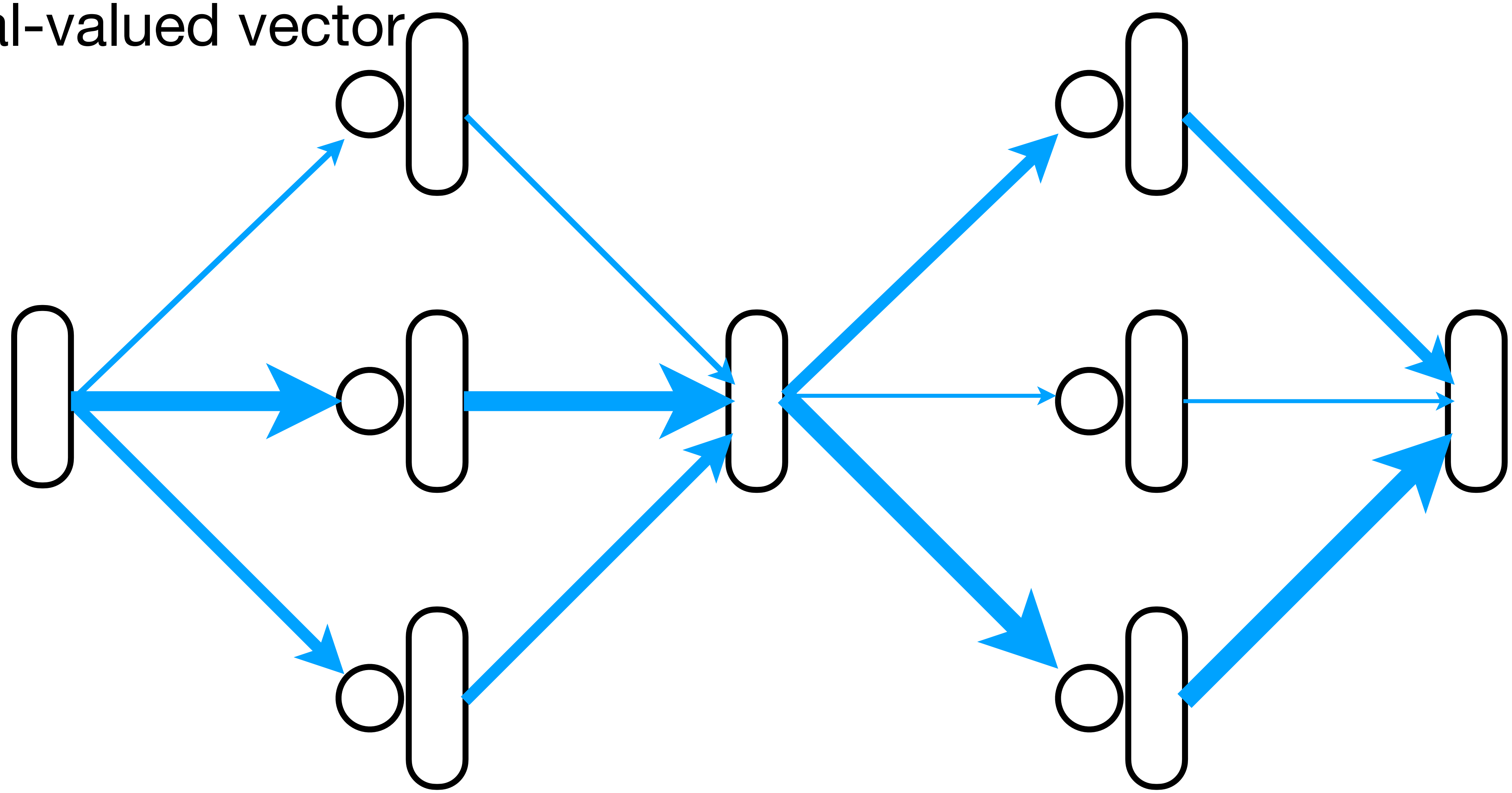
Exponential Search Space



- Gumbel-softmax straight-through (ST)
  - Forward: Sample one action
  - Backward: Relax by Gumbel-softmax

Exponential Search Space

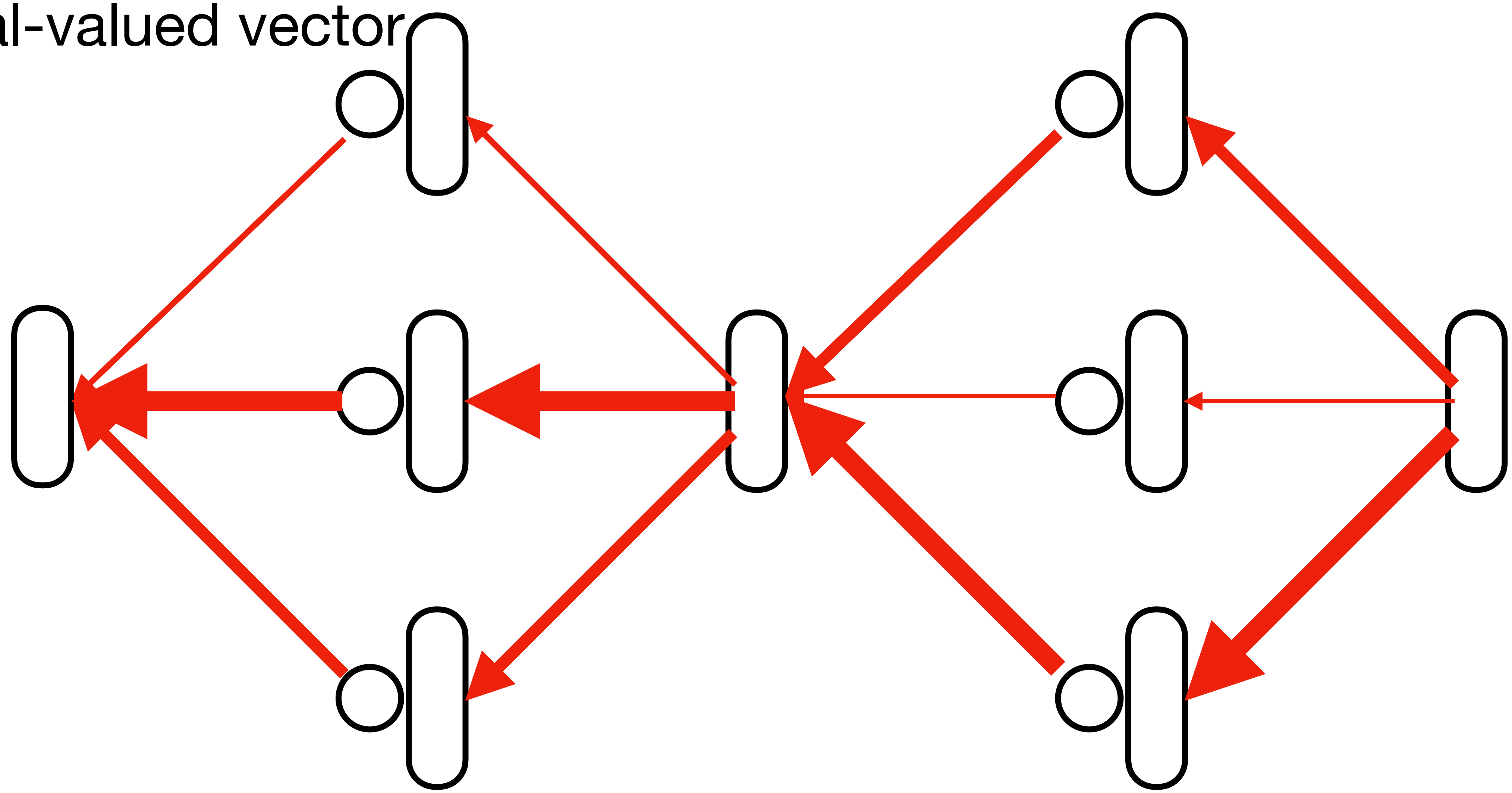
Discrete actions represented by
real-valued vector



- Gumbel softmax (non-ST)
  - Forward: Relax
  - Backward: Relax

Exponential Search Space

Discrete actions represented by real-valued vector



- Gumbel softmax (non-ST)
  - Forward: Relax
  - Backward: Relax

Gumbel vs. RL

Provable Mostly empirical

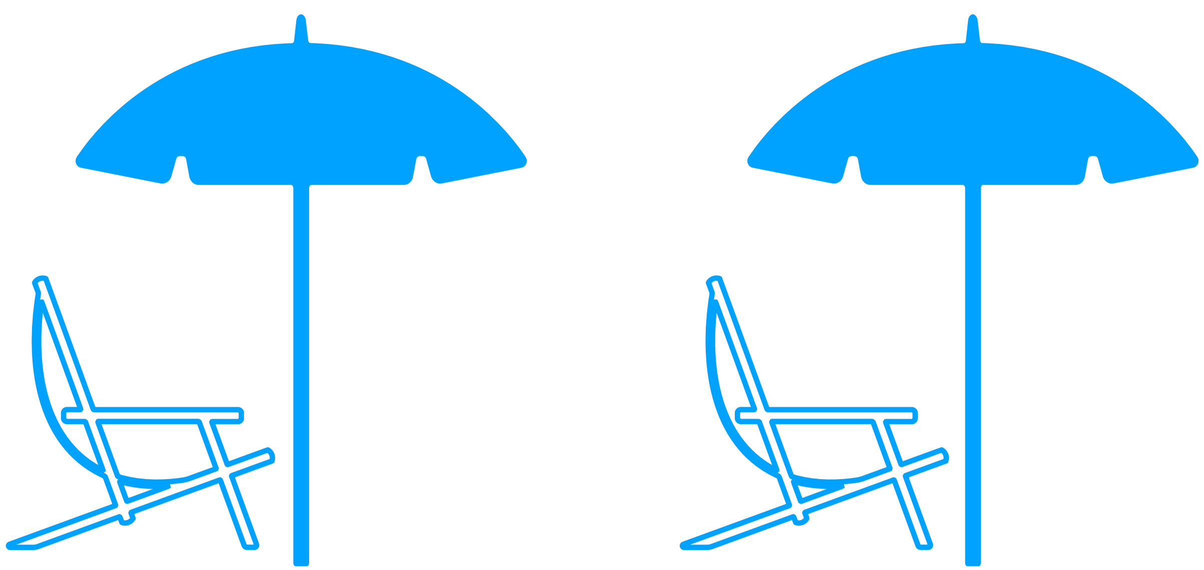
- **RL: unbiased**, high variance
  - Works with any reward (theoretically)
- **Gumbel: biased**, low variance (still involves sampling)
  - Works with differentiable loss

Gumbel vs. RL

Provable Mostly empirical

- **RL: unbiased**, high variance
  - Works with any reward (theoretically)
- **Gumbel: biased**, low variance (still involves sampling)
  - Works with differentiable loss

- **We may relax more**



Massage

maximize $$\log\left(\boxed{\sum_z p(z)}p(\boldsymbol{Y} \mid z, \boldsymbol{\theta})\right)$$

$\Downarrow$ generalize

maximize $$\mathop{\mathbb{E}}_{z \sim p_{\boldsymbol{\theta}}(z)} R(Y(z))$$

reparametrize $\Downarrow$

maximize $$\mathop{\mathbb{E}}_{\epsilon \in p(\epsilon)} J(Y(\nabla z_{\boldsymbol{\theta}}(\epsilon)))$$

relax

Massage

maximize $\log\left(\boxed{\sum_z p(z)} p(\boldsymbol{Y} \mid z, \boldsymbol{\theta})\right)$

$\Downarrow$ generalize

maximize $\boxed{\mathop{\mathbb{E}}_{z \sim p_{\boldsymbol{\theta}}(z)}} R(Y(z))$

reparametrize $\Downarrow$

maximize $\mathop{\mathbb{E}}_{\epsilon \in p(\epsilon)} J(Y(\nabla z_{\boldsymbol{\theta}}(\epsilon)))$

relax

maximize $J(Y(\mathbb{E}_{z \sim p_{\boldsymbol{\theta}}(z)}[\boldsymbol{z}]))$

Step-by-step Attention

Attention

- Your current querying state $\boldsymbol{q}$
- $z \in \{1, \cdots, n\}$: n discrete actions
  - Each could be represented as a continuous vector $\boldsymbol{z}_i$
- Attention mechanism

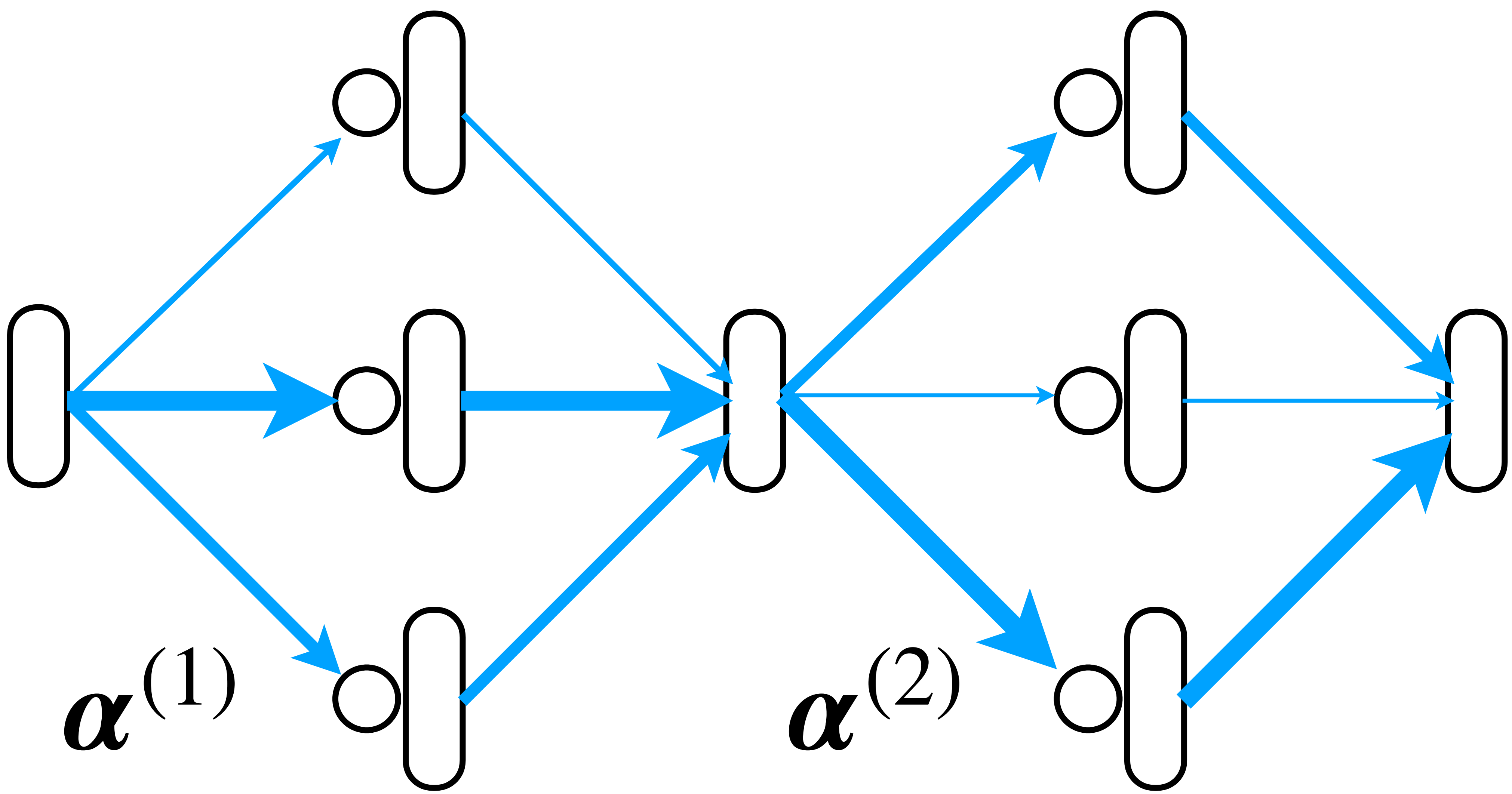
Unnormalized measure $\quad \tilde{\alpha}_i = \exp\{s(\boldsymbol{q}, \boldsymbol{z}_i)\}$

Attention probability $\quad \alpha_i = \dfrac{\tilde{\alpha}_i}{\sum_j \tilde{\alpha}_j}$

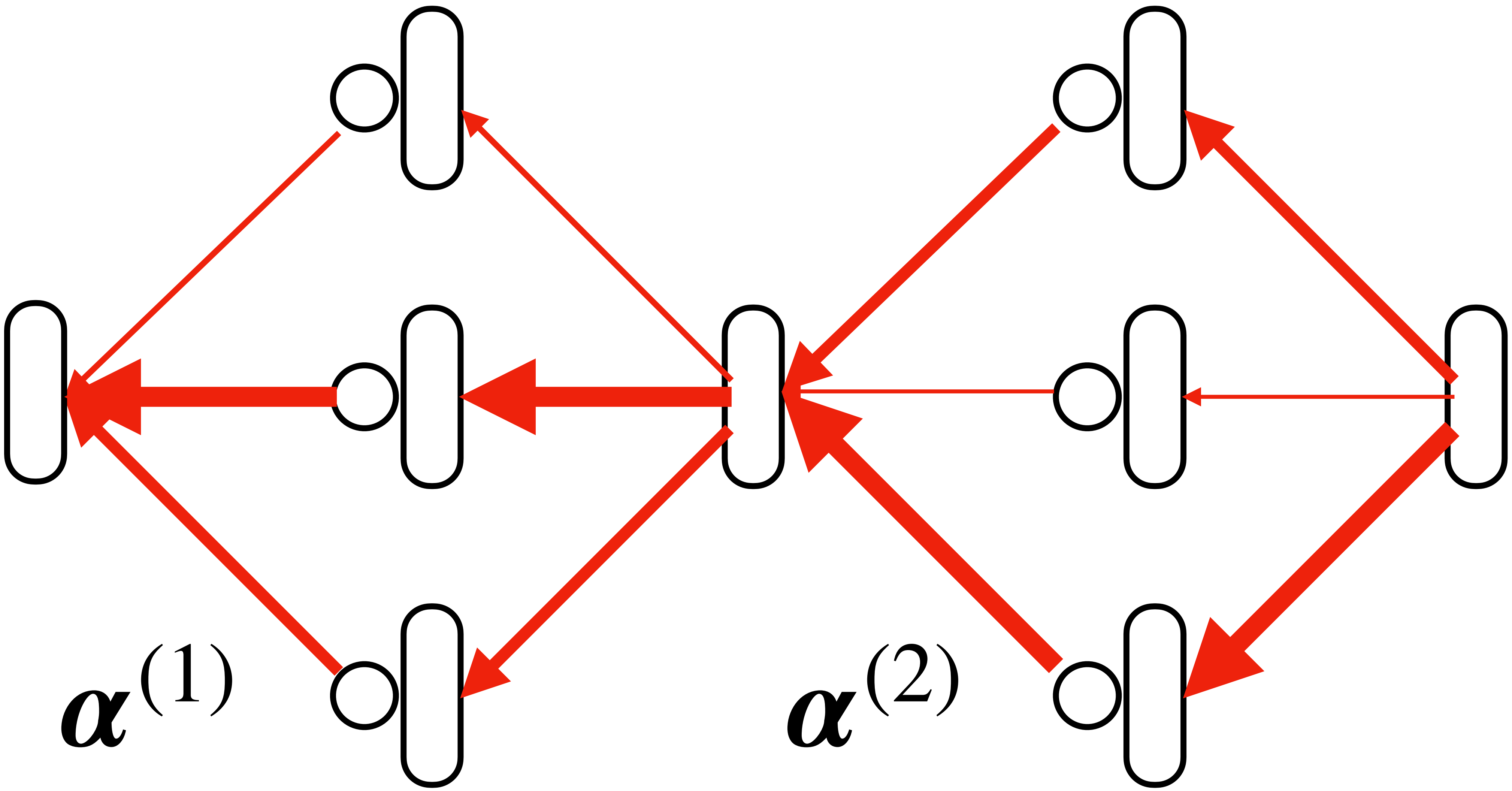
Attention content $\quad \boldsymbol{c} = \sum_i \alpha_i \boldsymbol{z}_i$

Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015

Step-by-step Attention



Step-by-step Attention



Attention vs Gumbel softmax

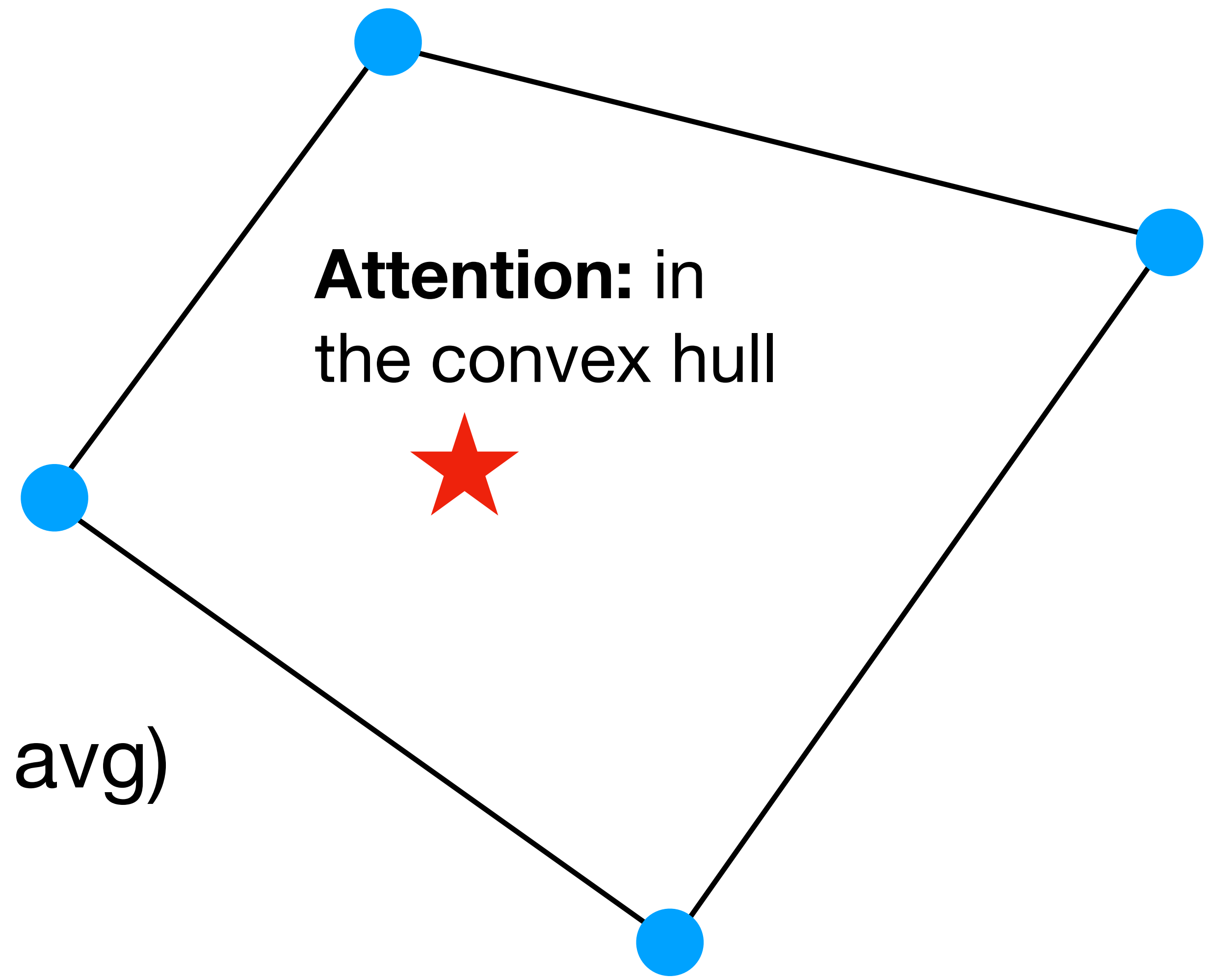
- Both relaxing **hard action** with **soft probability**
  - Attention: Directly using **predicted probability**
  - Gumbel: Using Gumbel-softmax distribution
    - Interpolation between **one-hot sample** and **uniform**
    - during which **predicted probability** is considered

Pros & Cons of Attention

- Pros
  - Easy to use and understand
  - No sampling is involved

Pros & Cons of Attention

- Pros
  - Easy to use and understand
  - No sampling is involved
- Cons
  - Landed in no-man's land (mode avg)

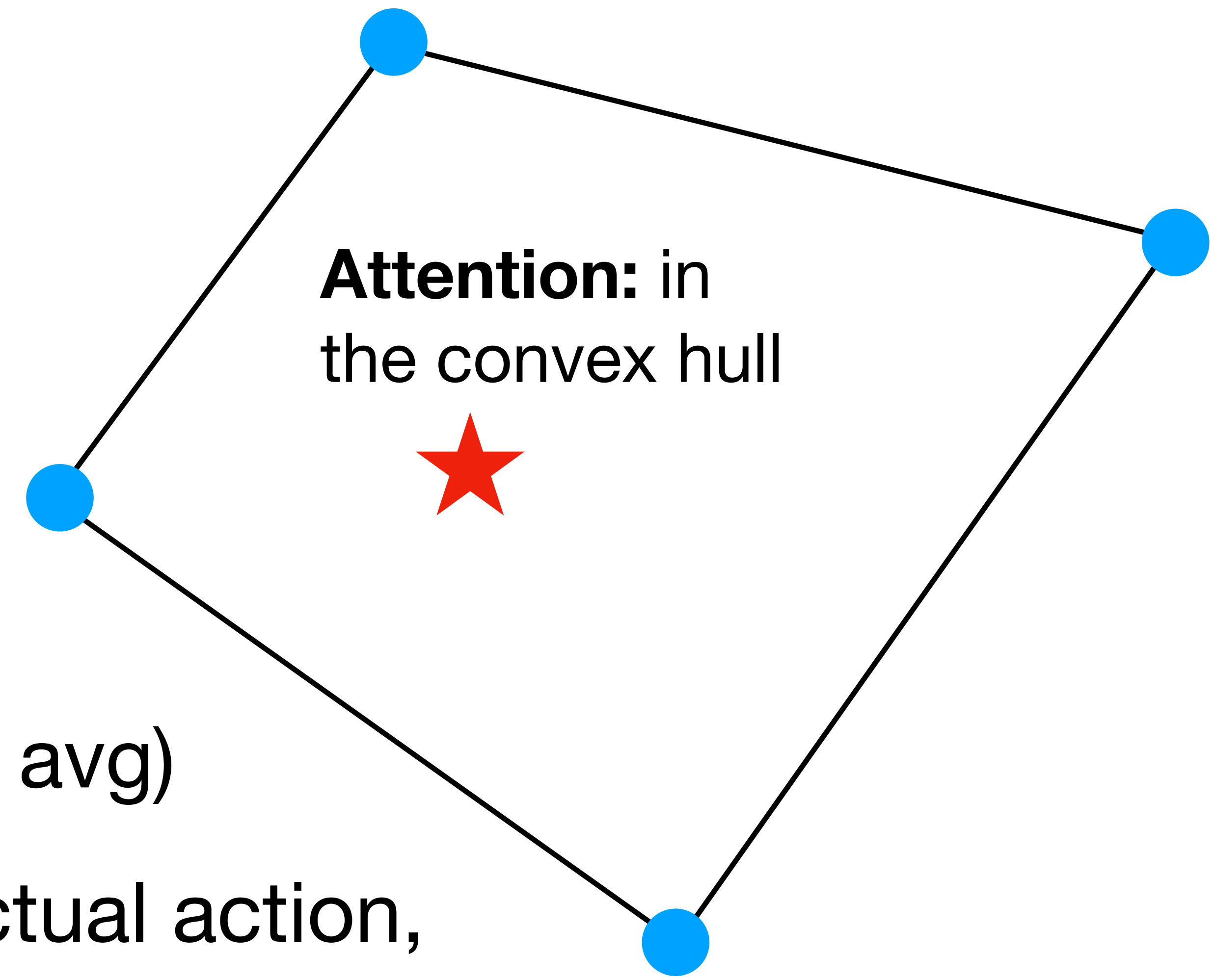


Pros & Cons of Attention

- Pros
  - Easy to use and understand
  - No sampling is involved
- Cons
  - Landed in no-man's land (mode avg)
    - If you **don't** care about the actual action,

      It's fine 😇

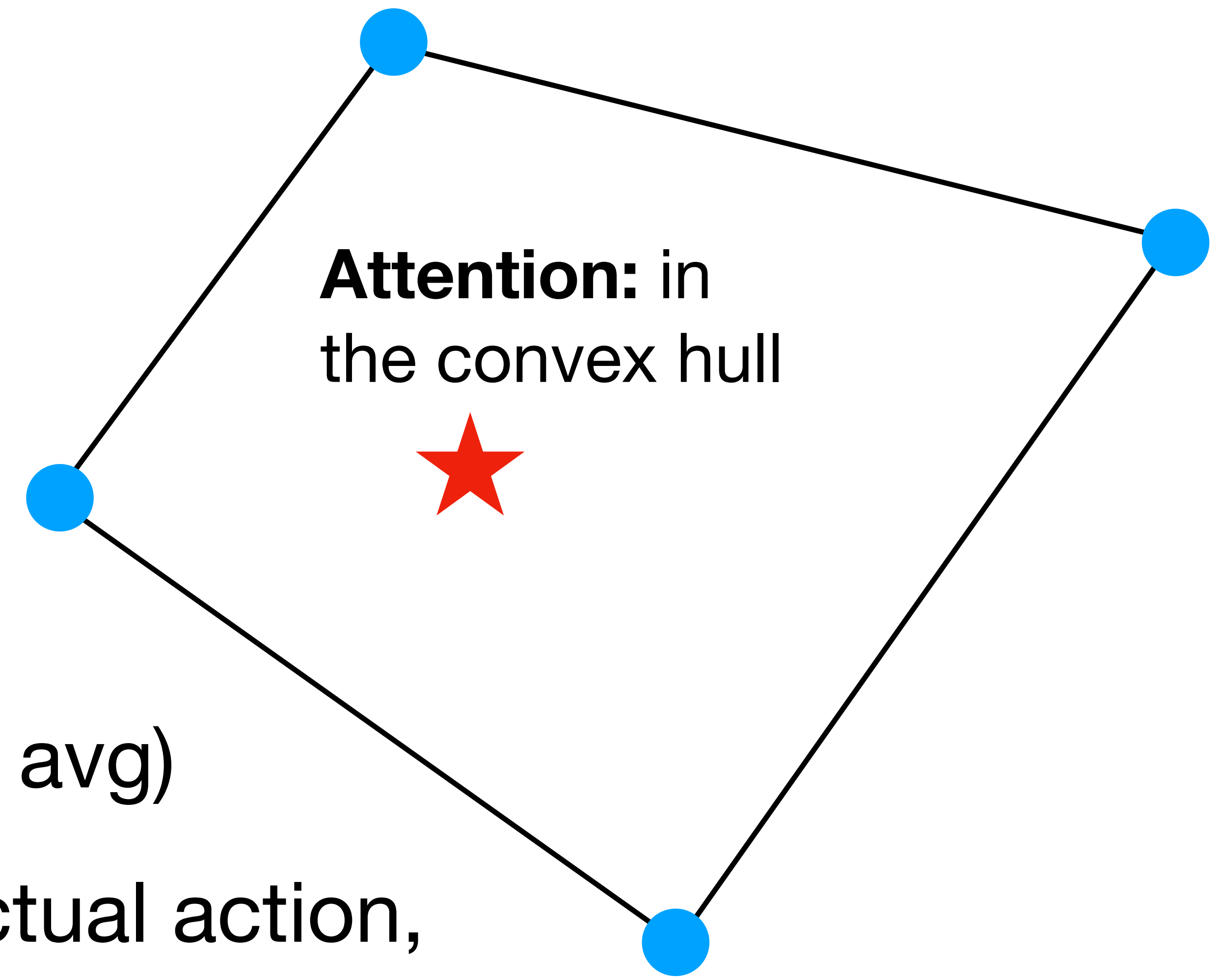
      - E.g., attentions in Transformer are all soft



Pros & Cons of Attention

- Pros
  - Easy to use and understand
  - No sampling is involved
- Cons
  - Landed in no-man's land (mode avg)
    - If you **don't** care about the actual action,

      It's fine 😇



This is not too wrong.
"Meaning is use" —Wittgenstein

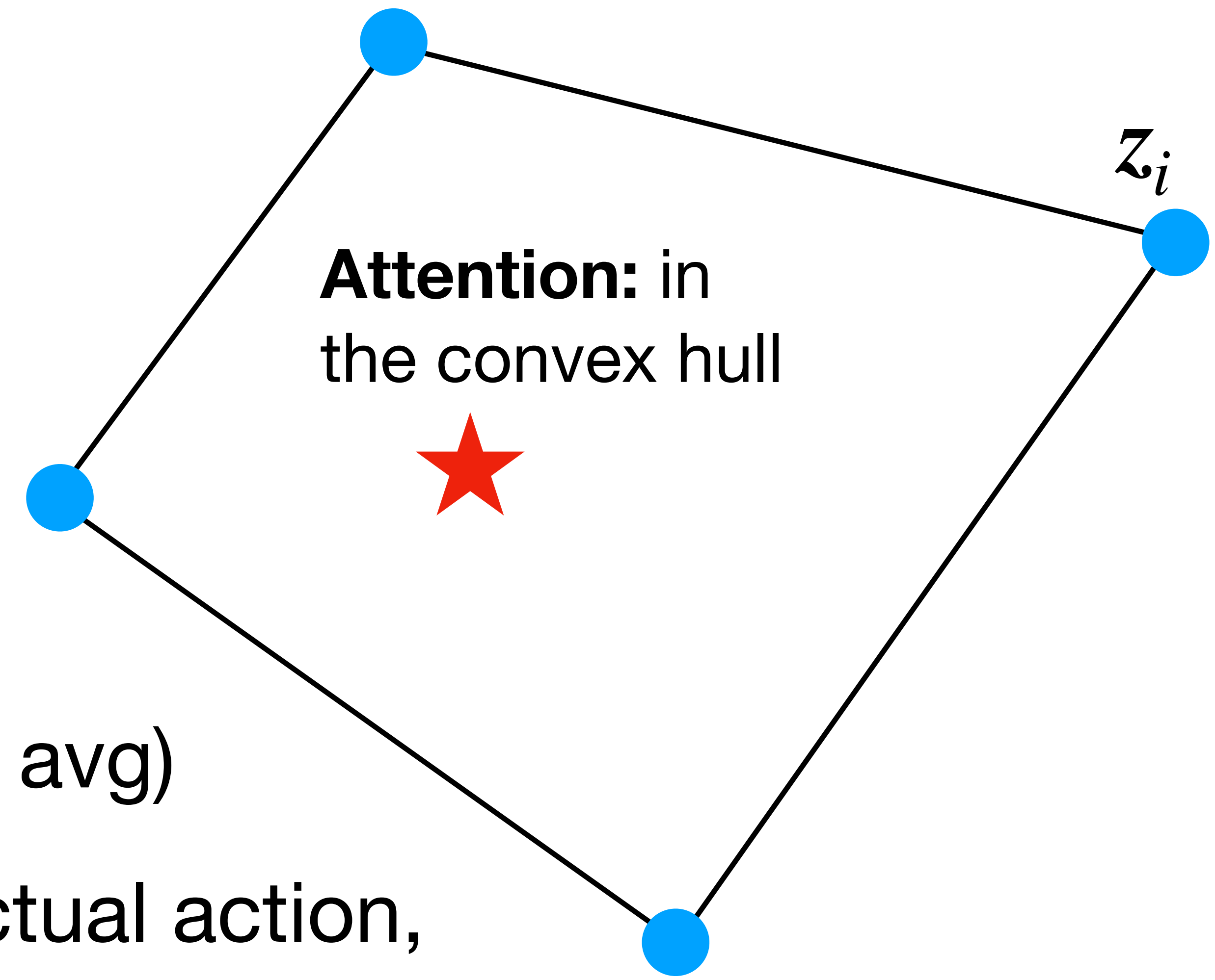
In machine learning,
how you train is how you predict

Pros & Cons of Attention

- Pros
  - Easy to use and understand
  - No sampling is involved
- Cons
  - Landed in no-man's land (mode avg)
    - If you **don't** care about the actual action,

      It's fine 😇
    - If you **do** care about the actual action,

      Discrepancy between training and prediction

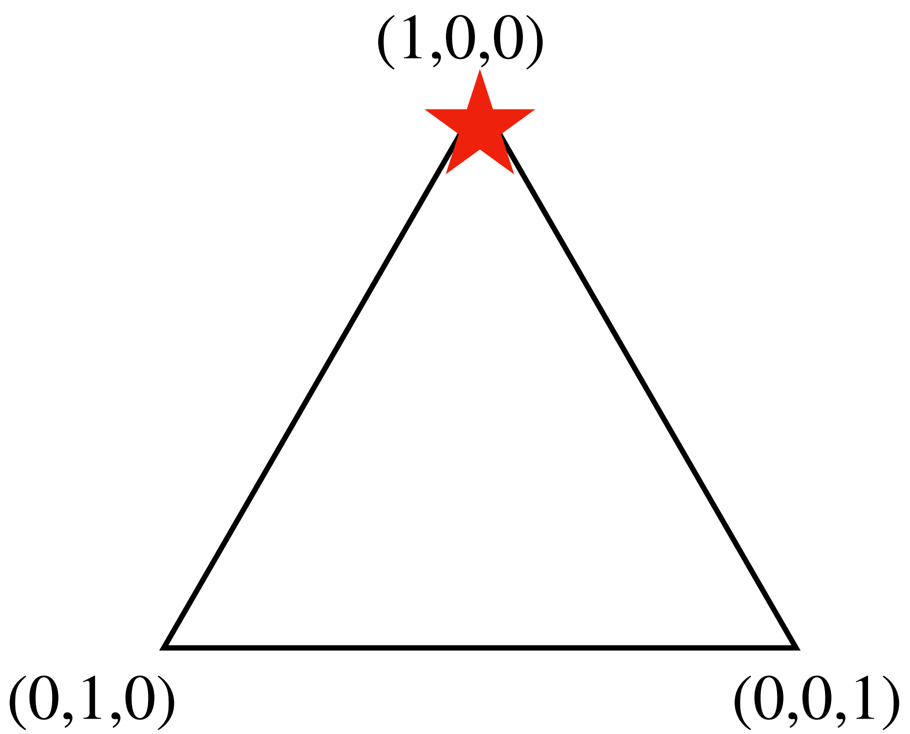


More Treatments of the Simplex

- Argmax

$$\boldsymbol{\alpha} = \mathrm{argmax}_{\boldsymbol{\alpha} \in \boldsymbol{\Delta}} \boldsymbol{s}^T \boldsymbol{\alpha}$$

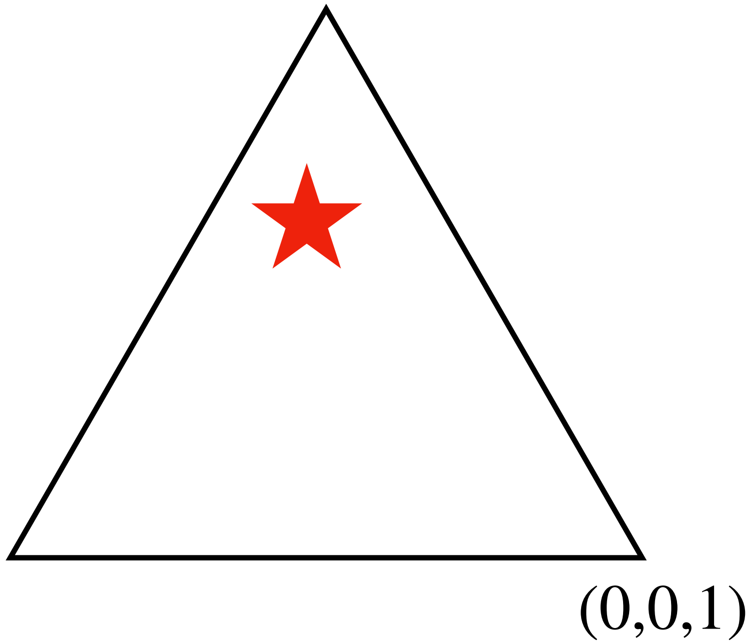
- Choose the largest element of $\boldsymbol{s}$
- Result in one-hot $\boldsymbol{\alpha}$ (assuming no ties)



More Treatments of the Simplex

- Softmax

$$\boldsymbol{\alpha} = \frac{\exp\{\boldsymbol{s}\}}{\sum_i \exp\{s_i\}}$$
$$= \operatorname{argmax}_{\boldsymbol{\alpha} \in \boldsymbol{\Delta}} \boldsymbol{s}^\top \boldsymbol{\alpha} + \mathscr{H}(\boldsymbol{\alpha})$$

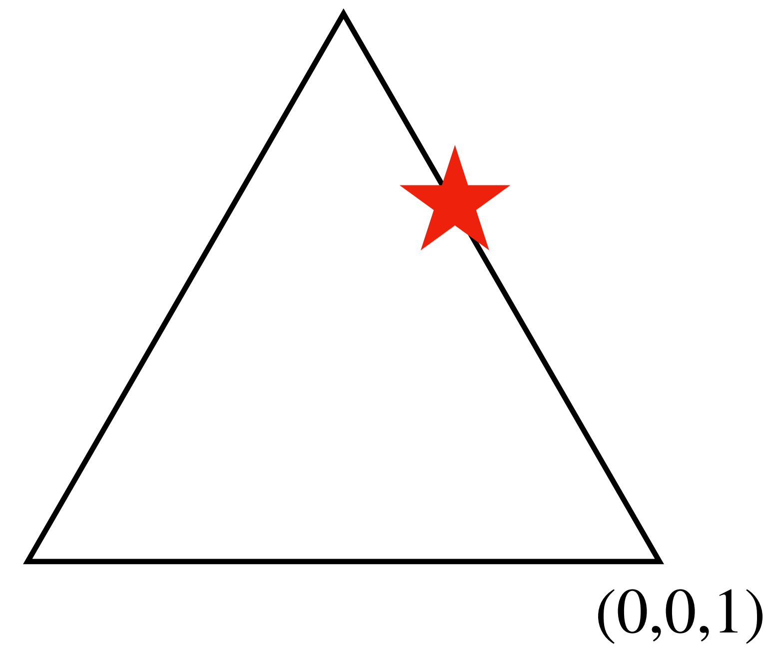


- Always dense

More Treatments of the Simplex

- Sparsemax

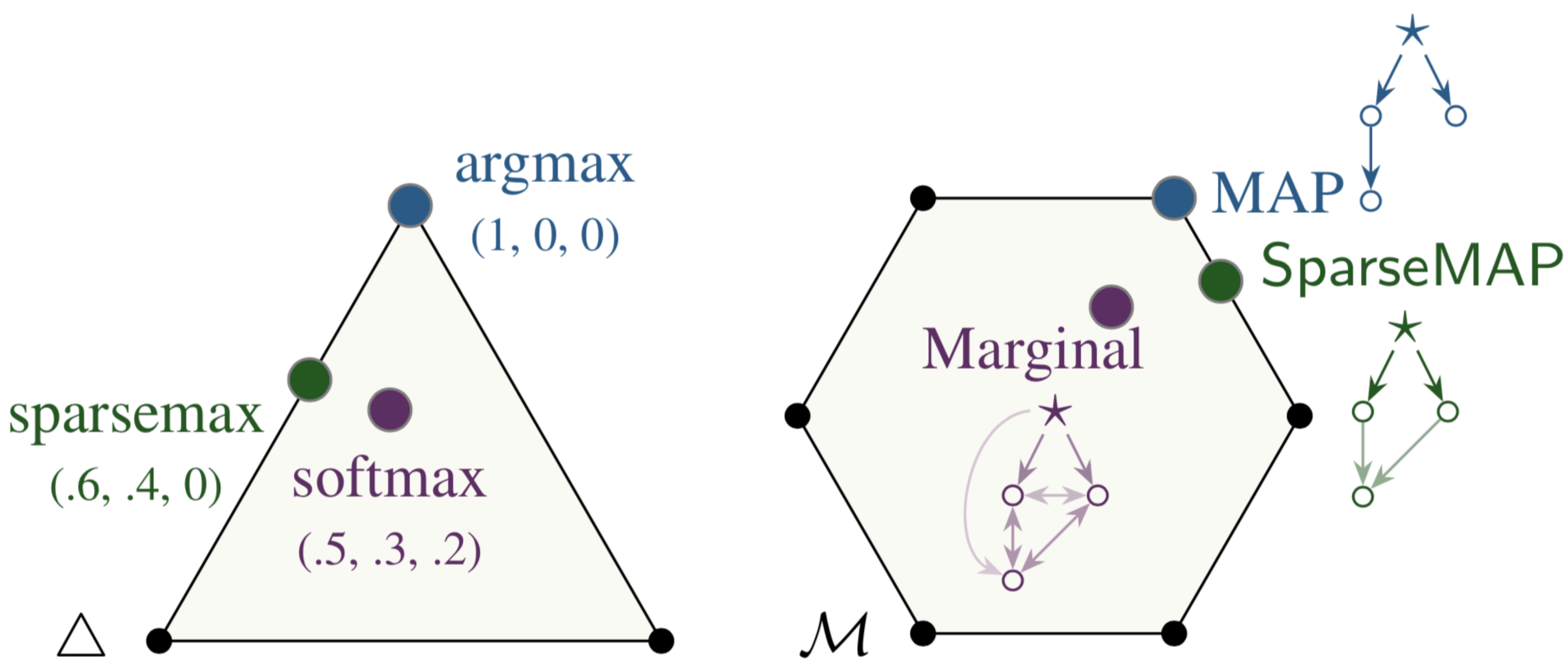
$$\boldsymbol{\alpha} = \mathrm{argmax}_{\boldsymbol{\alpha} \in \boldsymbol{\Delta}} \, \boldsymbol{s}^\top \boldsymbol{\alpha} - \frac{1}{2} \|\boldsymbol{\alpha}\|^2$$



- Denser than argmax
- Sparser than softmax

Martins, A. and Astudillo, R., June. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, 2016.

Extending Simplex to Polytope



- Structured prediction
  - A set of latent variables
  - Log-linear model on the set of (latent) variables

Niculae, V., Martins, A.F., Blondel, M. and Cardie, C. SparseMAP: Differentiable sparse structured inference. In *ICML*, 2018.

Massage

maximize $\log\left(\boxed{\sum_z p(z)}\, p(\boldsymbol{Y} \mid z, \boldsymbol{\theta})\right)$

generalize

maximize $\boxed{\mathbb{E}_{z \sim p_{\boldsymbol{\theta}}(z)}} R(Y(z))$

reparametrize

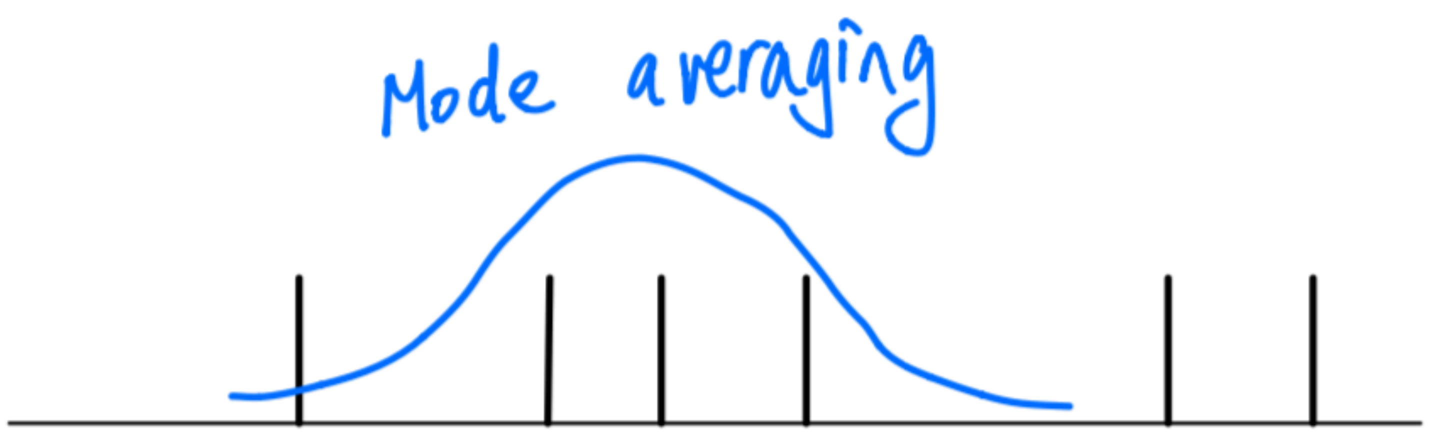
maximize $\mathbb{E}_{\epsilon \in p(\epsilon)} J(Y(z_{\boldsymbol{\theta}}(\epsilon)))$

∇ relax

maximize $J(Y(\mathbb{E}_{z \sim p_{\boldsymbol{\theta}}(z)}[\boldsymbol{z}]))$

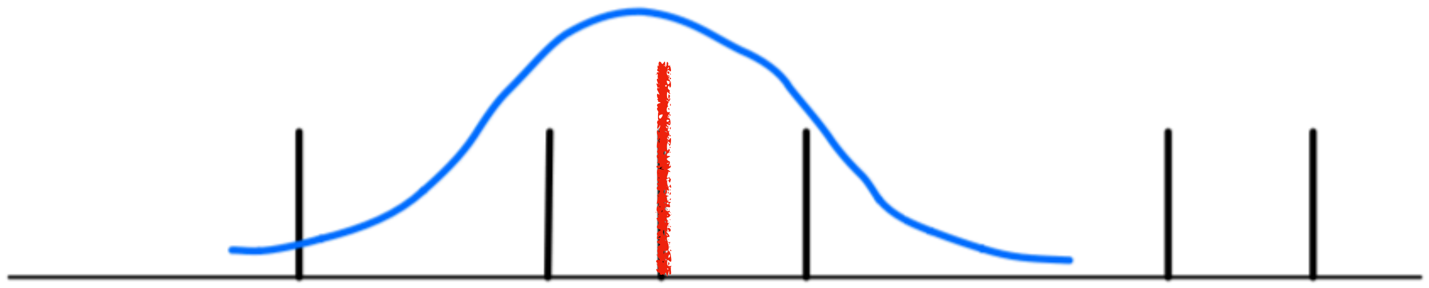
Combining Mode Avg/Sampling

- First, do mode averaging
  - Exploring all modes simultaneously
  - Having a general sense of the search space
- Then, do mode sampling
  - To learn more accurate actions



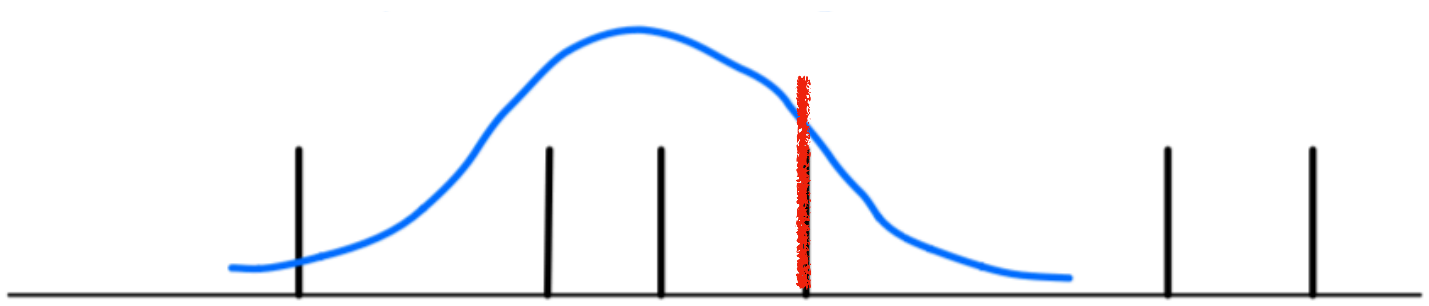
Combining Mode Avg/Sampling

- First, do mode averaging
  - Exploring all modes simultaneously
  - Having a general sense of the search space
- Then, do mode sampling
  - To learn more accurate actions



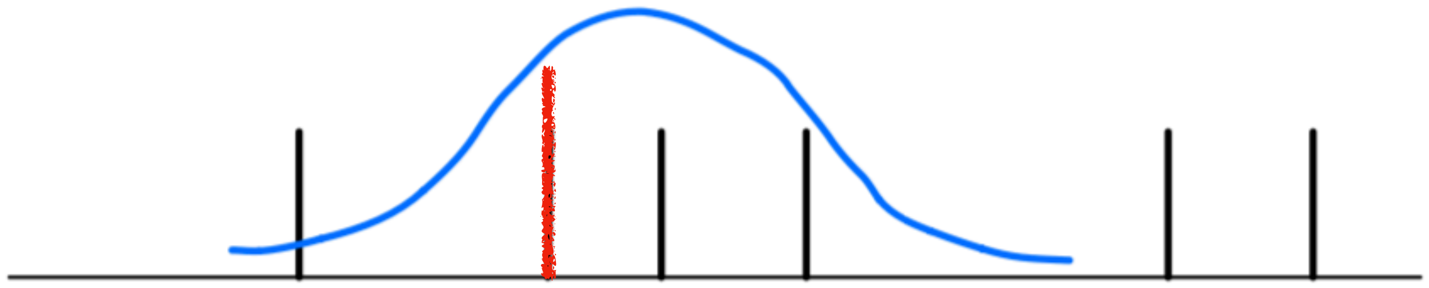
Combining Mode Avg/Sampling

- First, do mode averaging
  - Exploring all modes simultaneously
  - Having a general sense of the search space
- Then, do mode sampling
  - To learn more accurate actions



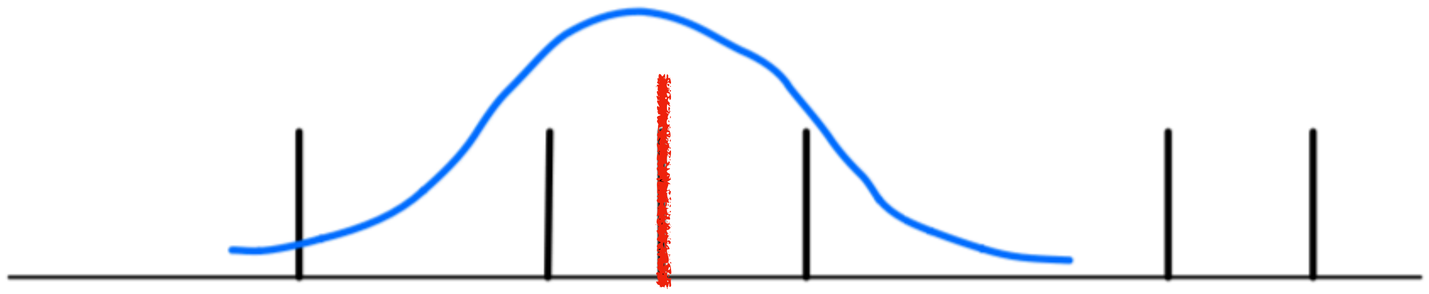
Combining Mode Avg/Sampling

- First, do mode averaging
  - Exploring all modes simultaneously
  - Having a general sense of the search space
- Then, do mode sampling
  - To learn more accurate actions



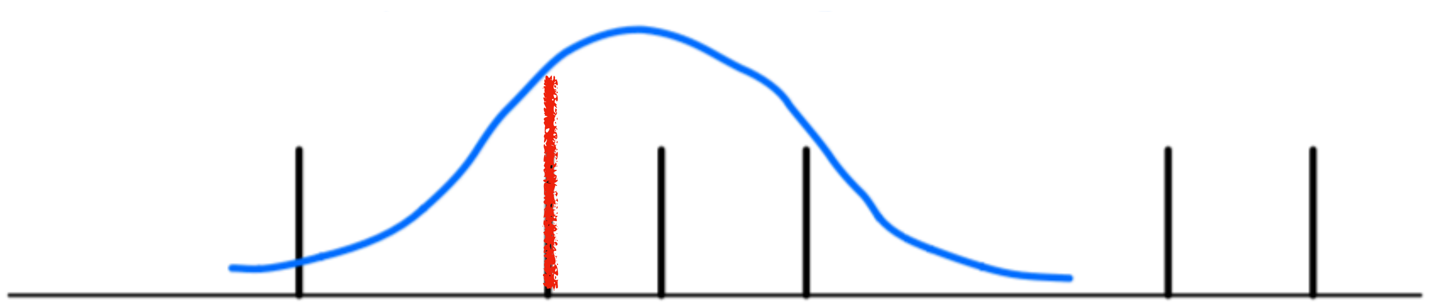
Combining Mode Avg/Sampling

- First, do mode averaging
  - Exploring all modes simultaneously
  - Having a general sense of the search space
- Then, do mode sampling
  - To learn more accurate actions



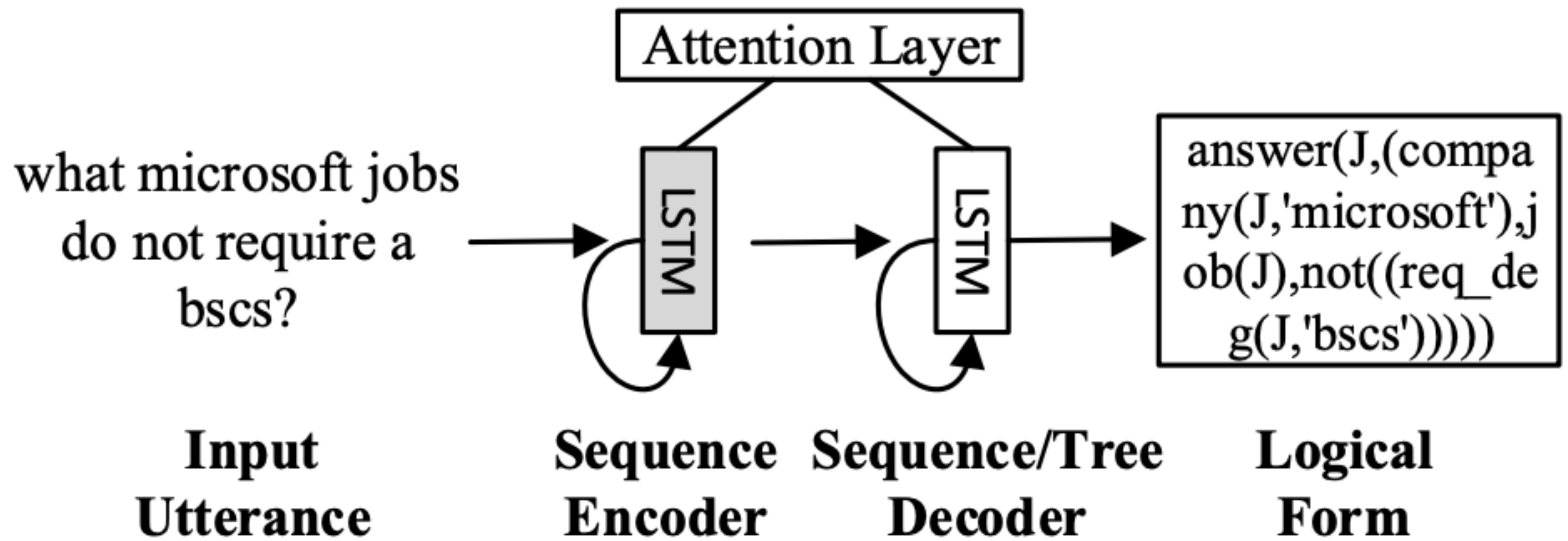
Combining Mode Avg/Sampling

- First, do mode averaging
  - Exploring all modes simultaneously
  - Having a general sense of the search space
- Then, do mode sampling
  - To learn more accurate actions



Application: Semantic Parsing

Semantic Parsing

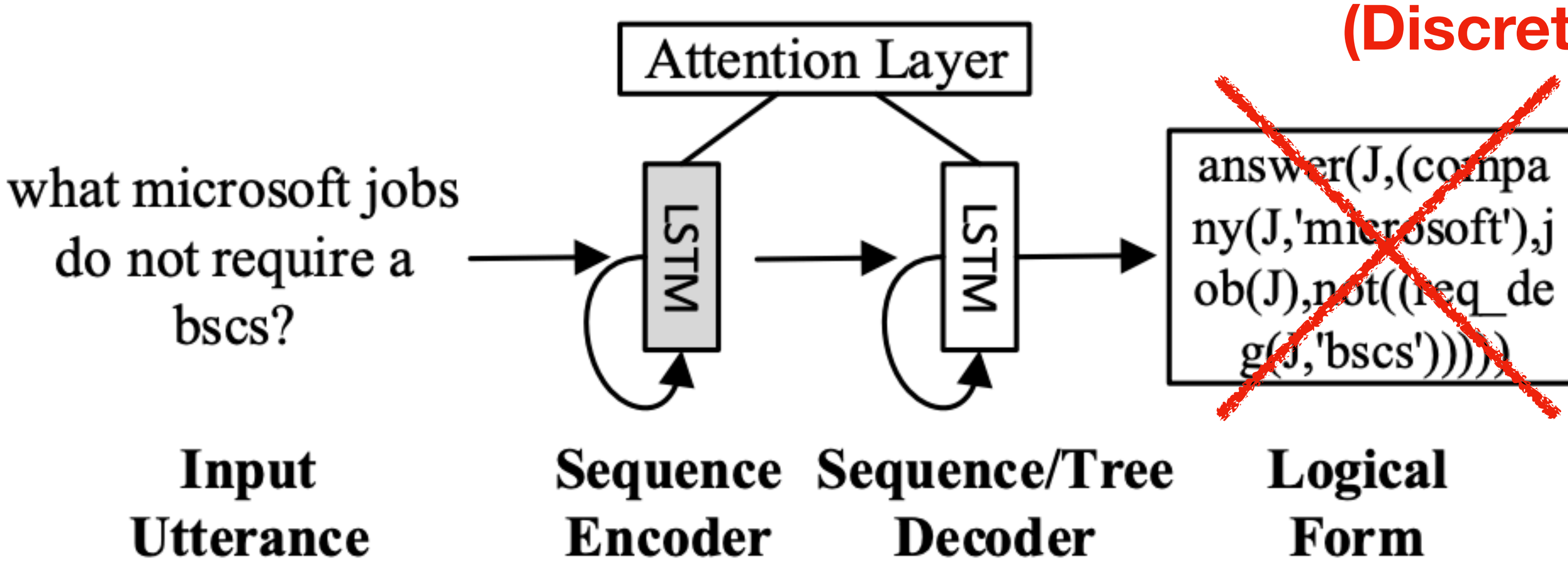


- Fully supervised setting:
  - Input natural language, and
  - Output logical forms
- Both are known during training

Dong, Li, and Mirella Lapata. Language to logical form with neural attention. In *ACL*, 2016.

Weakly Supervised setting

Unknown during training
(Discrete latent space)



Company	Req	Loc	$
A			
B			
C			

Supervision Signal: Result is Correct/Incorrect?

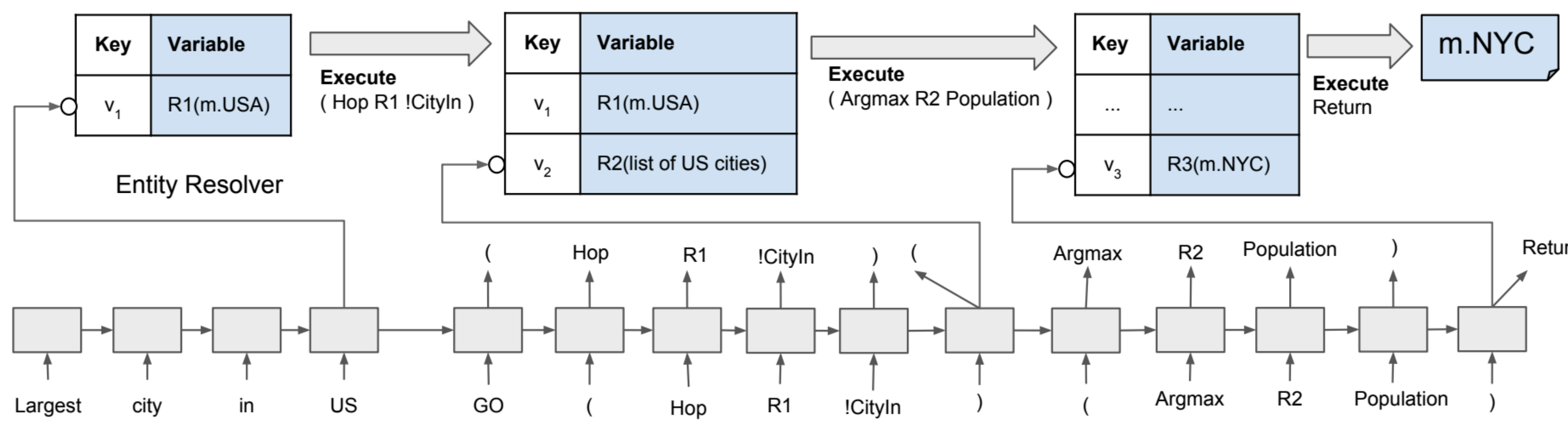
RL Approach

Predefined primitive operators

$$(\textit{Hop } r\ p\ ) \Rightarrow \{e_2 | e_1 \in r, (e_1, p, e_2) \in \mathbb{K}\}$$
$$(\textit{ArgMax } r\ p\ ) \Rightarrow \{e_1 | e_1 \in r, \exists e_2 \in \mathcal{E} : (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \geq e\}$$
$$(\textit{ArgMin } r\ p\ ) \Rightarrow \{e_1 | e_1 \in r, \exists e_2 \in \mathcal{E} : (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \leq e\}$$
$$(\textit{Filter } r_1\ r_2\ p\ ) \Rightarrow \{e_1 | e_1 \in r_1, \exists e_2 \in r_2 : (e_1, p, e_2) \in \mathbb{K}\}$$

Table 1: Interpreter functions. r represents a variable, p a property in Freebase. $\geq$ and $\leq$ are defined on numbers and dates.

Seq2Seq-like model



RL training
(BS better than sampling)

$$J^{RL}(\theta) = \sum_x \mathbb{E}_{P_\theta(a_{0:T}|x)}[R(x, a_{0:T})],$$

$$\nabla_\theta J^{RL}(\theta) = \sum_x \sum_{a_{0:T}} P_\theta(a_{0:T} \mid x) \cdot [R(x, a_{0:T}) - B(x)] \cdot \nabla_\theta \log P_\theta(a_{0:T} \mid x),$$

Liang, C., Berant, J., Le, Q., Forbus, K.D. and Lao, N.. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*, 2017.

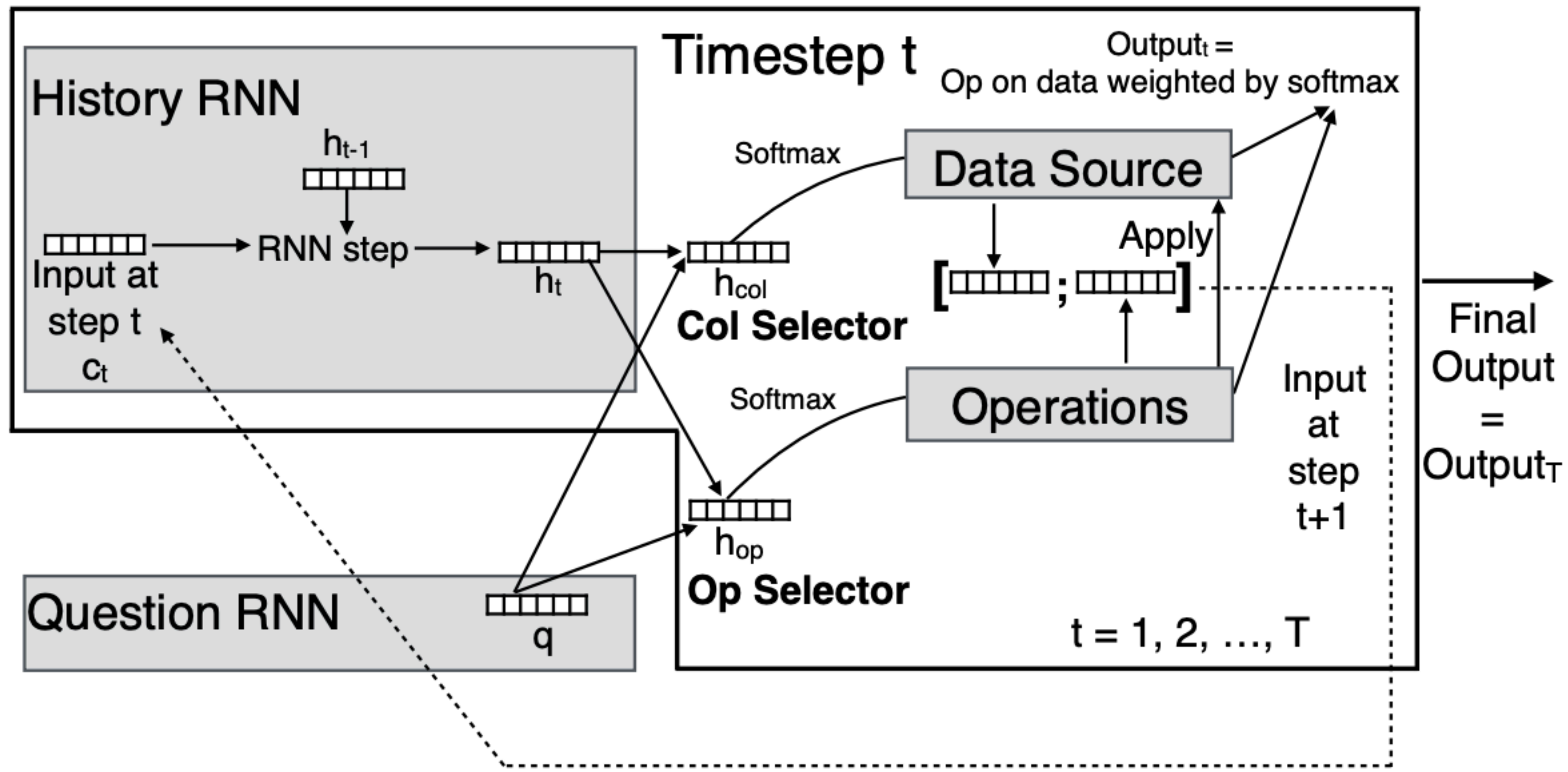
MLE

Method	**Approximation of $E_q[\cdot]$**	**Exploration strategy**	**Gradient weight $q(\mathbf{z})$**
REINFORCE	Monte Carlo integration	independent sampling	$p_\theta(\mathbf{z} \mid x)$
BS-MML	numerical integration	beam search	$p_\theta(\mathbf{z} \mid x, R(\mathbf{z}) \neq 0)$
RANDOMER	numerical integration	randomized beam search	$q_\beta(\mathbf{z})$

- Show close relationship between RL and MLE

Guu, K., Pasupat, P., Liu, E.Z. and Liang, P. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In ACL, 2017.

Attention on Execution Results



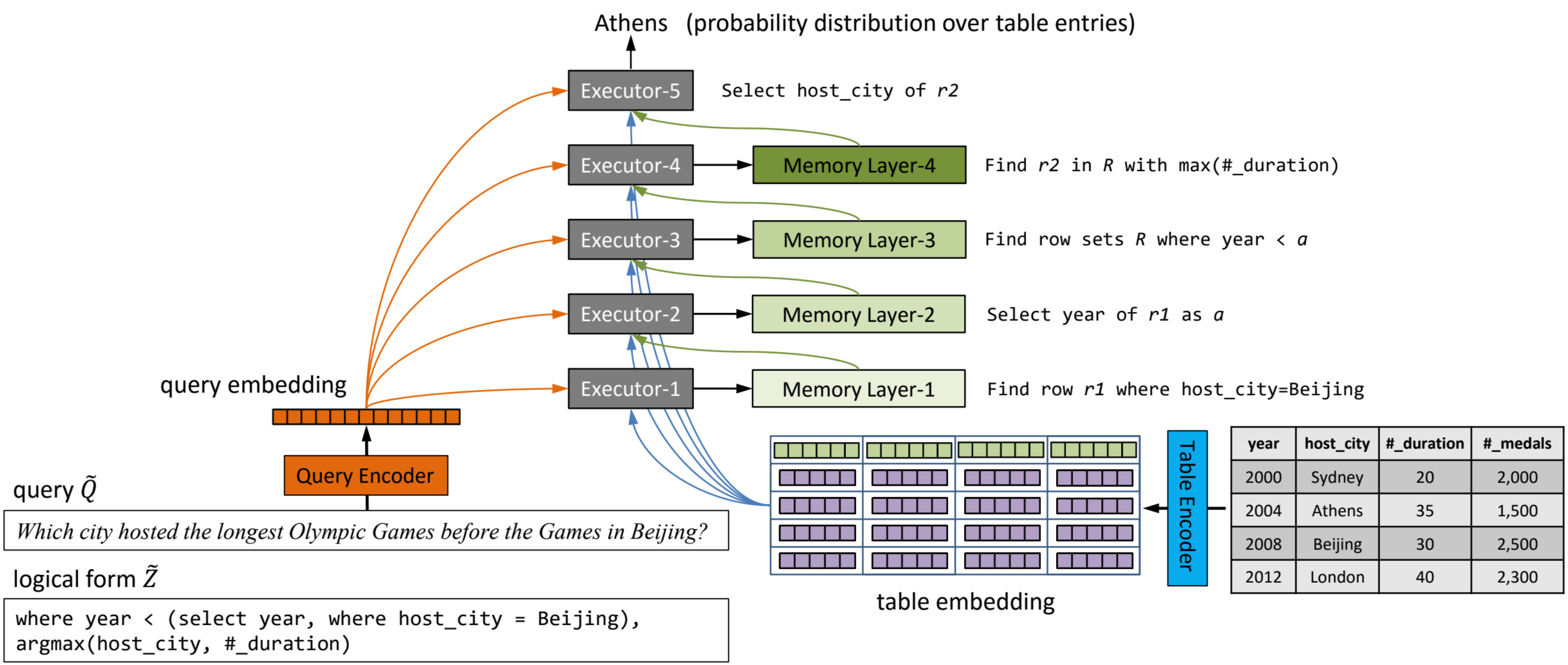
$$scalar\_answer_t = \alpha_t^{op}(\text{count})\, count_t + \alpha_t^{op}(\text{difference})\, diff_t + \sum_{j=1}^{C} \alpha_t^{col}(j)\alpha_t^{op}(\text{sum})\, sum_t[j],$$

$$lookup\_answer_t[i][j] = \alpha_t^{col}(j)\alpha_t^{op}(\text{assign})\, assign_t[i][j], \forall(i,j) i = 1, 2, \ldots, M, j = 1, 2, \ldots, C$$

Primitive operator + Step-by-step attn on results

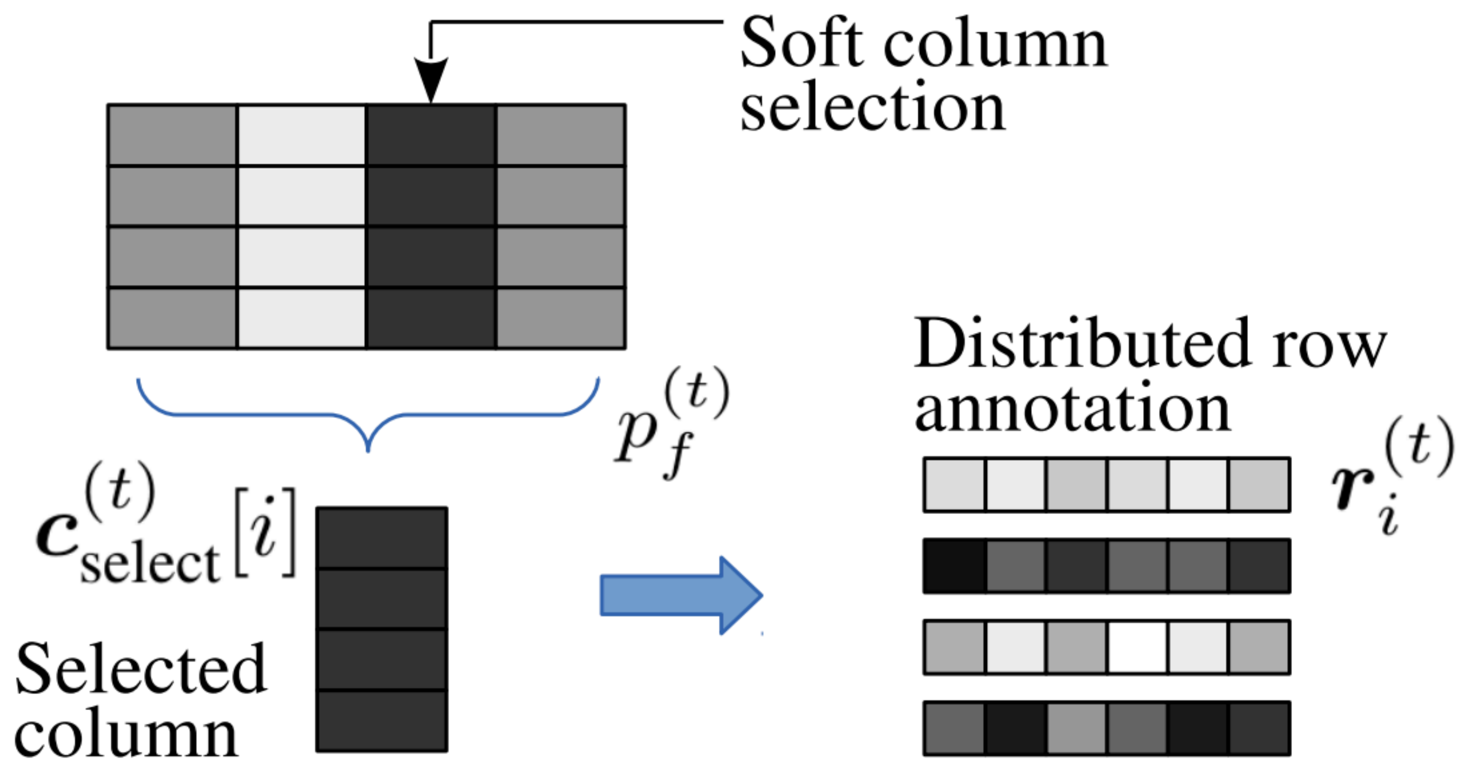
Neelakantan, A., Le, Q.V. and Sutskever, I. Neural programmer: Inducing latent programs with gradient descent. In *ICLR*, 2016.

Attention as Execution Itself



Yin, P., Lu, Z., Li, H. and Kao, B., 2015. Neural enquirer: Learning to query tables with natural language. In *IJCAI*, 2016.

Neural Executor

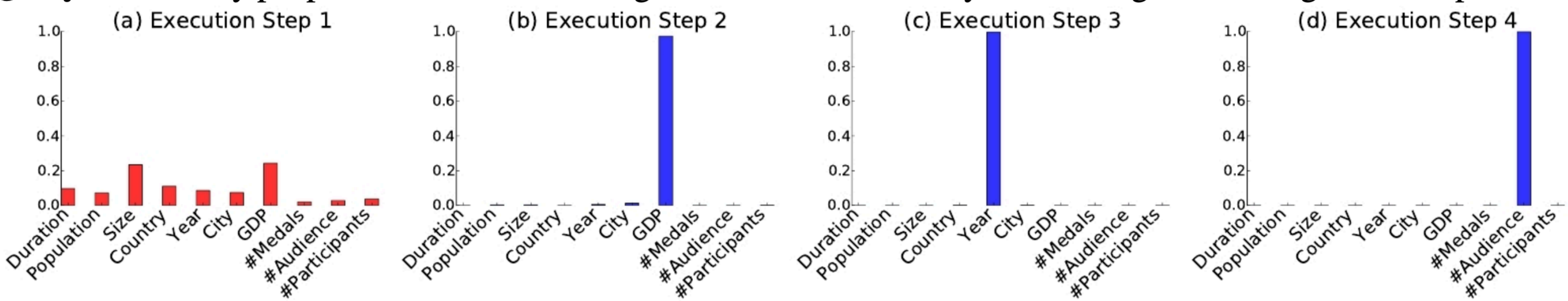


- Attention-based column selection
- Distributed representation for row selection
  - Not subject to primitive operators
  - Not fully explainable either

Yin, P., Lu, Z., Li, H. and Kao, B., 2015. Neural enquirer: Learning to query tables with natural language. In *IJCAI*, 2016.

Attention as Execution Itself

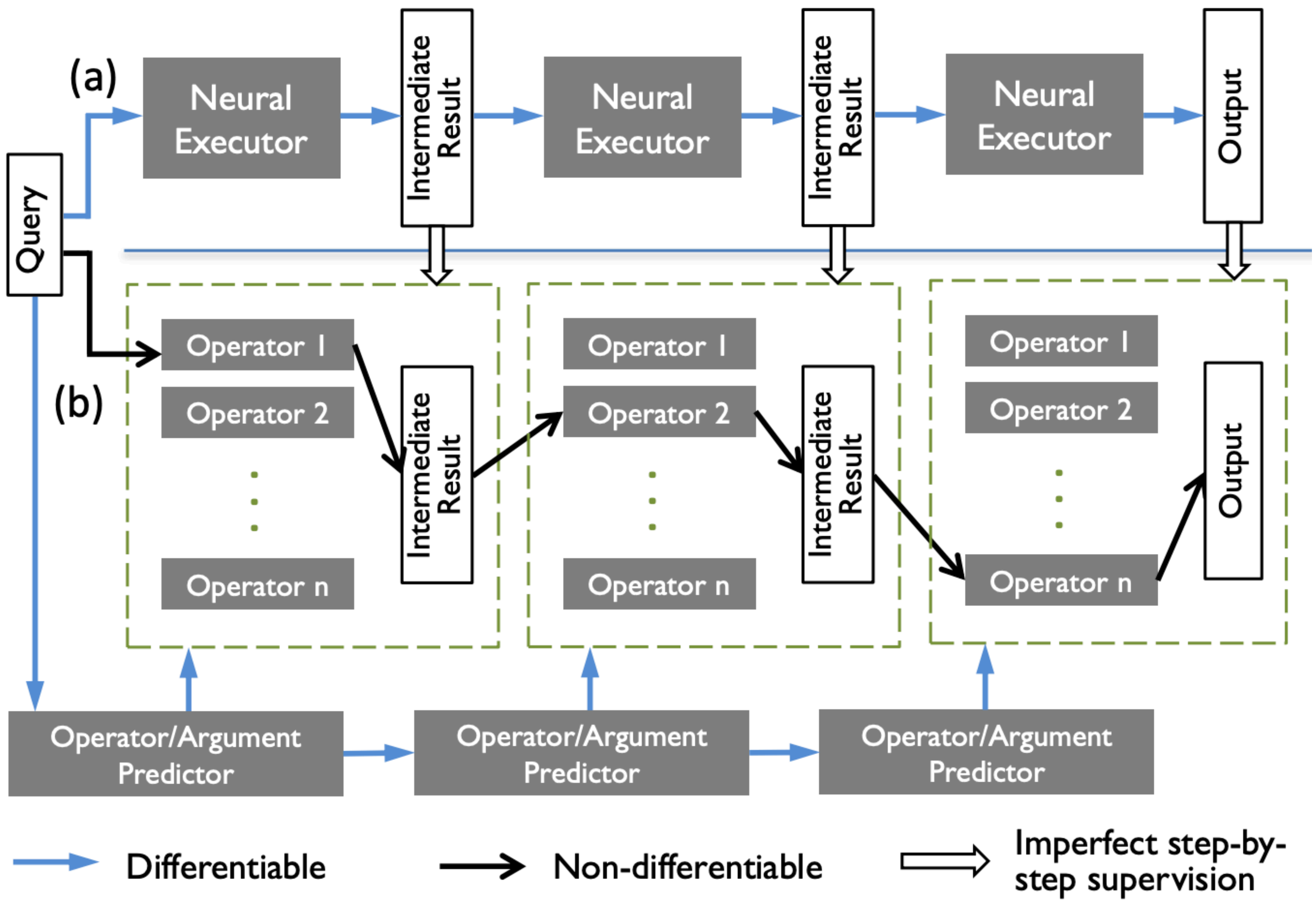
Query: How many people watched the earliest game whose host country GDP is larger than the game in Cape Town?



Step-by-step attention does learn meaningful things

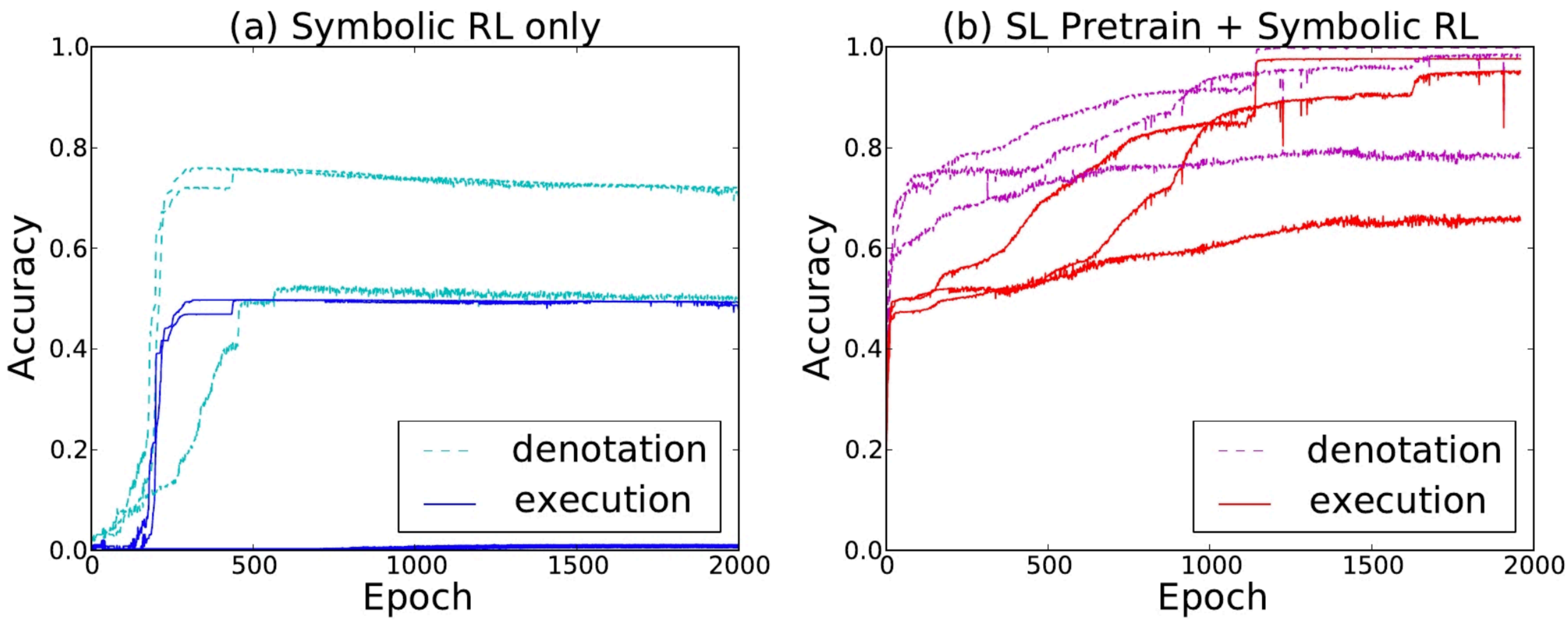
Yin, P., Lu, Z., Li, H. and Kao, B., 2015. Neural enquirer: Learning to query tables with natural language. In *IJCAI*, 2016.

Attention + RL



Lili Mou, Zhengdong Lu, Hang Li, Zhi Jin. Coupling distributed and symbolic execution for natural language queries. In *ICML*, 2017.

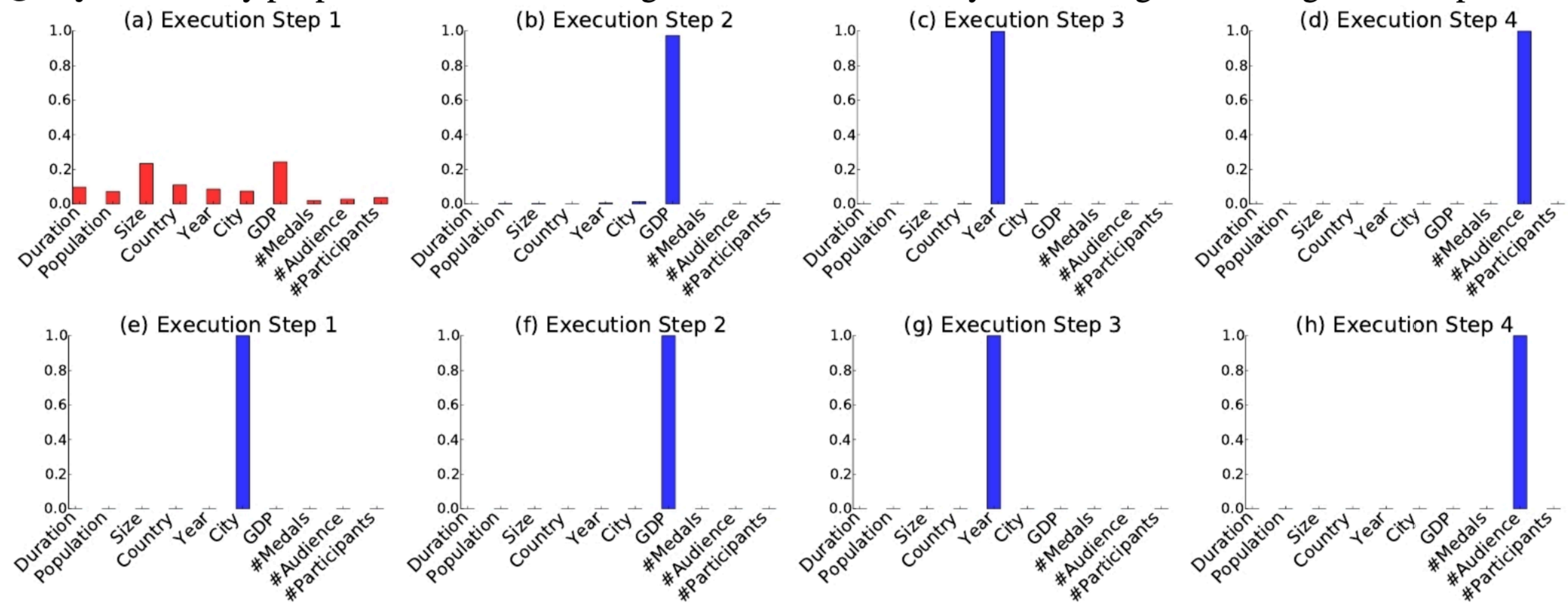
Attention-based initialization is important



Lili Mou, Zhengdong Lu, Hang Li, Zhi Jin. Coupling distributed and symbolic execution for natural language queries. In *ICML*, 2017.

Attention-based initialization is important

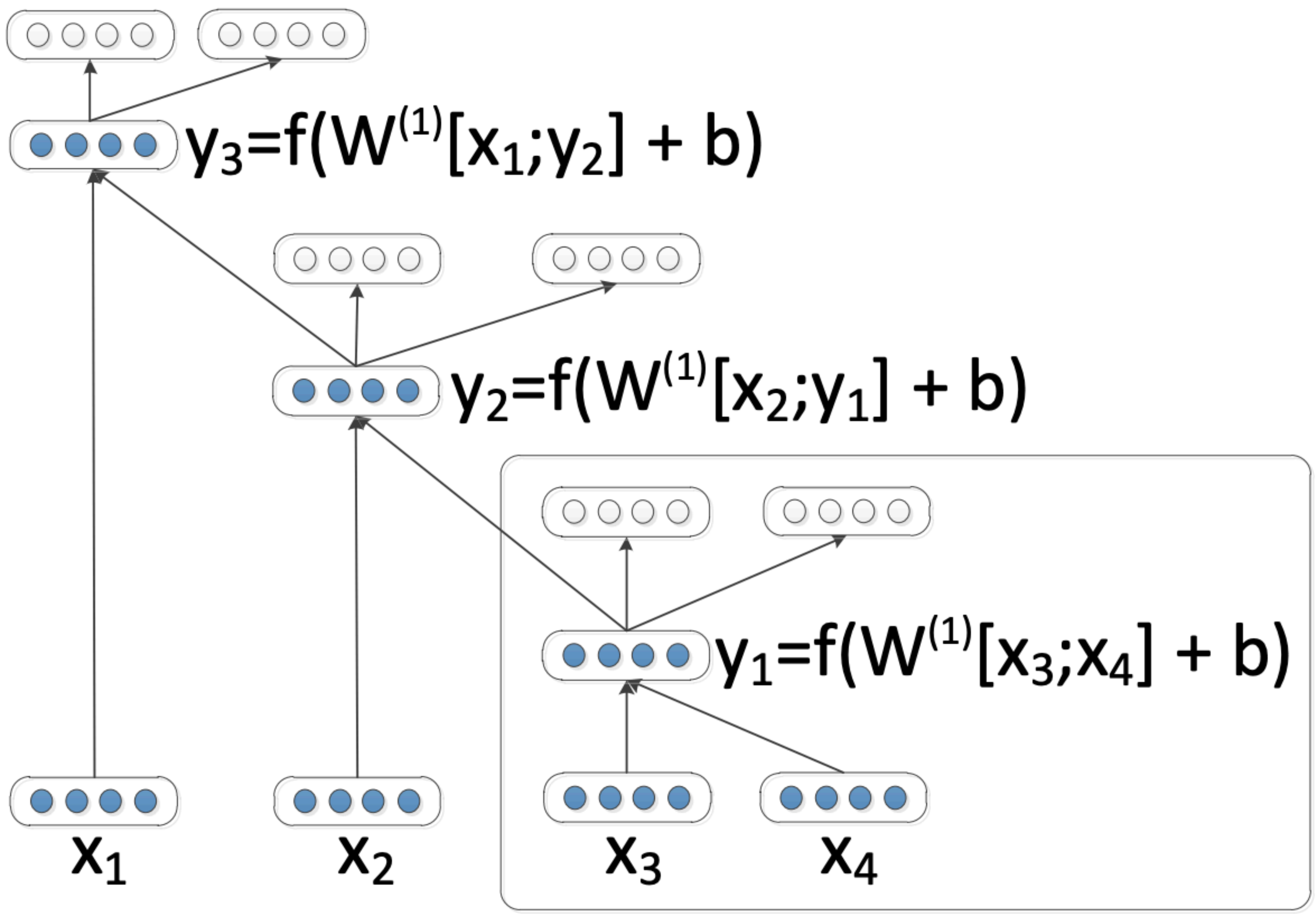
Query: How many people watched the earliest game whose host country GDP is larger than the game in Cape Town?



Lili Mou, Zhengdong Lu, Hang Li, Zhi Jin. Coupling distributed and symbolic execution for natural language queries. In *ICML*, 2017.

Application: Syntactic Parsing (Unsupervised)

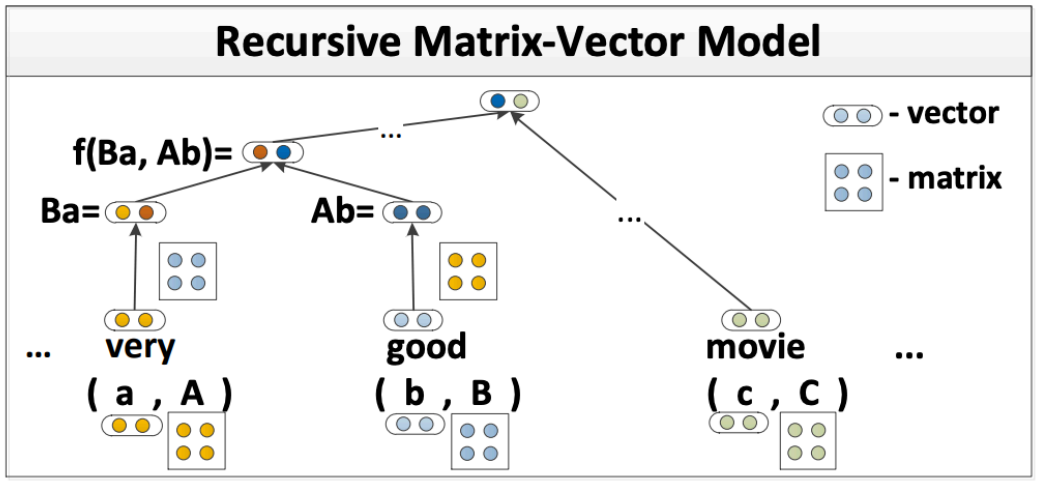
Recursive Autoencoder

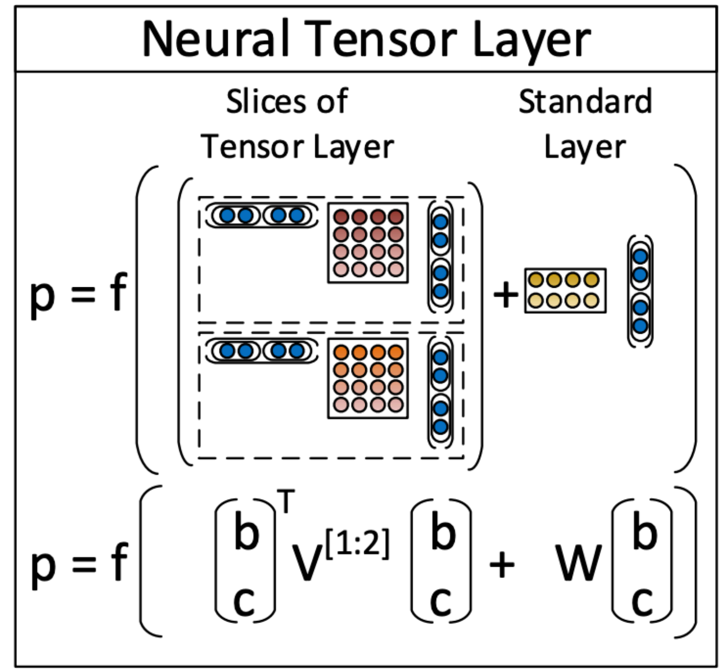


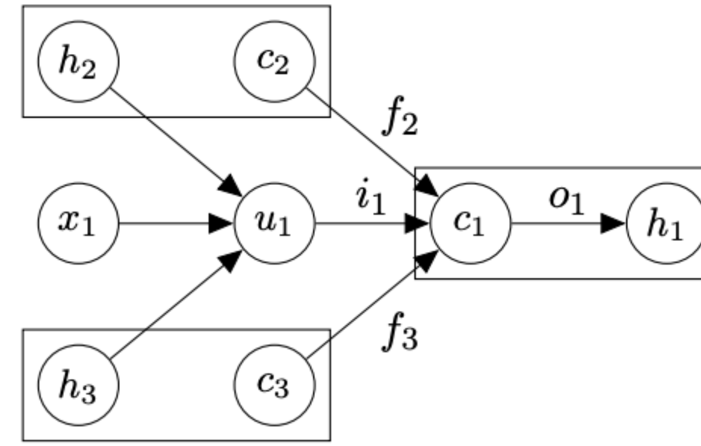
Induce tree structures by minimizing reconstruction on an AE

Socher, Richard, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, 2011.

Recursive Neural Network







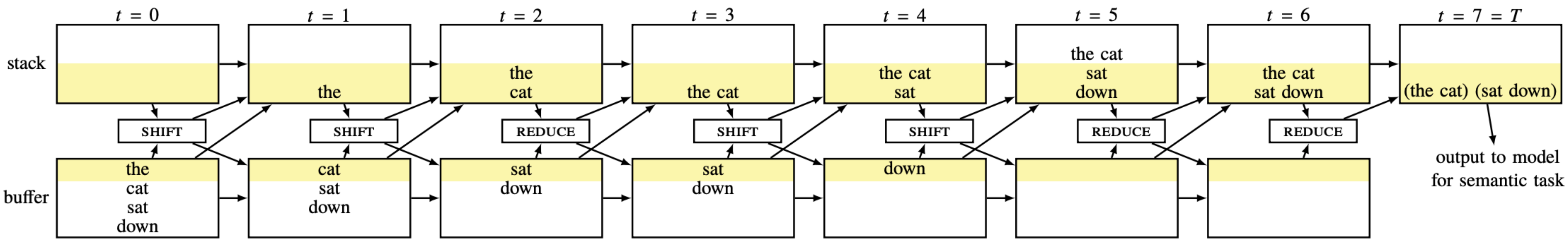
- Parsing by auto-encoding never worked
- Standard RecursiveNN is based on external parse trees

I.e., Tree structures are constant

Sheng, Socher, et al. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, 2015.
Socher, R., et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.
Socher R., et al. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, 2012.

SPINN

Stack-augmented Parser-Interpreter Neural Network

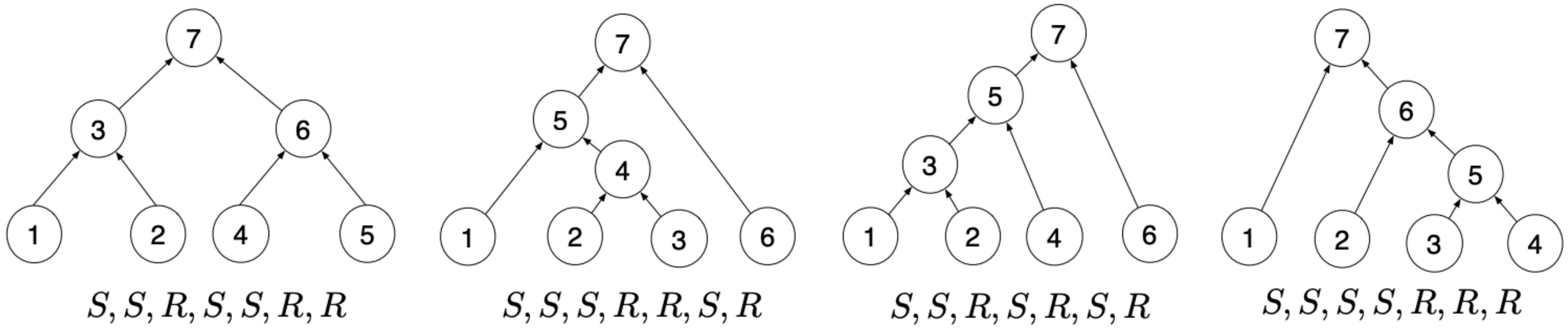


(b) The fully unrolled SPINN for *the cat sat down*, with neural network layers omitted for clarity.

- Shift-reduce parser jointly trained with downstream task
- Supervision provided by Standford Parser

Bowman, S.R., Gauthier, J., Rastogi, A., Gupta, R., Manning, C.D. and Potts, C., 2016. A fast unified model for parsing and sentence understanding. In *ACL*, 2016.

RL-SPINN



- Still shift-reduce parser
- Semi-supervised or unsupervised
- Trained by RL

$$\mathcal{R}(\mathbf{W}) = \mathbb{E}_{\pi(\mathbf{a},\mathbf{s};\mathbf{W}_R)}\left[\sum_{t=1}^{T} r_t a_t\right]$$

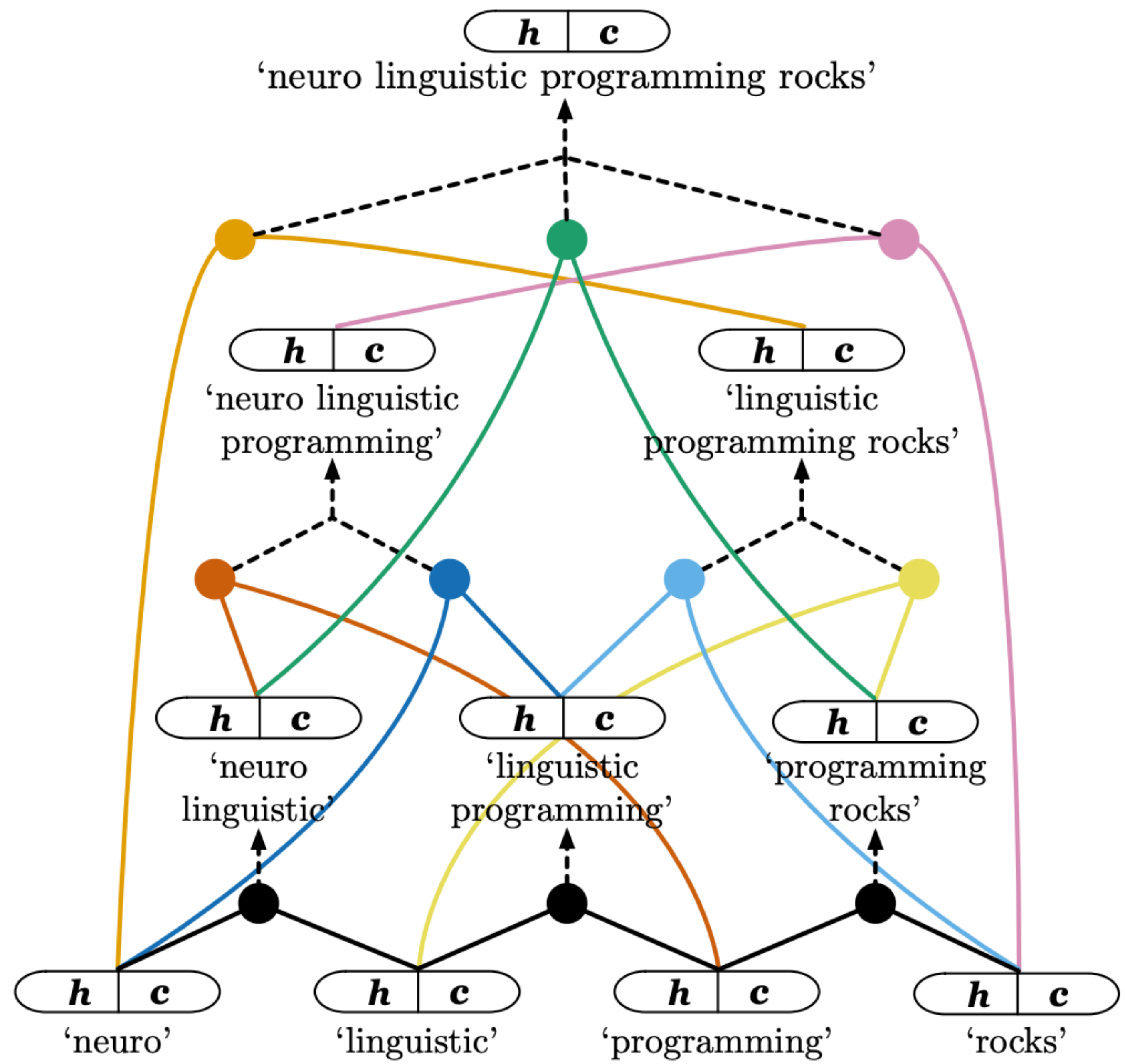
Yogatama, D., Blunsom, P., Dyer, C., Grefenstette, E. and Ling, W.. Learning to compose words into sentences with reinforcement learning. In *ICLR*, 2017.

Chart-style Parser

- Implicitly considering all possible trees
- Not exact marginalization
- Step-by-step fusion/attention

$$s_i = \mathrm{softmax}(e_i/t),$$

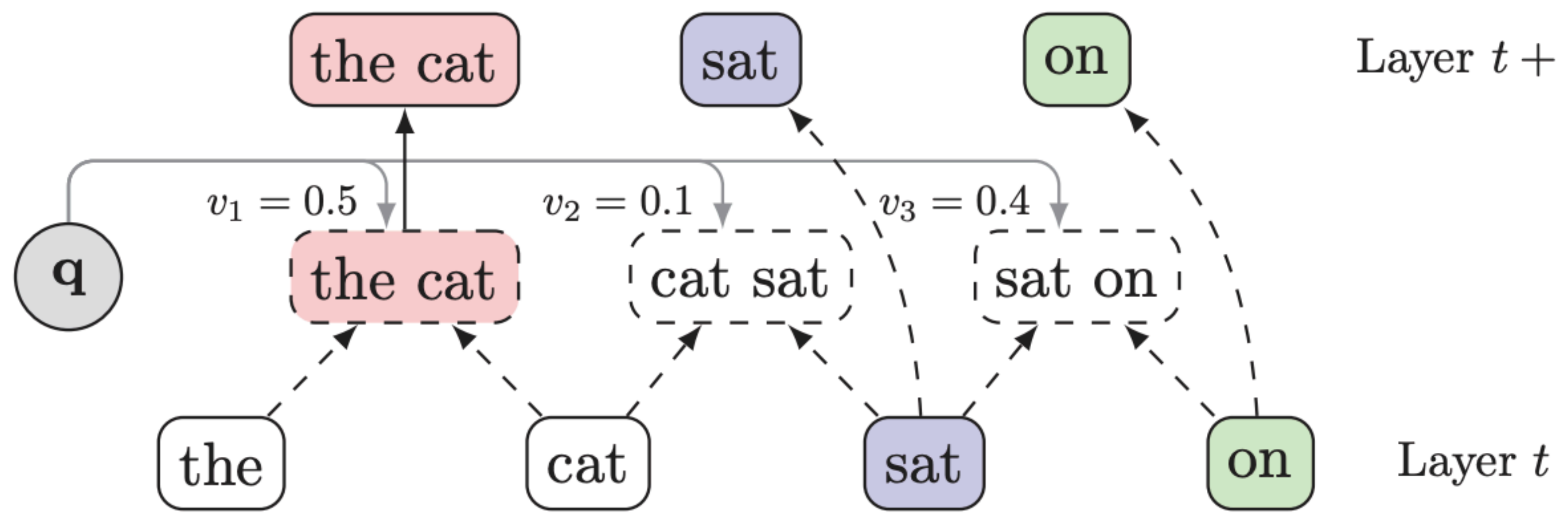
$$\boldsymbol{c} = \sum_{i=1}^{n} s_i \boldsymbol{c}_i, \qquad \boldsymbol{h} = \sum_{i=1}^{n} s_i \boldsymbol{h}_i.$$

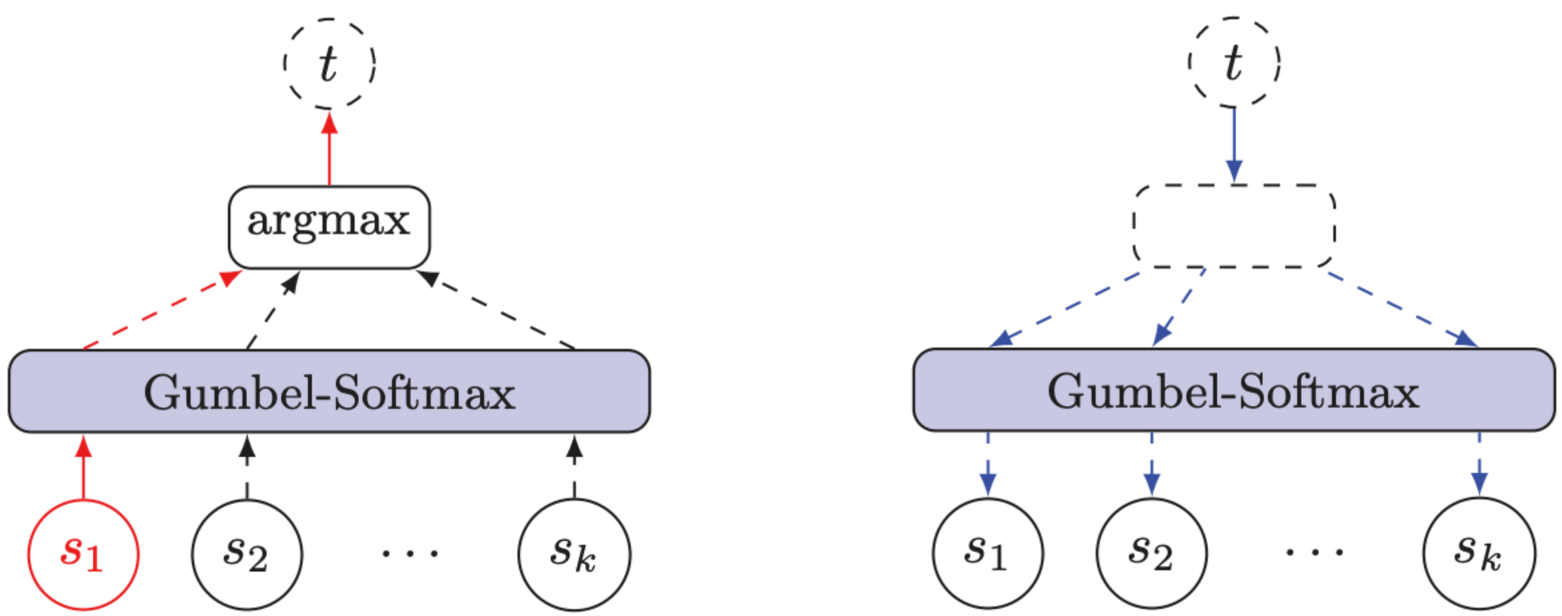


Maillard, J., Clark, S. and Yogatama, D. Jointly learning sentence embeddings and syntax with unsupervised tree-LSTMs. *NLE*, 2019.

Pyramid

- ST-Gumbel





Choi, J., Yoo, K.M. and Lee, S.G. Learning to compose task-specific tree structures. In *AAAI*, 2018.

Main issues with these models

[William et al., TACL'18]

- Trees are not consistent across random init.
- Do not resemble real trees

[Shi et al., EMNLP'18]

- All trees are similar to downstream performance
- Balanced trees are slightly better

Williams, A., Drozdov, A. and Bowman, S.R. Do latent tree learning models identify meaningful structure in sentences? *TACL*, 2018.

Shi, H., Zhou, H., Chen, J. and Li, L., 2018. On tree-based neural sentence modeling. In *EMNLP*, 2018.

Proximal Policy Optimization

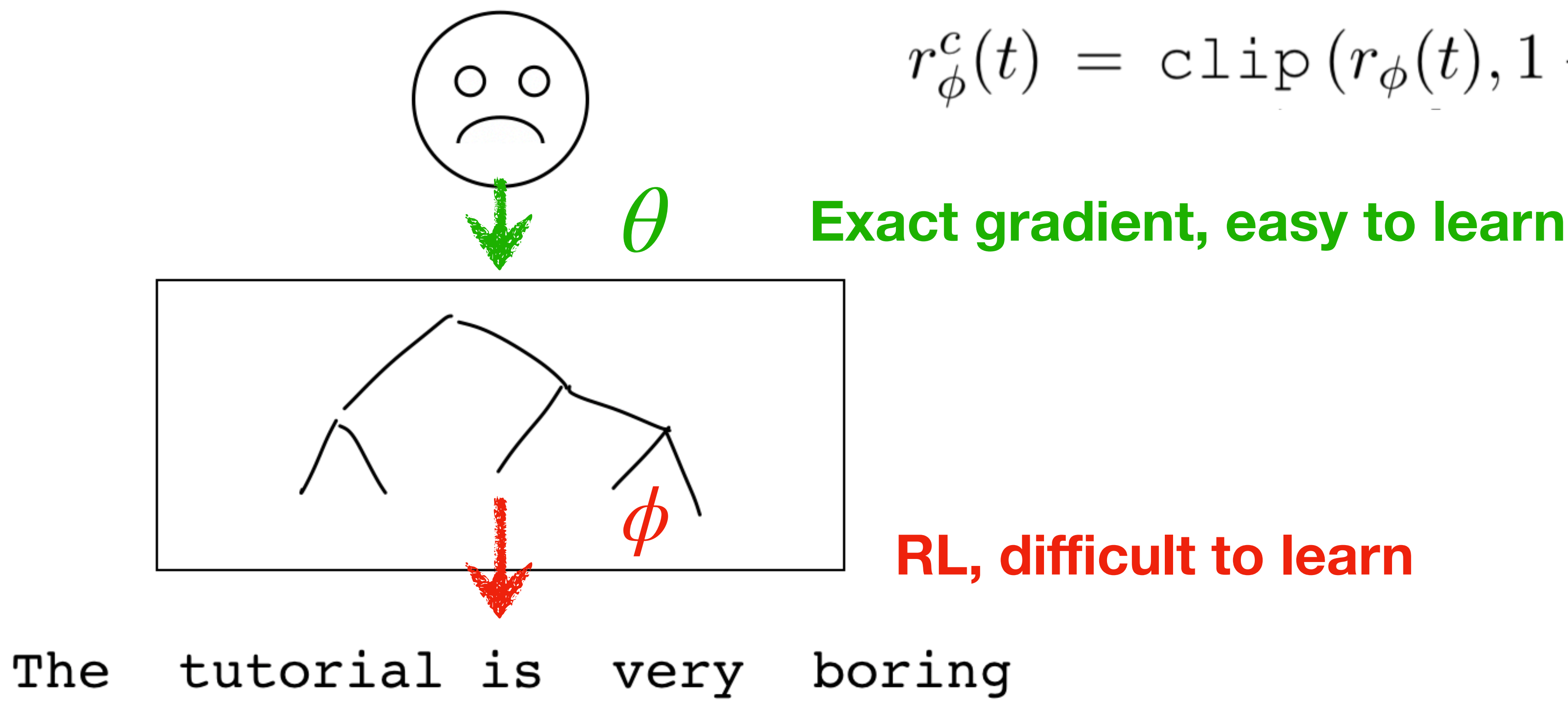
- Train the policy K steps

$$\hat{\mathbb{E}}_t \left[r_\phi(t) \ell(f_\theta(x,t), y) \right] \qquad r_\phi(t) = \frac{p_\phi(t|x)}{p_{\phi_{\text{old}}}(t|x)}$$

- Clip gradient

$$\hat{\mathbb{E}}_t \left[\max \left\{ r_\phi(t) \ell \left(f_\theta(x,t), y \right), r_\phi^c(t) \ell \left(f_\theta(x,t), y \right) \right\} \right]$$

$$r_\phi^c(t) = \texttt{clip}\left(r_\phi(t), 1-\epsilon, 1+\epsilon \right)$$



Havrylov, S., Kruszewski, G. and Joulin, A., 2019. Cooperative learning of disjoint syntax and semantics. In *NAACL-HLT*, 2019.

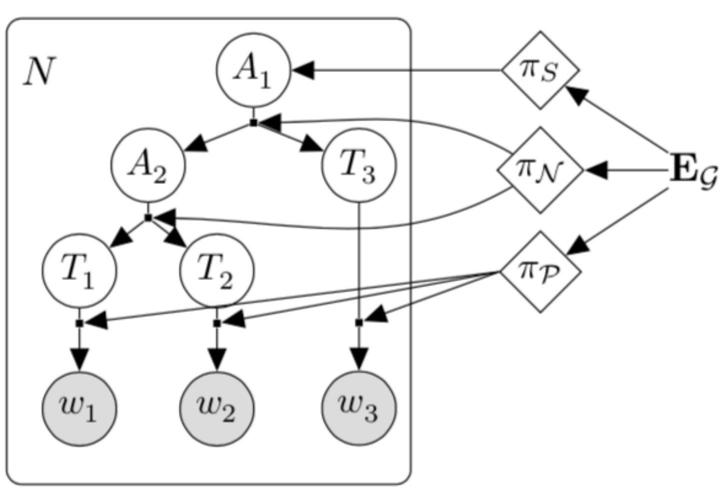
Compound PCFG

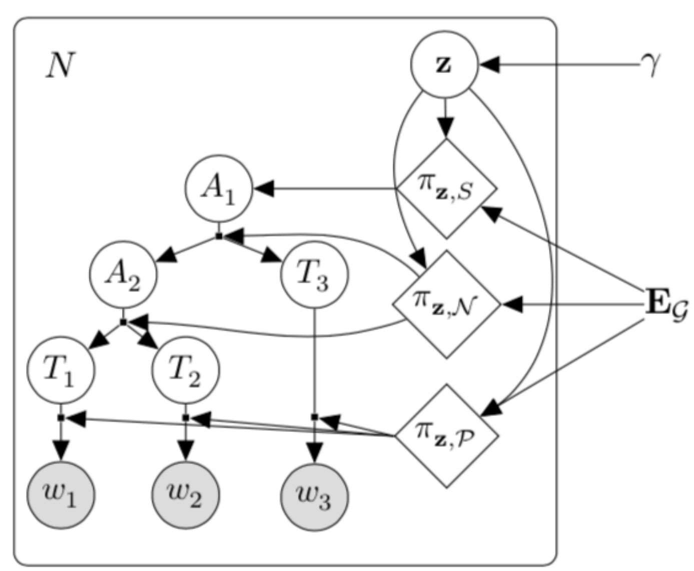
- Over-parametrize PCFG into a Gaussian continuous space
  - Shown to be easier to train and more linguistically plausible

$$\log p_\theta(\boldsymbol{x}) = \log\left(\int p_\theta(\boldsymbol{x} \mid \mathbf{z}) p_\gamma(\mathbf{z})\, \mathrm{d}\mathbf{z}\right)$$
$$= \log\left(\boxed{\int} \boxed{\sum_{t \in \mathcal{T}_\mathcal{G}(\boldsymbol{x})}} p_\theta(\boldsymbol{t} \mid \mathbf{z}) p_\gamma(\mathbf{z})\, \mathrm{d}\mathbf{z}\right)$$

VAE

Exact by inside all

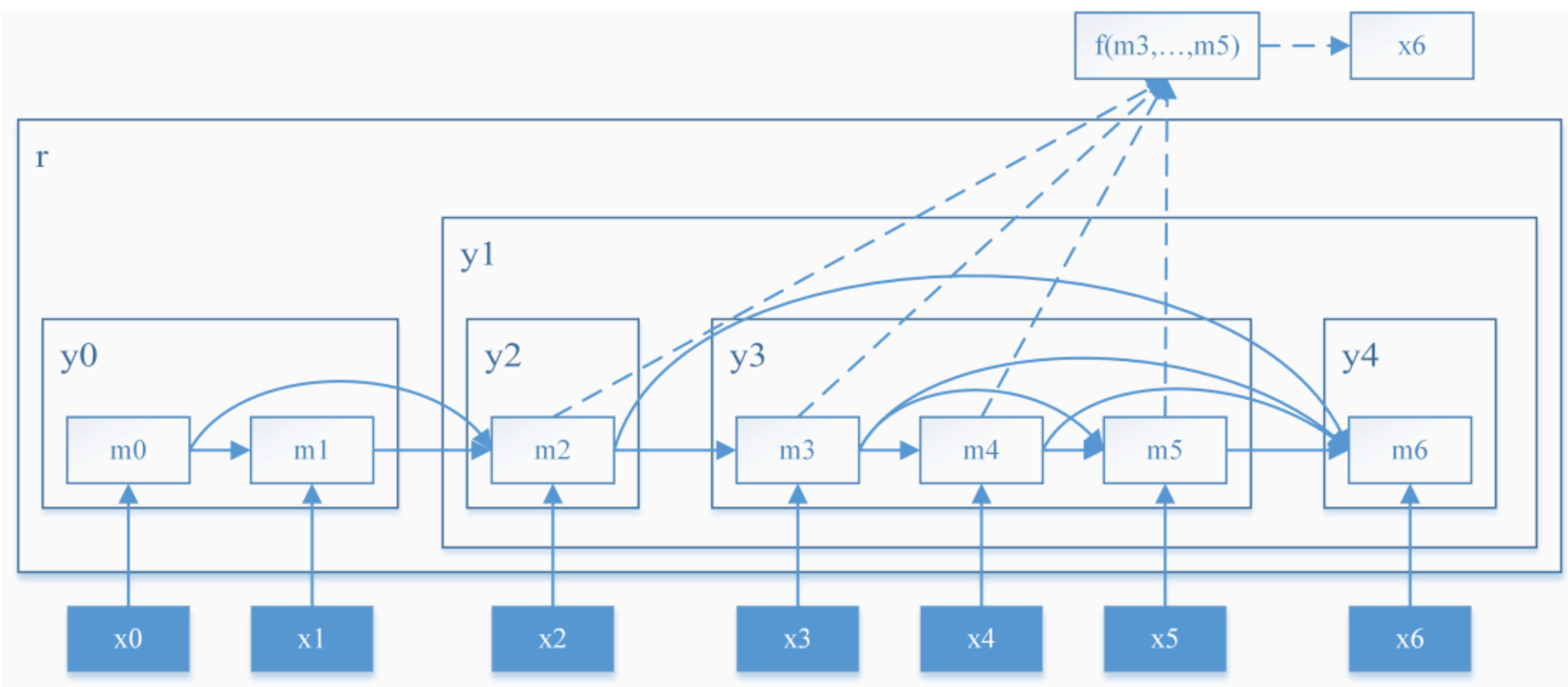




Kim, Y., Dyer, C. and Rush, A.M., 2019. Compound Probabilistic Context-Free Grammars for Grammar Induction. In *ACL*, 2019.

PRPN **Parsing-Reading-Predict Networks**

- Language modeling is important
- Structured attention, based on “syntactic distance”

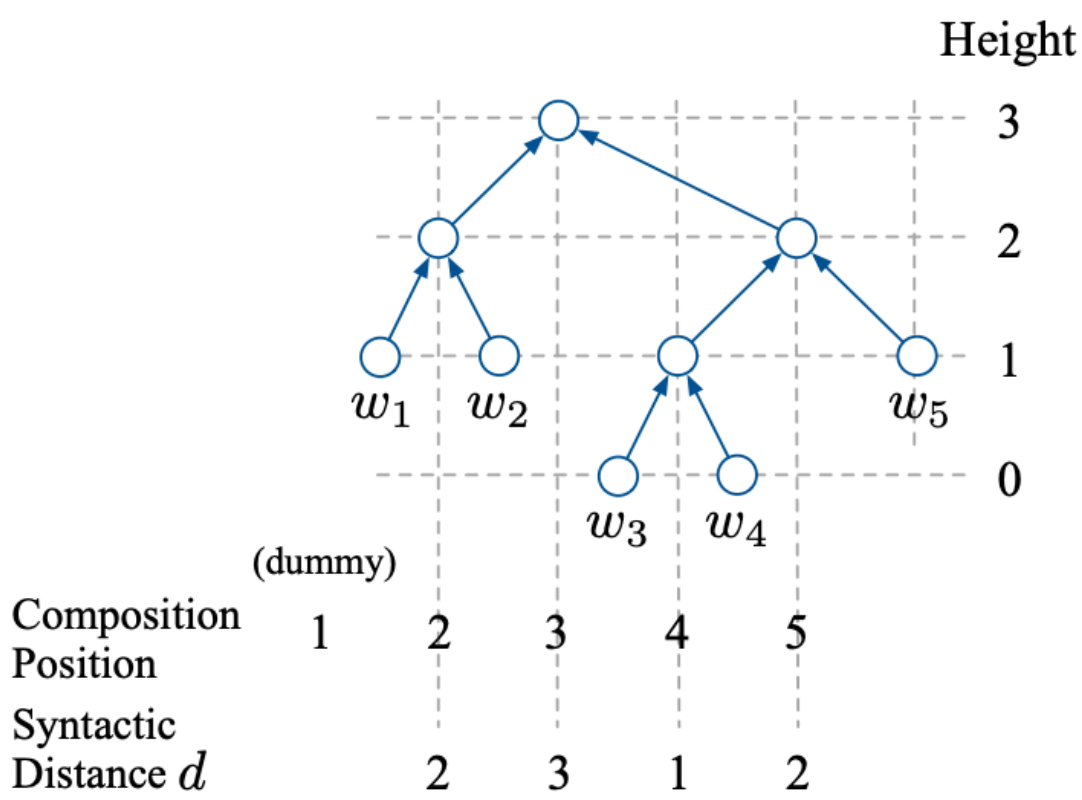


Shen, Y., Lin, Z., Huang, C.W. and Courville, A. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*, 2018.

PRPN Parsing-Reading-Predict Networks

- Syntactic distance d (learned in an unsupervised way)

Difference of d: $$\alpha_j^t = \frac{\text{hardtanh}(\tau(\widehat{d}_t - \widehat{d}_j)) + 1}{2} \in [0,1]$$



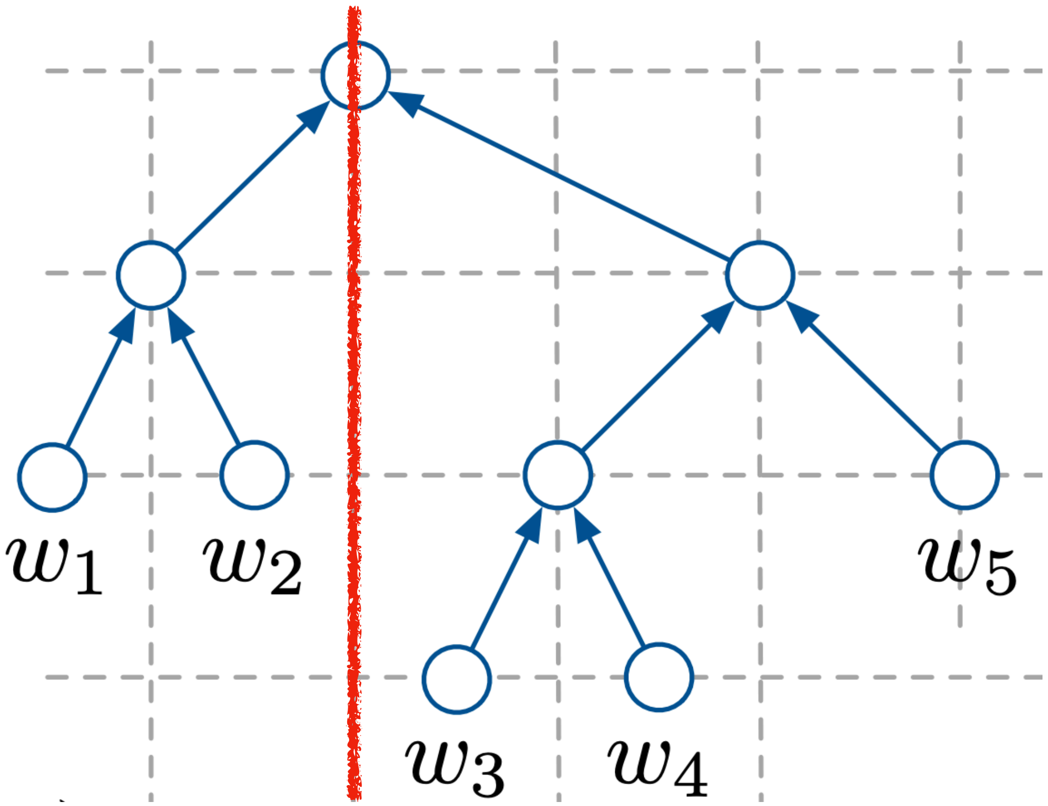
Multiplicative accumulation $$g_i^t = \prod_{j=i+1}^{t-1} \alpha_j^t$$

Reweigh self-attn. $$s_i^t = \frac{g_i^t}{\sum_{i=1}^{t-1} g_i^t} \widetilde{s}_i^t$$

Shen, Y., Lin, Z., Huang, C.W. and Courville, A. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*, 2018.

PRPN **Parsing-Reading-Predict Networks**

- Prediction



$$(w_1, w_2) \,|\, (w_3 w_4 w_5)$$

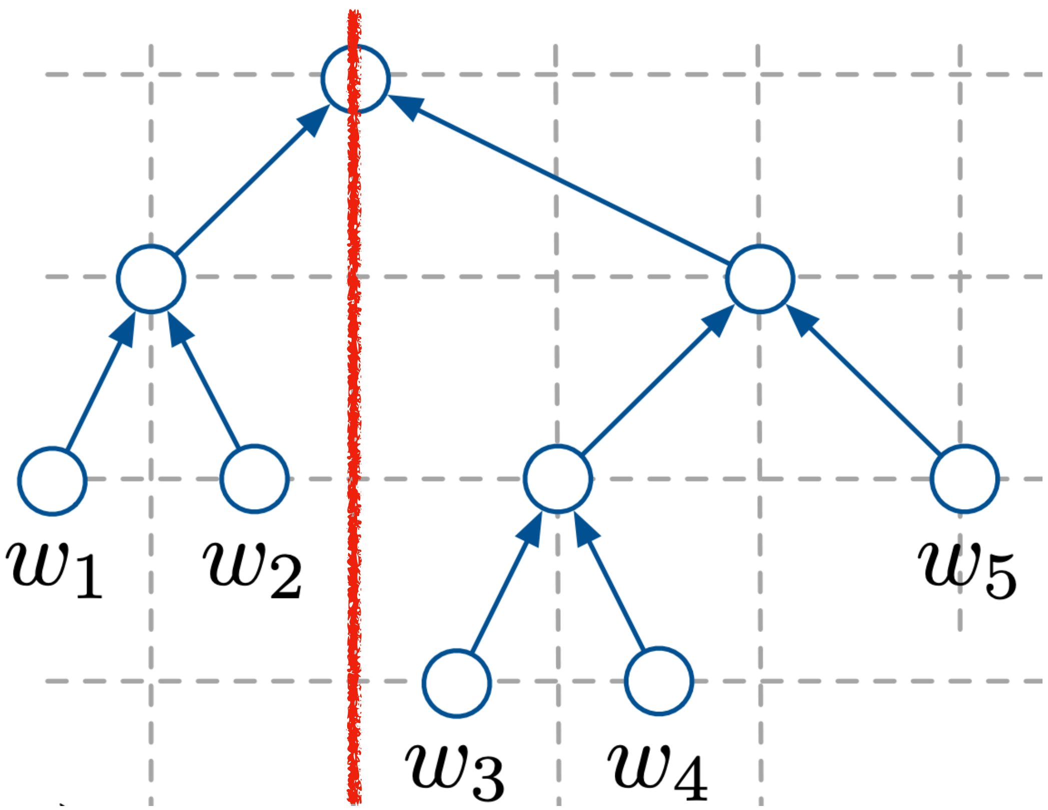
Intuitive way/In paper

Shen, Y., Lin, Z., Huang, C.W. and Courville, A. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*, 2018.

PRPN

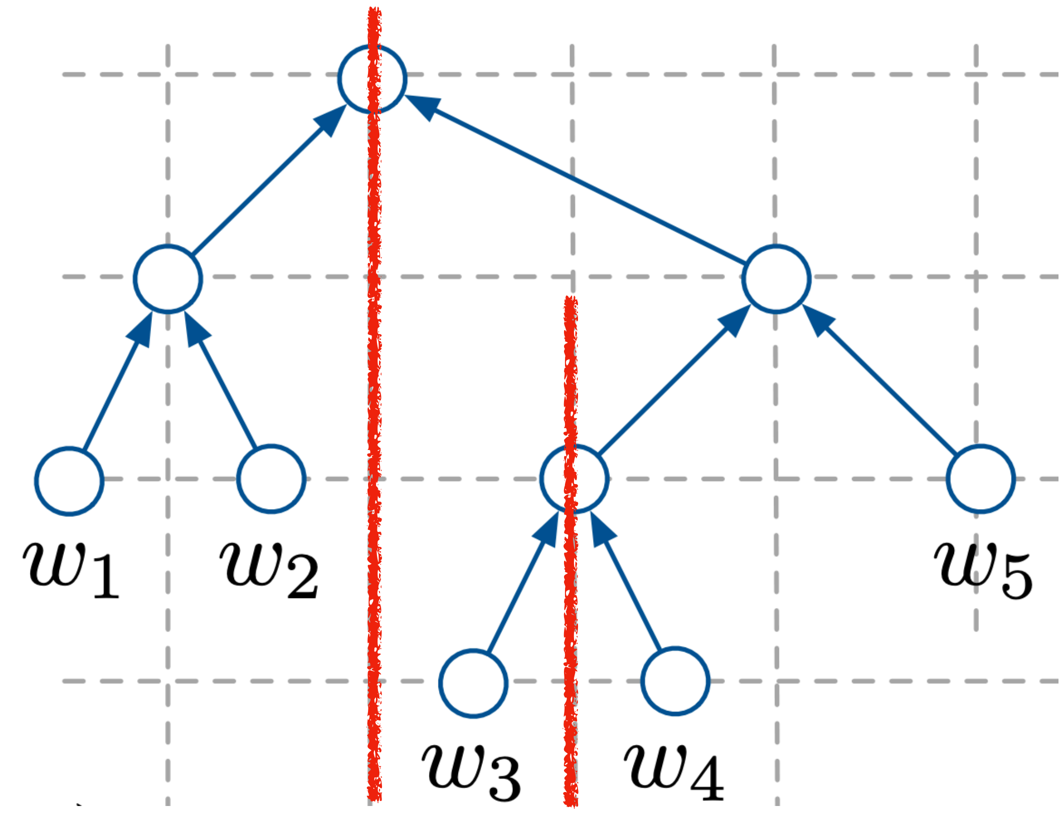
Parsing-Reading-Predict Networks

- Prediction



$(w_1, w_2) \,|\, (w_3 w_4 w_5)$

Intuitive way/In paper



$(w_1, w_2) \,|\, (w_3 \,|\, (w_4 w_5))$

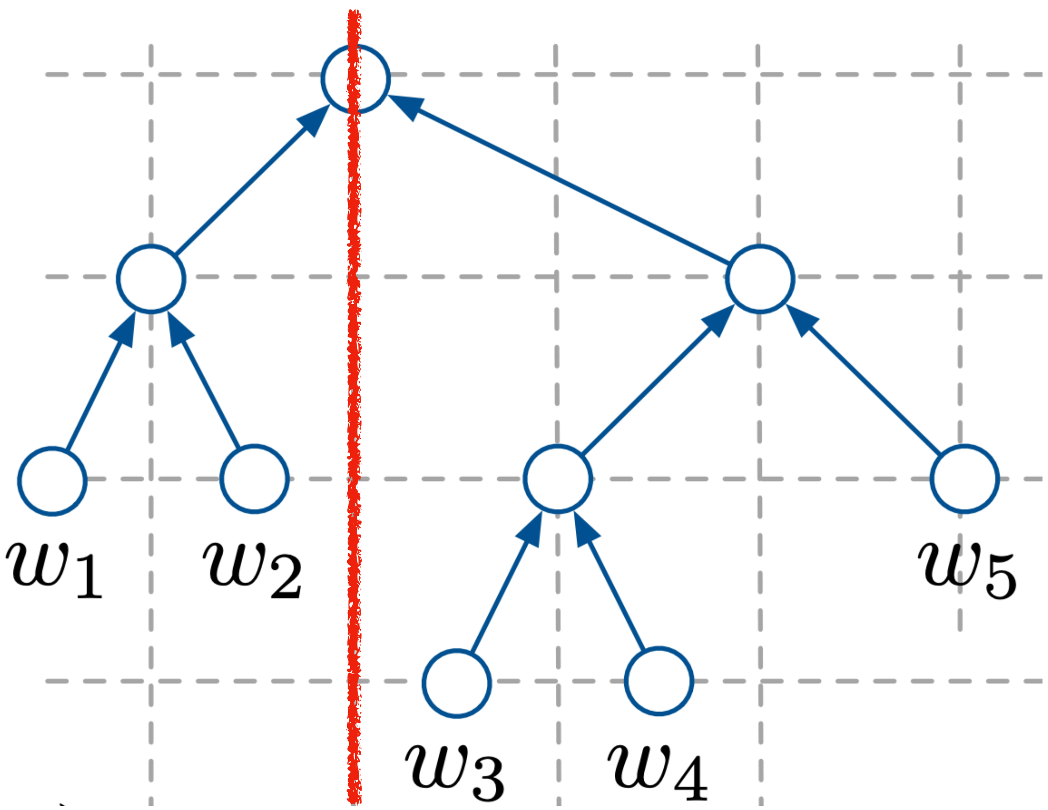
In Appendix/Code

Shen, Y., Lin, Z., Huang, C.W. and Courville, A. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*, 2018.

PRPN

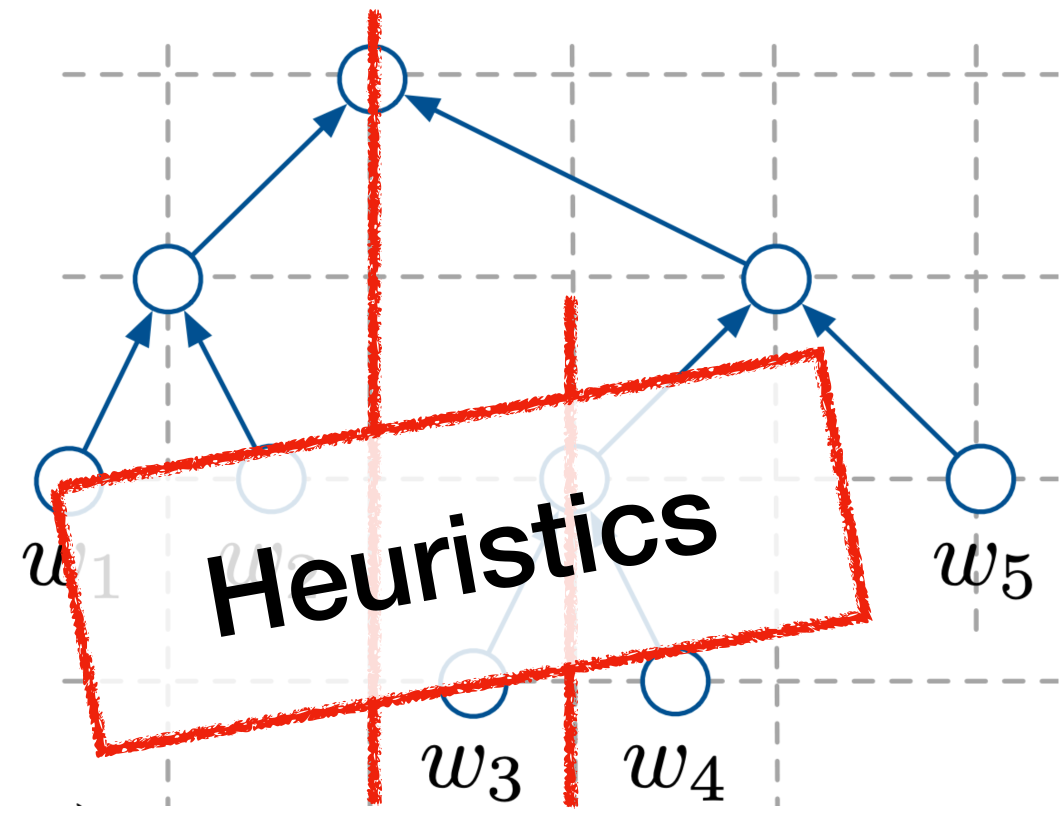
Parsing-Reading-Predict Networks

- Prediction



$(w_1, w_2) \,|\, (w_3 w_4 w_5)$

Intuitive way/In paper

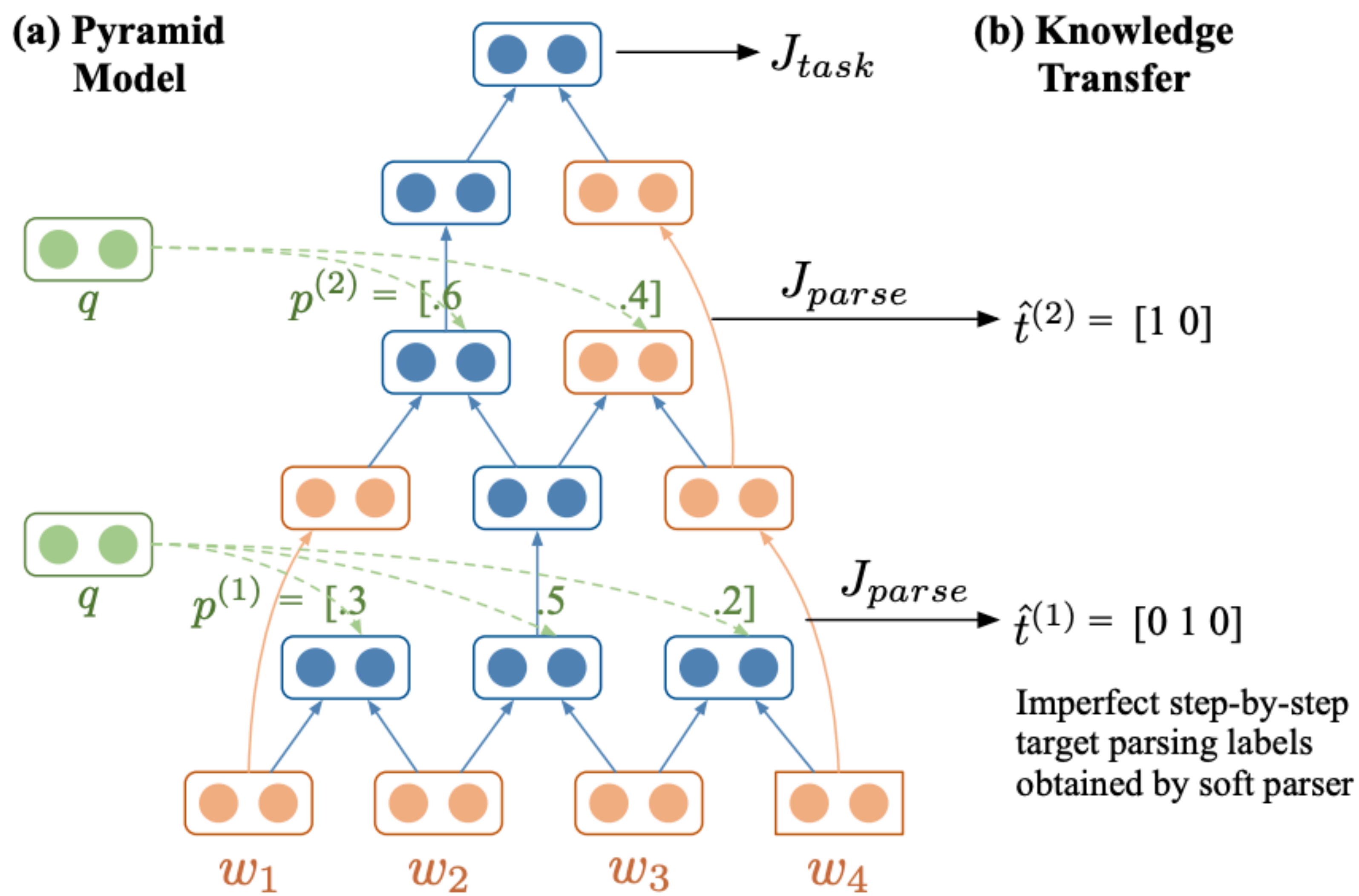


$(w_1, w_2) \,|\, (w_3 \,|\, (w_4 w_5))$

In Appendix/Code

Shen, Y., Lin, Z., Huang, C.W. and Courville, A. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*, 2018.

Combining Both Worlds



- Step1: Step-by-step learning from PRPN
- Step2: Policy improvement by ST-Gumbel

Bowen Li, Lili Mou, Frank Keller. An imitation learning approach to unsupervised parsing. In *ACL*, 2019.

Results

Model	w/o Punctuation			w/ Punctuation		
	Mean F	Self-agreement	RB-agreement	Mean F	Self-agreement	RB-agreement
Left-Branching	20.7	-	-	18.9	-	-
Right-Branching	**58.5**	-	-	18.5	-	-
Balanced-Tree	39.5	-	-	22.0	-	-
ST-Gumbel	36.4	57.0	33.8	21.9	56.8	**38.1**
PRPN	46.0	48.9	51.2	51.6	65.0	27.4
Imitation (SbS only)	45.9	49.5	62.2	52.0	**70.8**	20.6
Imitation (SbS + refine)	53.3†	**58.2**	**64.9**	**53.7**†	67.4	21.1

- Our results show
  - Language modeling is good, but semantic oriented tasks also help
  - ST-Gumbel works if meaningful initialized

Bowen Li, Lili Mou, Frank Keller. An imitation learning approach to unsupervised parsing. In *ACL*, 2019.

Summary

MLE	maximize	$\log\left(\sum_z p(z)p(\boldsymbol{Y}\mid z,\boldsymbol{\theta})\right)$
RL	maximize	$\mathop{\mathbb{E}}_{z\sim p_{\boldsymbol{\theta}}(z)} R(Y(z))$
Gumbel softmax	maximize	$\mathop{\mathbb{E}}_{\epsilon\in p(\epsilon)} J(Y(\ z_{\boldsymbol{\theta}}(\epsilon)\ ))$
Attention	maximize	$J(Y(\mathbb{E}_{z\sim p_{\boldsymbol{\theta}}(z)}[\boldsymbol{z}]))$

- Case studies
  - Weakly supervised semantic parsing
  - Unsupervised syntactic parsing

amii

UNIVERSITY OF ALBERTA

ByteDance 字节跳动

References

- Bishop CM. *Pattern Recognition and Machine Learning*. Springer, 2006.
- Eisner, Jason. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, 2016.
- Guu K, Pasupat P, Liu EZ, Liang P. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *ACL*, 2017.
- Kingma DP, Welling M. Auto-encoding variational Bayes. In *ICLR*, 2014.
- Sutton RS, Barto AG. Introduction to Reinforcement Learning. 1998.
- Gumbel EJ. Statistical theory of extreme values and some practical applications: a series of lectures. US Government Printing Office; 1948.
- Jang E, Gu S, Poole B. Categorical reparameterization with Gumbel-softmax. In *ICLR*, 2017.
- Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.
- Martins, A. and Astudillo, R., June. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, 2016.
- Niculae, V., Martins, A.F., Blondel, M. and Cardie, C. SparseMAP: Differentiable sparse structured inference. In *ICML*, 2018.
- Dong, Li, and Mirella Lapata. Language to logical form with neural attention. In *ACL*, 2016.
- Liang, C., Berant, J., Le, Q., Forbus, K.D. and Lao, N.. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*, 2017.
- Neelakantan, A., Le, Q.V. and Sutskever, I. Neural programmer: Inducing latent programs with gradient descent. In *ICLR*, 2016.

References

- Yin, P., Lu, Z., Li, H. and Kao, B., 2015. Neural enquirer: Learning to query tables with natural language. In *IJCAI*, 2016.
- Lili Mou, Zhengdong Lu, Hang Li, Zhi Jin. Coupling distributed and symbolic execution for natural language queries. In ICML, 2017.
- Socher, Richard, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, 2011.
- Bowman, S.R., Gauthier, J., Rastogi, A., Gupta, R., Manning, C.D. and Potts, C., 2016. A fast unified model for parsing and sentence understanding. In *ACL*, 2016.
- Yogatama, D., Blunsom, P., Dyer, C., Grefenstette, E. and Ling, W.. Learning to compose words into sentences with reinforcement learning. In *ICLR*, 2017.
- Maillard, J., Clark, S. and Yogatama, D.. Jointly learning sentence embeddings and syntax with unsupervised tree-LSTMs. *NLE*, 2019.
- Choi, J., Yoo, K.M. and Lee, S.G. Learning to compose task-specific tree structures. In *AAAI*, 2018.
- Williams, A., Drozdov, A. and Bowman, S.R. Do latent tree learning models identify meaningful structure in sentences? *TACL*, 2018.
- Havrylov, S., Kruszewski, G. and Joulin, A., 2019. Cooperative learning of disjoint syntax and semantics. In *NAACL-HLT*, 2019.
- Shen, Y., Lin, Z., Huang, C.W. and Courville, A. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*, 2018.
- Bowen Li, Lili Mou, Frank Keller. An imitation learning approach to unsupervised parsing. In *ACL*, 2019.
- Kim, Y., Dyer, C. and Rush, A.M., 2019. Compound Probabilistic Context-Free Grammars for Grammar Induction. In *ACL*, 2019.