Neural Networks in NLP: The Curse of Indifferentiability

Lili Mou
doublepower.mou@gmail.com
http://sei.pku.edu.cn/~moull12

Outline

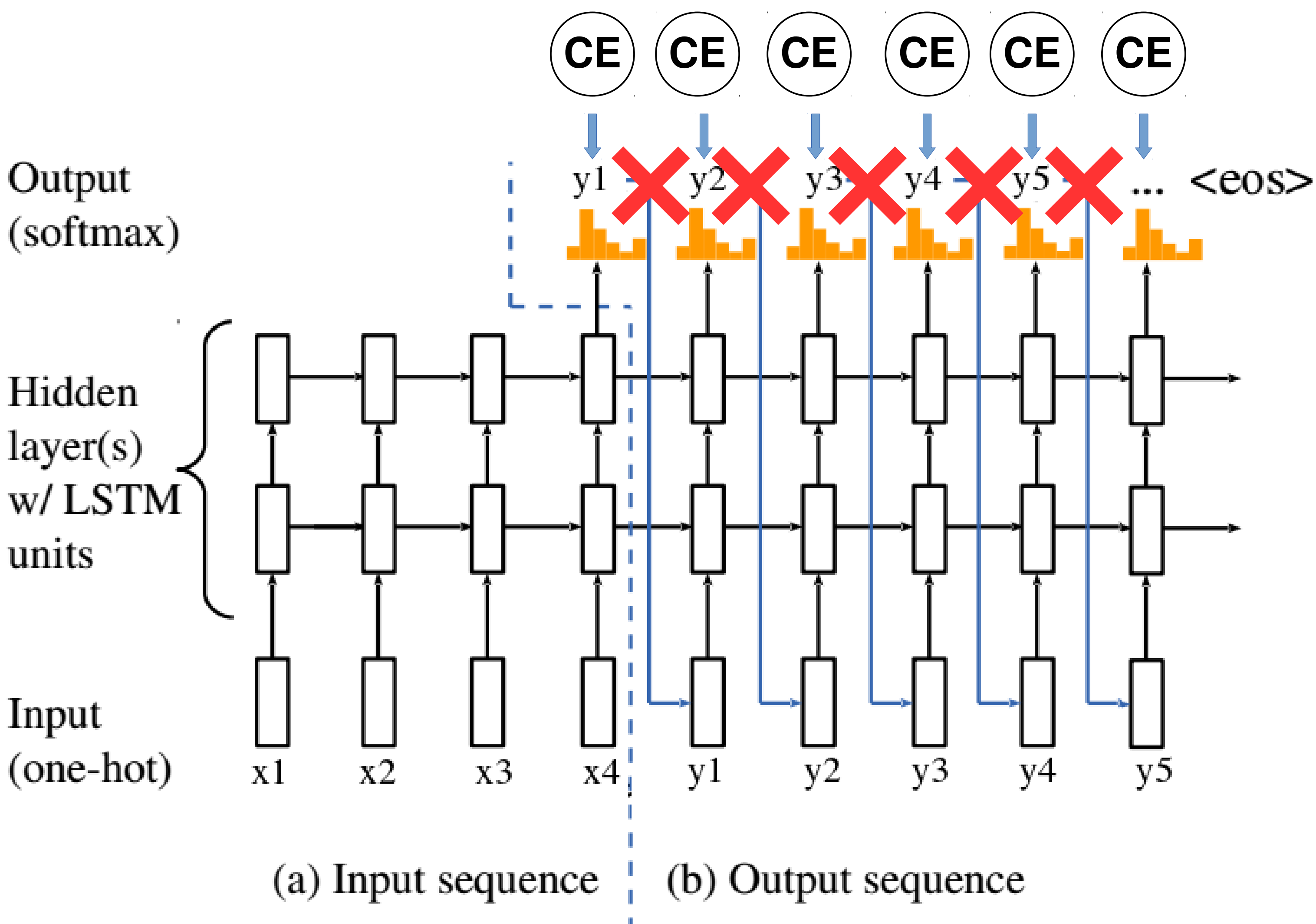
- Preliminary
- **Indifferentiability, solutions, and applications**
  - **The curse of indifferentiability**
  - Solutions: Attention, reinforcement learning, etc.
  - Applications: Sequence-level objective, SeqGAN, etc.
- A case study in semantic parsing

The Curse of Indifferentiability

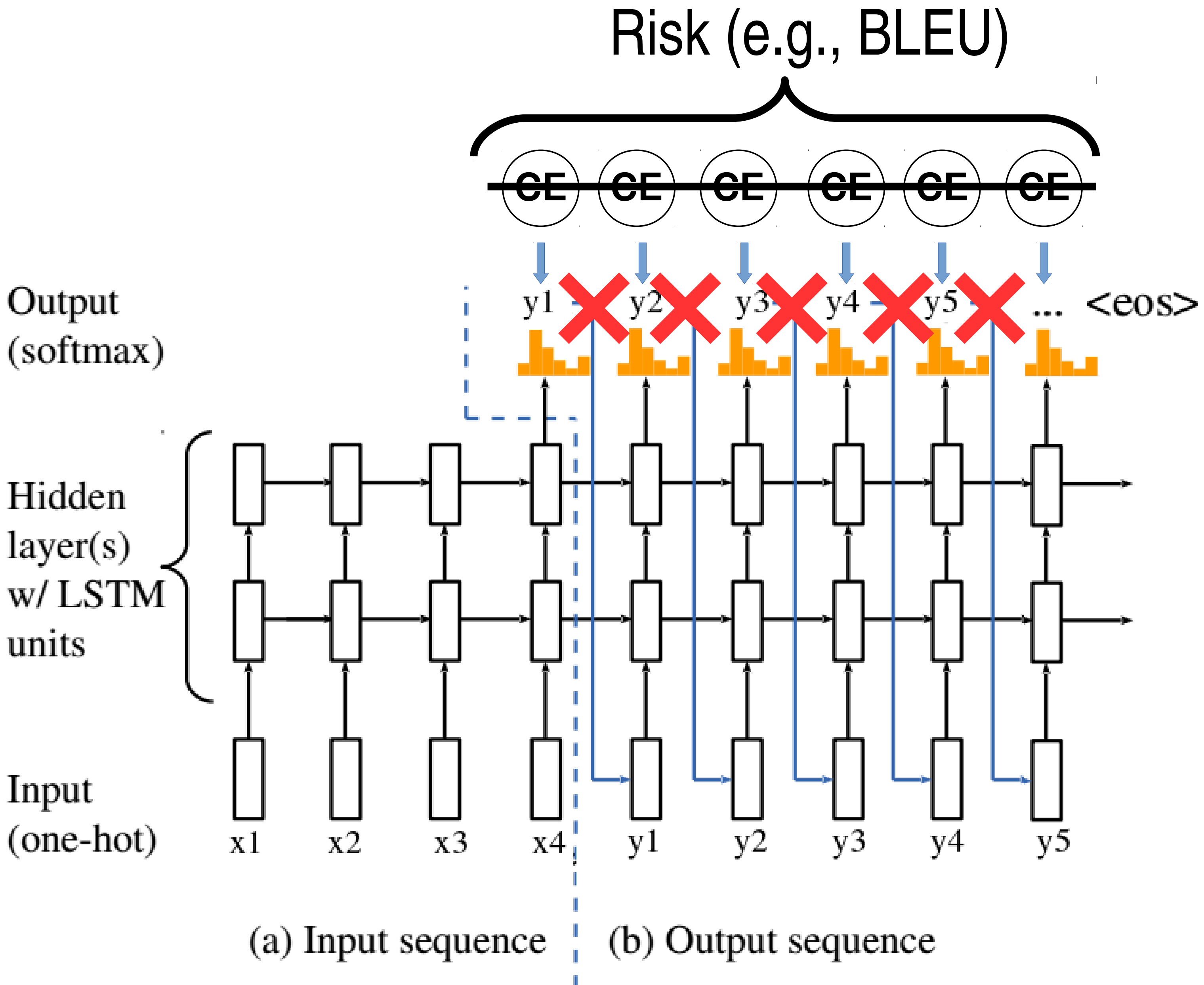
- Characters are discrete!
- Words are discrete!
- Phrases are discrete!
- Sentences are discrete!
- Paragraphs are discrete!
- All symbols are discrete!

- Word embeddings are continuous but are nothing!

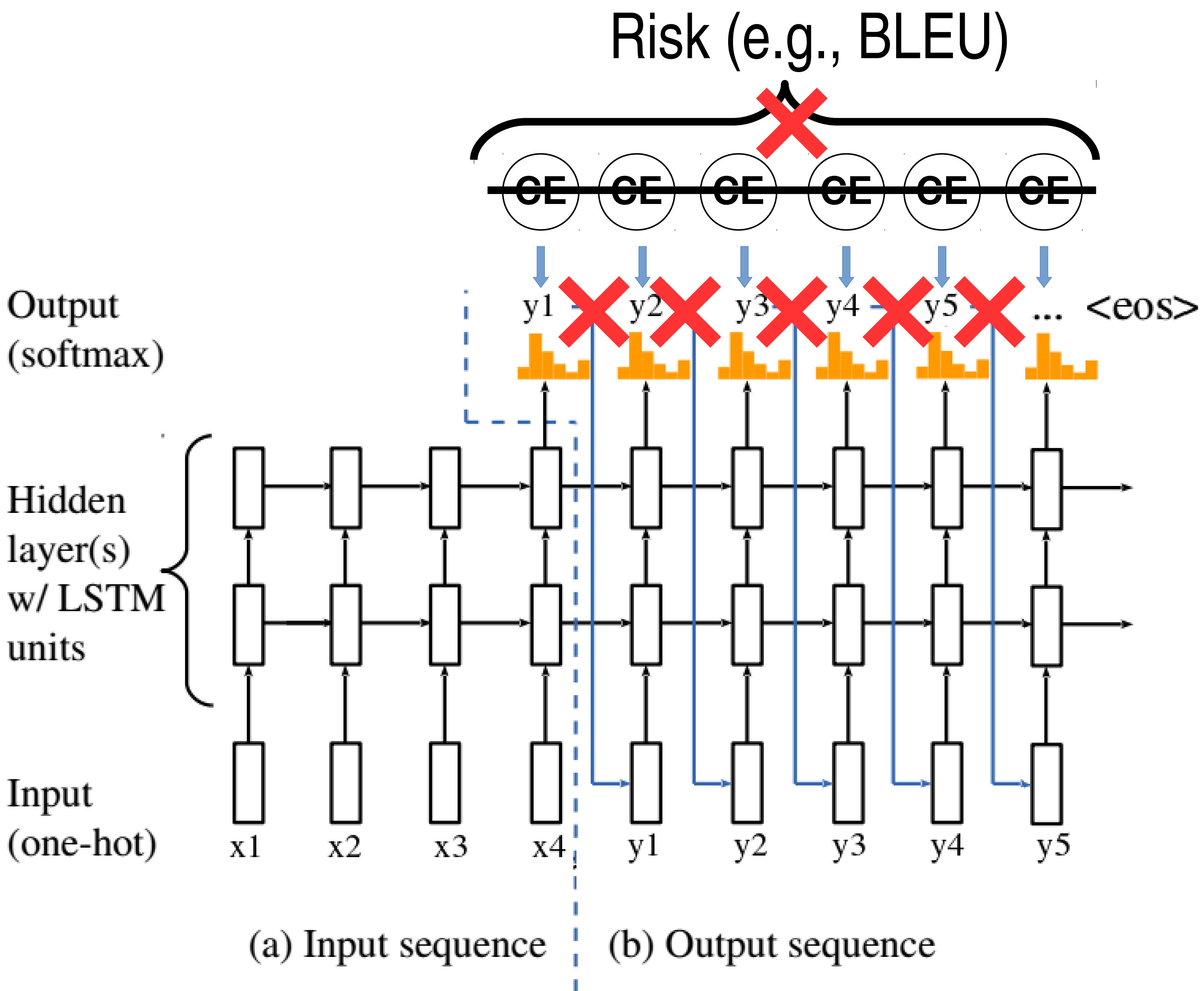
Indifferentiability



Indifferentiability



Indifferentiability



Indifferentiability

- Input: word embeddings 😋
- Output: argmax p(word) 🙀
- Risk: a function of output 🙀

Outline

- Preliminary
- **Indifferentiability, solutions, and applications**
  - The curse of indifferentiability
  - **Solutions: Attention, reinforcement learning, etc.**
  - Applications: Sequence-level objective, SeqGAN, etc.
- A case study in semantic parsing

Solution: Attempt #1

Classification of a particular word

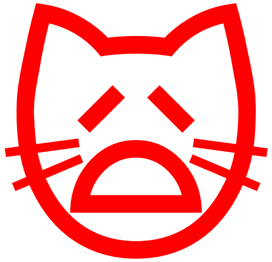
=> Regression of word embeddings

Solution: Attempt #1

Classification of a particular word

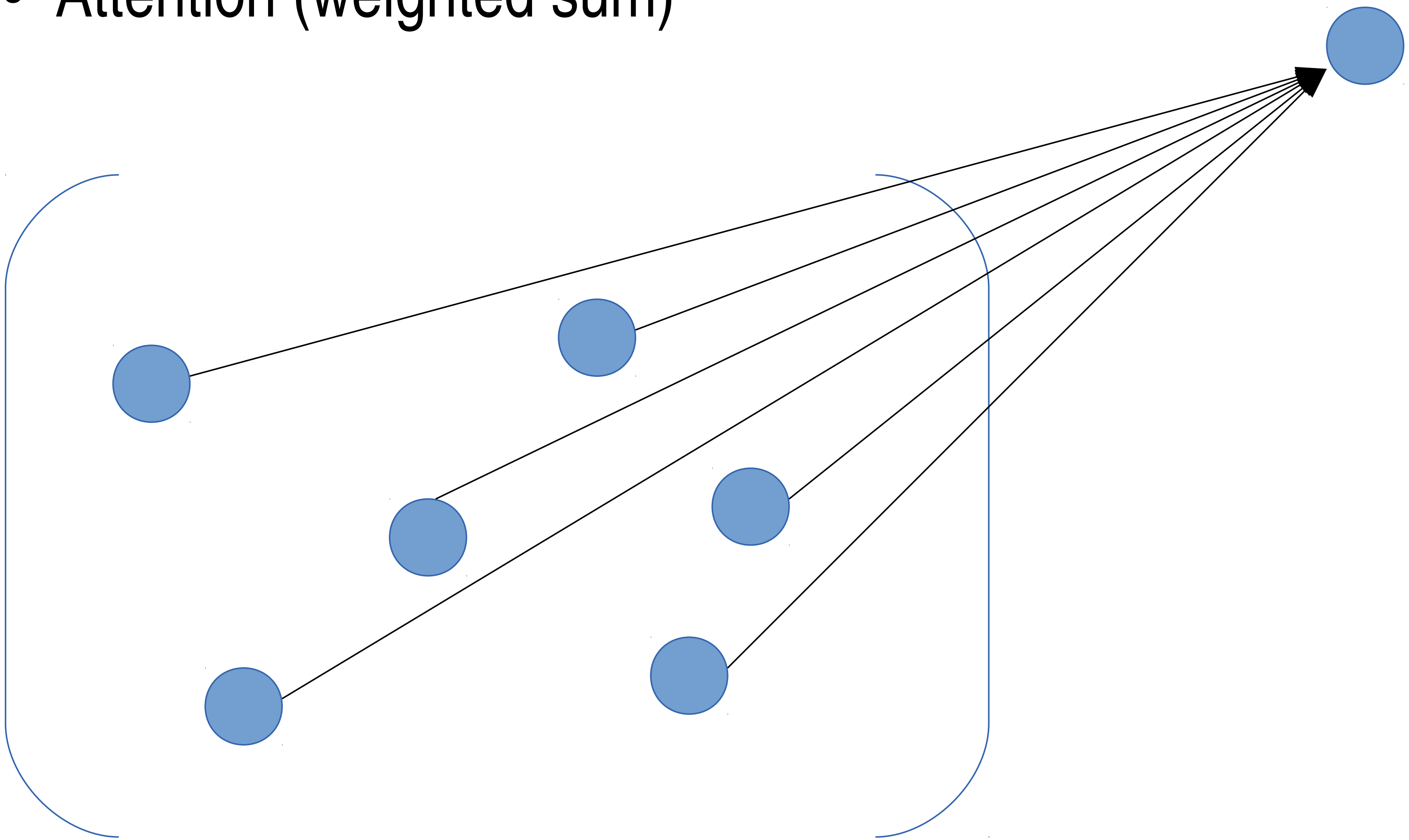
=> Regression of word embeddings

- Total failure (but why?)



Solution: Attempt #2

- Attention (weighted sum)



Solution: Attempt #3

- Reinforcement learning (Trial-and-error)
  - Sample an action (sequence)
  - See what the reward is

REINFORCE

Ranzato, Marc'Aurelio, et al. "Sequence Level Training with Recurrent Neural Networks." ICLR, 2016.

- Define an external cost function on a generated sequence
- Generate words by sampling
- Take the derivative of generated samples

$$L_\theta = -\sum_{w_1^g,\ldots,w_T^g} p_\theta(w_1^g,\ldots,w_T^g) r(w_1^g,\ldots,w_T^g) = -\mathbb{E}_{[w_1^g,\ldots w_T^g]\sim p_\theta} r(w_1^g,\ldots,w_T^g)$$

- $\partial J = \sum_{\mathbf{w}} [\partial\, p(\mathbf{w}|\ldots)]\, r(\mathbf{w}) = \sum_{\mathbf{w}} p(\mathbf{w})[\partial \log p(\mathbf{w})]\, r(\mathbf{w})$

Caveats

- REINFORCE may be extremely difficult to train
  - Hard to get started
  - Poor local optima
  - Sensitive to hyperparameters
- Supervised pretraining

Solution: Attempt #4

- Gumble softmax
  - Sample from a class distribution

$$z = \text{one\_hot}\left(\arg\max_i \left[g_i + \log \pi_i\right]\right)$$

where $g = -\log(-\log(\mathrm{u}))$

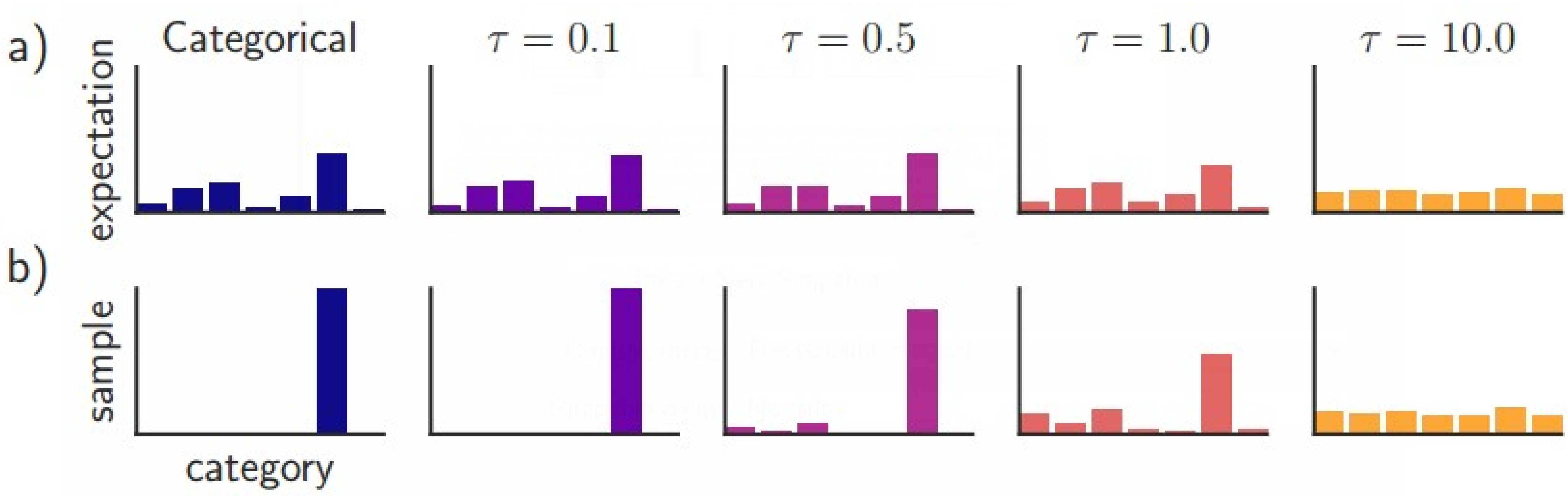
  - Softmax approximation

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)}$$

Jang, Eric, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax." ICLR, 2017.

Solution: Attempt #4

- Interpolation between onehot and uniform (with class distribution information)



Outline

- Preliminary
- **Indifferentiability, solutions, and applications**
  - The curse of indifferentiability
  - Solutions: Attention, reinforcement learning, etc.
  - **Applications: Sequence-level obj., SeqGAN, etc.**
- A case study in semantic parsing

Application: Sequence-Level Objective

- REINFORCE towards BLEU
- Annealing
  - For 1..T words
  - Supervised training: 1..t
  - RL: t+1..T

Results

TASK	XENT	DAD	E2E	MIXER
summarization	13.01	12.18	12.78	**16.22**
translation	17.74	20.12	17.77	**20.73**
image captioning	27.8	28.16	26.42	**29.16**

Shen, Shiqi, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. "Minimum risk training for neural machine translation." ACL, 2016.

System	Training	MT06	MT02	MT03	MT04	MT05	MT08
MOSES	MERT	32.74	32.49	32.40	33.38	30.20	25.28
RNNSEARCH	MLE	30.70	35.13	33.73	34.58	31.76	23.57
	MRT	37.34	40.36	40.93	41.37	38.81	29.23

Application: SeqGAN

Yu, Lantao, Weinan Zhang, Jun Wang, and Yong Yu. "Seqgan: sequence generative adversarial nets with policy gradient." In AAAI. 2017.

Generative Adversarial Network

- Two agents:
  - **G**enerative model: Generate new samples that are as similar as the data
  - **D**iscriminative model: Distinguish samples in disguise
- Each agent takes a step in turn

Objective of GAN

$$\min_G \max_D V(D,G) = \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{V(D,G)}$$

- G(z): A generated sample from distribution z
- D(x) = Estimated (by **D**) prob. that x is a real data sample
  - D(x)=1: **D** regards x as a training sample w.p.1
  - D(x)=0: **D** regards x as a generative sample w.p.1

Objective of GAN

$$\min_G \max_D V(D,G) = \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{V(D,G)}$$

Algorithm

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$\max_D V$

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Curse of Indifferentiability

$$\underbrace{\min_G \underbrace{\max_D V(D,G)}} = \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{V(D,G)}$$

Algorithm

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$\max_D V$

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Solution

- REINFORCE!

Table 2: Chinese poem generation performance comparison.

Algorithm	Human score	p-value	BLEU-2	p-value
MLE	0.4165	0.0034	0.6670	$< 10^{-6}$
SeqGAN	**0.5356**		**0.7389**	
Real data	0.6011		0.746	

Table 3: Obama political speech generation performance.

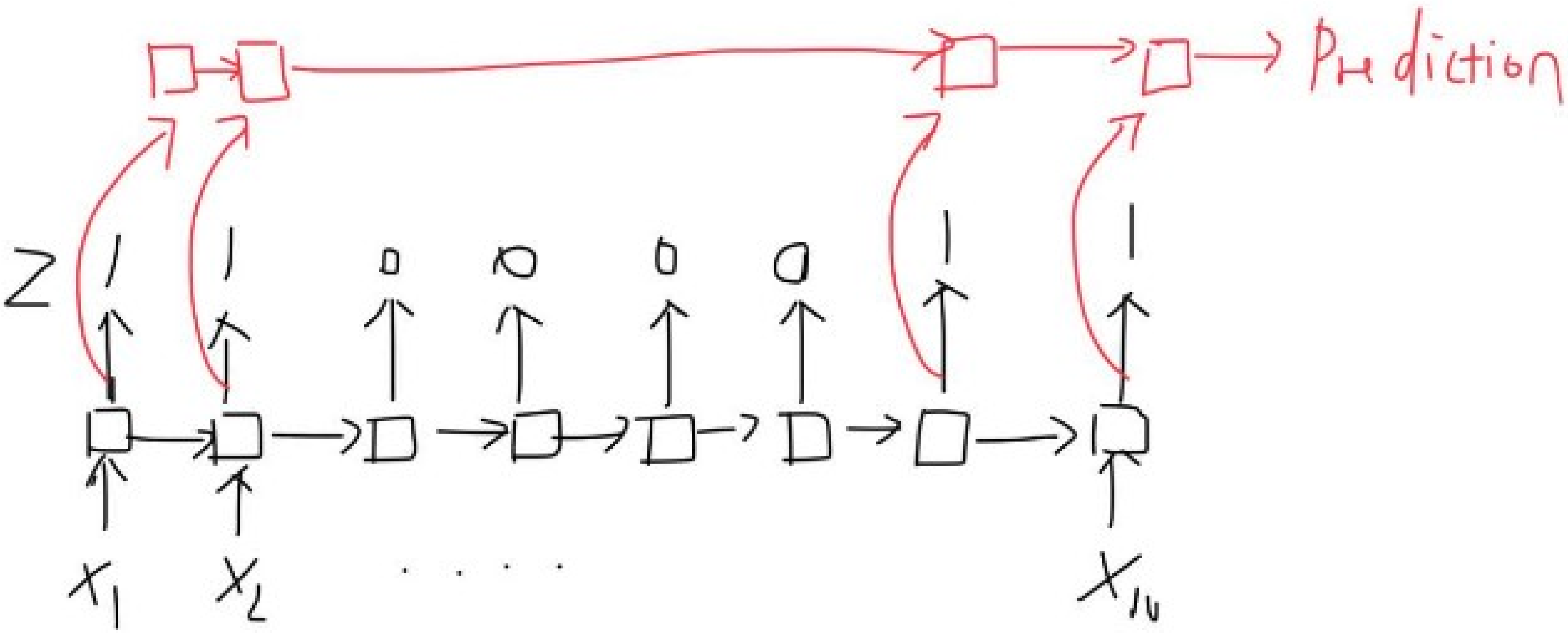
Algorithm	BLEU-3	p-value	BLEU-4	p-value
MLE	0.519	$< 10^{-6}$	0.416	0.00014
SeqGAN	**0.556**		**0.427**	

Table 4: Music generation performance comparison.

Algorithm	BLEU-4	p-value	MSE	p-value
MLE	0.9210	$< 10^{-6}$	22.38	0.00034
SeqGAN	**0.9406**		**20.62**	

- Does SeqGAN provide a more powerful density estimator?

Application: Rationale neural predictions



Lei, Tao, Regina Barzilay, and Tommi Jaakkola.
"Rationalizing neural predictions." EMNLP, 2016.

Objective

- $\mathcal{L}(\mathbf{z}, \mathbf{x}, \mathbf{y}) = \|\mathbf{enc}(\mathbf{z}, \mathbf{x}) - \mathbf{y}\|_2^2$
- $\Omega(\mathbf{z}) = \lambda_1 \|\mathbf{z}\| + \lambda_2 \sum_t |\mathbf{z}_t - \mathbf{z}_{t-1}|$
- $\text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y}) = \mathcal{L}(\mathbf{z}, \mathbf{x}, \mathbf{y}) + \Omega(\mathbf{z})$

Training

- REINFORCE!

Results

Red: appearance
Blue: Smell
Green: Palate

very dark beer . pours a nice finger and a half of creamy foam and stays throughout the beer . smells of coffee and roasted malt . has a major coffee-like taste with hints of chocolate . if you like black coffee , you will love this porter . creamy smooth mouthfeel and definitely gets smoother on the palate once it warms . it 's an ok porter but i feel there are much better one 's out there .