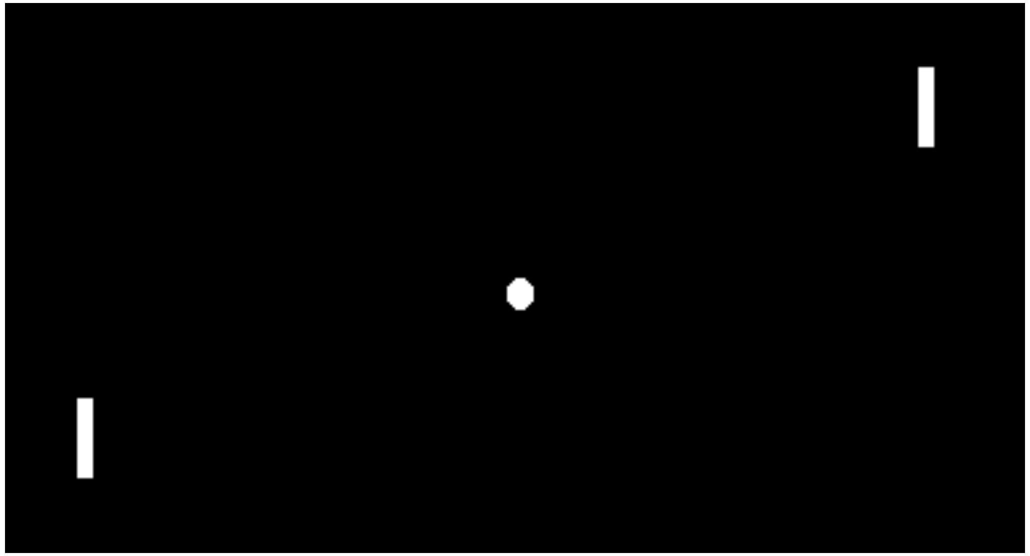
Reinforcement Learning with Policy Gradient

Contents

1. Popular application of RL
2. Human Vs RL agent
3. The basic RL with PG algorithm
4. RL rescues non-differentiable computation
5. Conclusion

Popular application of RL (1/5)



ATARI games

AlphaGo

Human:

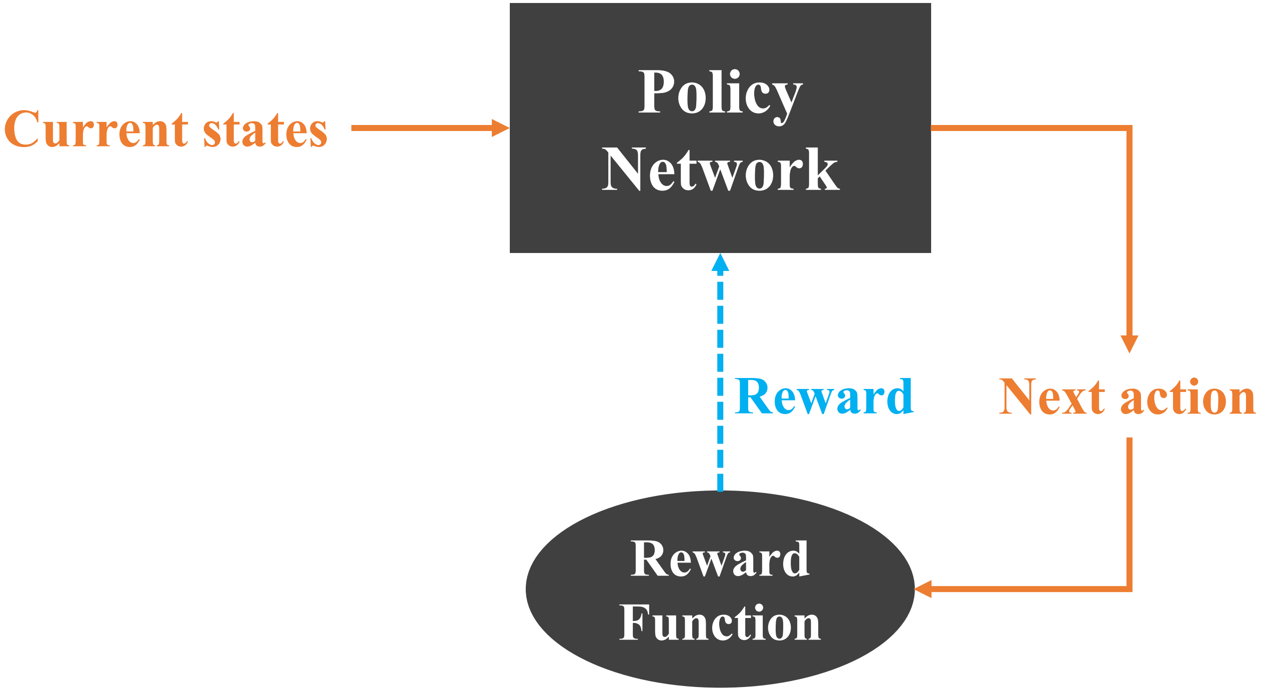
- Learn from experience
- Learn from rules
- Learn from knowledge base

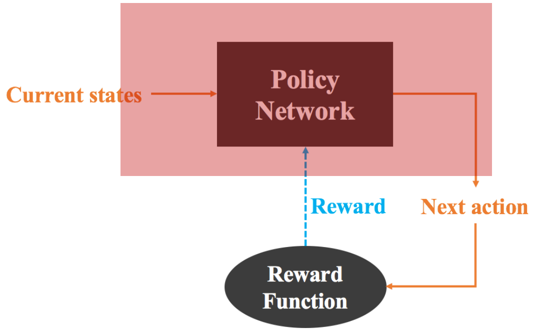
Vs

RL agent:

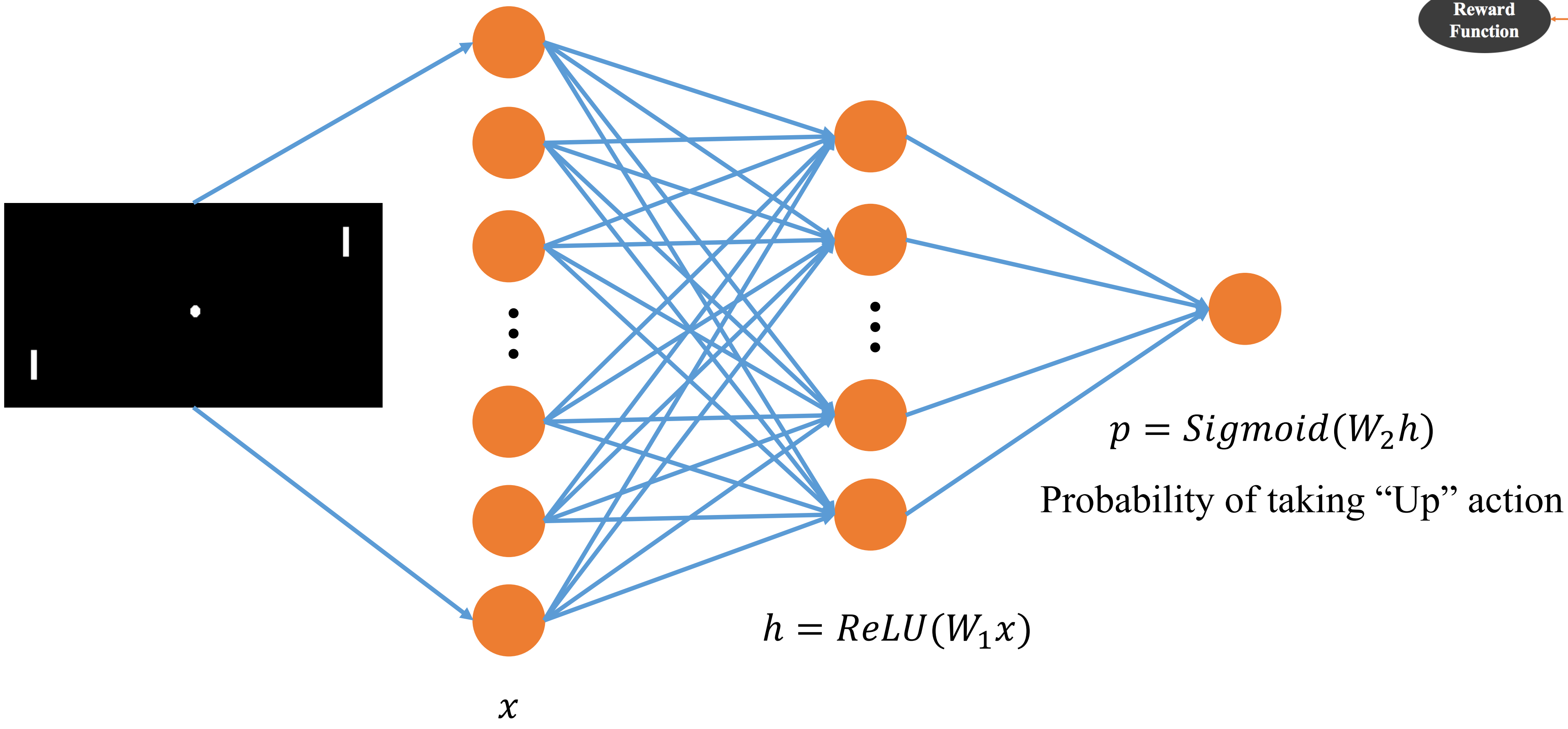
- Learn from experience

Overall Structure of RL with PG

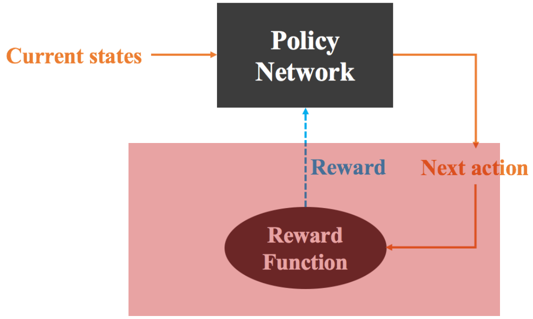


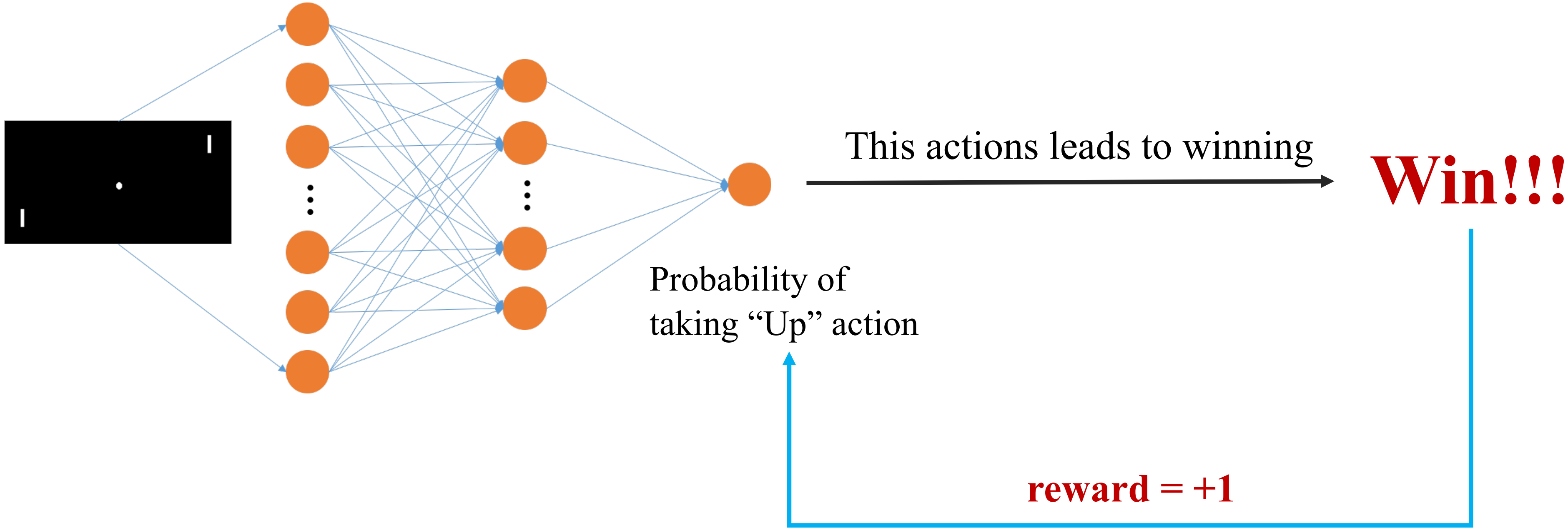


Detailed Feed Forward



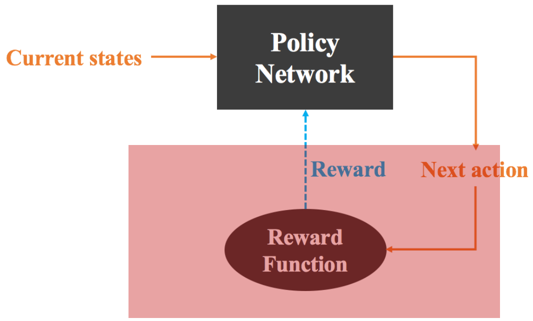
Detailed Back Propagation

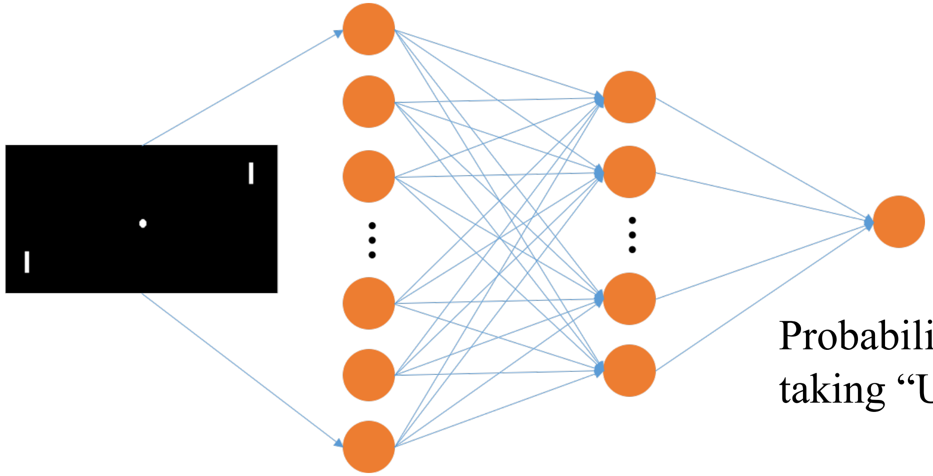




Reward for winning is +1
Reward for losing is -1
Otherwise is 0

Detailed Back Propagation





Probability of
taking “Up” action

This actions leads to winning

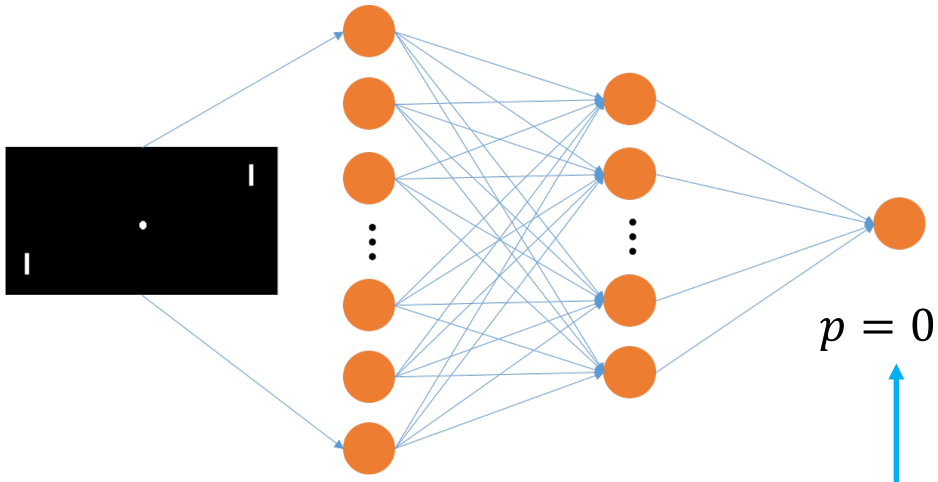
Win!!!

Reward for winning is +1
Reward for losing is -1
Otherwise is 0

reward = +1

How to combine the reward into the gradients in NN???

Supervised Learning Vs Reinforcement Learning



$p = 0.8$

Back propagate through the NN

Generated action = “Up”

Up	**0.8**
Down	0.2

X-entropy loss

Ground truth = “Up”

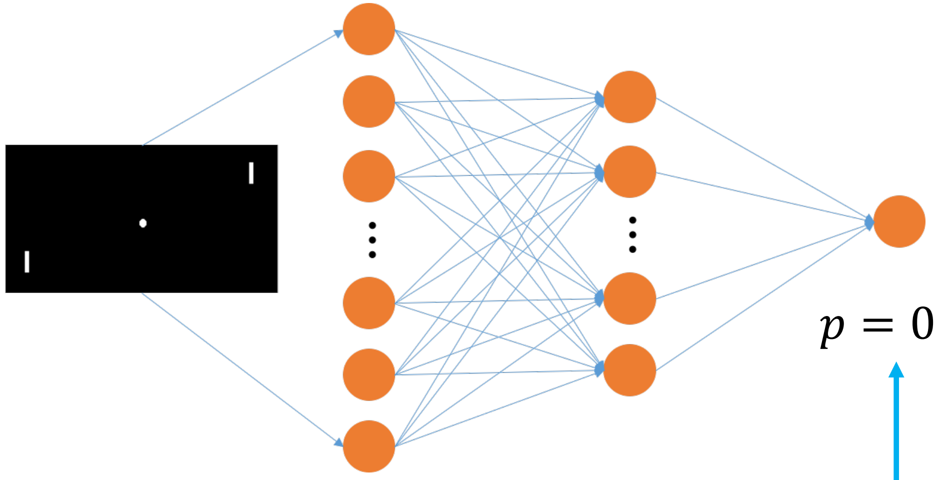
Up	**1.0**
Down	0.0

Gradients

Up	-0.2
Down	0.2

The gradient from loss to probability

Supervised Learning Vs **Reinforcement Learning**



$p = 0.8$

Back propagate through the NN

Generated action = "Up"

Up	**0.8**
Down	0.2

Win!!!

Reward = +1

Fake gradients

Up	1
Down	0

The **fake** gradient (**reward**) from **reward function** to probability

Mathematical backup of “fake gradient”

Some notations:

$$f(x): Reward\ function$$

$$p(x): The\ probability\ of\ taking\ an\ action$$

The goal is to:

$$Max\{E_{x\sim p(x)}[f(x)]\}$$

Calculate the partial derivation of $f(x)$ on θ

$$\nabla_\theta E_x[f(x)] = \nabla_\theta \sum_x p(x) f(x)$$

definition of expectation

Calculate the partial derivation of $f(x)$ on θ

$$\nabla_\theta E_x[f(x)] = \boxed{\nabla_\theta} \boxed{\sum_x} p(x) f(x)$$

definition of expectation

Calculate the partial derivation of $f(x)$ on θ

$$\nabla_\theta E_x[f(x)] = \nabla_\theta \sum_x p(x) f(x) \qquad \text{definition of expectation}$$

$$= \sum_x \nabla_\theta p(x) f(x) \qquad \text{swap sum and gradient}$$

Calculate the partial derivation of $f(x)$ on θ

$$\nabla_\theta E_x[f(x)] = \nabla_\theta \sum_x p(x) f(x)$$ definition of expectation

$$= \sum_x \boxed{\nabla_\theta p(x)} f(x)$$ swap sum and gradient

$$\boxed{p(x) \frac{\nabla_\theta p(x)}{p(x)}}$$

Calculate the partial derivation of $f(x)$ on θ

$$\nabla_\theta E_x[f(x)] = \nabla_\theta \sum_x p(x) f(x)$$ definition of expectation

$$= \sum_x \nabla_\theta p(x) f(x)$$ swap sum and gradient

$$= \sum_x p(x) \frac{\nabla_\theta p(x)}{p(x)} f(x)$$ both multiply and divide by $p(x)$

Calculate the partial derivation of $f(x)$ on θ

$$\nabla_\theta E_x[f(x)] = \nabla_\theta \sum_x p(x) f(x)$$ definition of expectation

$$= \sum_x \nabla_\theta p(x) f(x)$$ swap sum and gradient

$$= \sum_x p(x) \boxed{\frac{\nabla_\theta p(x)}{p(x)}} f(x)$$ both multiply and divide by $p(x)$

$$\updownarrow$$

$$\boxed{\nabla_\theta \log[p(x)]}$$

Calculate the partial derivation of $f(x)$ on θ

$$\nabla_\theta E_x[f(x)] = \nabla_\theta \sum_x p(x) f(x)$$ definition of expectation

$$= \sum_x \nabla_\theta p(x) f(x)$$ swap sum and gradient

$$= \sum_x p(x) \frac{\nabla_\theta p(x)}{p(x)} f(x)$$ both multiply and divide by $p(x)$

$$= \sum_x p(x) \nabla_\theta \log p(x) f(x)$$ use the fact that $\nabla_\theta \log(z) = \frac{1}{z} \nabla_\theta z$

Calculate the partial derivation of $f(x)$ on θ

$$
\begin{aligned}
\nabla_\theta E_x[f(x)] &= \nabla_\theta \sum_x p(x) f(x) && \text{definition of expectation} \\
&= \sum_x \nabla_\theta p(x) f(x) && \text{swap sum and gradient} \\
&= \sum_x p(x) \frac{\nabla_\theta p(x)}{p(x)} f(x) && \text{both multiply and divide by } p(x) \\
&= \sum_x p(x) \boxed{\nabla_\theta \log p(x) f(x)} && \text{use the fact that } \nabla_\theta \log(z) = \frac{1}{z} \nabla_\theta z
\end{aligned}
$$

Calculate the partial derivation of $f(x)$ on θ

$$\begin{aligned}
\nabla_\theta E_x[f(x)] &= \nabla_\theta \sum_x p(x) f(x) && \text{definition of expectation} \\
&= \sum_x \nabla_\theta p(x) f(x) && \text{swap sum and gradient} \\
&= \sum_x p(x) \frac{\nabla_\theta p(x)}{p(x)} f(x) && \text{both multiply and divide by } p(x) \\
&= \sum_x p(x) \nabla_\theta \log p(x) f(x) && \text{use the fact that } \nabla_\theta \log(z) = \frac{1}{z} \nabla_\theta z \\
&= E_x[f(x) \nabla_\theta \log p(x)] && \text{definition of expectation}
\end{aligned}$$

The gradient of $f(x)$ *on* $\theta \approx f(x) *$ the gradient of log probability

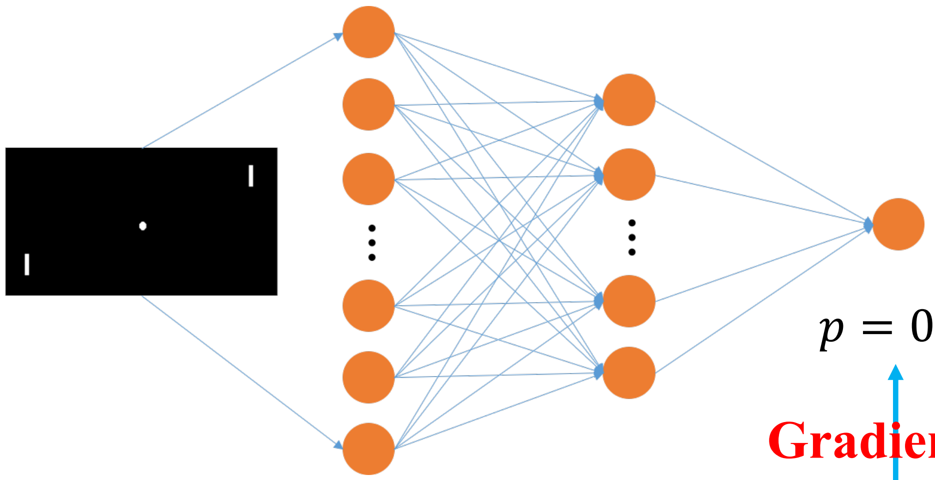
Partial derivation of $f(x)$ on θ:

$$\nabla_\theta E_x[f(x)] = E_x[f(x)\nabla_\theta \log p(x)]$$

The loss function:

$$L(\theta) = \sum f(x) \log p(x;\theta)$$

Supervised Learning Vs **Reinforcement Learning**



$p = 0.8$

Back propagate through the NN

Generated action = “Up”

Up	**0.8**
Down	0.2

Win!!!

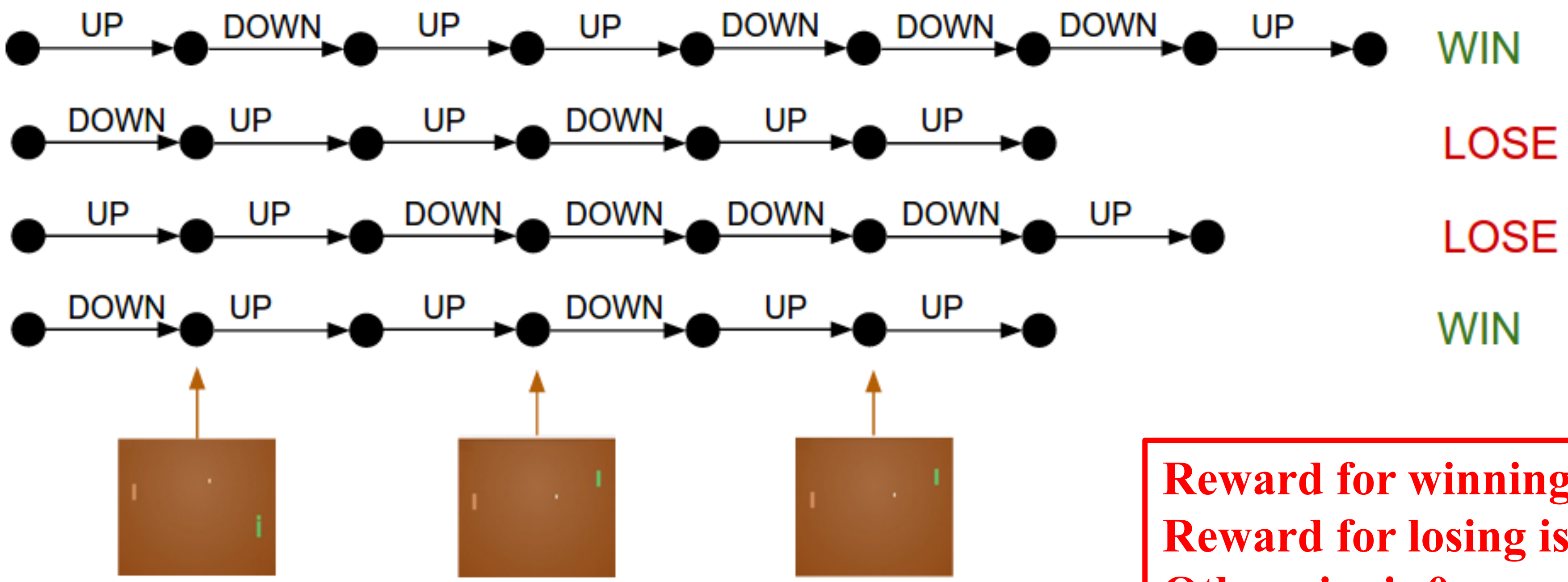
Reward = +1

Up	1
Down	0

Gradient ≈ reward

The reward could be regarded as gradient from loss to log probability

Calculate rewards manually

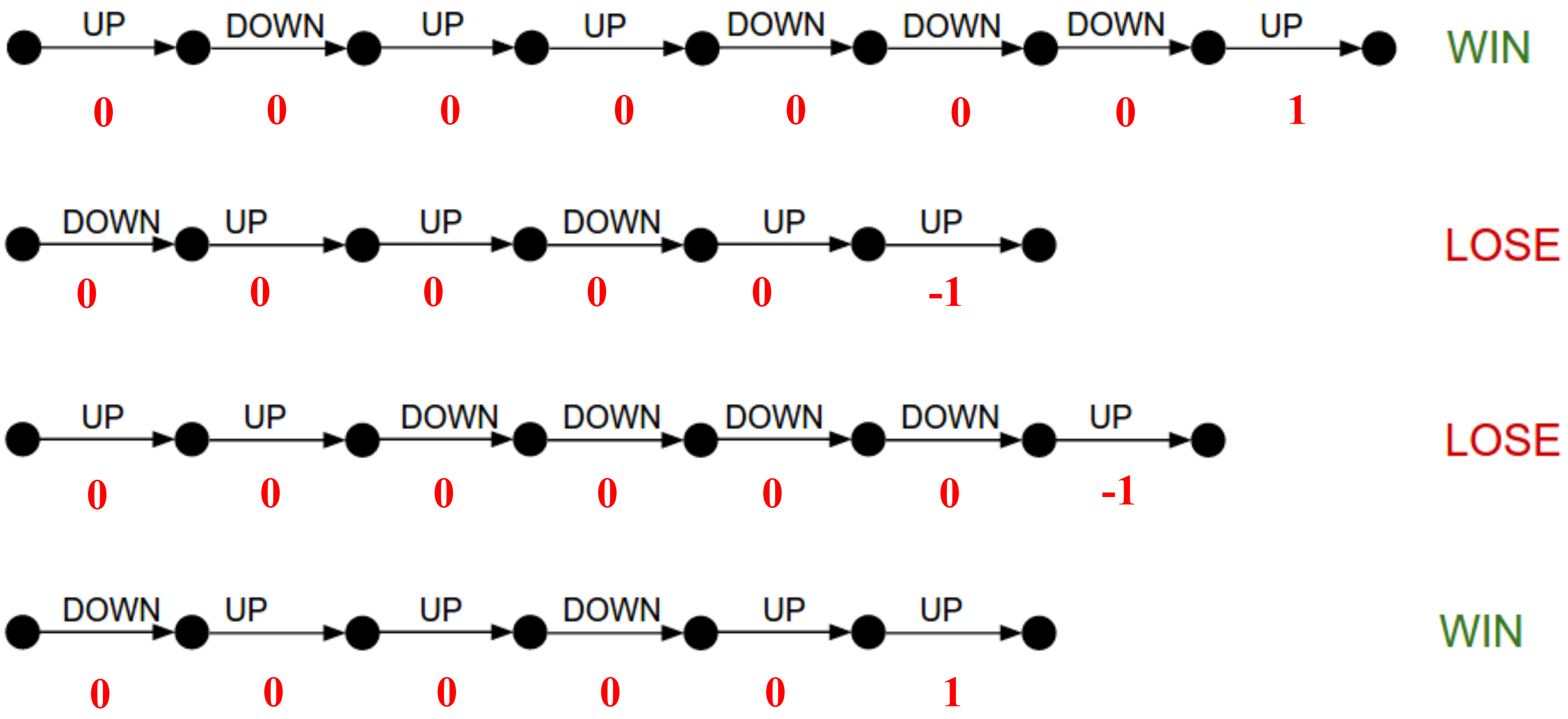


Reward for winning is +1
Reward for losing is -1
Otherwise is 0

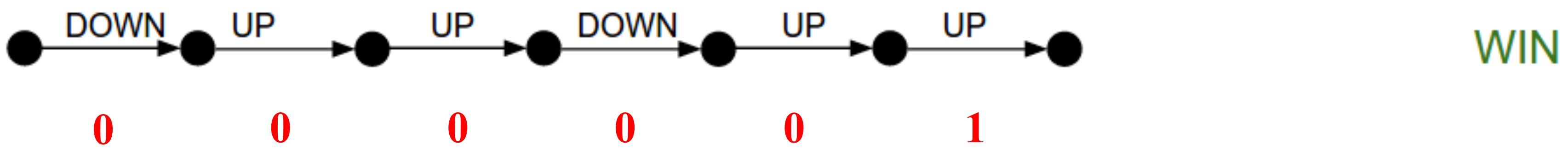
What are the rewards for these four sets of policies?

Reward for winning is +1
Reward for losing is -1
Otherwise is 0

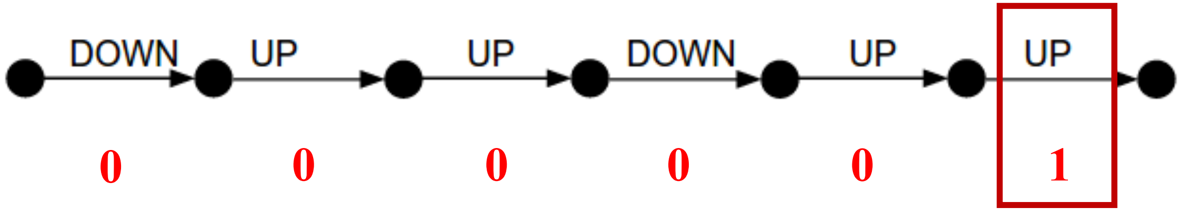
Calculate rewards manually



Discounted rewards



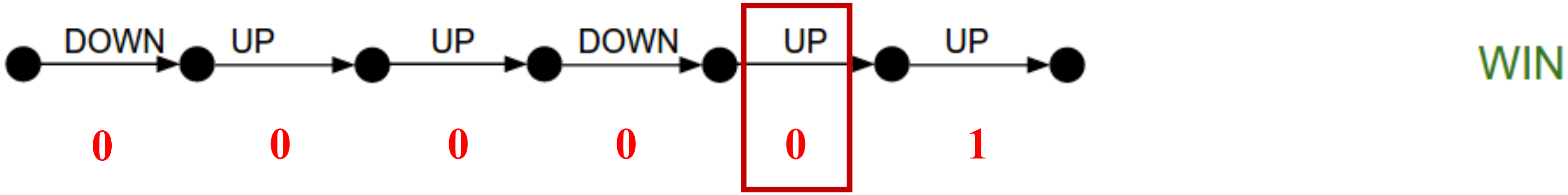
Discounted rewards



WIN

$r_5 = \gamma * 0 + r_5 = 1$ $(\gamma = 0.99)$

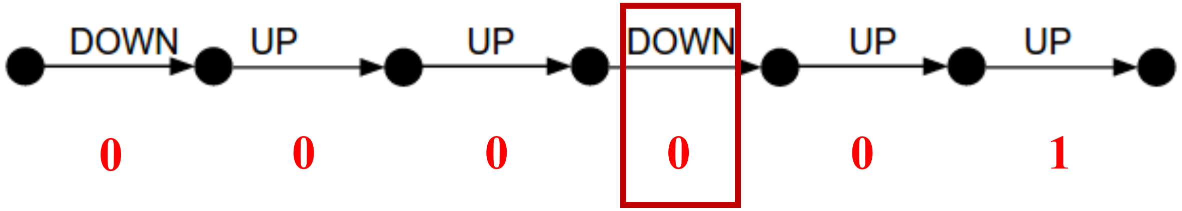
Discounted rewards



$$r_5 = \gamma * 0 + r_5 = 1 \qquad (\gamma = 0.99)$$

$$r_4 = \gamma * r_5 + r_4 = 0.99$$

Discounted rewards



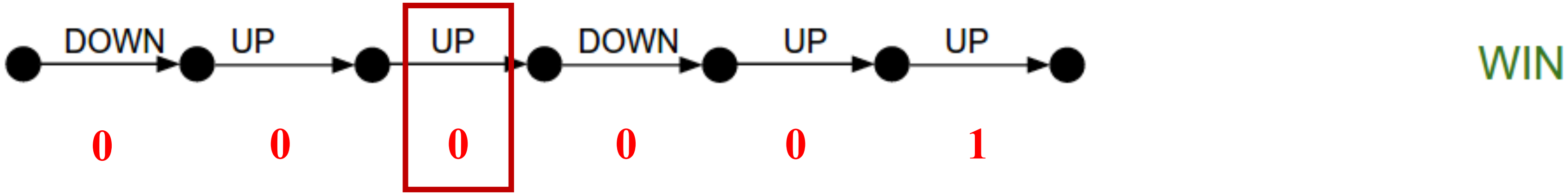
WIN

$$r_5 = \gamma * 0 + r_5 = 1 \qquad (\gamma = 0.99)$$

$$r_4 = \gamma * r_5 + r_4 = 0.99$$

$$r_3 = \gamma * r_4 + r_3 = 0.99^2$$

Discounted rewards



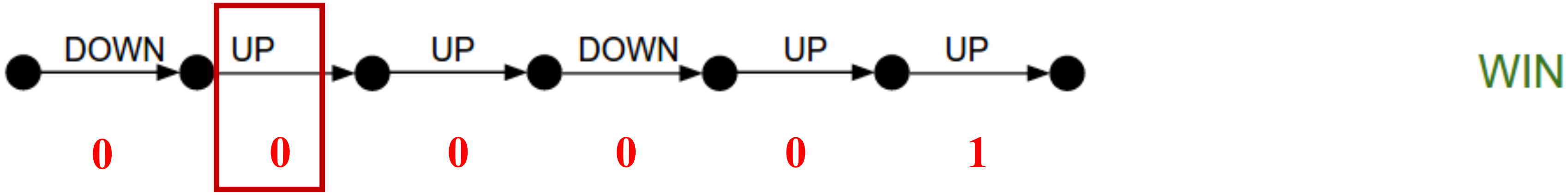
$$r_5 = \gamma * 0 + r_5 = 1 \qquad (\gamma = 0.99)$$

$$r_4 = \gamma * r_5 + r_4 = 0.99$$

$$r_3 = \gamma * r_4 + r_3 = 0.99^2$$

$$r_2 = \gamma * r_3 + r_2 = 0.99^3$$

Discounted rewards



$$r_5 = \gamma * 0 + r_5 = 1 \qquad (\gamma = 0.99)$$

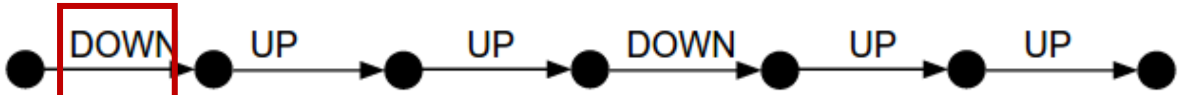
$$r_4 = \gamma * r_5 + r_4 = 0.99$$

$$r_3 = \gamma * r_4 + r_3 = 0.99^2$$

$$r_2 = \gamma * r_3 + r_2 = 0.99^3$$

$$r_1 = \gamma * r_2 + r_1 = 0.99^4$$

Discounted rewards



WIN

0 0 0 0 0 1

$$r_5 = \gamma * 0 + r_5 = 1 \qquad (\gamma = 0.99)$$

$$r_4 = \gamma * r_5 + r_4 = 0.99$$

$$r_3 = \gamma * r_4 + r_3 = 0.99^2$$

$$r_2 = \gamma * r_3 + r_2 = 0.99^3$$

$$r_1 = \gamma * r_2 + r_1 = 0.99^4$$

$$r_0 = \gamma * r_1 + r_0 = 0.99^5$$

Discounted rewards

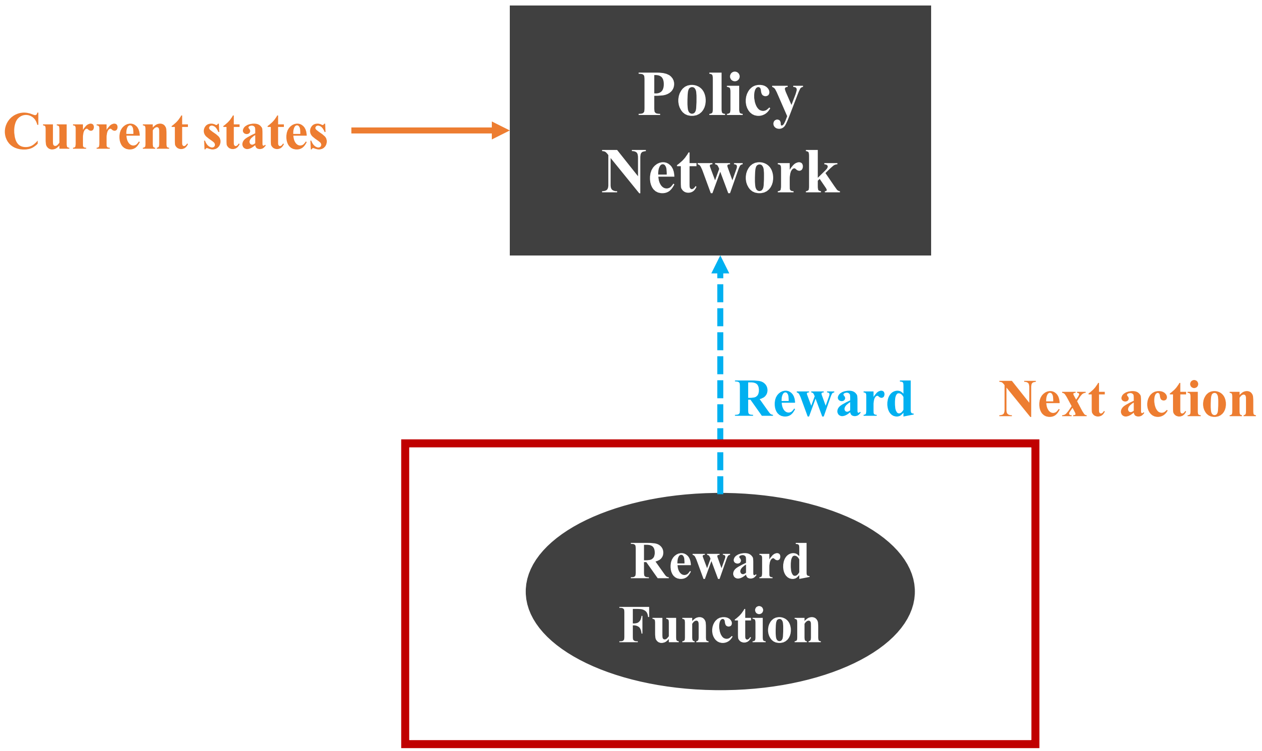
WIN

Original rewards:	0	0	0	0	0	1
Discounted rewards:	0.99^5	0.99^4	0.99^3	0.99^2	0.99	1

Congratulations!!!

We have finished the basic RL with PG

Hope I have made it clear ☺



No care whether it is differentiable or not

Sequence Level Training With Recurrent Neural Networks

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, Wojciech Zaremba

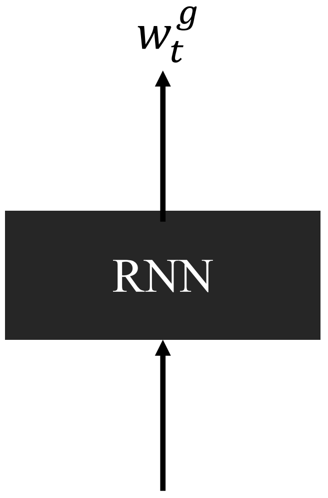
Existing limitations in text generation

Existing limitations in text generation

1. The input data distribution in training is different from that in testing
2. The evaluation metric in training is different from that in testing
3. The evaluation metric in training is word-level based

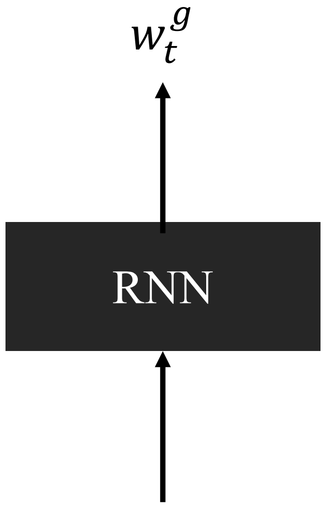
Input data distribution

In training:



w_{t-1} from data distribution

In testing:



w_{t-1}^g from model distribution

Evaluation Metric

In training:

1. Cross entropy loss
2. Word-level based

In testing:

1. BLEU or ROUGE-2…
2. Sequence-level based

Evaluation Metric

In training:

1. Cross entropy loss
2. Word-level based

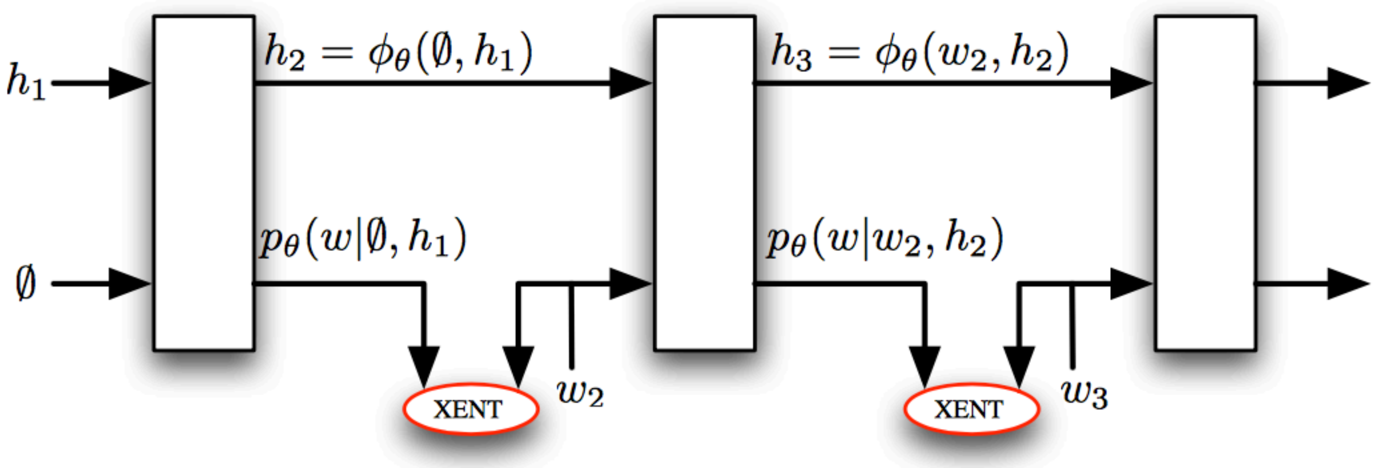
In testing:

1. BLEU or ROUGE-2…
2. Sequence-level based

The longer you trained, the more this kind of errors accumulated ☹

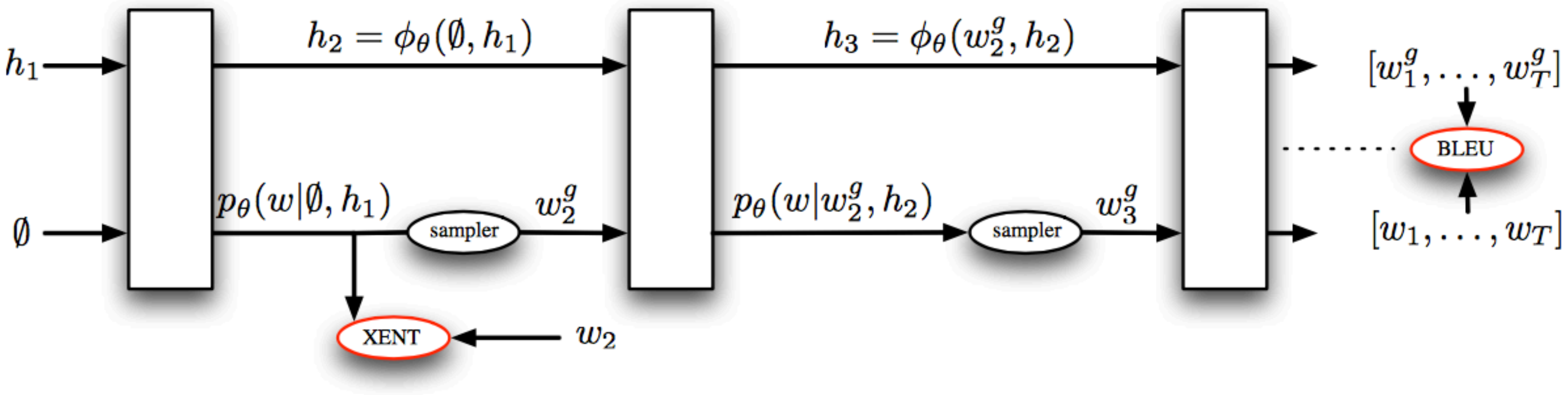
MIXER here to rescue

(Mixed Incremental Cross-Entropy Reinforce)

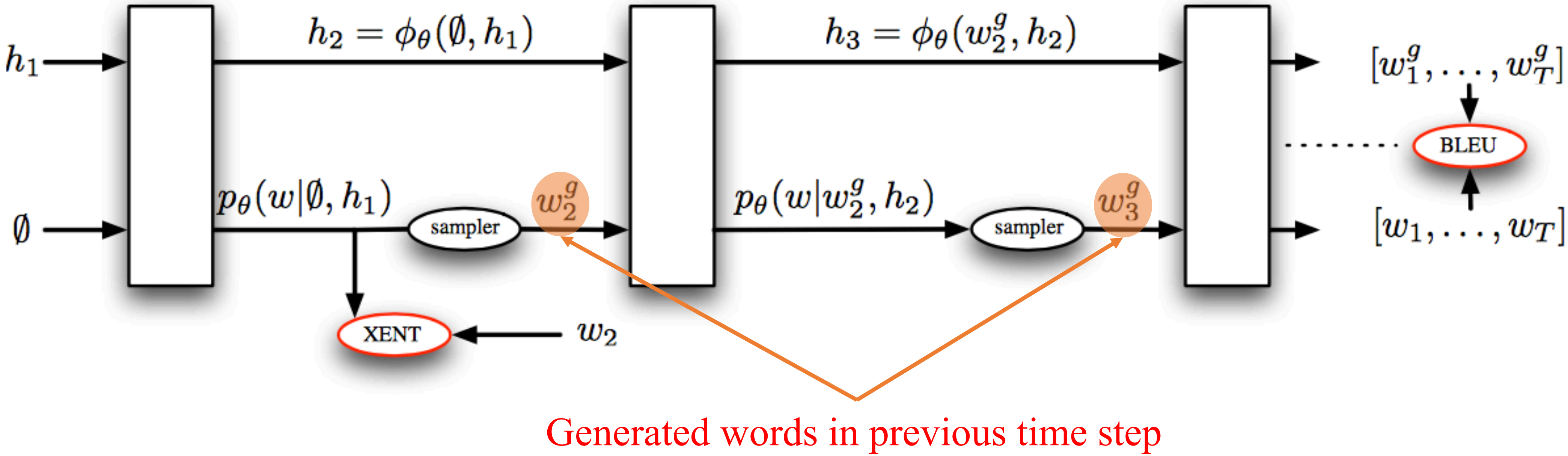


The unfolded normal RNN for text generation

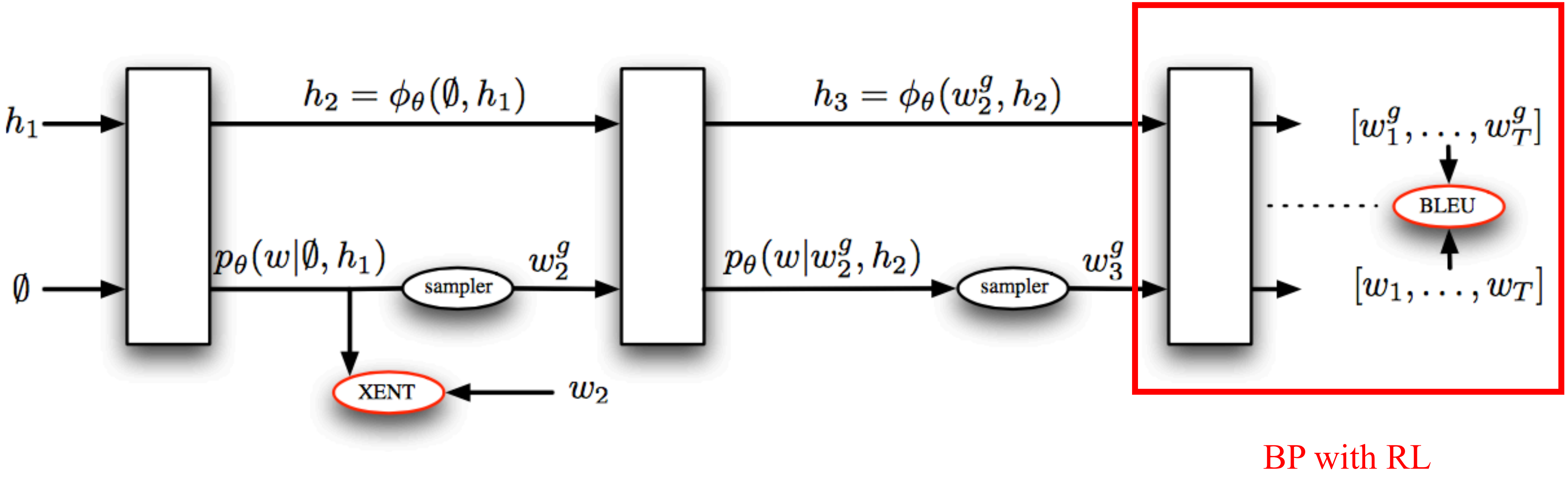
MIXER Overall Structure



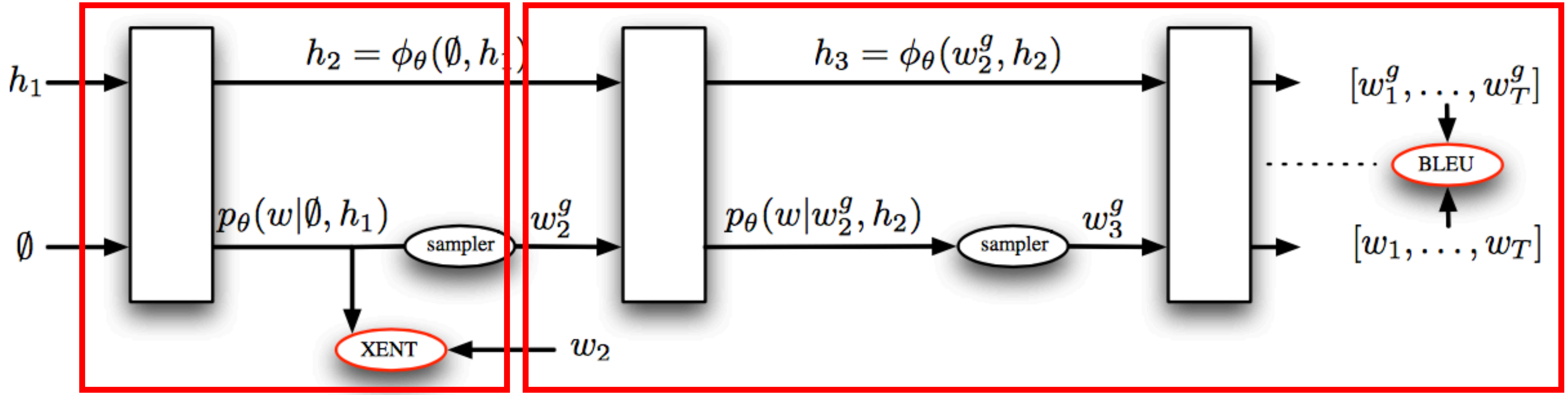
MIXER Overall Structure



MIXER Overall Structure



MIXER Overall Structure



First s steps trained with normal RNN

Next $\mathrm{T} - s$ steps trained with RNN-RL

Partial derivation of $\boldsymbol{f(x)}$ on $\boldsymbol{\theta}$:

$$\nabla_\theta E_x[f(x)] = E_x[f(x)\nabla_\theta \log p(x)]$$

The loss function:

$$L(\theta) = \sum f(x) \log p(x; \theta)$$

RL gradient details

Objective function:

$$L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s)]$$

Partial derivation:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w_s \sim p_\theta}[r(w^s)\nabla_\theta \log p_\theta(w^s)]$$

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w_s \sim p_\theta}[(r(w^s) - b)\nabla_\theta \log p_\theta(w^s)]$$

Chain rule:

$$\nabla_\theta L(\theta) = \sum_{t=1}^{T} \frac{\partial L(\theta)}{\partial s_t} \frac{\partial s_t}{\partial \theta}$$

$$\frac{\partial L(\theta)}{\partial s_t} \approx (r(w^s) - b)\boxed{(p_\theta(w_t|h_t) - 1_{w_t^s})}$$

Normal multi-class derivation

RL gradient details

Objective function:

$$L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s)]$$

Partial derivation:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w_s \sim p_\theta}[r(w^s)\nabla_\theta \log p_\theta(w^s)]$$

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w_s \sim p_\theta}[(r(w^s) - \boxed{b})\nabla_\theta \log p_\theta(w^s)]$$

Chain rule:

$$\nabla_\theta L(\theta) = \sum_{t=1}^{T} \frac{\partial L(\theta)}{\partial s_t} \frac{\partial s_t}{\partial \theta}$$

$$\frac{\partial L(\theta)}{\partial s_t} \approx (r(w^s) - \boxed{b})(p_\theta(w_t|h_t) - 1_{w_t^s})$$

Average reward at time t+1

Tricks in MIXER

The search space for RL agent is too large in text generation.

Therefore the RNN is initialized with optimal parameters, which means it is pre-trained using normal RNN structures

Existing limitations in text generation

1. The input data distribution in training is different from that in testing
2. The evaluation metric in training is different from that in testing
3. The evaluation metric in training is word-level based

Addressed by MIXER

1. The input data distribution in training is similar to that in testing
2. The evaluation metric in training is the same to that in testing, BLEU…
3. The evaluation metric in training is sequence-level based, BLEU…

Conclusions

1. Reinforcement learning agent learns from experience with interacting with reward function;
2. The reward function is no need to be differentiable;
3. Reinforcement learning has applied in lots of tasks in text generation

Thank you ☺