Implement Adversarial Training for Text Generation

jcykcai

Maximum Likelihood Estimate (MLE)

- Do not leave GAN alone
- The story starts from Maximum Likelihood Estimate (MLE) training
- The problems of MLE:
  - Exposure bias
  - Goal Mismatch (Perplexity vs. Task-specific goal)

Reinforcement Learning (RL)

- Better Objective, BELU?
- Take sequence generation procedure as a **sequential decision** making process
  - Reward: BELU? ROUGE? F-score?
  - Action: word selection at each step
  - State: partially-generated sentence

Reinforcement Learning (RL)

- An example (REINFORCE + Policy Gradient)
  - Train a generation model G to generate response based on $x \in D_x$.

$$J(\theta) = \mathbb{E}_{x \in D_x} \mathbb{E}_{y \in G(x)} S(x, y)$$

$$\nabla J(\theta) = \mathbb{E}_{x \in D_x} \mathbb{E}_{y \in G(x)} \nabla \log p_\theta(y \mid x) S(x, y)$$

Generative Neural Network (GAN)

- The GAN framework
- Does not use a manually designed objective
- Directly aims at narrowing the gap between the distributions of real data and the generated.

Generative Neural Network (GAN)

- A discriminator is trained to differentiate the true data and the generated data.

- A generative model is trained to fool the discriminator.

- A minmax game

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

Non-differentiable Problem

- GANs require differentiation through generated units.
- In the case of text generation, generated units are discrete
  - A distribution over whole vocabulary => word index
  - Break the the back-propagation

Non-differentiable Problem

- Typically, in a word-by-word generation process.
- The hidden state h_t of the decoder is used to draw a distribution over the vocabulary.
- $d_t = \textbf{softmax}(Wh_t + b)$
- The generated word y_t could be decided by argmax function or random sample, making y_t a one-hot vector.

$$\arg\max(x)_i = \mathbb{I}[x_i = \max(x)]$$

Non-differentiable Problem

- Solutions
  - Continuous approximation
  - Gumbel Softmax trick
  - Reinforcement Learning

Continuous Approximation

- Output a mix of words, rather than a single word.
- Apply the distribution d_t directly
- The mix of words: $\sum_{i=1}^{|V|} e_i \cdot (\textbf{softmax}(Wh_t + b))_i$

Continuous Approximation

- Approach argmax function

$$\sum_{i=1}^{|V|} e_i \cdot (\textbf{softmax}((Wh_t + b)/\tau))_i$$

- $\tau \in (0,\infty)$ is the temperature.

- The softmax function approaches argmax as $\tau \to 0$, and it becomes uniform when $\tau \to \infty$

Gumbel Softmax trick

- Gumbel transforms sampling from a categorical distribution to an optimization problem.

- It is proved that: $y \sim \textbf{softmax}(a)$ is equivalent to

$y = \arg\max(g + a), g \sim$ **Gumbel i.i.d.**

- The Gumbel distribution is defined by:

$$g_i = -\log(-\log(u_i)), u_i \sim \textbf{Uniform}(0,1)$$

Gumbel Softmax trick

$$y = \arg\max(g + a), g \sim \textbf{Gumbel i.i.d.}$$

- Solve the argmax by temperature trick

$$y = \textbf{softmax}((g + a)/\tau), g \sim \textbf{Gumbel i.i.d.}$$

- Straight Through (ST) Gumbel

```
def gumbel_softmax(logits, temperature, hard = False, eps = 1e-20):
    sample = Variable(torch.rand(logits.size())).cuda()
    gumbel_logits = (logits - torch.log(- torch.log(sample + eps) +eps))/ temperature
    y = F.softmax(gumbel_logits, dim = -1)
    if hard:
        hard_y = torch.zeros_like(y)
        _. idx = torch.max(y, dim= -1, keepdim = True)
        hard_y.scatter_(-1, idx, 1.0)
        y = (hard_y -y).detach_() + y
    return y
```

Reinforcement Learning

- Recall the GAN formulation

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

- G is trained to maximize $\mathbb{E}_{\mathbf{z}\in p_{\mathbf{z}}(\mathbf{z})}[\log D(G(\mathbf{z}))]$
- Take $\log D(G(\mathbf{z}))$ as reward in RL settings.

Reinforcement Learning

- An example (REINFORCE + Policy Gradient)
  - Train a generation model G to generate response based on $x \in D_x$.

$$J(\theta) = \mathbb{E}_{x \in D_x} \mathbb{E}_{y \in G(x)} S(x, y)$$

$$\nabla J(\theta) = \mathbb{E}_{x \in D_x} \mathbb{E}_{y \in G(x)} \nabla \log p_\theta(y \mid x) S(x, y)$$

  - Now $S(x, y) = D(x, y)$ is implemented by the adversarial trained discriminator.

Reinforcement Learning

- Baseline model to reduce the variance of the estimate.

$$\nabla J(\theta) = \mathbb{E}_{x \in D_x} \mathbb{E}_{y \in G(x)} \nabla \log p_\theta(y \mid x)(S(x, y) - baseline(x, y))$$

- Intermediate rewards
  - Monte Carlo Search
  - Apply full awards to partially decoded sentence

Back to our start point

- MLE is important.
- Most successful models are pretrained using MLE and alternately update itself using MLE and Adversarial Training.

References

- Goodfellow, Ian, et al. "Generative adversarial nets." *NIPS*. 2014

- Yu, Lantao, et al. "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient." *AAAI*. 2017.

- Maddison, Chris J., Andriy Mnih, and Yee Whye Teh. "The concrete distribution: A continuous relaxation of discrete random variables." *arXiv preprint arXiv: 1611.00712* (2016).

- Jang, Eric, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax." *arXiv preprint arXiv:1611.01144* (2016).

- Xu, Zhen, et al. "Neural response generation via gan with an approximate embedding layer." *EMNLP*. 2017.

- Li, Jiwei, et al. "Adversarial learning for neural dialogue generation." *EMNLP.* 2017.