1 Research Question

Research question: Discrete generation on text with GAN.

Difficulties:

- Hard to define a differential loss function on the generated discrete text sequence.

2 Method Overview

- Overview:
  - Use importance sampling to produce a differential continuous loss on the generator
  - In addition to the previous technique, use multiple tricks, including Monte Carlo Tree Search, to further improve the performance.

3 Policy gradient

- Setting: RNN generator that can produce text sequences $X_t = \{x_{t1}, x_{t2}..x_{tm}\}$. Let $G(z_t; \theta) = \mathbb{P}(X = X_t : \{x_{t1}, x_{t2}..x_{tm}\})$ represents the generator.

- Problem: $x_i$ after softmax is a single index, can’t back-propagate to the RNN model.

- Reinforcement Learning:
  - Use RL to train the RNN model.
  - Treat the prediction from discriminator $D$ as the reward $r$.
  - Use $r$ to update the parameter $\theta$ in $G$

- Policy Gradient
Suppose in total $t$ trails, each time a sample $X_t$ is generated, and $D$ gives reward $r(x_t)$, we have:

$$\nabla L_G(\theta) = \nabla \theta \mathbb{E}_t [r(X_t)]$$

$$= \nabla \theta \sum_t \mathbb{P}(X = X_t) r(X_t)$$

$$= \sum_t \nabla \theta \mathbb{P}(X = X_t) r(X_t)$$

$$= \sum_t \mathbb{P}(X = X_t) \nabla \theta \log \mathbb{P}(X = X_t) r(X_t)$$

$$= \mathbb{E}_t [\nabla \theta \log \mathbb{P}(X = X_t) r(X_t)]$$

$$= \mathbb{E}_t [\nabla \theta \log G(z_t; \theta) r(X_t)]$$

(1)

Following equation 1, the gradient on the model parameter can be estimated using the reward over samples.

In real practise to reduce variance, people use $\sum_t (\frac{r_t}{\sum_t r_t} - b)$ instead of using $\sum_t r_t$ directly, where $b$ is a scalar "baseline". In the paper, they varies $b$ from 0 to 1.

Similar technique is also used in SeqGAN[1] and many other discrete generation papers.

4 Importance Sampling

- Importance sampling: Estimating properties of a particular distribution, while only having samples generated from a different distribution than the distribution of interest.

- In GAN, suppose $p_d(X)$ is the distribution of real data. $p_\theta(X)$ is the distribution of generator. The optimal $D$ will have the property $D(X) = \frac{p_d}{p_d + p_\theta}$.

- Therefore, $p_d = \frac{D(X)}{1 - D(X)} p_\theta$.

- Let $r_D(X) = \frac{D(X)}{1 - D(X)}$. Keep an older copy of $p_\theta$ as $p'$ whose parameter is $\theta'$. Then define $q(X) = \frac{r_D(X)}{Z(\theta')} p'(X)$, $q(X)$ is a "fixed" distribution that approximates the data distribution disregard the moving of $p_\theta(X)$. $Z(\theta')$ is a normalization term.
• Define the loss

\[ L_G(\theta) = \text{KL}(q(X)||p_\theta(X)) \]

\[ = \mathbb{E}_{q(X)}[\log p_\theta(X)] - H(q(X)) \]

\[ = \mathbb{E}_{q'(X)}[\frac{q_\theta(X)}{p'(X)} \log p_\theta(X)] - H(q(X)) \]

\[ \approx \frac{1}{Z(\theta')} \mathbb{E}_{p_\theta(X)}[r_D(X) \log p_\theta(X)] - H(q(X)) \]  

(2)

\[ H(q(X)) \text{ and } Z(\theta') \text{ are not relevant with } p_\theta. \]

• The loss used in iteration becomes \( \mathbb{E}_{p_\theta(X)}[r_D(X) \log p_\theta(X)] \)

• Exactly the same form with policy gradient, if reward \( r = \frac{D(X)}{1-D(X)} \)

5 Algorithm of MaliGAN-Basic

Algorithm 1 MaliGAN

Require: A generator \( p \) with parameters \( \theta \). A discriminator \( D(x) \) with parameters \( \theta_d \). A baseline \( b \).

1: \textbf{for} number of training iterations \textbf{do}
2: \hspace{1em} \textbf{for} \( k \) steps \textbf{do}
3: \hspace{2em} Sample a minibatch of samples \( \{x_i\}_{i=1}^m \) from \( p_\theta \).
4: \hspace{2em} Sample a minibatch of samples \( \{y_i\}_{i=1}^m \) from \( p_d \).
5: \hspace{2em} Update the parameter of discriminator by taking gradient ascent of discriminator loss

\[ \sum_i \nabla_{\theta_d} \log D(y_i) + \sum_i \nabla_{\theta_d} \log(1 - D(x_i)) \]

6: \hspace{2em} \textbf{end for}
7: \hspace{2em} Sample a minibatch of samples \( \{x_i\}_{i=1}^m \) from \( p_\theta \).
8: \hspace{2em} Update the generator by applying gradient update

\[ \sum_{i=1}^m (\sum_i r_D(x_i) - b) \nabla \log p_\theta(x_i) \]

9: \hspace{1em} \textbf{end for}

6 Tricks

6.1 Monte Carlo Tree Search

• Use a cumulative loss to estimate the total loss

\[ E_{p_\theta}(r_D(X)\nabla p(X)) = E_{p_\theta}(\sum_{t=1}^L Q(a_t, s_t)\nabla p_\theta(a_t|s_t)) \]
• I have doubt with this part in the paper. I believe simple Monte Carlo method is enough for the estimation. Applying MCTS will falsely give a biased estimation over the reward.

6.2 Mixed MLE training
• Variance of training is too large.
• Fix the input to use the training data for $N$ time steps, only let the model generating the remaining $T - N$ time steps. $N$ gradually reduce, from $T$ to 0.

6.3 Single Real-data based re-normalization
• Two layer estimation to reduce variance.
• In each mini-batch, this paper first draws a mini-batch of samples (e.g. 32) of high-level latent variables, then for each high level value draws a number of low-level data samples.
7 Algorithm of MaliGAN with tricks

Algorithm 2 Sequential MaliGAN with Mixed MLE
Training

Require: A generator $p$ with parameters $\theta$.
A discriminator $D(x)$ with parameters $\theta_d$.
Maximum sequence length $T$, step size $K$.
A baseline $b$, sampling multiplicity $m$.

1: $N = T$
2: Optional: Pretrain model using pure MLE with some epochs.
3: for number of training iterations do
4: $N = N - K$
5: for k steps do
6: Sample a minibatch of sequences $\{y_i\}_{i=1}^m$ from $p_d$.
7: While keeping the first $N$ steps the same as $\{y_i\}_{i=1}^m$, sample a minibatch of sequences $\{x_i\}_{i=1}^m$ from $p_\theta$ from
time step $N$.
8: Update the discriminator by taking gradient ascend of
discriminator loss.
\[
\sum_i [\nabla_{\theta_d} \log D(y_i)] + \sum_i [\nabla_{\theta_d} \log(1 - D(x_i))]
\]
9: end for
10: Sample a minibatch of sequences $\{x_i\}_{i=1}^m$ from $p_d$.
11: For each sample $x_i$ with length larger than $N$ in the mini-
batch, clamp the generator to the first $N$ words of $s$, and freely run the model to generate $m$ samples $x_{i,j}, j = 1, \cdots m$ till the end of the sequence.
12: Update the generator by applying the mixed MLE-Mali
gradient update
\[
\nabla L_G^N \approx \sum_{i=1,j=1}^{m,n} \left( \frac{r_D(x_{i,j})}{\sum_j r_D(x_{i,j})} - b \right) \nabla \log p_\theta(x_{i,j}^{>N} | x_{i,j}^{\leq N})
+ \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^N p_\theta(a_{i,t}^j | s_i^t)
\]
13: end for

8 Evaluation:

8.1 Dataset
Discrete MNIST, Poem Generation Dataset and Penn Treebank (PTB) Dataset.
- Discrete MNIST: Treat MNIST as a discrete task.
- Penn Treebank (PTB) Dataset:
8.2 Models in comparison

- MLE
- SeqGAN[1]
- MaliGAN-Basic
- MaliGAN-Full

8.3 Result

1. Result of Poetry generation dataset.

   Table 1. Experimental results on Poetry Generation task. The result of SeqGAN is directly taken from (Yu et al., 2017).

<table>
<thead>
<tr>
<th>Model</th>
<th>Poem-5 BLEU-2</th>
<th>Poem-5 PPL</th>
<th>Poem-7 BLEU-2</th>
<th>Poem-7 PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE</td>
<td>0.6934</td>
<td>564.1</td>
<td>0.3186</td>
<td>192.7</td>
</tr>
<tr>
<td>SeqGAN</td>
<td>0.7389</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MaliGAN-basic</td>
<td>0.7406</td>
<td>548.6</td>
<td>0.4892</td>
<td>182.2</td>
</tr>
<tr>
<td>MaliGAN-full</td>
<td>0.7628</td>
<td>542.7</td>
<td>0.5526</td>
<td>180.2</td>
</tr>
</tbody>
</table>

2. Basic version of MaliGAN doesn’t show much improvement on MLE.

3. According to paper, MCTS works very slow, limits the capacity on large dataset.

4. Result of Penn Treebank Dataset.

   Table 2. Experimental results on PTB. Note that we evaluate the models in sentence-level.

<table>
<thead>
<tr>
<th></th>
<th>MLE</th>
<th>MaliGAN-basic</th>
<th>MaliGAN-full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid-Perplexity</td>
<td>141.9</td>
<td>131.6</td>
<td>128.0</td>
</tr>
<tr>
<td>Test-Perplexity</td>
<td>138.2</td>
<td>125.3</td>
<td>123.8</td>
</tr>
</tbody>
</table>

References