RL and GAN for Sentence Generation and Chat-bot

Hung-yi Lee
Outline

• Policy Gradient

• SeqGAN
  • Two techniques: MCMC, partial
  • Experiments: SeqGAN and dialogue

• Original GAN
  • MadliGAN
  • Gumbel
Review: Chat-bot

• Sequence-to-sequence learning

Training data:

A: OOO
B: XXX
A: Δ Δ Δ

Encoder
Input sentence
history information

Generator
output sentence

A: Δ Δ Δ
A: OOO
B: XXX
Review: Encoder

Hierarchical Encoder

Encoder

to generator
Review: Generator

can be different with attention mechanism
Review: Training Generator

\[ C = \sum_{t} C_t \]

Minimizing cross-entropy of each component

Reference:

A

B

\[ C_1 \]

A

B

\[ C_2 \]

A

B

\[ C_3 \]

A

B

: condition from decoder

&lt;BOS&gt;
Review: Training Generator

Training data: \((h, \hat{x})\)

\[
C = \sum_{t} C_t
\]

\(\hat{x}_t\): t-th word, \(\hat{x}_{1:t}\): first t words of \(\hat{x}\)

\[
C_t = -\log P_\theta(\hat{x}_t | \hat{x}_{1:t-1}, h)
\]

\[
C = -\sum_{t} \log P(\hat{x}_t | \hat{x}_{1:t-1}, h)
\]

\[
= -\log P(\hat{x}_1 | h) P(\hat{x}_t | \hat{x}_{1:t-1}, h)
\]

\[
= -\log P(\hat{x} | h)
\]

Maximizing the likelihood of generating \(\hat{x}\) given \(h\)
RL for Sentence Generation

Introduction

• Machine obtains feedback from user

How are you?
Bye bye 😊

-10

Hello
Hi 😊

3

• Chat-bot learns to maximize the *expected reward*
Maximizing Expected Reward

\[ \theta^* = \arg \max_{\theta} \bar{R}_\theta \]

Maximizing expected reward

\[ \bar{R}_\theta = \sum_h P(h) \sum_x R(h, x) P_\theta(x|h) \]

Randomness in generator

Probability that the input/history is \( h \)
Maximizing Expected Reward

\[ \theta^* = \arg \max_{\theta} \bar{R}_\theta \]

Maximizing expected reward

\[ \bar{R}_\theta = \sum_h P(h) \sum_x R(h, x) P_{\theta}(x|h) = E_{h \sim P(h)} \left[ E_{x \sim P_{\theta}(x|h)}[R(h, x)] \right] \]

\[ = E_{h \sim P(h), x \sim P_{\theta}(x|h)}[R(h, x)] \approx \frac{1}{N} \sum_{i=1}^{N} R(h^i, x^i) \]

Sample: \((h^1, x^1), (h^2, x^2), \ldots, (h^N, x^N)\)
Policy Gradient

\[
\bar{R}_\theta = \sum_h P(h) \sum_x R(h, x) P_\theta(x|h) \approx \frac{1}{N} \sum_{i=1}^{N} R(h^i, x^i)
\]

\[
\nabla R_\theta = \sum_h P(h) \sum_x R(h, x) \nabla P_\theta(x|h) \approx \frac{1}{N} \sum_{i=1}^{N} R(h^i, x^i) \nabla \log P_\theta(x|h)
\]

\[
= \sum_h P(h) \sum_x R(h, x) P_\theta(x|h) \frac{\nabla P_\theta(x|h)}{P_\theta(x|h)}
\]

\[
= \sum_h P(h) \sum_x R(h, x) P_\theta(x|h) \nabla \log P_\theta(x|h)
\]

\[
= E_{h \sim P(h), x \sim P_\theta(x|h)} [R(h, x) \nabla \log P_\theta(x|h)]
\]

\[
\frac{d \log(f(x))}{dx} = \frac{1}{f(x)} \frac{df(x)}{dx}
\]
Policy Gradient

• Gradient Ascent

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{i=1}^{N} R(h^i, x^i) \nabla \log P_{\theta}(x^i|h^i)$$

$R(h^i, x^i)$ is positive

$\Rightarrow$ After updating $\theta$, $P_{\theta}(x^i|h^i)$ will increase

$R(h^i, x^i)$ is negative

$\Rightarrow$ After updating $\theta$, $P_{\theta}(x^i|h^i)$ will decrease
## Implementation

<table>
<thead>
<tr>
<th></th>
<th>Maximum Likelihood</th>
<th>Reinforcement Learning</th>
</tr>
</thead>
</table>
| Objective Function   | \[
\frac{1}{N} \sum_{i=1}^{N} \log P_\theta(\hat{x}^i|h^i)
\] | \[
\frac{1}{N} \sum_{i=1}^{N} R(h^i,x^i)\log P_\theta(x^i|h^i)
\] |
| Gradient             | \[
\frac{1}{N} \sum_{i=1}^{N} \nabla \log P_\theta(\hat{x}^i|h^i)
\] | \[
\frac{1}{N} \sum_{i=1}^{N} R(h^i,x^i)\nabla \log P_\theta(x^i|h^i)
\] |
| Training Data        | \{(h^1,\hat{x}^1), ..., (h^N,\hat{x}^N)\} | \{(h^1,x^1), ..., (h^N,x^N)\} |
|                      | \[
R(h^i,\hat{x}^i) = 1
\] | Sampling as training data weighted by \(R(h^i,x^i)\) |
Implementation

\[ \theta^t \]

\[
\begin{align*}
(h^1, x^1) & \quad R(h^1, x^1) \\
(h^2, x^2) & \quad R(h^2, x^2) \\
\vdots & \quad \vdots \\
(h^N, x^N) & \quad R(h^N, x^N)
\end{align*}
\]

\[ \theta^0 \] can be well pre-trained from \( \{(h^1, \hat{x}^1), \ldots, (h^N, \hat{x}^N)\} \)

New Objective:

\[
\theta^{t+1} \leftarrow \theta^t + \eta \nabla R_{\theta^t}
\]

\[
\frac{1}{N} \sum_{i=1}^{N} R(h^i, x^i) \log P_{\theta}(x^i|h^i)
\]

\[
\frac{1}{N} \sum_{i=1}^{N} R(h^i, x^i) \nabla \log P_{\theta^t}(x^i|h^i)
\]
Add a Baseline

If $R(h^i, x^i)$ is always positive

$$\frac{1}{N} \sum_{i=1}^{N} R(h^i, x^i) \log \nabla P_{\theta}(x^i | h^i)$$

**Ideal case** $P_{\theta}(x|h)$

- $(h, x^1)$
- $(h, x^2)$
- $(h, x^3)$

**Due to Sampling**

- Not sampled

- $(h, x^1)$
- $(h, x^2)$
- $(h, x^3)$

Because it is probability ...
Add a Baseline

If $R(h^i, x^i)$ is always positive

$$\frac{1}{N} \sum_{i=1}^{N} R(h^i, x^i) \log \nabla P_\theta(x^i|h^i)$$

There are several ways to obtain the baseline $b$. 

There are several ways to obtain the baseline $b$. 

Not sampled

Add baseline
Alpha GO style training!

• Let two agents talk to each other

How old are you?  
See you.  
See you.  
See you.  

How old are you?  
I am 16.  
I though you were 12.  
What make you think so?

Using a pre-defined evaluation function to compute $R(h,x)$
Example Reward

• The final reward $R(h, x)$ is the weighted sum of three terms $r_1(h, x)$, $r_2(h, x)$ and $r_3(h, x)$

$$R(h, x) = \lambda_1 r_1(h, x) + \lambda_2 r_2(h, x) + \lambda_3 r_3(h, x)$$
Example Results

<table>
<thead>
<tr>
<th>Baseline mutual information model (Li et al. 2015)</th>
<th>Proposed reinforcement learning model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reinforcement learning?

Start with observation $s_1$

Observation $s_2$

Observation $s_3$

Action $a_1$: “right”

Obtain reward $r_1 = 0$

Action $a_2$: “fire”

Obtain reward $r_2 = 5$

(kill an alien)
Reinforcement learning?

The action we take influence the observation in the next step.

Action taken

Actions set

observation


reward: R(“BAA”, reference)
Reinforcement learning?

• One can use any advanced RL techniques here.
• For example, actor-critic
SeqGAN


Basic Idea – Sentence Generation

- Code $z$ sampled from prior distribution
- Sampling from RNN at each time step also provides randomness
- $\text{Original GAN}$
Algorithm – Sentence Generation

- Initialize generator Gen and discriminator Dis
- In each iteration:
  - Sample real sentences $x$ from database
  - Generate sentences $\tilde{x}$ by Gen
  - Update Dis to increase $Dis(x)$ and decrease $Dis(\tilde{x})$

- Update Gen such that

```
Generator --> Discriminator --> scalar
```

\uparrow
Basic Idea – Chat-bot

Input sentence/history h → Chatbot → response sentence x

Input sentence/history h → Discriminator → Real or fake

Response sentence x → Discriminator → Real or fake

Conditional GAN

Algorithm – Chat-bot

- Initialize generator Gen and discriminator Dis
- In each iteration:
  - Sample real history $h$ and sentence $x$ from database
  - Sample real history $h'$ from database, and generate sentences $\tilde{x}$ by $\text{Gen}(h')$
  - Update Dis to increase $Dis(h, x)$ and decrease $Dis(h', \tilde{x})$
- Update Gen such that...
Can we do backpropagation?
Tuning generator will not change the output.
Alternative: improved WGAN
Reinforcement Learning

- Consider the output of discriminator as reward
  - Update generator to increase discriminator = to get maximum reward

\[
\nabla \tilde{R}_\theta \approx \frac{1}{N} \sum_{i=1}^{N} \left( D(h^i, x^i) - b \right) \nabla \log P_\theta (x^i | h^i)
\]

- Different from typical RL
  - The discriminator would update
Reward for Every Generation Step

\[ \nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{i=1}^{N} (D(h^i, x^i) - b) \nabla \log P_\theta(x^i|h^i) \]

\( h^i = \text{"What is your name?"} \quad D(h^i, x^i) - b \text{ is negative} \)
\( x^i = \text{"I don't know"} \quad \text{Update } \theta \text{ to decrease } \log P_\theta(x^i|h^i) \)

\[ \log P_\theta(x^i|h^i) = \log P(x_1^i|h^i) + \log P(x_2^i|h^i, x_1^i) + \log P(x_3^i|h^i, x_{1:2}^i) \]

\[ P(\text{"I"}|h^i) \]

\( h^i = \text{"What is your name?"} \quad D(h^i, x^i) - b \text{ is positive} \)
\( x^i = \text{"I am John"} \quad \text{Update } \theta \text{ to increase } \log P_\theta(x^i|h^i) \)

\[ \log P_\theta(x^i|h^i) = \log P(x_1^i|h^i) + \log P(x_2^i|h^i, x_1^i) + \log P(x_3^i|h^i, x_{1:2}^i) \]

\[ P(\text{"I"}|h^i) \]
Reward for Every Generation Step

\[ h^i = \text{"What is your name?"} \quad x^i = \text{"I don't know"} \]

\[ \log p_{\theta}(x^i | h^i) = \log p(x_1^i | h^i) + \log p(x_2^i | h^i, x_1^i) + \log p(x_3^i | h^i, x_1^i, x_2^i) \]

\[ p(\text{"I"} | h^i) \quad p(\text{"don't"} | h^i, \text{"I"}) \quad p(\text{"know"} | h^i, \text{"I don't"}) \]

\[ \nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} (Q(h^i, x_{1:t}^i) - b) \nabla \log p_{\theta}(x_t^i | h^i, x_{1:t-1}^i) \]

Method 1. Monte Carlo (MC) Search

Method 2. Discriminator For Partially Decoded Sequences
Monte Carlo Search

• How to estimate $Q(h^i, x^i_{1:t})$?

\[
Q("What is your name?", "I")
\]

\[
\begin{align*}
  h^i & \quad x^i_1 \\
  x^A & = \text{I am John} \quad D(h^i, x^A) = 1.0 \\
  x^B & = \text{I am happy} \quad D(h^i, x^B) = 0.1 \\
  x^C & = \text{I don’t know} \quad D(h^i, x^C) = 0.1 \\
  x^D & = \text{I am superman} \quad D(h^i, x^D) = 0.8 \\
\end{align*}
\]

A roll-out generator for sampling is needed

\[
Q(h^i, "I") = 0.5
\]
Rewarding Partially Decoded Sequences

- Training a discriminator that is able to assign rewards to both fully and partially decoded sequences
  - Break generated sequences into partial sequences

\[
Q(h, x_{1:t})
\]

- \(h=\text{“What is your name?”}, x=\text{“I am john”}\)
- \(h=\text{“What is your name?”}, x=\text{“I am”}\)
- \(h=\text{“What is your name?”}, x=\text{“I”}\)
- \(h=\text{“What is your name?”}, x=\text{“I don’t know”}\)
- \(h=\text{“What is your name?”}, x=\text{“I don’t”}\)
- \(h=\text{“What is your name?”}, x=\text{“I”}\)
Teacher Forcing

- The training of generative model is unstable
  - This reward is used to promote or discourage the generator’s own generated sequences.
  - Usually it knows that the generated results are bad, but does not know what results are good.

- Teacher Forcing

  Training Data for SeqGAN: \[ \{(h^1, x^1), \ldots, (h^N, x^N)\} \]
  Obtained by sampling weighted by \( D(h^i, x^i) \)

  Adding more Data: \[ \{(h^1, \hat{x}^1), \ldots, (h^N, \hat{x}^N)\} \]
  Real data
  Consider \( D(h^i, \hat{x}^i) = 1 \)
Experiments in paper

• Sentence generation: Synthetic data
  • Given an LSTM
  • Using the LSTM to generate a lot of sequences as “real data”
  • Generator learns from the “real data” by different approaches
  • Generator generates some sequences
  • Using the LSTM to compute the negative log likelihood (NLL) of the sequences
    • Smaller is better
Experiments in paper
- Synthetic data

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Random</th>
<th>MLE</th>
<th>SS</th>
<th>PG-BLEU</th>
<th>SeqGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLL</td>
<td>10.310</td>
<td>9.038</td>
<td>8.985</td>
<td>8.946</td>
<td><strong>8.736</strong></td>
</tr>
<tr>
<td>p-value</td>
<td>$&lt; 10^{-6}$</td>
<td>$&lt; 10^{-6}$</td>
<td>$&lt; 10^{-6}$</td>
<td>$&lt; 10^{-6}$</td>
<td>$&lt; 10^{-6}$</td>
</tr>
</tbody>
</table>

**Learning curve**

- SeqGAN
- MLE
- Schedule Sampling
- PG-BLEU

NLL by oracle vs. Epochs
(a) $g$-steps=100, $d$-steps=1, $k=10$

(b) $g$-steps=30, $d$-steps=1, $k=30$

(c) $g$-steps=1, $d$-steps=1, $k=10$

(d) $g$-steps=1, $d$-steps=5, $k=3$
Experiments in paper
- Real data

Table 2: Chinese poem generation performance comparison.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Human score</th>
<th>p-value</th>
<th>BLEU-2</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE</td>
<td>0.4165</td>
<td></td>
<td>0.6670</td>
<td>&lt; $10^{-6}$</td>
</tr>
<tr>
<td>SeqGAN</td>
<td>0.5356</td>
<td>0.0034</td>
<td>0.7389</td>
<td></td>
</tr>
<tr>
<td>Real data</td>
<td>0.6011</td>
<td></td>
<td>0.746</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Obama political speech generation performance.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>BLEU-3</th>
<th>p-value</th>
<th>BLEU-4</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE</td>
<td>0.519</td>
<td>&lt; $10^{-6}$</td>
<td>0.416</td>
<td>0.00014</td>
</tr>
<tr>
<td>SeqGAN</td>
<td>0.556</td>
<td></td>
<td>0.427</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Music generation performance comparison.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>BLEU-4</th>
<th>p-value</th>
<th>MSE</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE</td>
<td>0.9210</td>
<td>&lt; $10^{-6}$</td>
<td>22.38</td>
<td>0.00034</td>
</tr>
<tr>
<td>SeqGAN</td>
<td>0.9406</td>
<td></td>
<td>20.62</td>
<td></td>
</tr>
</tbody>
</table>
## Results - Chat-bot

<table>
<thead>
<tr>
<th>Setting</th>
<th>adver-win</th>
<th>adver-lose</th>
</tr>
</thead>
<tbody>
<tr>
<td>single-turn</td>
<td>0.62</td>
<td>0.18</td>
</tr>
<tr>
<td>multi-turn</td>
<td>0.72</td>
<td>0.10</td>
</tr>
</tbody>
</table>

### Examples:

<table>
<thead>
<tr>
<th>Input</th>
<th>Vanilla-MLE</th>
<th>Vanilla-Sample</th>
<th>REINFORCE</th>
<th>REGS Monte Carlo</th>
</tr>
</thead>
<tbody>
<tr>
<td>tell me ... how long have you had this falling sickness?</td>
<td>i 'm not a doctor</td>
<td>well everything you did was totally untrue</td>
<td>i don ’t know how long it ’s been</td>
<td>A few months, I guess</td>
</tr>
<tr>
<td>so i had the doctors test sammy ’s response to conditioning.</td>
<td>sammy wrote the test sammy wrote the test</td>
<td>objects pick objects up objects objects objects objects objects objects</td>
<td>i ’m not sure that ’s relevant</td>
<td>so he took the pills</td>
</tr>
<tr>
<td>what are you going to do?</td>
<td>i ’m going to the movies</td>
<td>get him outta here first !</td>
<td>i ’m going to get you</td>
<td>i ’m going to get you a new car</td>
</tr>
<tr>
<td>they fear your power your intellect.</td>
<td>you ’re the only one who knows what ’s going on</td>
<td>when they are conquered and you surrender they will control all of us</td>
<td>i ’m afraid i ’m not ready yet</td>
<td>i ’m not afraid of your power</td>
</tr>
</tbody>
</table>
To Learn More ...
Algorithm – MaliGAN

Maximum-likelihood Augmented Discrete GAN

• Initialize generator Gen and discriminator Dis

• In each iteration:
  • Sample real sentences $x$ from database
  • Generate sentences $\tilde{x}$ by Gen
  • Update Dis to maximize
    \[
    \sum_x \log D(x) + \sum_{\tilde{x}} \log(1 - D(\tilde{x}))
    \]

  • Update Gen by gradient
  \[
  \frac{1}{N} \sum_{i=1}^{N} \left( \frac{r_D(x^i)}{\sum_{i=1}^{N} r_D(x^i)} - b \right) \nabla \log P_\theta(x^i) \\
  \]

\[
D(h^i, x^i) = \frac{D(x^i)}{1 - D(x^i)}
\]
To learn more ......

• Professor forcing
  • Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, Yoshua Bengio, “Professor Forcing: A New Algorithm for Training Recurrent Networks”, NIPS, 2016

• Handling discrete output by methods other than policy gradient
  • MaliGAN, Boundary-seeking GAN