Sequence Generation
Hung-yi Lee
李宏毅
Outline

• RNN with Gated Mechanism
• Sequence Generation
• Conditional Sequence Generation
• Tips for Generation
RNN with Gated Mechanism
Recurrent Neural Network

- Given function $f$: $h', y = f(h, x)$
  
  No matter how long the input/output sequence is, we only need one function $f$.
Deep RNN

\[ h', y = f_1(h, x) \quad b', c = f_2(b, y) \]
Bidirectional RNN

\[ h', a = f_1(h, x) \quad b', c = f_2(b, x) \]

\[ y = f_3(a, c) \]
Naïve RNN

• Given function $f$: $h', y = f(h, x)$

\[
\begin{align*}
    h' &= \sigma(W^h h + W^i x) \\
    y &= \sigma(W^o h')
\end{align*}
\]
LSTM

- \( c^t \) changes slowly
- \( c^t \) is \( c^{t-1} \) added by something
- \( h^t \) changes faster
- \( h^t \) and \( h^{t-1} \) can be very different
\[ y^t \]

\[ c^{t-1} \rightarrow LSTM \rightarrow c^t \]

\[ h^{t-1} \rightarrow LSTM \rightarrow h^t \]

\[ x^t \]

\[ z = \tanh(W) \]

\[ z^i = \sigma(W) \]

\[ z^f = \sigma(W) \]

\[ z^o = \sigma(W) \]

Input gate

Forget gate

Output gate
\[ z = \tanh( \theta ) \]

"peephole"

obtained by the same way
\[ c^t = z^f \odot c^{t-1} + z^i \odot z \]
\[ h^t = z^o \odot \text{tanh}(c^t) \]
\[ y^t = \sigma(W'h^t) \]
GRU

\[ h^t = z \bigodot h^{t-1} + (1 - z) \bigodot h' \]
LSTM: A Search Space Odyssey

1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)

Standard LSTM works well
Simply LSTM: coupling input and forget gate, removing peephole
Forget gate is critical for performance
Output gate activation function is critical
Sequence Generation
Generation

• Sentences are composed of characters/words
• Generating a character/word at each time by RNN

```
x: [0 1 0 0 0 0 ...... 0]
y: [0 0 0 0 0 0.7 0.3 ...... 0]
```

![Diagram of RNN model]

- Distribution over the token
  (sampling from the distribution to generate a token)
- The token generated at the last time step.
  (represented by 1-of-N encoding)
Generation

• Sentences are composed of characters/words
• Generating a character/word at each time by RNN

\[
y^1: P(w | <BOS>) \\
y^2: P(w | <BOS>, 床) \\
y^3: P(w | <BOS>, 床, 前)
\]

Until <EOS> is generated
Generation

Training

Training data: 春眠不覺曉
Generation

- Images are composed of pixels
- Generating a pixel at each time by RNN

Consider as a sentence
blue red yellow gray ......

Train a RNN based on the “sentences”
Generation - PixelRNN

- Images are composed of pixels

3 x 3 images
Conditional Sequence Generation
Conditional Generation

• We don’t want to simply generate some random sentences.
• Generate sentences based on conditions:

**Caption Generation**

Given condition: "A young girl is dancing."

**Chat-bot**

Given condition: "Hello"

"Hello. Nice to see you."
Conditional Generation

- Represent the input condition as a vector, and consider the vector as the input of RNN generator

Image Caption Generation

A vector

A

woman

. (period)

Input image

CNN

<BOS>
Conditional Generation

- Represent the input condition as a vector, and consider the vector as the input of RNN generator.
- E.g. Machine translation / Chat-bot

Information of the whole sentences

Encoder 💻Jointly train 💻Decoder

**Sequence-to-sequence learning**
Conditional Generation

M: Hello
U: Hi
M: Hi

Need to consider longer context during chatting

https://www.youtube.com/watch?v=e2MpOmyQJw4

M: Hello
U: Hi

Dynamic Conditional Generation

Encoder

Decoder
Machine Translation

- Attention-based model

\[ \alpha = h^T W z \]

What is \textit{match}?

Jointly learned with other part of the network

\[ \alpha \]

\[ h \]

\[ z \]

Design by yourself

- Cosine similarity of \( z \) and \( h \)
- Small NN whose input is \( z \) and \( h \), output a scalar
Machine Translation

- Attention-based model

\[ c^0 = \sum \hat{\alpha}_0^i h^i \]
\[ = 0.5h^1 + 0.5h^2 \]
Machine Translation

- Attention-based model
Machine Translation

• Attention-based model

\[ c^1 = \sum \hat{\alpha}_i^1 h^i \]
\[ = 0.5h^3 + 0.5h^4 \]
Machine Translation

- Attention-based model

The same process repeat until generating <EOS>
Speech Recognition


<table>
<thead>
<tr>
<th>Model</th>
<th>Clean WER</th>
<th>Noisy WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLDNN-HMM [22]</td>
<td>8.0</td>
<td>8.9</td>
</tr>
<tr>
<td>LAS</td>
<td>14.1</td>
<td>16.5</td>
</tr>
<tr>
<td>LAS + LM Rescoring</td>
<td>10.3</td>
<td>12.0</td>
</tr>
</tbody>
</table>
Image Caption Generation

- A vector for each region

CNN

$z^0 \xrightarrow{\text{match}} 0.7$
Image Caption Generation

A vector for each region

CNN

worded sum

$z^0 \rightarrow z^1$

Word 1

$0.7 \rightarrow 0.1 \rightarrow 0.1 \rightarrow 0.0 \rightarrow 0.0$
Image Caption Generation

A vector for each region

CNN

Word 1 ➔ Word 2

Weighted sum

\[ z^0 \rightarrow z^1 \rightarrow z^2 \]
Image Caption Generation

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Image Caption Generation

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.

A person is standing on a beach with a surfboard.

A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

Ref: A man and a woman ride a motorcycle
A man and a woman are talking on the road

Ref: A woman is frying food
Someone is frying a fish in a pot

Tips for Generation
Attention

Bad Attention: each input component has approximately the same attention weight

\[ w_1 \quad w_2 \ (\text{woman}) \quad w_3 \quad w_4 \ (\text{woman}) \quad \ldots \quad \text{no cooking} \]

Good Attention: each input component has approximately the same attention weight

E.g. Regularization term: \[ \sum_i \left( \tau - \sum_t \alpha_t^i \right)^2 \]

For each component \quad Over the generation
Mismatch between Train and Test

- **Training**

\[ C = \sum_t C_t \]

Minimizing cross-entropy of each component

: condition

Reference:
Mismatch between Train and Test

• **Generation**

We do not know the reference

Testing: The inputs are the outputs of the last time step.

Training: The inputs are reference.

*Exposure Bias*
One step wrong

May be totally wrong

Never explore .......

一步错，步步错
When we try to decrease the loss for both steps 1 and 2 ....

Training is matched to testing.

In practice, it is hard to train in this way.
Scheduled Sampling
Beam Search

The green path has higher score.
Not possible to check all the paths
Beam Search

Keep several best path at each step
Beam size = 2
Beam Search

The size of beam is 3 in this example.

https://github.com/tensorflow/tensorflow/issues/654#issuecomment-169009989
Better Idea?

I am ...... ✓
You are ...... ✓

I are ...... ×
You am ...... ×
Object level v.s. Component level

- Minimizing the error defined on component level is not equivalent to improving the generated objects.

Ref: The dog is running fast

\[ C = \sum_t C_t \]

Cross-entropy of each step

Optimize object-level criterion instead of component-level cross-entropy.

Object-level criterion: \( R(y, \hat{y}) \)

\( y \): generated utterance, \( \hat{y} \): ground truth
Reinforcement learning?

Start with observation $s_1$

Observe $s_2$

Observe $s_3$

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.

reward: $R(\text{“BAA”}, \text{reference})$