Recurrent Neural Networks II

CS 4391 Introduction Computer Vision
Professor Yu Xiang
The University of Texas at Dallas

Some slides of this lecture are courtesy Stanford CS231n
Recurrent Neural Networks

Output label

\[
\begin{align*}
Y_{t-1} & \quad \uparrow \\
Y_t & \quad \uparrow \\
Y_{t+1} & \quad \uparrow
\end{align*}
\]

Input

\[
\begin{align*}
X_{t-1} & \quad \uparrow \\
X_t & \quad \uparrow \\
X_{t+1} & \quad \uparrow
\end{align*}
\]

Internal state (memory)
Vanilla RNN

\[ h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t) \]

\[ = \tanh \left( (W_{hh} \quad W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \]

\[ = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \]

\[ y_t = W_{hy} h_t \]
Long Short Term Memory (LSTM)

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

LSTM

Input gate
forget gate
output gate
Sigmoid

Cell
Hidden state

\[ c_t = f \odot c_{t-1} + i \odot g \]
\[ h_t = o \odot \tanh(c_t) \]

Store Cell and hidden states
**Long Short Term Memory (LSTM)**

- **$g$: Gate gate, how much to write to cell**
- **$i$: Input gate, whether to write to cell**
- **$f$: Forget gate, whether to erase cell**
- **$o$: Output gate, how much to reveal cell**

$$
\begin{align*}
\begin{pmatrix}
  i \\
  f \\
  o \\
  g
\end{pmatrix} &= \begin{pmatrix}
  \sigma \\
  \sigma \\
  \sigma \\
  \tanh
\end{pmatrix} W \begin{pmatrix}
  h_{t-1} \\
  x_t
\end{pmatrix} \\
ct &= f \odot ct_{-1} + i \odot g \\
h_t &= o \odot \tanh(ct)
\end{align*}
$$
Long Short Term Memory (LSTM)

- Make the RNN easier to preserve information over many steps
  - E.g., $f = 1$ and $i = 0$
  - This is difficult for vanilla RNN

- LSTM does not guarantee that there is no vanishing or exploding gradient

- It provides an easier way to learn long-distance dependencies
Gated Recurrent Unit (GRU)

\[ z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \]
\[ r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \]
\[ \hat{h}_t = \phi_h(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h) \]
\[ h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \]

- \( x_t \): input vector
- \( h_t \): output vector
- \( \hat{h}_t \): candidate activation vector
- \( z_t \): update gate vector
- \( r_t \): reset gate vector
- \( W, U \) and \( b \): parameter matrices and vector

https://en.wikipedia.org/wiki/Gated_recurrent_unit
GRUs vs. LSTMs

• Both have a forget gate

• GRU has fewer parameters, no output gate

• GRUs have similar performance compared to LSTMs, have shown better performance on certain datasets
Recurrent Neural Networks

- Many to many:
  - E.g., action recognition on video frames

- One to many:
  - E.g., image captioning, image -> sequences of words

- Many to one:
  - E.g., action prediction, sequences of frames -> action class

- Many to many:
  - E.g., Video Captioning
    Sequence of video frames -> caption
Recurrent Units on CNN Features

Convolution + ReLU
Max Pooling
Concatenation
Deconvolution
Addition
Recurrent Units

RGB Image
Time t
Depth Image

RGB Image
Time t+1
Depth Image

Labels
data association

DA-RNN. Xiang & Fox, RSS’17

4/10/2024
Yu Xiang
Machine Translation

- Translate a phrase from one language to another
  - E.g., English phrase to French phrase

Google Translation

UT Dallas is a rising public research university in the heart of DFW.

UT Dallas est une université de recherche publique en plein essor au cœur de DFW.

13 words 15 words
Machine Translation

- **Input**
  \[ x = (x_1, x_2, \ldots, x_T) \]

- **Output**
  \[ y = (y_1, y_2, \ldots, y_{T'}) \]  \( T \neq T' \)

Not one to one mapping

RNN

\( y_{t-1} \)
\( y_t \)
\( y_{t+1} \)

\( x_{t-1} \)
\( x_t \)
\( x_{t+1} \)
RNN Encoder-Decoder

\[ x_1 \rightarrow h_1 \rightarrow h_2 \rightarrow \cdots \rightarrow h_T \rightarrow c \]

\[ h_t = f(h_{t-1}, x_t) \]
\[ c = h_T \]

\[ y_{T'} \leftarrow \cdots \leftarrow y_2 \leftarrow y_1 \leftarrow y_0 \quad [\text{START}] \]

\[ s_t = f(s_{t-1}, y_{t-1}, c) \]
\[ y_t = g(s_t, y_{t-1}, c) \]

RNN Encoder-Decoder

- Encoder: \( h_t = f(h_{t-1}, x_t) \) \( c = h_T \)
- Decoder: \( s_t = f(s_{t-1}, y_{t-1}, c) \) \( y_t = g(s_t, y_{t-1}, c) \)

- Pros
  - Can deal with different input size and output size

- Cons
  - The fixed length embedding cannot handle long sentence well (long-distance dependencies)
Limitations of RNNs

• The sequential computation of hidden states precludes parallelization within training examples

• Cannot handle long sequences well
  • Truncated back-propagation due to memory limits

  • Difficult to capture dependencies in long distances
Summary

• RNNs can be used for sequential data to capture dependencies in time

• LSTMs and GRUs are better than vanilla RNNs

• It is difficult to capture long-term dependencies in RNNs

• Use transformers (in future lectures)
Further Reading

• Stanford CS231n, lecture 10, Recurrent Neural Networks
  http://cs231n.stanford.edu/

• Long Short Term Memory
  https://www.researchgate.net/publication/13853244_Long_Short-term_Memory