Recurrent Neural Networks

CS598PS - Machine Learning for Signal Processing
Nov. 6, 2015
Outline

- Why do we need recurrency in the network?
- A plain Recurrent Neural Network (RNN)
- Long Short-Term Memory (LSTM)
- Applications of RNNs
In the ordinary neural networks, input samples (and outputs) are assumed to be independent from each other.
- The network forgets what happened before.

What if the data samples are sequential?
- And the sequence is meaningful for the job?

We’ve already seen some similar stories before..
- GMM VS HMM
Is Concatenation a Good Idea?

- A network can take a series of input samples, rather than one by one

Is this good enough?
- What happens when we have to concatenate too many samples (long term dependence)?
- What if the length of input sequences varies, e.g. in translation?
- Need to know the maximum length of the dependency
Introducing Recurrency

- Recurrent Neural Networks (RNN)

RNN shares weights across time
- Some of them are in-between hidden units

\[
\sigma(Wx_t)
\]

\[
A_{z_t} = (Wx_t + Uz_{t-1})
\]

\[
z_t = \sigma(A_{z_t})
\]

\[
A_{o_t} = Vz_t
\]

\[
o_t = \sigma(A_{o_t})
\]
What’s Going on in an RNN?

- The function we want to learn: \( y_t = x_{t-1} \cdot x_t \)

\[
\begin{align*}
Y_t & : +1 \quad -1 \quad -1 \quad +1 \quad -1 \quad +1 \\
O_t & : +0.97 \quad -0.95 \quad -0.96 \quad +0.97 \quad -0.95 \quad +0.90 \\
\hat{z}_{1,t} & : +0.12 \quad -0.94 \quad +0.97 \quad +0.24 \quad -0.94 \quad +0.98 \\
\hat{z}_{2,t} & : +0.83 \\
x_t & : +1 \quad +1 \quad -1 \quad +1 \quad +1 \quad -1 \quad -1 \\

W = \begin{bmatrix} -2.21 \\ -2.29 \end{bmatrix}, \quad U = \begin{bmatrix} 0.53 & 1.97 \\ 0.06 & 1.61 \end{bmatrix}, \quad b_x = \begin{bmatrix} 2.32 \\ -0.83 \end{bmatrix}, \quad V = \begin{bmatrix} -3.42 \\ 3.28 \end{bmatrix}, \quad b_y = 2.16
\end{align*}
\]
Back Propagation Through Time (BPTT)

\[
A_{z_t} = (Wx_t + Uz_{t-1})
\]
\[
z_t = \sigma(A_{z_t})
\]
\[
A_{o_t} = Vz_t
\]
\[
o_t = \sigma(A_{o_t})
\]

\[
\frac{\partial E}{\partial A_{o_t}} = \frac{\partial E}{\partial o_t} \frac{\partial o_t}{\partial A_{o_t}} = d_{o_t} = (o_t - y_t) \odot \sigma'(A_{o_t})
\]

\[
\frac{\partial E}{\partial V} = \frac{\partial E}{\partial o_t} \frac{\partial o_t}{\partial A_{o_t}} \frac{\partial A_{o_t}}{\partial V} = \nabla V = d_{o_t} z_t^T
\]

\[
\frac{\partial E}{\partial A_{z_t}} = \frac{\partial E}{\partial o_t} \frac{\partial o_t}{\partial A_{o_t}} \frac{\partial A_{o_t}}{\partial z_t} \frac{\partial z_t}{\partial A_{z_t}} = d_{z_t} = \sigma'(A_{z_t}) \odot (V^T d_{o_t})
\]

\[
\frac{\partial E}{\partial A_{z_{t-1}}} = \frac{\partial E}{\partial o_t} \frac{\partial o_t}{\partial A_{o_{t-1}}} \frac{\partial A_{o_{t-1}}}{\partial z_{t-1}} \frac{\partial z_{t-1}}{\partial A_{z_{t-1}}} \frac{\partial A_{z_{t-1}}}{\partial A_{z_t}} = d_{z_{t-1}} = \sigma'(A_{z_{t-1}}) \odot U^T d_{z_t}
\]
Back Propagation Through Time (BPTT)

\[
\begin{align*}
\frac{\partial \mathcal{E}}{\partial A_{z_1}} &= d_{z_1} = \sigma'(A_{z_1}) \odot \left( V^T d_{o_1} + U^T d_{z_2} \right) \\
&= \sigma'(A_{z_1}) \odot \left( V^T d_{o_1} + U^T \left( \sigma'(A_{z_2}) \odot \left( V^T d_{o_2} + U^T d_{z_3} \right) \right) \right) \\
&= \sigma'(A_{z_1}) \odot \left( V^T d_{o_1} + U^T \left( \sigma'(A_{z_2}) \odot \left( V^T d_{o_2} + U^T \left( \sigma'(A_{z_3}) \odot \left( V^T d_{o_3} + U^T d_{z_4} \right) \right) \right) \right) \right) \\
&= \cdots
\end{align*}
\]

- We keep multiplying some small numbers to the error backpropagated!
Vanishing Gradients During BPTT

- **An example:** \( y_t = x_{t-2} \)
  - For simplicity, we fix all active weights to be 1
  - \( tanh \) activation

\[
U = \begin{bmatrix}
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\quad U = \begin{bmatrix}
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\quad V = \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix}
\]

\[
W = \begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix}
\quad W = \begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix}
\quad W = \begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix}
\]

\[
y_t = x_{t-2} + 1 - 0.76 + 0.64 + 0.76 - 0.64 + 0.56
\]
Vanishing Gradients During BPTT

- An example: \( y_t = x_{t-2} \)

* When the learning rate is 0.1
Vanishing Gradients During BPTT

- Let’s improve the ordinary RNN
  - The flow of the error backpropagation is blocked with a nonlinearity

\[ h_{t-1} \xrightarrow{\text{tanh}} h_t \xrightarrow{\text{tanh}} h_{t+1} \]
Long Short-Term Memory (LSTM)

For the LSTM, the state at time \( t \) is given by:

\[
\begin{align*}
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
\tilde{C}_t &= \text{tanh}(W_c x_t + U_c h_{t-1}) \\
i_t &= \sigma(W_i x_t + U_i h_{t-1}) \\
f_t &= \sigma(W_f x_t + U_f h_{t-1}) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1}) \\
h_t &= o_t \odot \text{tanh}(\tilde{C}_t)
\end{align*}
\]

* Figure inspired by http://colah.github.io/posts/2015-08-Understanding-LSTMs/*
There’s a thing called ‘Theano’, which does this for you
Another Delay Function with LSTM

- The function to learn: \( y_t = x_{t-10} \)
- 10 hidden units
Network Topologies Using LSTM

- **One-to-one**
  - Input: fixed-size
  - Output: fixed-size
  - e.g. image classification

- **One-to-many**
  - Input: fixed-size
  - Output: sequence
  - e.g. image captioning

- **Many-to-one**
  - Input: sequence
  - Output: fixed-size
  - e.g. sentiment analysis, speech recognition

- **Many-to-many**
  - Input: synced sequence
  - Output: synced sequence
  - e.g. video frame classification, phoneme classification, source separation

- **Many-to-many**
  - Input: sequence
  - Output: sequence
  - e.g. translation
Language Modeling

- Train an LSTM network on Shakespeare (4.4MB)
  - 3 layers X 512 units
  - Sample a character from the softmax output (probabilities)

---

PANDARUS:
Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCEN'TIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
Classification - GoT

Sequences for training

Sequences for testing

Tyrion is Lannister likes cats
Cersei is Lannister likes cats
Jamie is Lannister likes cats
Robb is Stark likes dogs
Arya is Stark likes dogs
Bran is Stark likes dogs
Tywin is Lannister
Sansa is Stark likes dogs

Slide 17
Classification - GoT

- The network doesn’t make a decision until it observes the keywords
Classification – GoT with Marital Status

Tyrion is Lannister
Tyrion married nobody
Tyrion likes cats

Tyrion is Lannister
Tyrion married Stark
Tyrion likes dogs

Cersei is Lannister
Cersei married nobody
Cersei likes cats

Cersei is Lannister
Cersei married Lannister
Cersei likes cats

Talisa is Maegyr
Talisa married nobody
Talisa likes cats

Talisa is Maegyr
Talisa married Stark
Talisa likes dogs

Robb is Stark
Robb married nobody
Robb likes dogs

Robb is Stark
Robb married Maegyr
Robb likes dogs
The network waits more to see who married who

Unless he or she is Stark
Classification – Two Words

- Trained on four utterances per class
  - 3 hidden units, MFCC features, outputs at all frames (pooling)

- LSTM is confused in the first place, but immediately decides the class with the first syllable
Classification – Let’s Fool LSTM

- “One” VS “One Two”
  - 4 hidden units, MFCC features, outputs at all frames (pooling)
  - LSTM first thinks it’s “one”, but flips the decision if “two” follows
A Simple, But Tricky Separation

\[
\begin{align*}
Y^{tr} & \quad \text{Train} \\
& \quad \text{LSTM} \\
\end{align*}
\]

\[
\begin{align*}
X^{tr} & \quad \text{LSTM} \\
S_1^{tr} & \quad + \quad S_2^{tr} \\
\end{align*}
\]

\[
\begin{align*}
O^{tst} & \quad \text{Test} \\
& \quad \text{LSTM} \\
\end{align*}
\]

\[
\begin{align*}
X^{tst} & \quad \text{Test} \\
S_1^{tst} & \quad \text{G.T.} \\
S_2^{tst} & \\
\end{align*}
\]
Gated Recurrent Units

- A much simpler, yet powerful gating mechanism
**Bidirectional RNN (or LSTM)**

- Addresses bidirectional order in the sequence
Recap.

- Recurrent neural networks efficiently learns temporal structure
- Gating techniques in LSTM can prevent gradient vanishing problem
- Many interesting applications
What’s next

- We’ll explore some more application areas
  - And some of them will be covered by guest lecturers

- Next Wednesday,
  - Neuro/bio signals
Reading materials

- **Some nice web materials**
  - [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
  - [http://karpathy.github.io/2015/05/21/rnn-effectiveness/](http://karpathy.github.io/2015/05/21/rnn-effectiveness/)
  - [http://www.deeplearning.net/tutorial/lstm.html](http://www.deeplearning.net/tutorial/lstm.html)

- **Sequence-to-sequence learning**

- **GRU**

- **BLSTM**