Recurrent Neural Networks (RNN) and Long-Short-Term-Memory (LSTM)

Yuan YAO
HKUST
Summary

- We have shown:
  - CNN Architectures: LeNet5, Alexnet, VGG, GoogleNet, Resnet
- Now
  - Recurrent Neural Networks
  - LSTM
- Reference:
  - Feifei Li, Stanford cs231n
Recurrent Neural Networks
“Vanilla” Neural Network

one to one

Vanilla Neural Networks
Recurrent Neural Networks: Process Sequences

- One to one
- One to many
- Many to one
- Many to many
- Many to many

E.g. Image Captioning
image -> sequence of words
Recurrent Neural Networks: Process Sequences

- **one to one**
- **one to many**
- **many to one**
- **many to many**

*Example: Sentiment Classification*

sequence of words -> sentiment
Recurrent Neural Networks: Process Sequences

e.g. Machine Translation
seq of words -> seq of words
Recurrent Neural Networks: Process Sequences

e.g. Video classification on frame level
Sequential Processing of Non-Sequence Data

Classify images by taking a series of “glimpses”


Figure copyright Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra, 2015. Reproduced with permission.
Recurrent Neural Network
Recurrent Neural Network

usually want to predict a vector at some time steps
We can process a sequence of vectors $x$ by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

- **new state**
- **old state**
- **input vector at some time step**
- **some function with parameters $W$**
We can process a sequence of vectors $\mathbf{x}$ by applying a recurrence formula at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.
The state consists of a single "hidden" vector $h$: 

$$ h_t = f_W(h_{t-1}, x_t) $$

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

$$ y_t = W_{hy} h_t $$

Or, 

$$ y_t = \text{softmax}(W_{hy} h_t) $$
RNN: Computational Graph
Time invariant systems

RNN: Computational Graph

Re-use the same weight matrix at every time-step

\[ h_0 \xrightarrow{f_W} h_1 \xrightarrow{f_W} h_2 \xrightarrow{f_W} h_3 \xrightarrow{\ldots} h_T \]

\[ W \xrightarrow{x_1} x_2 \xrightarrow{x_3} \ldots \]
RNN: Computational Graph: Many to Many
RNN: Computational Graph: Many to Many

\[
\begin{align*}
  h_0 & \rightarrow f_W & h_1 & \rightarrow f_W & h_2 & \rightarrow f_W & h_3 & \rightarrow \ldots & h_T \\
  x_1 & \rightarrow W & x_2 & \rightarrow W & x_3 & \rightarrow W & L_1 & \rightarrow L_2 & \rightarrow L_3 & \rightarrow L_T \\
  y_1 & \rightarrow L_1 & y_2 & \rightarrow L_2 & y_3 & \rightarrow L_3 & y_T & \rightarrow L_T
\end{align*}
\]
RNN: Computational Graph: Many to One
RNN: Computational Graph: One to Many

\[ h_0 \xrightarrow{f_W} h_1 \xrightarrow{f_W} h_2 \xrightarrow{f_W} h_3 \xrightarrow{\ldots} h_T \]

\[ W \]

\[ x \]

\[ y_3 \]

\[ y_T \]
Sequence to Sequence: Many-to-one + one-to-many

**Many to one**: Encode input sequence in a single vector

**One to many**: Produce output sequence from single input vector
Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”
Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”
Example:
Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: “hello”
Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model
Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model
Example:
Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model
Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model
Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient
**Truncated Backpropagation through time**

Run forward and backward through chunks of the sequence instead of whole sequence.
Truncated Backpropagation through time

43

Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps.
Truncated Backpropagation through time
Example: Text->RNN

THE SONNETS
by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thyself alone,
And止 thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself dry fuel, to thy sweet self too cruel:
That art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thou own bud buried thy content,
And tender churl mak'st waste in niggardling;
Pity the world, or else this gluton be,
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held;
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserveth thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine
This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.

https://gist.github.com/karpathy/d4dee566867f8291f8291f086
at first:

"Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had oft en him. Pierre aking his soul came to the packs and drove up his father-in-law women.

"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennnc Phe lism thond hon at. MeiDimorotion in ther thize."

train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of
her heary, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.

train more

"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennnc Phe lism thond hon at. MeiDimorotion in ther thize."

train more
Image Captioning

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick
Convolutional Neural Network

Recurrent Neural Network
before:
\[ h = \tanh(W_{xh} \cdot x + W_{hh} \cdot h) \]

now:
\[ h = \tanh(W_{xh} \cdot x + W_{hh} \cdot h + W_{ih} \cdot v) \]
test image

sample

<END> token

=> finish.
Image Captioning: Example Results

- A cat sitting on a suitcase on the floor
- A cat is sitting on a tree branch
- A dog is running in the grass with a frisbee
- A white teddy bear sitting in the grass
- Two people walking on the beach with surfboards
- A tennis player in action on the court
- Two giraffes standing in a grassy field
- A man riding a dirt bike on a dirt track

Captions generated using neurtalk2
All images are CC0 Public Domain: cat suitcase, cat tree, dog, bear, surfers, tennis, giraffe, motorcycle
Image Captioning: Failure Cases

A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

A man in a baseball uniform throwing a ball

A bird is perched on a tree branch

A person holding a computer mouse on a desk

Captions generated using neuraltalk2. All images are CC0 Public domain: fur coat, handstand, spider web, baseball.
Vanilla RNN Gradient Flow

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

\[ = \tanh \begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \]

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

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

Backpropagation from $h_t$ to $h_{t-1}$ multiplies by $W$ (actually $W_{hh}^T$)

$$h_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t)$$

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

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

Bengio et al., "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al., "On the difficulty of training recurrent neural networks", ICML 2013
Vanilla RNN Gradient Flow

Computing gradient of $h_0$ involves many factors of $W$ (and repeated tanh)

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

\[
\begin{align*}
&h_0 \rightarrow x_1 \\
&h_1 \rightarrow x_2 \\
&h_2 \rightarrow x_3 \\
&h_3 \rightarrow x_4 \\
&h_4
\end{align*}
\]

Computing gradient of \( h_0 \) involves many factors of \( W \) (and repeated tanh)

Largest singular value > 1: 
**Exploding gradients**

Largest singular value < 1: 
**Vanishing gradients**

Bengio et al., “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

Computing gradient of \( h_0 \) involves many factors of \( W \) (and repeated tanh)

Largest singular value > 1: Exploding gradients
Largest singular value < 1: Vanishing gradients

Gradient clipping: Scale gradient if its norm is too big

\[
\text{grad}_\text{norm} = \text{np.sum}(\text{grad} * \text{grad})
\]
\[
\text{if grad}_\text{norm} > \text{threshold}:
\]
\[
\text{grad} *= (\text{threshold} / \text{grad}_\text{norm})
\]

Bengio et al. "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Vanilla RNN Gradient Flow

Computing gradient of $h_0$ involves many factors of $W$ (and repeated tanh)

Largest singular value > 1: Exploding gradients

Largest singular value < 1: Vanishing gradients

Change RNN architecture

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Long Short Term Memory (LSTM)
Long Short Term Memory (LSTM)

**Vanilla RNN**

\[ h_t = \tanh \left( W \left( h_{t-1} \right) \right) \]

**LSTM**

\[
\begin{pmatrix}
i \\
f \\
o \\
g \\
\end{pmatrix} =
\begin{pmatrix}
\sigma \\
\sigma \\
\sigma \\
\tanh \\
\end{pmatrix}
W
\begin{pmatrix}
h_{t-1} \\
x_t \\
\end{pmatrix}
\]

\[
c_t = f \odot c_{t-1} + i \odot g
\]

\[
h_t = o \odot \tanh(c_t)
\]

Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation 1997
Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

\[
\begin{align*}
\text{vector from below (} x \text{)} & \quad \text{vector from before (} h \text{)} \\
\begin{array}{c}
W \\
x \\
h
\end{array} & \quad \begin{array}{c}
sigmoid \\
sigmoid \\
sigmoid \\
tanh
\end{array} & \quad \begin{array}{c}
i \\
f \\
o \\
g
\end{array}
\end{align*}
\]

4h x 2h \quad 4h \quad 4*h

f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell

\[
\begin{pmatrix}
i \\
f \\
o \\
g
\end{pmatrix} = \begin{pmatrix}
\sigma \\
\sigma \\
\sigma \\
tanh
\end{pmatrix} W \begin{pmatrix}
h_{t-1} \\
\sigma
\end{pmatrix}
\]

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

[Hochreiter et al., 1997]

\[
\begin{align*}
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{f}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{align*}
\]

\[
\begin{pmatrix}
\mathbf{\hat{c}}_t \\
\mathbf{\hat{o}}_t \\
\mathbf{\hat{g}}_t
\end{pmatrix} = 
\begin{pmatrix}
\sigma & \sigma & \sigma \\
\sigma & \sigma & \tanh
\end{pmatrix}
\begin{pmatrix}
\mathbf{h}_{t-1} \\
\mathbf{x}_t
\end{pmatrix}
\]
Long Short Term Memory (LSTM): Gradient Flow
[Hochreiter et al., 1997]

Backpropagation from $c_t$ to $c_{t-1}$ only elementwise multiplication by $f$, no matrix multiply by $W$

\[
\begin{pmatrix}
i \\
f \\
o \\
g
\end{pmatrix} = \begin{pmatrix}
\sigma \\
\sigma \\
\sigma \\
tanh
\end{pmatrix} W \begin{pmatrix}
h_{t-1} \\
x_t
\end{pmatrix}
\]

\[
c_t = f \odot c_{t-1} + i \odot g
\]

\[
h_t = o \odot \tanh(c_t)
\]
Long Short Term Memory (LSTM): Gradient Flow
[Hochreiter et al., 1997]

Uninterrupted gradient flow!

Similar to ResNet!
Long Short Term Memory (LSTM): Gradient Flow
[Hochreiter et al., 1997]

Uninterrupted gradient flow!

In between:
Highway Networks
\[ g = T(x, W_T) \]
\[ y = g \circ H(x, W_H) + (1 - g) \circ x \]
Srivastava et al., “Highway Networks”, ICML DL Workshop 2015

Similar to ResNet!
Multilayer RNNs

\[ h^l_t = \tanh(W^l (h^{l-1}_t, h^{l-1}_{t-1})) \]

\( h \in \mathbb{R}^n \quad W^l \ [n \times 2n] \)

LSTM:

\[
\begin{pmatrix}
i \\
f \\
o \\
g
\end{pmatrix} = 
\begin{pmatrix}
sigm & sigm \\
sigm & tanh
\end{pmatrix}
W^l 
\begin{pmatrix}
h^{l-1}_t \\
h^{l-1}_{t-1}
\end{pmatrix}
\]

\[ c^l_t = f \odot c^l_{t-1} + i \odot g \]

\[ h^l_t = o \odot \tanh(c^l_t) \]
Other RNN Variants

**GRU** [Learning phrase representations using rnn encoder-decoder for statistical machine translation, Cho et al. 2014]

\[
\begin{align*}
  r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
  z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\
  \tilde{h}_t &= \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \\
  h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t
\end{align*}
\]


[An Empirical Exploration of Recurrent Network Architectures, Jozefowicz et al., 2015]

\[
\begin{align*}
  z &= \text{sign}(W_{xz}x_t + b_z) \\
  r &= \text{sign}(W_{xz}x_t + W_{hr}h_{t-1} + b_r) \\
  h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + \tanh(x_t) + b_h) \odot z \\
  &\quad + h_t \odot (1 - z)
\end{align*}
\]

**MUT1:**

\[
\begin{align*}
  z &= \text{sign}(W_{xz}x_t + W_{hr}h_{t-1} + b_z) \\
  r &= \text{sign}(W_{xz}x_t + W_{hr}h_{t-1} + b_r) \\
  h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{hh}x_t + b_h) \odot z \\
  &\quad + h_t \odot (1 - z)
\end{align*}
\]

**MUT2:**

\[
\begin{align*}
  z &= \text{sign}(W_{xz}x_t + W_{hr}h_{t-1} + b_z) \\
  r &= \text{sign}(W_{xz}x_t + W_{hr}h_{t-1} + b_r) \\
  h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{hh}x_t + b_h) \odot z \\
  &\quad + h_t \odot (1 - z)
\end{align*}
\]

**MUT3:**

\[
\begin{align*}
  z &= \text{sign}(W_{xz}x_t + W_{hr}h_{t-1} + b_z) \\
  r &= \text{sign}(W_{xz}x_t + W_{hr}h_{t-1} + b_r) \\
  h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{hh}x_t + b_h) \odot z \\
  &\quad + h_t \odot (1 - z)
\end{align*}
\]
Image Captioning with Attention

RNN focuses its attention at a different spatial location when generating each word


Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kinos, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Richard S. Zemel, and Yoshua Bencho, 2015. Reproduced with permission.
Image Captioning with Attention

Image Captioning with Attention

Image Captioning with Attention

Image: H x W x 3

CNN

Features: L x D

Distribution over L locations

h0

h1

a1

Weighted combination of features

Weighted features: D

z1

y1

First word

Image Captioning with Attention

Image Captioning with Attention

Image Captioning with Attention

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kin, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bencho, 2015. Reproduced with permission.
Image Captioning with Attention

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Richard S. Zemel, and Yoshua Bengio, 2015. Reproduced with permission.
Summary

- RNN is flexible in architectures
- Vanilla RNNs are simple but don’t work very well
- Common to use LSTM or GRU: their additive interactions improve gradient flow
  - Backward flow of gradients in RNN can explode or vanish.
  - Exploding is controlled with gradient clipping.
  - Vanishing is controlled with additive interactions
- Better/simpler architectures are a hot topic of current research
- Better understanding (both theoretical and empirical) is needed
Thank you!