CMPE 258, Deep Learning

Sequence learning & NLP

May 03, 2018

DMH 149A

Taehee Jeong

Ph.D., Data Scientist

SJSU SAN JOSÉ STATE UNIVERSITY

Final exam

Comprehensive exam

- Linear regression, logistic regression, DNN, CNN, RNN

Thursday, May 17

14:45-17:00

DMH 149A

Group Project schedule

Presentation date : 5/8, 5/10

Report (including code) due date : 5/13

Number of members : 1 to 4

Content: DNN, CNN, RNN related

Platform : Pandas, Numpy, tensorflow, keras (please discuss with me for others)

Grading policy:

- Content : 40 pts
  ; Creativity in data collection, Neural network architecture / algorithm, application (same quality as a conference paper)
- Presentation : 20 pts
- Report : 20 pts
- Code : 20 pts

Last class

- Character-based token → 1D convolution(word) + pooling(n-gram) → Neural Network → softmax
- Count vector or TF-IDF→ Neural Network → softmax
- Word vector (word2vec, glove) → Neural Network → softmax
- Word-embedding (word2vec, glove) → RNN

Character-level language model

Vocabulary = [a, b, c, …, z, [ ], ., ‘, ; , … , A, …, Z]

- Advantage:
  - Do not worry about unknown token.
- Disadvantage:
  - Much longer sequence
  - Is not as good as word level language models at capturing long range dependencies between how
  - the earlier parts of the sentence also affect the later part of the sentence.
  - more computationally expensive to train

Coursera: Deep learning Specialization, Andrew Ng

Example of Name entity recognition

Name entity recognition can be used to find people's names, companies names, times, locations, countries names, currency names, and so on.

x: Harry Potter and Hermione Granger invented a new spell.

y: 1 1 0 1 1 0 0 0 0

Coursera: Deep learning Specialization, Andrew Ng

Name entity recognition

x: Harry Potter and Hermione Granger invented a new spell.

Position of index $x^{<1>}$ $x^{<2>}$ $x^{<3>}$ . . . . . $x^{<t>}$ . . . . $x^{<9>}$

length $T_x = 9$

y: 1 1 0 1 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ . . . . $y^{<9>}$

length $T_y = 9$

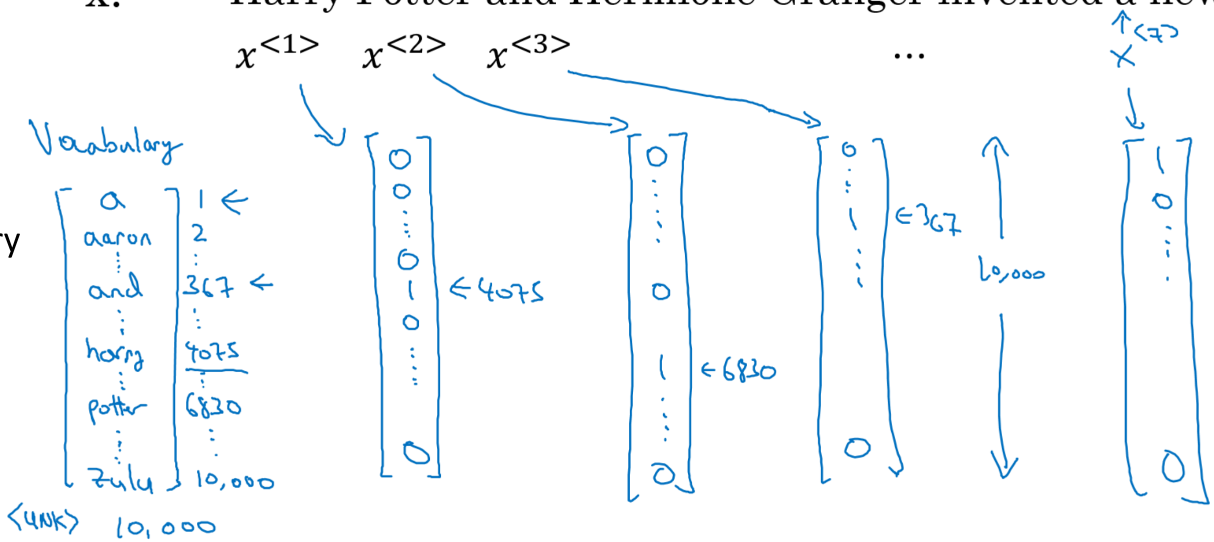
Coursera: Deep learning Specialization, Andrew Ng

One-hot vector to represent words

Map: $x^{<t>} \rightarrow y^{<t>}$

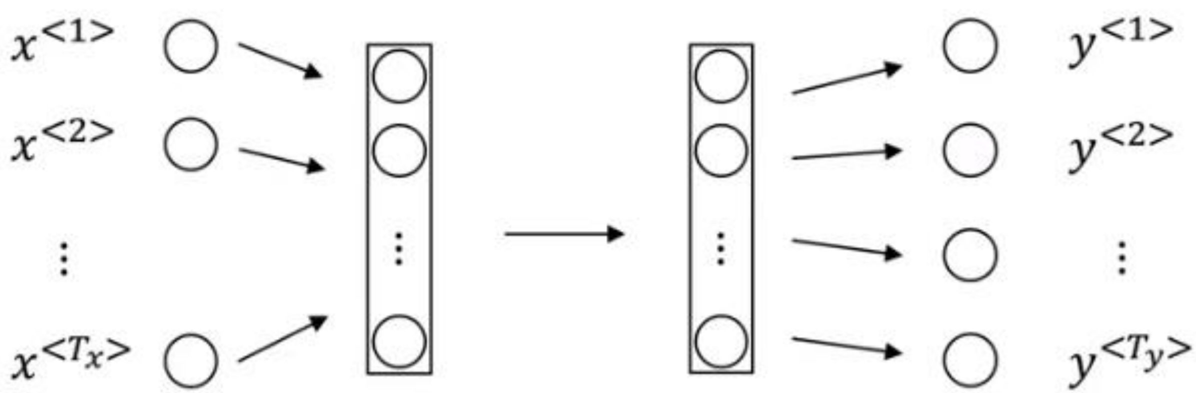
x: Harry Potter and Hermione Granger invented a new spell.

dictionary



Coursera: Deep learning Specialization, Andrew Ng

Why not a Neural Network?



Problems:

- Arbitrary sequence length : Inputs, outputs can be different lengths in different examples.
- Doesn't share features learned across different positions of text.
- Large number of parameters

Coursera: Deep learning Specialization, Andrew Ng

Language model

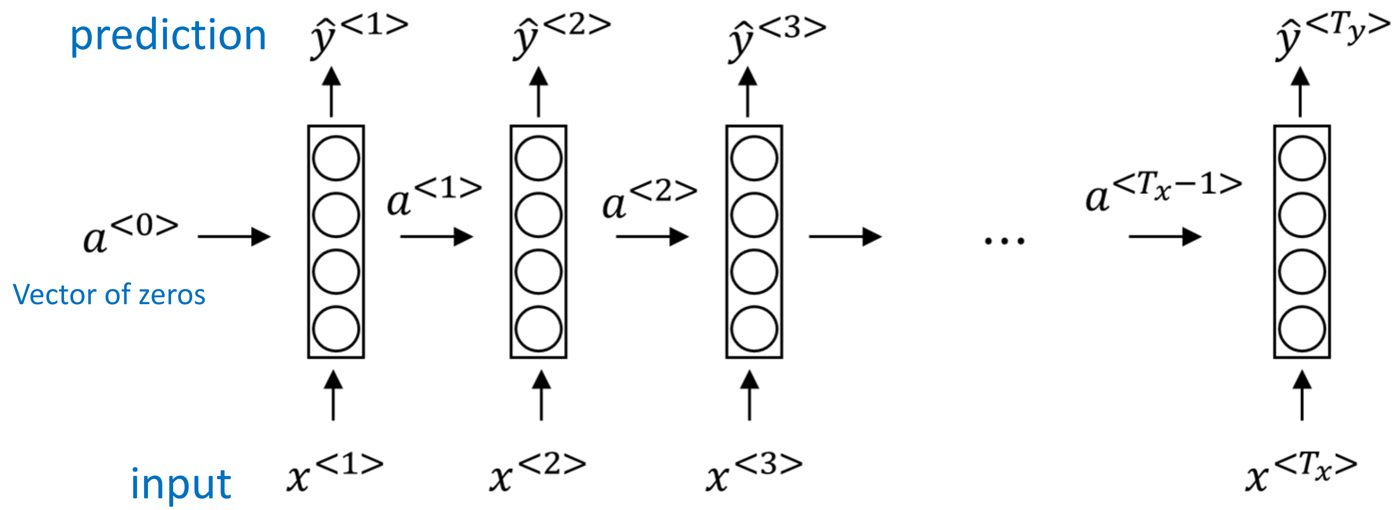
Cats sleep average 15 hours per a day. <EOS>

$$P(text) = P(x_0, \ldots, x_n) =$$
$$= P(x_0)P(x_1|x_0)P(x_2|x_0, x_1)\ldots P(x_n|\ldots)$$

$$\hat{P}(w_1^T) = \prod_{t=1}^{T} \hat{P}(w_t|w_1^{t-1})$$ A neural probabilistic language model , Bengio et. al., 2003

Coursera: Natural Language Processing, National Research University Higher School of Economics

Recurrent Neural Networks

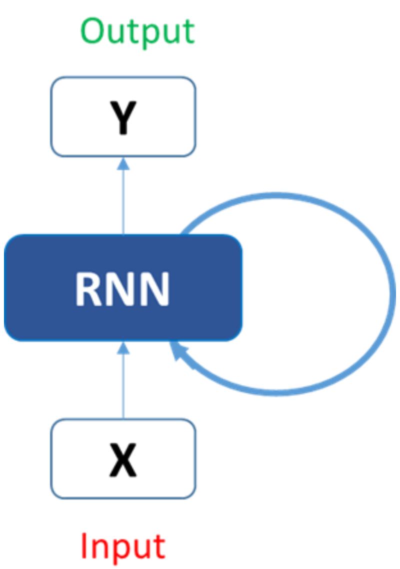


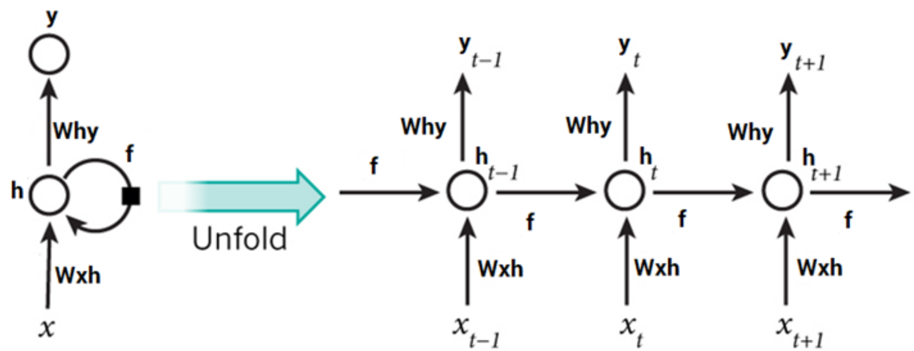
$x^{<t>}$: current input (word embedding)
$a^{<t-1>}$: activation value in previous hidden state
$a^{<t>}$: activation value in current hidden state
$y^{<t>}$: prediction (probability distribution for the next word)

Fixed number of inputs at each time step
→ Solve the arbitrary sequence length

Coursera: Deep learning Specialization, Andrew Ng

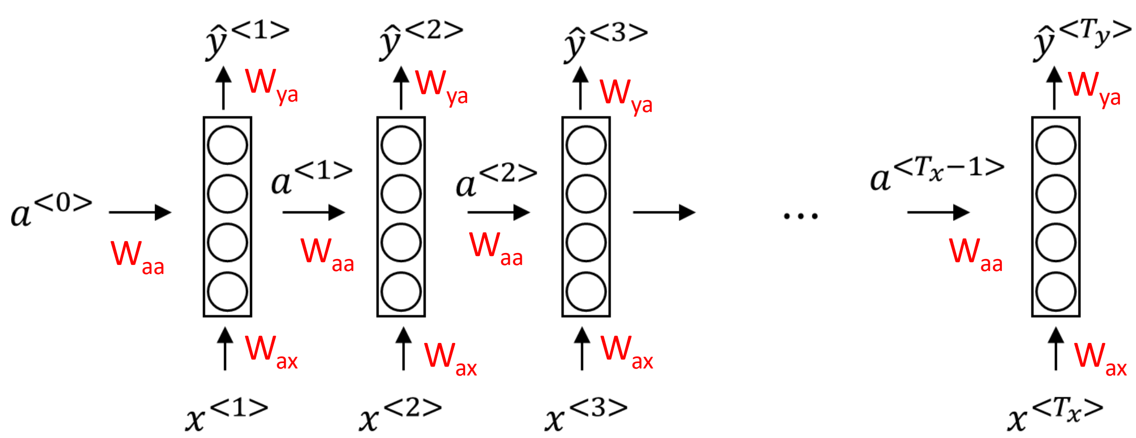
Recurrent Neural Networks





Fundamentals of Deep Learning – Introduction to Recurrent Neural Networks
DISHASHREE GUPTA,2017

Forward Propagation



$x^{<t>}$: current input
$a^{<t-1>}$: activation in previous hidden state
$a^{<t>}$: activation in current hidden state
$y^{<t>}$: prediction

g_1, g_2 : activation function
g_1 : tanh, ReLU
g_2 : sigmoid, softmax

$$\boxed{a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)}$$

$$a^{<1>} = g_1(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a)$$

$$\boxed{\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)}$$

$$\hat{y}^{<1>} = g_2(W_{ya}a^{<1>} + b_y)$$

All the parameters are shared across the different time steps → solve large number of parameters

Coursera: Deep learning Specialization, Andrew Ng

Simplified RNN notation

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

$$\boxed{a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)}$$

$$\boxed{\hat{y}^{<t>} = g(W_y a^{<t>} + b_y)}$$

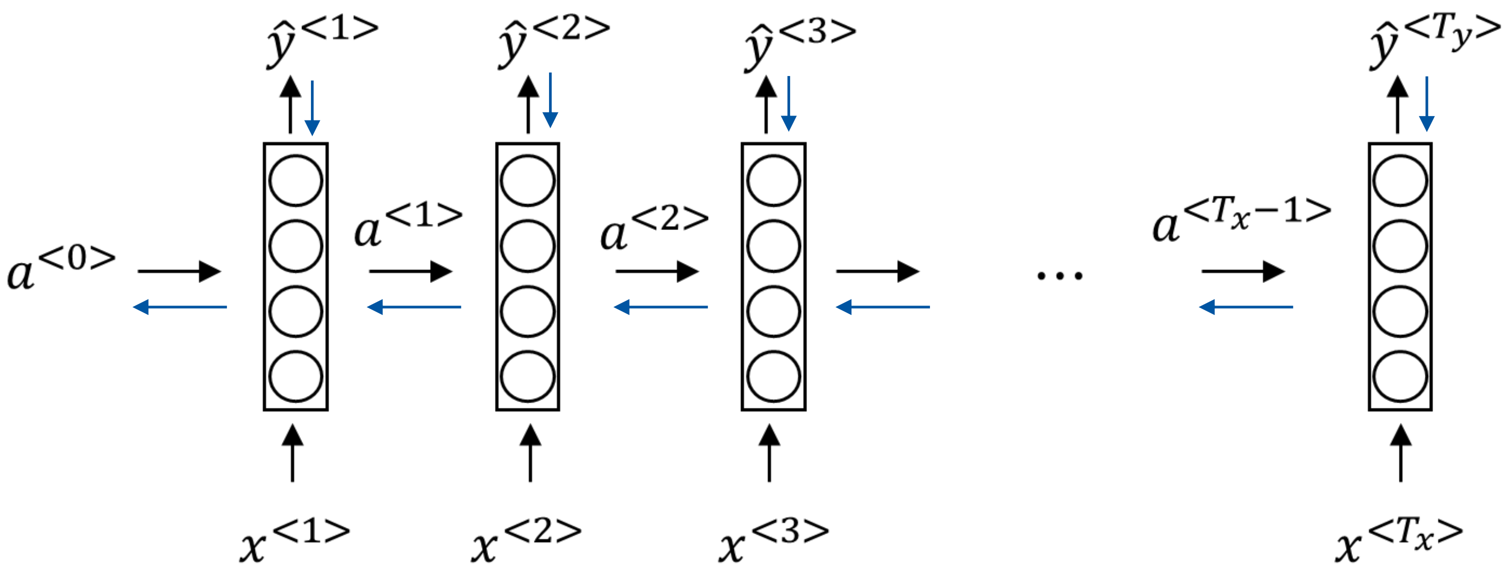
$$[W_{aa}; W_{ax}] = W_a$$

$$[W_{aa}; W_{ax}] \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = W_{aa}a^{<t-1>} + W_{ax}x^{<t>}$$

$x^{<t>}$: current input
$a^{<t-1>}$: activation in previous hidden state
$a^{<t>}$: activation in current hidden state
$y^{<t>}$: prediction

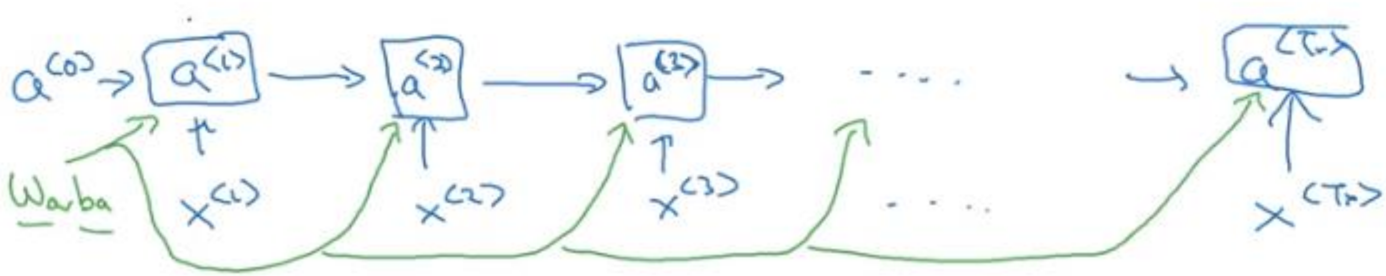
Coursera: Deep learning Specialization, Andrew Ng

Forward propagation and backpropagation



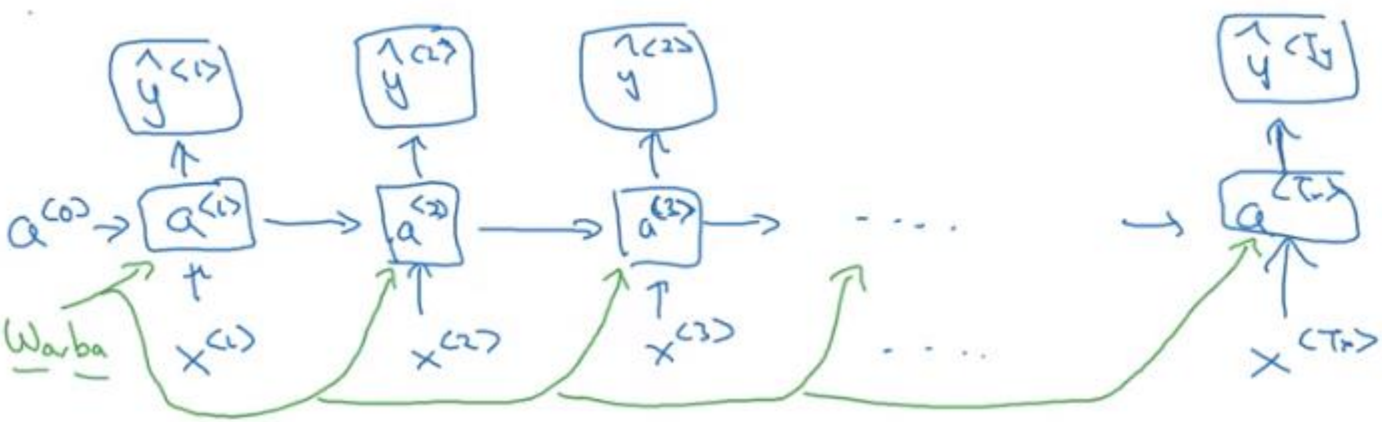
Coursera: Deep learning Specialization, Andrew Ng

Backpropagation through time



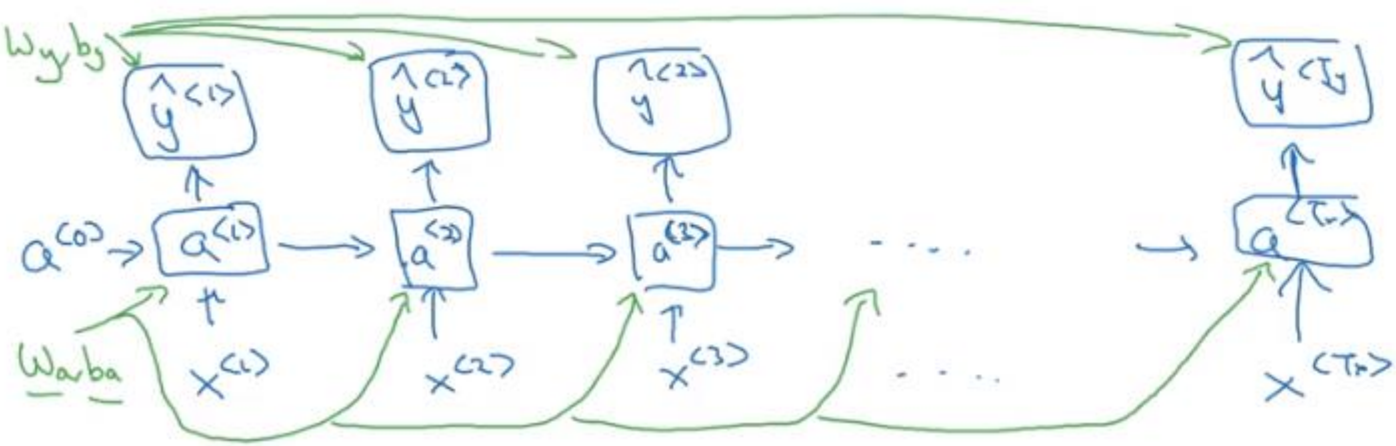
Coursera: Deep learning Specialization, Andrew Ng

Backpropagation through time



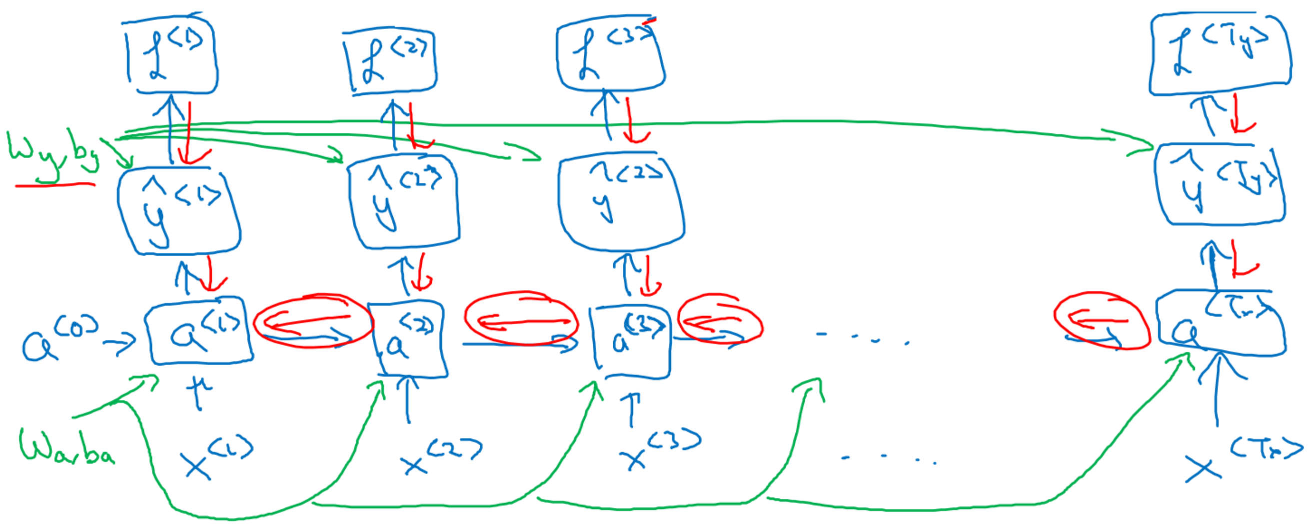
Coursera: Deep learning Specialization, Andrew Ng

Backpropagation through time



Coursera: Deep learning Specialization, Andrew Ng

Backpropagation through time(BPTT)

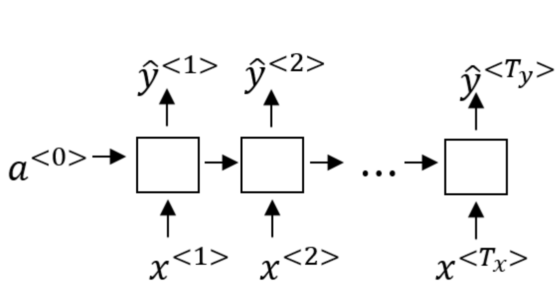


Cross-entropy loss $L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>}\log(\hat{y}^{<t>}) - (1 - y^{<t>})\log(1 - \hat{y}^{<t>})$

Total loss $$L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

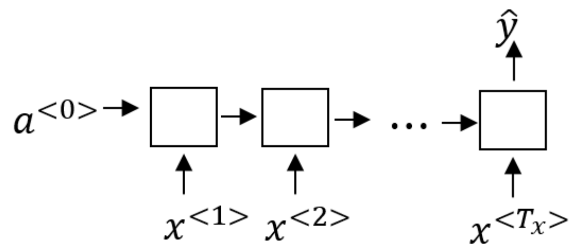
Coursera: Deep learning Specialization, Andrew Ng

RNN types



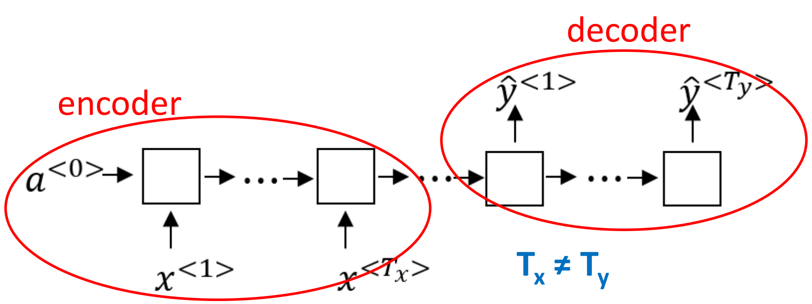
Many to many $T_x = T_y$

Name entity recognition



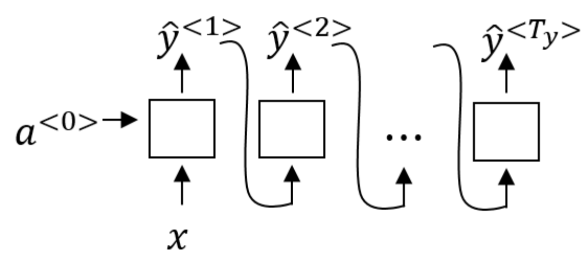
Many to one

Sentiment classification



Many to many

Machine translation

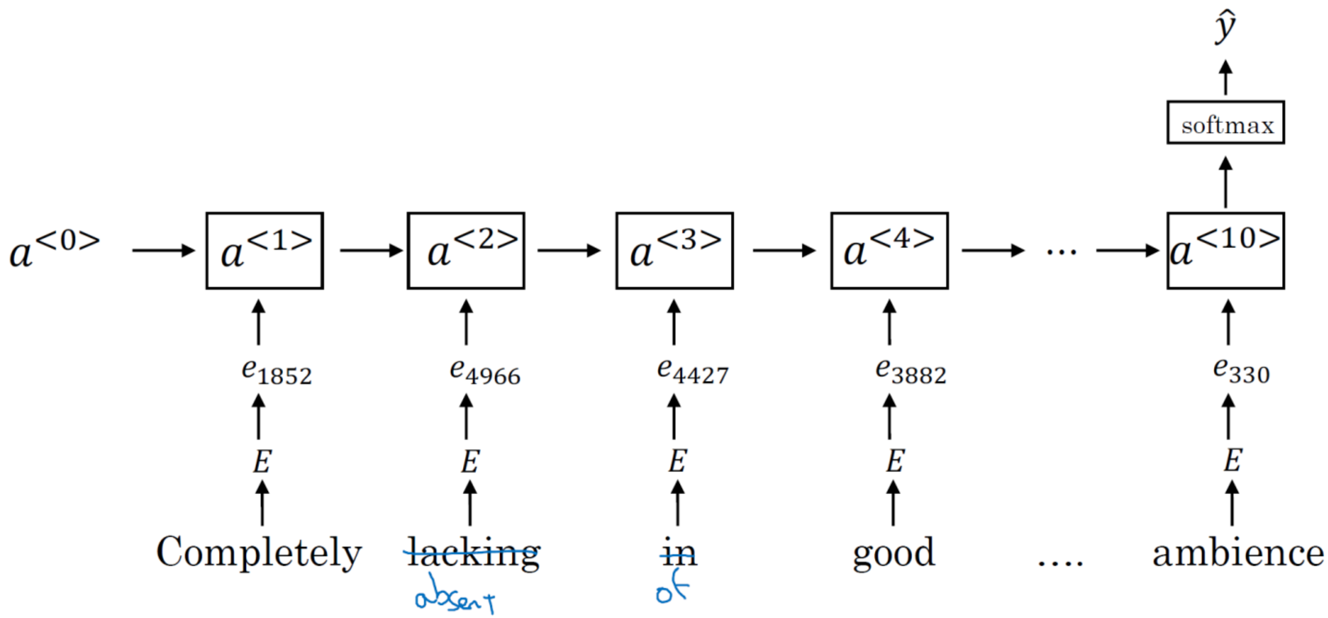


One to many

Music generation

Coursera: Deep learning Specialization, Andrew Ng

RNN for sentiment classification



Coursera: Deep learning Specialization, Andrew Ng

Long range of dependence

The cat, which already ate …, was full.

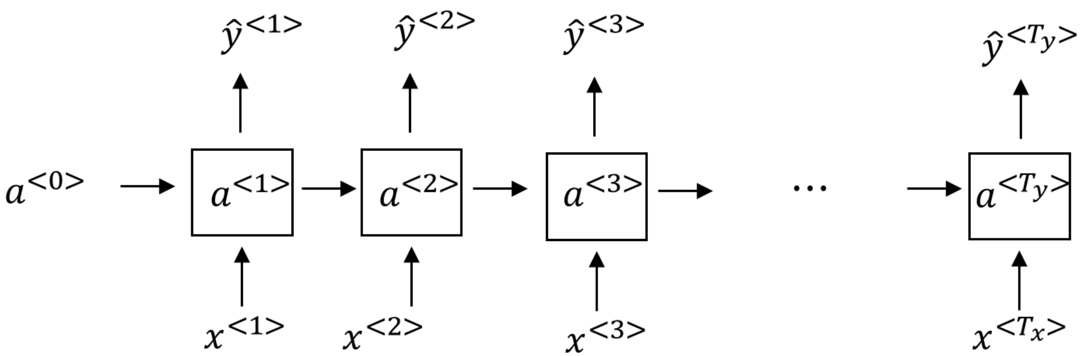
The cats, which already ate …, were full.

long range dependencies is how the earlier parts of the sentence affects the later part of the sentence.

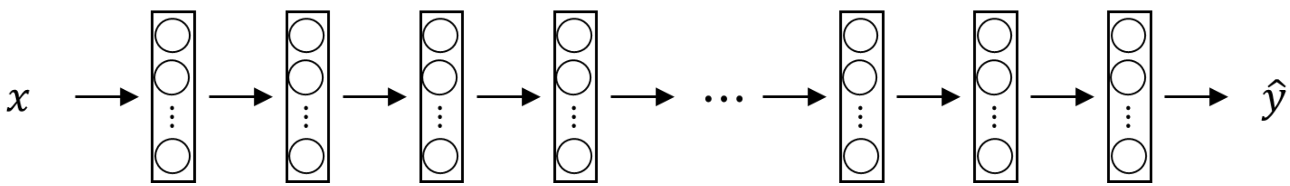
Vanishing gradients with RNNs

The cat, which already ate …, was full.

The cats, which already ate …, were full.



Not easy to capture long range of dependence with RNN.



Coursera: Deep learning Specialization, Andrew Ng

Gradients with RNNs

$$\left| \frac{\partial h_i}{\partial h_{i-1}} \right| < 1$$

- Gradients is vanishing after 3-4 time steps.
  - Not easy to capture long range of dependence between the earlier parts of the sentence and the later part of the sentence.

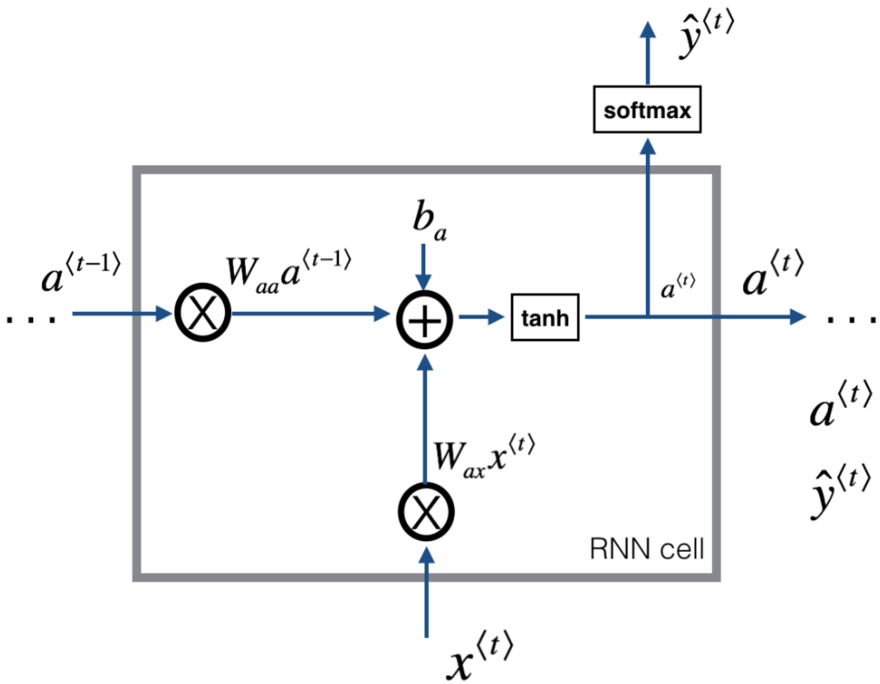
$$\left| \frac{\partial h_i}{\partial h_{i-1}} \right| > 1$$

- Exploding gradients:
  - Learning process becomes unstable
  - Gradient becomes NaN
  - Gradient clipping

Coursera: Natural Language Processing, National Research University Higher School of Economics

Coursera: Deep learning Specialization, Andrew Ng

RNN unit



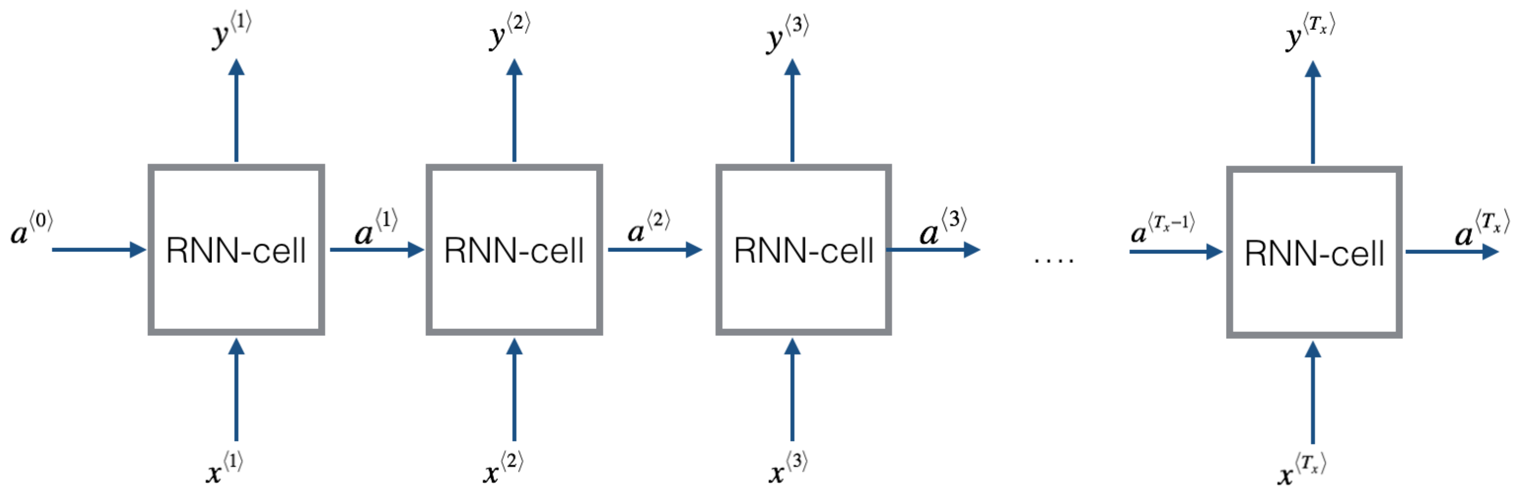
$x^{<t>}$: current input
$a^{<t-1>}$: activation in previous hidden state
$a^{<t>}$: activation in current hidden state
$\hat{y}^{<t>}$: prediction

$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

Coursera: Deep learning Specialization, Andrew Ng

RNN forward pass



Coursera: Deep learning Specialization, Andrew Ng

Gated Recurrent Unit (simplified GRU)

The cat, which already ate …, was full.

$c^{<t>}=1$ $c^{<t>}=1$

$\Gamma_u=1$ $\Gamma_u=0$ $\Gamma_u=0$ $\Gamma_u=1$

c : memory cell
$c^{<t>} = a^{<t>}$
$\tilde{c}^{<t>}$: candidate of activation
Γ_u: Update gate

$$\tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

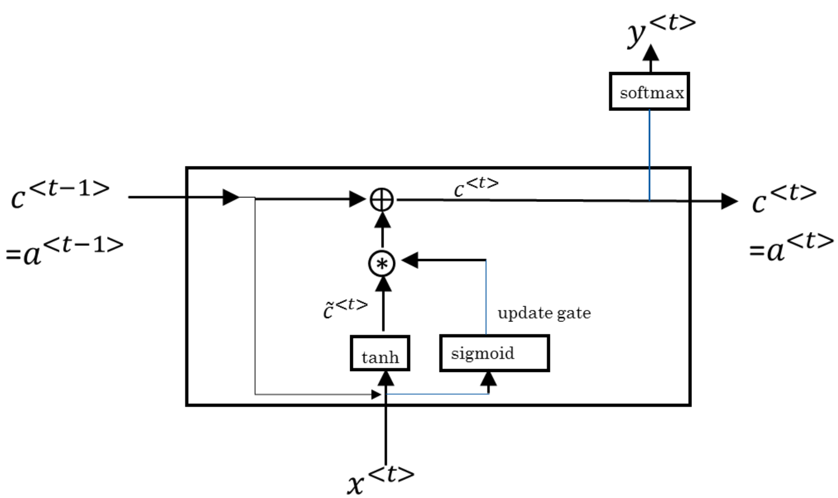
*:Element-wise multiplication

Coursera: Deep learning Specialization, Andrew Ng

On the properties of neural machine translation: Encoder-decoder approaches, Cho et al., 2014.

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, Chung et al., 2014.

Gated Recurrent Unit (GRU)



$$\tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$\Gamma_u \approx 0$, $c^{<t>} = c^{<t-1>}$

$c^{<t>}$ maintains many previous time-steps and helps vanishing gradient problem and allows long range dependencies.

Coursera: Deep learning Specialization, Andrew Ng

On the properties of neural machine translation: Encoder-decoder approaches, Cho et al., 2014.

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, Chung et al., 2014.

Full GRU

$\tilde{h}$ $\tilde{c}^{<t>} = \tanh(W_c[\Gamma_u * c^{<t-1>}, x^{<t>}] + b_c)$ Γ_r: relevance gate

u $\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$

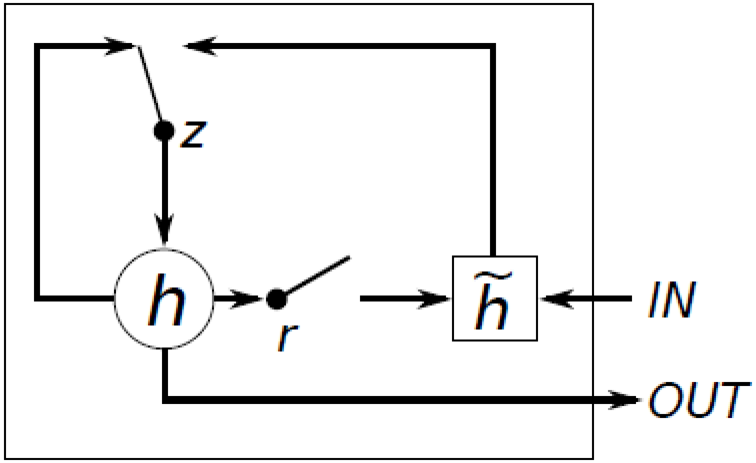
r $\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$

h $c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$

Relevance gate is how relevant $c^{<t-1>}$ is to compute $\tilde{c}^{<t>}$.

Coursera: Deep learning Specialization, Andrew Ng

Gated Recurrent Unit (GRU)



r : reset gate

z : update gates

h : activation in hidden state

$\hat{h}$: candidate activation in hidden state.

On the properties of neural machine translation: Encoder-decoder approaches, Cho et al., 2014.

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, Chung et al., 2014.

GRU vs. LSTM(long short term memory)

GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

update $\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$

forget $\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$

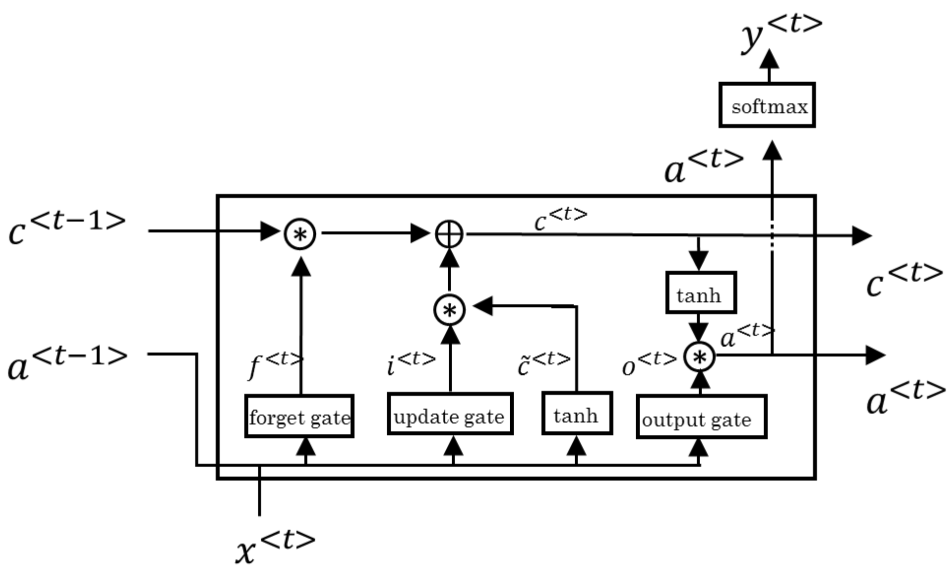
output $\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh(c^{<t>})$$

Coursera: Deep learning Specialization, Andrew Ng
Long short-term memory, Hochreiter & Schmidhuber 1997.

LSTM (long short term memory) cell



$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$

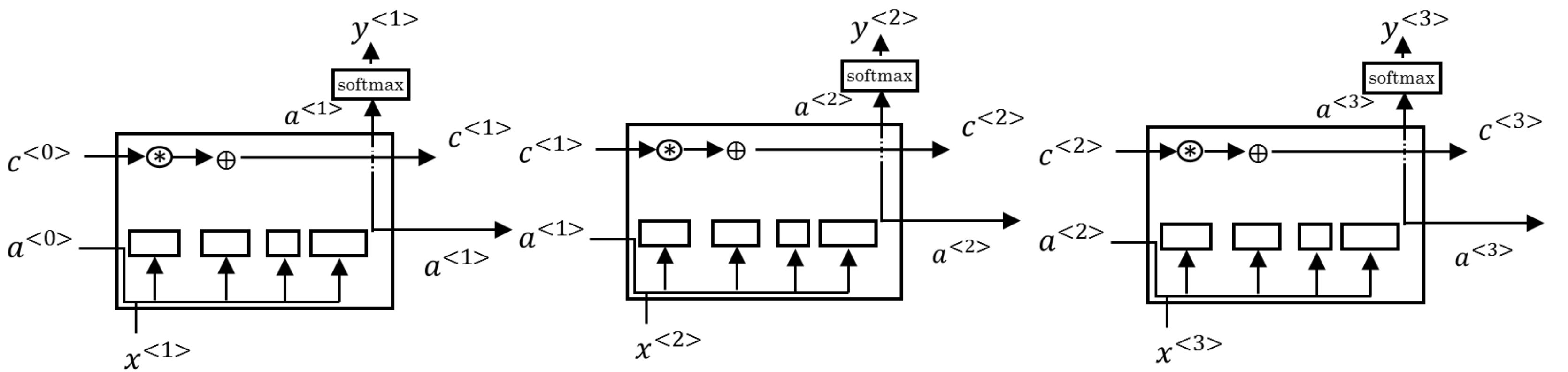
$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh(c^{<t>})$$

Coursera: Deep learning Specialization, Andrew Ng

LSTM forward



$\Gamma_f \approx 0$, $c^{<3>} = c^{<0>}$

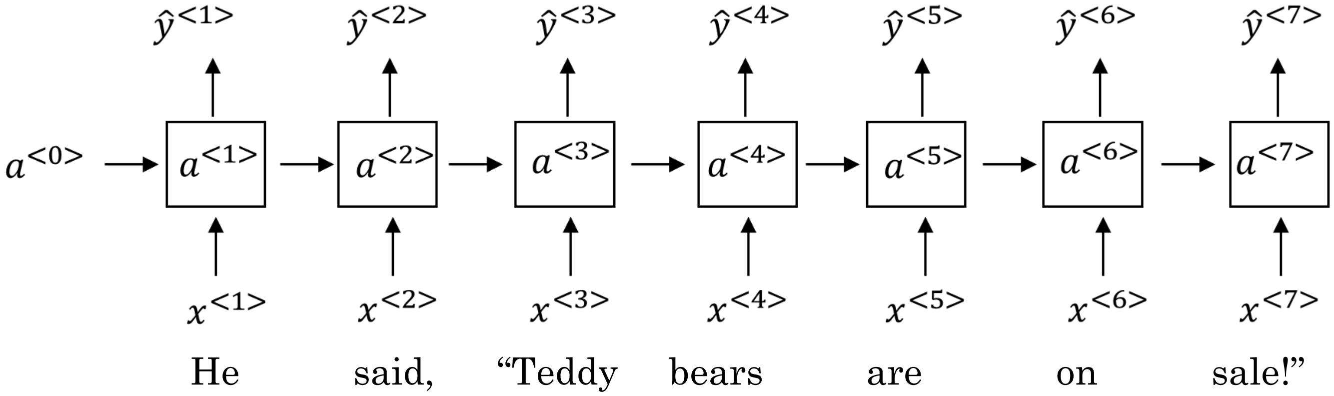
$c^{<t>}$ maintains many previous time-steps and helps vanishing gradient problem and allows long range dependencies.

Coursera: Deep learning Specialization, Andrew Ng

Unidirectional RNN

He said, "Teddy bears are on sale!"

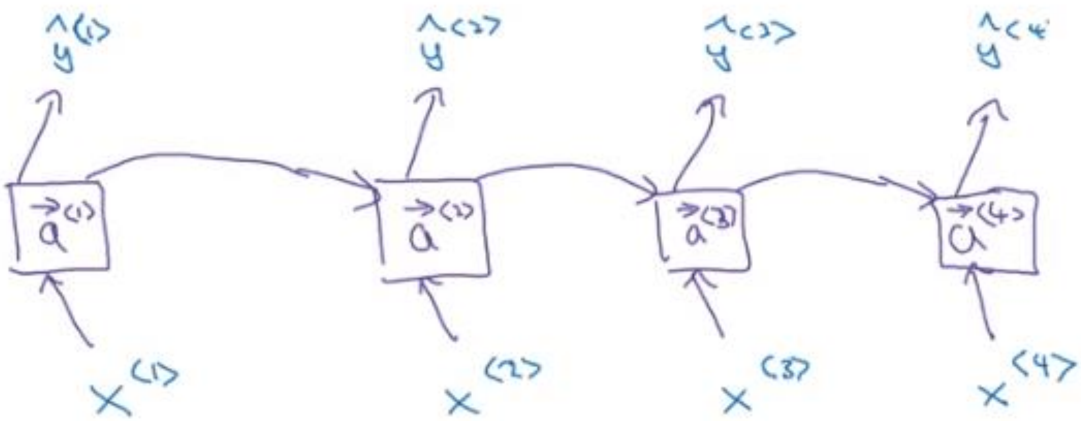
He said, "Teddy Roosevelt was a great President!"



Coursera: Deep learning Specialization, Andrew Ng

Bidirectional RNN

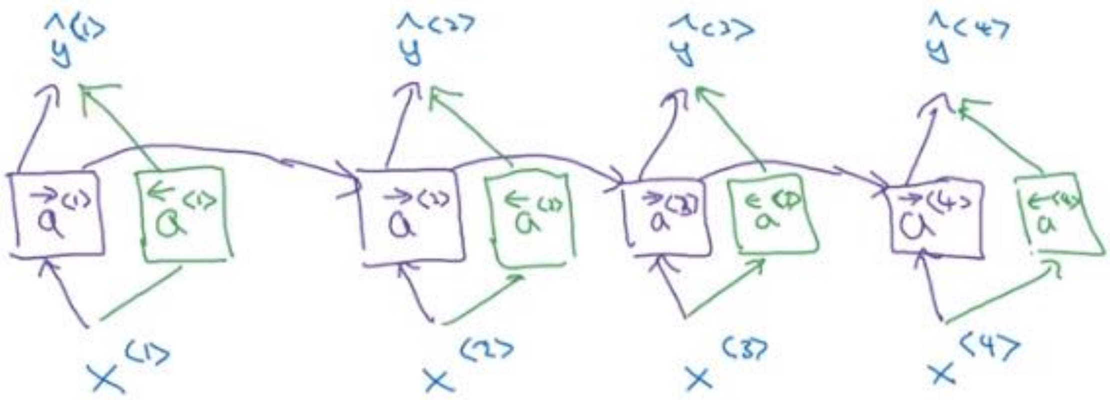
Getting information from the future



Coursera: Deep learning Specialization, Andrew Ng

Bidirectional RNN

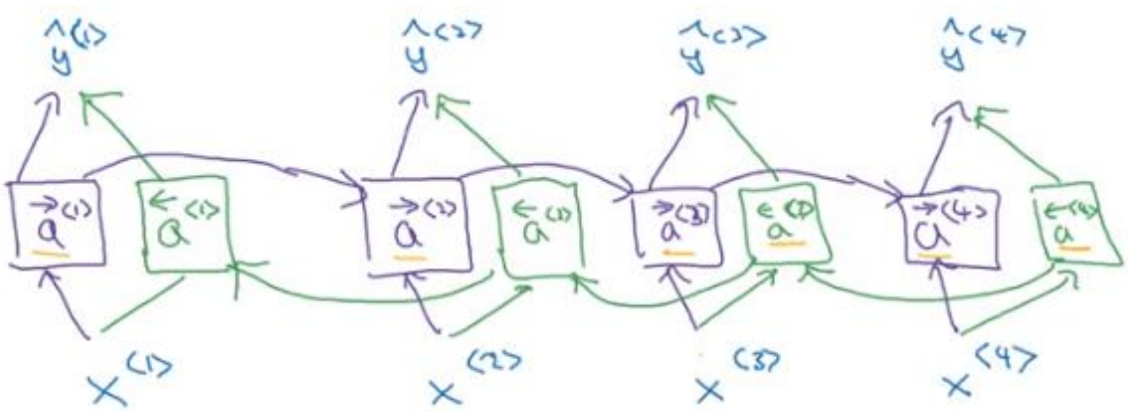
Getting information from the future



Coursera: Deep learning Specialization, Andrew Ng

Bidirectional RNN

Getting information from the future

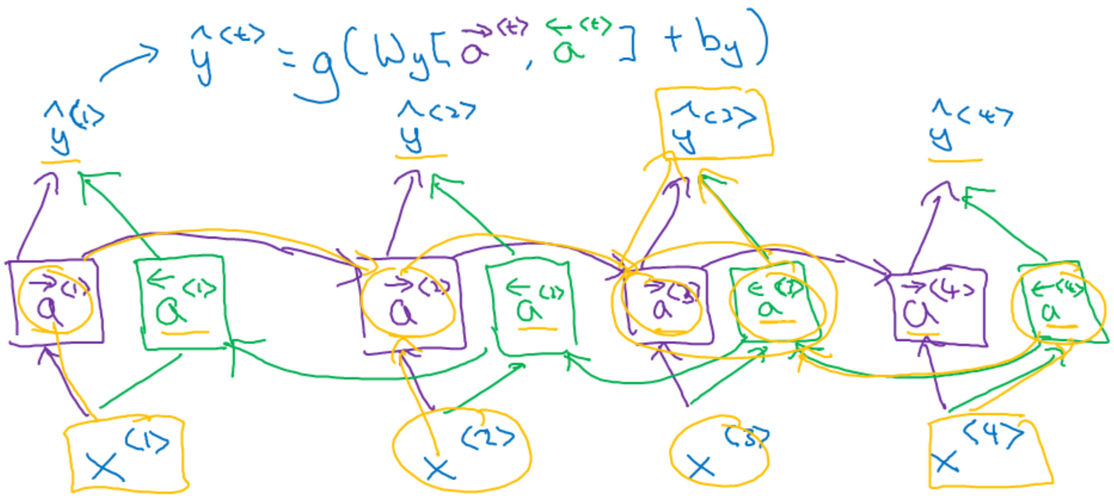


Coursera: Deep learning Specialization, Andrew Ng

SJSU SAN JOSÉ STATE UNIVERSITY

Bidirectional RNN

Getting information from the future

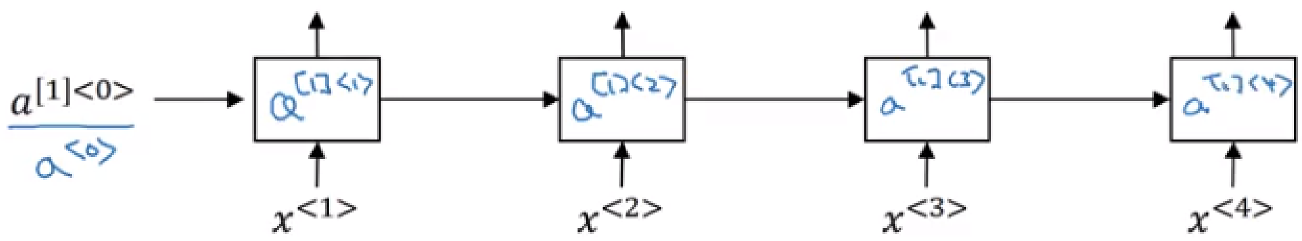


Coursera: Deep learning Specialization, Andrew Ng

BRNN with GRU/LSTM

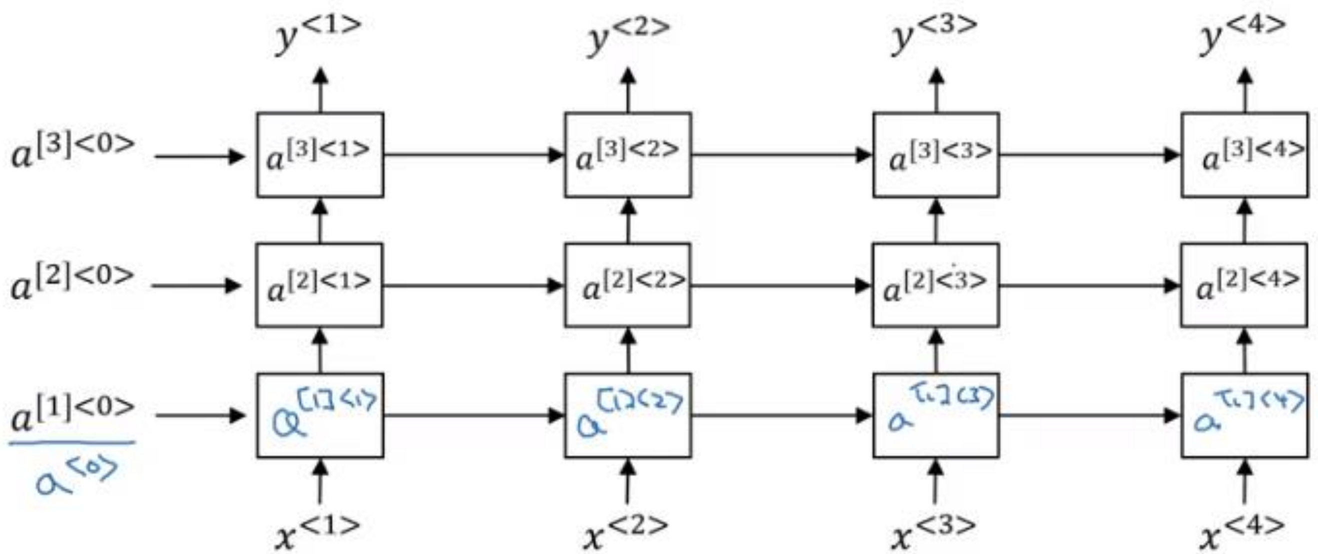
Entire sentence is needed to build BRNN.

Deep RNNs



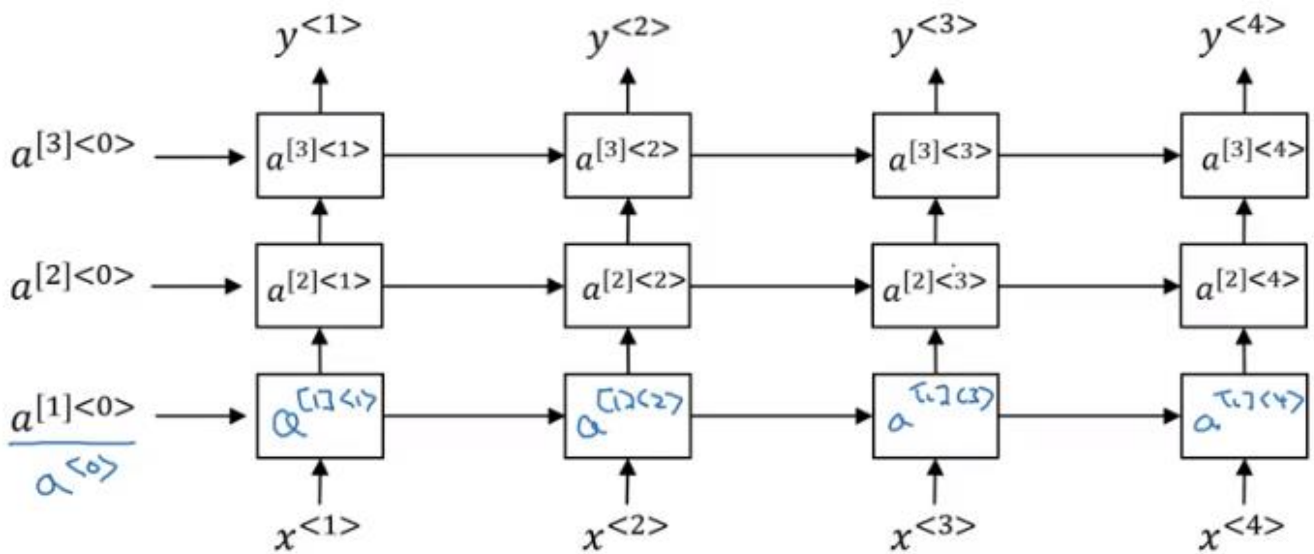
Coursera: Deep learning Specialization, Andrew Ng

Deep RNNs



Coursera: Deep learning Specialization, Andrew Ng

Deep RNNs



$$a^{[2]<3>} = g(W_a^{[2]}[\, a^{[2]<2>}, a^{[1]<3>}] + b_a^{[2]})$$

Coursera: Deep learning Specialization, Andrew Ng

Summary

- RNN
- Backpropagation through time (BPTT)
- RNN type: many to many, one to many, many to one
- Gated Recurrent Unit (GRU)
- long short term memory (LSTM)
- Bidirectional RNN
- Deep RNN