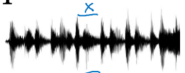
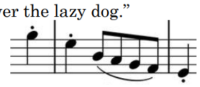
Sequence Models Week 1

Examples of sequence data

Speech recognition		"The quick brown fox jumped over the lazy dog."
Music generation	∅	
Sentiment classification	"There is nothing to like in this movie."	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	Do you want to sing with me?
Video activity recognition		Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	Yesterday, Harry Potter met Hermione Granger.

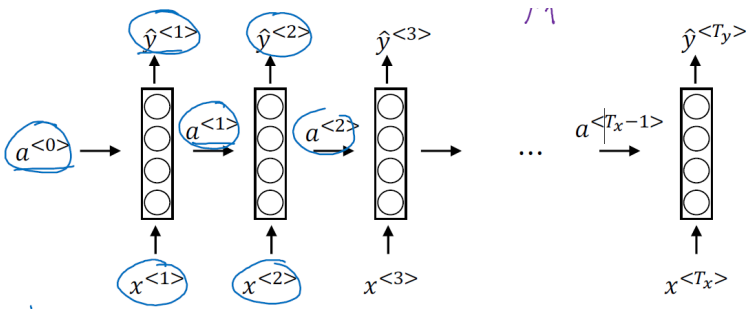
Recurrent Neural Network Model

Why not a standard network?

Problem:

- Inputs, outputs can be different lengths in different examples
- Doesn't share features learned across different positions of text.

RNN model:



Drawback: 只使用前面的信息来预测后面的.
如 $x^{<3>}$, 只能使用 $x^{<1>}$, $x^{<2>}$ 的信息, 但后面的 $x^{<4>}$, ... 等信息则也是有用的.

Forward Propagation:

$a^{<0>} = \vec{0}$

$$a^{<1>} = g_1(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a) \leftarrow \tanh/\text{relu}$$

$$\hat{y}^{<1>} = g_2(W_{ya}a^{<1>} + b_y) \leftarrow \text{sigmoid/softmax}$$

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

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

Try to simplify it to suit the more simpler network

$a^{<t>} = g_1(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a)$ → $a^{<t>} = g(W_a [a^{<t-1>}, x^{<t>}] + b_a)$

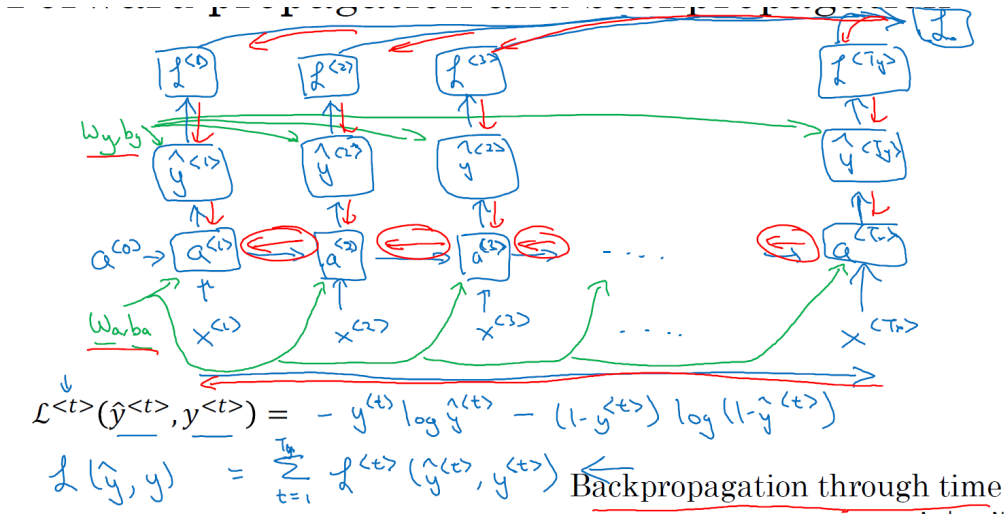
$\hat{y}^{<t>} = g(W_{ya} a^{<t>} + b_y)$

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

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

$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$

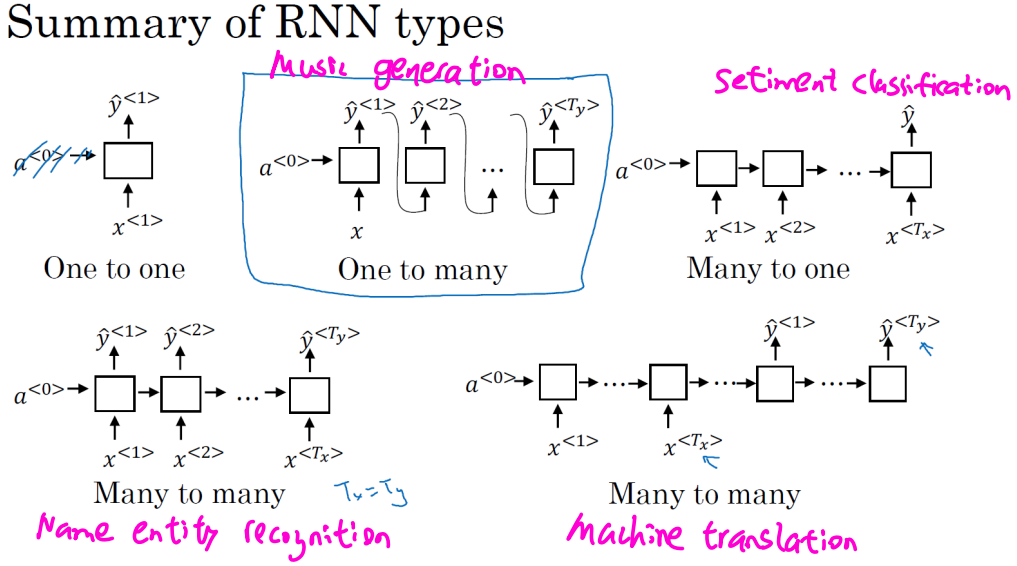
Backpropagation Through Time



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

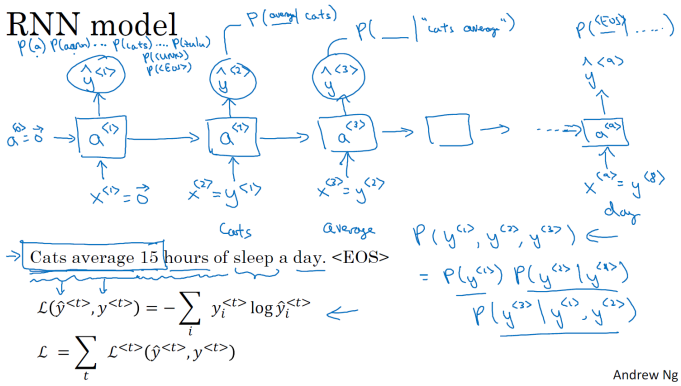
$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$ ← Backpropagation through time

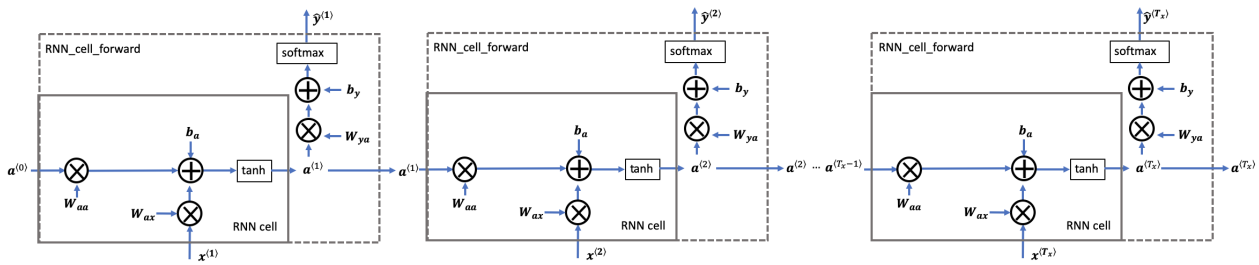
Different Types of RNNs



Language Model and Sequence Generation

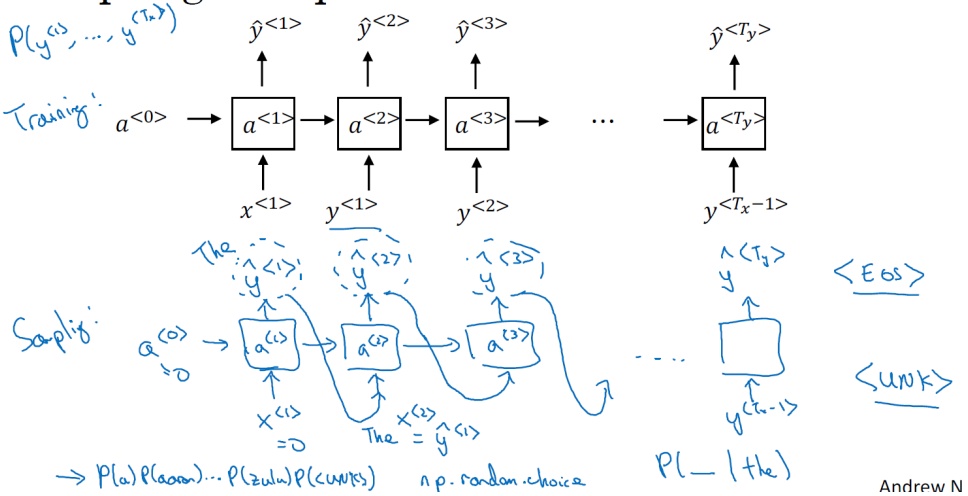
Training set: large corpus of english text.





Sampling Novel Sequences

- Character-level
- word-level



Vanishing Gradient with RNNs

explode / vanish
why?

The basic RNNs cannot catch the long-range dependency.

Recap: Why vanishing Gradient?

① 梯度消失和梯度爆炸是指前面几层的梯度，因为链式法则不断乘小于(大于)1的数，导致梯度非常小(大).

② sigmoid 导数的范围在 [0, 0.25]

③ 如何解决，对一般 CNN →
- 用 ReLU 代替 sigmoid
- 用 BN 层
- 用 ResNet

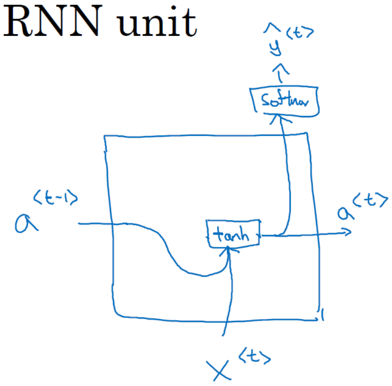
How to solve?

explode → gradient clipping

vanish → ? GRU / LSTM

Gated Recurrent Unit (GRU)

RNN unit



tanh

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

GRU ☆

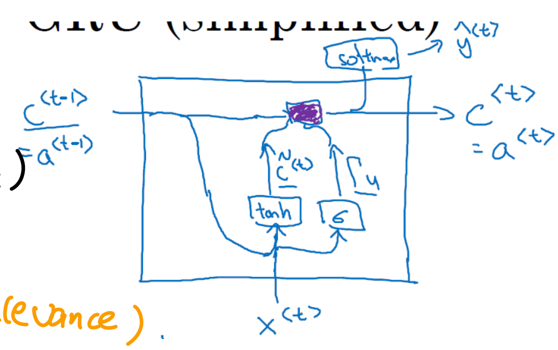
c = memory cell

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

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

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

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



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

Long Short Term Memory (LSTM)

GRU	LSTM
$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$	$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$
$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$	$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$
$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$	$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$ forget gate
$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$	$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$ update gate
$a^{<t>} = c^{<t>}$	$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$ output gate
	$a^{<t>} = \Gamma_o * \tanh(c^{<t>})$

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

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

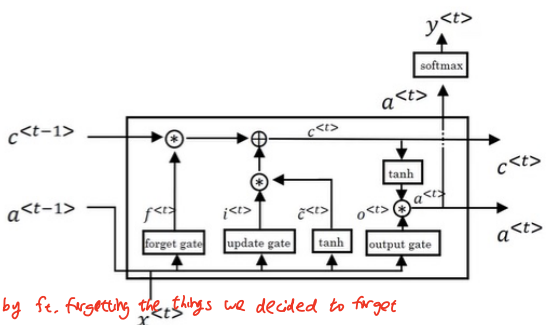
$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$ forget gate

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

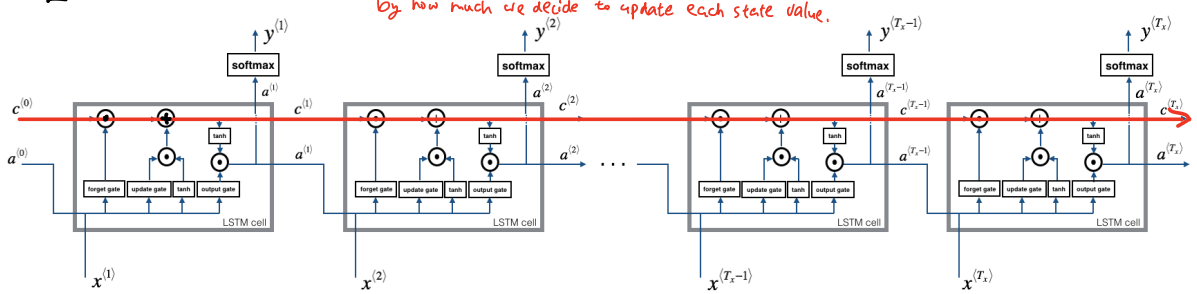
$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$

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

a_next



multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $\Gamma_u * \tilde{c}^{<t>}$. This is a new candidate values, scaled by how much we decide to update each state value.



- a forget gate: decide which inputs units should be remembered and passed along. It is a tensor with values between 0 and 1
- an update gate: again a tensor containing values between 0 and 1. It decides on what information to throw away, and what new information to add.
- output gate: decide what gets as the output of the time step.

Bidirectional RNNs

Deep RNNs

Sequence Model Week 2 Word Representation

Word representation

Transfer learning and word embeddings

1. Learn word embeddings from large text corpus.
   (Or download pre-trained embedding online)
2. Transfer embedding to new task with smaller training set
3. Continue to finetune the word embeddings with new data.

Properties of word embeddings

t-SNE

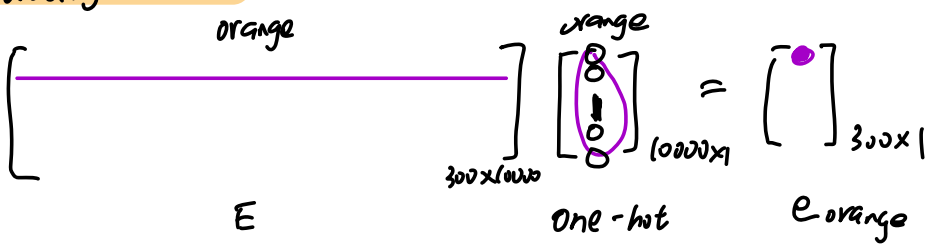
$$e_{man} - e_{woman} = e_{king} - e_{w}$$

Find word w $\Rightarrow$ $\arg\max_{w}$ similarity$(e_w, e_{king} + e_{woman} - e_{man})$

cosine similarity

$$Sim(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$

Embedding matrix

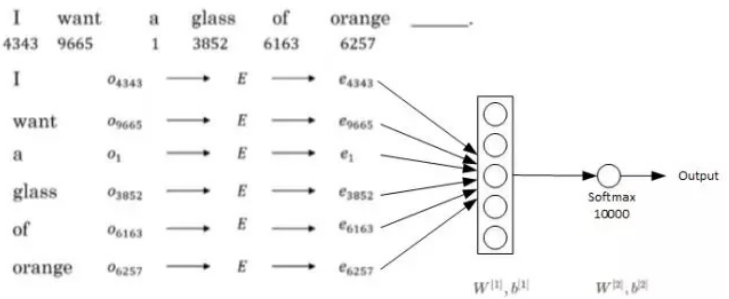


For a given word w, the embedding matrix E is a matrix that maps its 1-hot representation to its embedding e_w as follows:

$$e_w = E \cdot O_w$$

Learning Word Embeddings: Word2Vec & Glove

Learning Word Embeddings



Word2Vec

Word2Vec is a framework aimed at learning word embeddings by estimating the likelihood that a given word is surrounded by other word. Popular models include Skip-gram, negative Sampling and CBOW

Skip-gram

The skip-gram word2Vec model is a supervised learning task that learns word embeddings by accessing the likelihood of any given target word t happening with a context word c.

Softmax: $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

$$P(t|c) = \frac{\exp(\theta_t^T e_c)}{\sum_{j=1}^{|V|} \exp(\theta_j^T e_c)}$$

, where θ_t is a parameter associated with t.

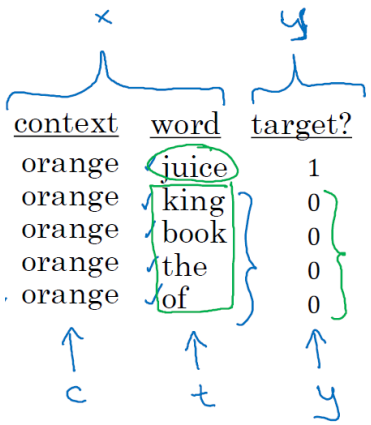
Drawback: Summing over the whole vocabulary in the denominator of the softmax part makes this model computationally expensive

Some methods to improve: hierarchical softmax model.

Negative Sampling

It is set of binary classifiers using logistic regression that aim at assessing how a given context and a given target words are likely to appear simultaneously. with the models being trained on sets of k negative examples and 1 positive example

$$P(y=1 \mid c, t) = \sigma(\theta_t^T e_c)$$



Sentiment classification

Sentiment classification problem

The dessert is excellent.	★★★★☆
Service was quite slow.	★★☆☆☆
Good for a quick meal, but nothing special.	★★★☆☆
Completely lacking in good taste, good service, and good ambience.	★☆☆☆☆